

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ
КАФЕДРА ТЕОРІЇ ТА ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

ПРОГРАМУВАННЯ ТЕОРІЯ ТА ПРАКТИКА

**ЗБІРНИК МАТЕРІАЛІВ
ЗА РЕЗУЛЬТАТАМИ ІТ-ПРОЄКТУ
МІЖДИСЦИПЛІНАРНОЇ ІНТЕГРАЦІЇ**

2024–2025 навчальний рік

За загальною редакцією

Л.Л. Омельчук, О.М. Ткаченка, О.В. Шишацької, Н.Г. Русіної



УДК 004.9(08)

П78

Рецензенти:

канд. фіз.-мат. наук, проф. І. М. Вергунова,
канд. фіз.-мат. наук, доц. Д. В. Затула

*Рекомендовано до друку вченою радою
факультету комп'ютерних наук та кібернетики
(протокол № 16 від 24 червня 2025 року)*

*Ухвалено науково-методичною комісією
факультету комп'ютерних наук та кібернетики
Київського національного університету імені Тараса Шевченка
(протокол № 11 від 23 червня 2025 року)*

Загальна редакція:

Омельчук Л. Л. – кандидат фізико-математичних наук, доцент кафедри теорії та технології програмування

Ткаченко О. М. – кандидат технічних наук, доцент кафедри теорії та технології програмування

Шишацька О. В. – кандидат фізико-математичних наук, доцент кафедри теорії та технології програмування

Русіна Н. Г. – кандидат педагогічних наук, доцент кафедри теорії та технології програмування

Програмування: теорія та практика : зб. матер. за результатами
П78 ІТ-проєкту міждисциплінарної інтеграції. 2024–2025 навчальний рік / за заг. ред.
Л. Л. Омельчук, О. М. Ткаченка, О. В. Шишацької, Н. Г. Русіної. – Київ:
Видавничо-поліграфічний центр "Київський університет", 2025. – 223 с.

ISBN 978-966-933-341-4

Збірник містить публікації студентів за результатами міждисциплінарного проєкту інтеграції на прикладі курсів "Основи управління ІТ-проєктами", "Розробка ПЗ під мобільні платформи", "Soft Skills в інформаційних технологіях", "Інформаційні технології та правовий захист", "Інструментальні середовища та технології програмування" (спеціальність "Комп'ютерні науки").

УДК 004.9(08)

*Матеріали подано в авторській редакції, відповідальність
за достовірність фактів, цитат, посилань на джерела та вживання назв документів,
власних імен тощо несуть автори публікацій.*

ISBN 978-966-933-341-4

© Кафедра теорії та технології програмування, 2025
© Київський національний університет імені Тараса Шевченка,
ВПЦ "Київський університет", 2025

TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV
FACULTY OF COMPUTER SCIENCE AND CYBERNETICS
DEPARTMENT OF THEORY AND TECHNOLOGY OF PROGRAMMING

PROGRAMMING THEORY AND PRACTICE

**MATERIALS
OF THE INTERDISCIPLINARY INTEGRATION
IT PROJECT RESULTS**

2024-2025 academic year

Edited by L. Omelchuk, O. Tkachenko, O. Shyshatska, N. Rusina



UDC 004.9(08)
P78

Reviewers:

PhD in Computer Science, Prof. I. Vergunova,
PhD in Computer Science, Assoc. Prof. D. Zatula

*Recommended for printing by the Academic Council
of the Faculty of Computer Science and Cybernetics
(Protocol № 16, June 24, 2025)*

*Approved by the Scientific and Methodological Commission
of the Faculty of Computer Science and Cybernetics
of Taras Shevchenko National University of Kyiv
(Protocol No. 11 June 23, 2025)*

General Editing:

Omelchuk L. – PhD Computer Science, Associate Professor at the Department of Theory and Technology of Programming

Tkachenko O. – PhD in Computer Science, Associate Professor at the Department of Theory and Technology of Programming

Shyshatska O. – PhD in Computer Science, Associate Professor at the Department of Theory and Technology of Programming

Rusina N. – PhD in Pedagogics, Associate Professor at the Department of Theory and Technology of Programming

Programming: Theory and Practice. Materials of the interdisciplinary integration
P78 IT project results. 2024-2025 academic year / Edited by L.Omelchuk, O.Tkachenko,
O.Shyshatska, N. Rusina. – Kyiv: Publishing and Polygraphic Center "Kyiv University",
2025. – 223 p.

ISBN 978-966-933-341-4

The collection includes student publications based on the results of the interdisciplinary project of integration on the example of courses "Fundamentals of IT project management", "Software development for mobile platforms", "Soft Skills in Information Technologies", "Information Technologies and Legal Protection", "Development Tools and Programming Technologies" (specialty "Computer Science").

UDC 004.9(08)

*The materials are presented in the author's edition.
Authors of the publications are responsible for the accuracy of facts, quotations,
references to sources, and the use of document titles, personal names, etc.*

ISBN 978-966-933-341-4

© Department of Theory and Technology of Programming, 2025
© Taras Shevchenko National University of Kyiv,
Publishing and Polygraphic Center "Kyiv University", 2025

ЗМІСТ

ПЕРЕДМОВА ВІД РЕДАКТОРІВ	7
ІНФОРМАЦІЙНО-КРАУДФАНДИНГОВА ПЛАТФОРМА ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ВОЛОНТЕРІВ "ВОЛОНТЕР+"	
<i>Богдан ЧЕБАН, Денис ШЕВЕЛЬ, Артем БАРИШ, Максим ХОДАКОВ, Євгеній СЬОМКІН</i>	9
РОЗРОБКА СИСТЕМИ "BRIEFLY"	
<i>Кирило БЕДРАК, Олександр ОСТРЕНКО, Матвій КОВАЛИК, Яна МАЛАШКІНА, Юліана КАРПЦОВА, Анастасія МОЗГОВА, Андрій ТОЛСТОП'ЯТИЙ, Владислав ШПАРАГОВСЬКИЙ</i>	54
ВЕБ-ЗАСТОСУНОК "ЗРУЧНОРОЗКЛАД"	
<i>Денис АСТАФ'ЄВ, Сергій ХАЛІК, Костянтин ДЕМЕНТЬЄВ, Арсеній САГАЛЯНОВ, Олександр ПІЩАНЕЦЬКИЙ, Діана СУШКО, Євгенія БАЖМАЄВА</i>	93
МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ СПРОЩЕННЯ ДОГЛЯДУ ЗА ДОМАШНІМИ ТВАРИНАМИ	
<i>Єгор БАЧИНСЬКИЙ, Наталія ГРИЦИШИН, Аріна ЗАБРОДІНА, Олександр КОРНІЄНКО, Катерина ЛИСЕНКО, Максим МАРКОВ, Наталія СЕМЙОН, Антон ТІХОНОВ</i>	116
ЗАСТОСУНОК ДЛЯ ПІДГОТОВКИ ДО НМТ "OSVITA+"	
<i>Олександр СТАРЖИНСЬКИЙ, Владислав ТАРАН, Катерина ШЕМЕТ, Яна УТОЧКІНА, Іван БОЖЕВСЬКИЙ, Максим СОМОВ, Євген ЧОПЕНКО, Владислав КУЧИНСЬКИЙ</i>	156
DOCSDIRECT: ПРОФЕСІЙНА ПЛАТФОРМА СУПЕРВІЗІЙ ДЛЯ ПСИХОЛОГІВ, ПСИХОТЕРАПЕВТІВ ТА ПСИХІАТРІВ	
<i>Володимир КРАВЧУК, Михайло СТАТНІК, Наталія МУСКА, Ігор КОСТЮК, Нікіта МАСИЧ, Микита ІВАНОВ, Катерина КИСІЛЬ, Лаура МОСКАЛЕНКО</i>	195

CONTENTS

FOREWORD FROM THE EDITORS	7
INFORMATION AND CROWDFUNDING PLATFORM TO SUPPORT VOLUNTEER ACTIVITIES "VOLUNTEER+"	
<i>Bohdan CHEBAN, Denys SHEVEL, Artem BARYSH, Maksym KHODAKOV, Yevhenii SOMKIN</i>	9
DEVELOPMENT OF THE "BRIEFLY" SYSTEM	
<i>Kyrylo BEDRAK, Oleksandr OSTRENKO, Matviy KOVALYK, Yana MALASHKINA, Yuliana KARPTSOVA, Anastasiia MOZHOVA, Andriy TOLSTOPIATY, Vladyslav SHPARAHOVSKY</i>	54
WEB APPLICATION "ZRUCHNOROZKLAD"	
<i>Denys ASTAFIEV, Serhiy KHALIK, Kostiantyn DEMENTIEV, Arsenii SAHALIANOV, Oleksandr PISHCHANETSKIY, Diana SUSHKO, Yevheniia BAZHMAEVA</i>	93
MOBILE APPLICATION FOR SIMPLIFYING PET CARE	
<i>Yehor BACHYNSKYI, Nataliia HRYTSYSHYN, Arina ZABRODINA, Oleksandr KORNIENKO, Kateryna LYSENKO, Maksym MARKOV, Nataliia SEMION, Anton TIKHONOV</i>	116
APPLICATION FOR NMT PREPARATION "OSVITA+"	
<i>Oleksandr STARZHYNKYI, Vladyslav TARAN, Kateryna SHEMET, Yana UTOCHKINA, Ivan BOZHEVSKYI, Maksym SOMOV, Yevhen CHOPENKO, Vladyslav KUCHYNSKYI</i>	156
DOCDIRECT: AN ADVANCED SUPERVISION PLATFORM FOR PSYCHOLOGISTS, PSYCHOTHERAPISTS AND PSYCHIATRISTS	
<i>Volodymyr KRAVCHUK, Mykhailo STATNIK, Natalia MUSKA, Igor KOSTIUK, Nikita MASYCH, Mykyta IVANOV, Kateryna KYSIL, Laura MOSKALENKO</i>	195

ПЕРЕДМОВА ВІД РЕДАКТОРІВ

Шановні читачі!

До вашої уваги представлено п'ятий щорічний випуск збірника *"Програмування: теорія та практика"*, що є результатом реалізації міждисциплінарного ІТ-проєкту інтеграції у 2024–2025 навчальному році. У ньому вміщено публікації студентських команд, які протягом двох семестрів працювали над розв'язанням реальних задач у межах командної розробки програмного забезпечення, поєднуючи знання з ІТ-дисциплін освітньо-професійної програми першого (бакалаврського) рівня вищої освіти *"Інформатика"* (спеціальність *"Комп'ютерні науки"*) та програми *"Прикладна (комп'ютерна) лінгвістика та англійська"* (спеціальність *"Філологія"*, спеціалізація *"Прикладна лінгвістика"*).

У першому семестрі обов'язкову участь у проєкті брали студенти 4-го курсу ОПП *"Інформатика"* у межах вибіркової дисципліни *"Основи управління ІТ-проєктами"* (викладач: Шишацька О.В.) та обов'язкової дисципліни *"Інформаційні технології"* (лектор: Ткаченко О.М.; викладачі лабораторних занять: Свистунов А.О., Терещенко Я.В.). Крім того, за бажанням долучалися студенти 3-го курсу з вибіркової дисципліни *"WEB-технології"* (викладач: Свистунов А.О.) та студенти ОПП *"Прикладна (комп'ютерна) лінгвістика та англійська мова"* – з дисципліни *"Бази даних та бази знань"* (викладач: Терещенко Я.В.).

У другому семестрі обов'язкову участь у проєкті брали студенти 4-го курсу ОПП *"Інформатика"* з дисциплін *"Soft Skills в інформаційних технологіях"* (викладач: Ткаченко О.М.) та *"Інформаційні технології та правовий захист"* (викладач: Русіна Н.Г.). За бажанням до команд долучалися студенти 2-го курсу, які вивчали дисципліну *"Інструментальні середовища та технології програмування"* (лектор і викладач лабораторних занять: Омельчук Л.Л.; викладачі лабораторних занять: Свистунов А.О., Шишацький А.В., Русіна Н.Г., Галавай О.М.), а також студенти з ОПП *"Прикладна (комп'ютерна) лінгвістика та англійська"*.

Представлені проєкти охоплюють широкий спектр тем: застосунок для волонтерів; інструмент для підсумовування повідомлень, текстів і документів; система автоматизації складання та управління навчальними розкладами; застосунок для догляду за домашніми тваринами; тренажер для підготовки до НМТ; вебплатформа супервізій для фахівців з психології.

Усі розробки створювалися з урахуванням повного життєвого циклу програмного продукту – від ідеї до реалізації MVP та публічного захисту. Студенти не лише поглибили технічні компетентності, а й удосконалили навички управління проєктами, командної роботи, презентації результатів і критичного мислення.

Менторами, які щотижня супроводжували роботу студентських команд, виступили викладачі кафедри теорії та технології програмування: Омельчук Л.Л., Ткаченко О.М., Шишацька О.В., Русіна Н.Г., Свистунов А.О., Шишацький А.В., Терещенко Я.В., Галавай О.М.

На фінальний захист проєктів було запрошено представників ІТ-галузі та закладів вищої освіти України.

Щиро дякуємо всім учасникам – студентам, викладачам і рецензентам – за активну та натхненну працю. Висловлюємо особливу подяку колегам з Навчально-наукового інституту філології, зокрема гаранту ОПІ *"Прикладна (комп'ютерна) лінгвістика та англійська мова"* Зубань О.М., за плідну співпрацю, яка ще раз довела ефективність міждисциплінарного підходу в сучасній IT-освіті.

Сподіваємося, що представлений збірник стане джерелом натхнення для майбутніх досліджень, спільних ініціатив і нових освітніх форматів.

З повагою

Редакційна колегія:

Людмила **ОМЕЛЬЧУК**

Олексій **ТКАЧЕНКО**

Олена **ШИШАЦЬКА**

Наталія **РУСІНА**

ІНФОРМАЦІЙНО-КРАУДФАНДИНГОВА ПЛАТФОРМА ДЛЯ ПІДТРИМКИ ДІЯЛЬНОСТІ ВОЛОНТЕРІВ "ВОЛОНТЕР+"

*Богдан ЧЕБАН, Денис ШЕВЕЛЬ, Артем БАРИШ,
Максим ХОДАКОВ, Євгеній СЬОМКІН*

Дана робота описує розробку та впровадження платформи "Волонтер+", що об'єднує військових, волонтерів і донорів для оперативної, прозорої координації гуманітарної допомоги в умовах кризових ситуацій. Система забезпечує створення та обробку запитів, збір коштів, звітність і аналітику, використовуючи сучасні веб-технології, інтеграцію з платіжними сервісами і високі стандарти безпеки. Платформа підвищує ефективність розподілу ресурсів і залучення громадськості до волонтерства.

ВОЛОНТЕРСТВО, КРАУДФАНДИНГ, ПЛАТФОРМА, КООРДИНАЦІЯ ДОПОМОГА,
БЛАГОДІЙНІСТЬ, ВІЙСЬКОВОСЛУЖБОВЦІ, БЕЗПЕКА, АНАЛІЗ

INFORMATION AND CROWDFUNDING PLATFORM TO SUPPORT VOLUNTEER ACTIVITIES "VOLUNTEER+"

*Bohdan CHEBAN, Denys SHEVEL, Artem BARYSH, Maksym KHODAKOV,
Yevhenii SOMKIN*

This paper presents the development and implementation of the Volunteer+ platform that unites military personnel, volunteers, and donors to ensure fast and transparent coordination of humanitarian aid in crisis situations. The system enables creation and processing of requests, fundraising, reporting, and analytics, leveraging modern web technologies, mobile apps, payment-gateway integration, and robust security. The platform increases efficiency of resource distribution and fosters public engagement in volunteering.

VOLUNTEERING, CROWDFUNDING, PLATFORM, COORDINATION, AID, CHARITY,
MILITARY PERSONNEL, SECURITY, ANALYSIS

ВСТУП

Постановка задачі. Створити єдину веб-платформу, що дає можливість військовим оперативно формувати запити на підтримку, а волонтерам і донорів – швидко їх знаходити та прозоро закривати.

Актуальність теми. Умови воєнних дій та надзвичайних ситуацій вимагають термінової координації ресурсів; існуючі рішення не гарантують прозорості фінансів і зручної взаємодії всіх сторін.

Методи дослідження. Аналіз конкурентного середовища, проведення глибоких інтерв'ю з кінцевими користувачами, SWOT-аналіз, прототипування UI/UX, тестування MVP.

Цілі та завдання. Прикладні: забезпечити швидку та прозору координацію допомоги; підвищити довіру донорів і волонтерів до процесу; зменшити час від запиту до закриття потреби. Професійні: закріпити знання з веб-технологій (React, Spring Boot, MariaDB), поглибити досвід інтеграції платіжних сервісів та реалізації безпечної аутентифікації та авторизації.

1. ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

Опис процесу пошуку ідеї продукту

Процес генерування та валідації концепції "Волонтер+" складався з кількох етапів:

- *онлайн дослідження*: аналіз наукових публікацій і звітів із гуманітарної допомоги для виявлення типових проблем у кризових зонах [1], огляд існуючих ІТ рішень (GoFundMe, Team Rubicon, Razom for Ukraine) з акцентом на їхній функціонал, UX/UI, швидкість реагування та прозорість фінансових потоків [2];

- *користувацькі інтерв'ю та опитування*: проведено декілька інтерв'ю з представниками різних груп: військові, волонтери та інші, онлайн опитування для ранжування ключових болючих точок: відсутність централізованого обміну запитами, непрозорість витрат, низька мотивація волонтерів через брак фідбеку;

- *фокус-групи та воркшопи*: два воркшопи за участі фахівців із іт та представників благодійних фондів для пошуку інноваційних підходів до інтеграції платіжних систем, звітності та комунікацій, карта емпатії користувачів для формалізації їхніх очікувань і страхів.

SWOT аналіз і бізнес-моделювання

У межах стратегічного аналізу було проведено оцінку внутрішніх і зовнішніх чинників, що впливають на розвиток платформи "Волонтер+". Результати SWOT-аналізу наведено на рис. 1.

SWOT-аналіз

	Корисно для досягнення цілі	Шкодить to achieving the objective
Внутрішні властивості організації	Сильні сторони: <ul style="list-style-type: none">Актуальність через потребу в прозорій інформації про військо.Можливість створити унікальний рейтинг для стимулювання покращень і конкуренції.	Слабкі сторони: <ul style="list-style-type: none">Важко отримати достовірні дані від військових та держструктур.Високі вимоги до безпеки даних.
Зовнішні властивості оточення	Можливості: <ul style="list-style-type: none">Потенційне партнерство з Міністерством Оборони України для офіційної підтримки.Розширення платформи для підтримки ветеранів та	Ризики: <ul style="list-style-type: none">Стати жертвами ворожого ІПСОКонкуренція з подібними платформами, якщо такі з'являться.

Рис. 1. SWOT-аналіз концепції Волонтер+

Для оцінки життєздатності ініціативи розроблено попередню Business Model Canvas, що охоплює ключові елементи функціонування платформи:

- *ключові партнери*: волонтерські організації, ЗСУ, ІТ-компанії, фонди підтримки, банківські установи;

- *ключові види діяльності*: розробка та підтримка платформи, модерація контенту, обробка запитів, підтримка користувачів.

- *ціннісні пропозиції*: швидке реагування на потреби військових, прозорість зборів, безпечна та надійна комунікація;

- *взаємовідносини з клієнтами*: підтримка 24/7, верифікація користувачів, зворотний зв'язок, автоматичне інформування про статус.
- *канали збуту*: веб-платформа, соціальні мережі, електронна пошта, потенційно – мобільний застосунок;
- *цільові сегменти*: волонтери, військові підрозділи, донори, адміністрації бригад;
- *ключові ресурси*: команда розробників, база користувачів, система зборів і звітності, хостинг, підтримка даних;
- *структура витрат*: розробка, сервери, підтримка, перевірка користувачів, реклама, юридичний супровід;
- *джерела доходу/фінансування*: добровільні внески, гранти, донати на розвиток платформи, партнерські програми.

Опис обраної сфери для створення продукту

Для створення продукту було обрано сферу цифрової координації гуманітарної та волонтерської допомоги в кризових умовах.

Сфера цифрового волонтерства є доволі новою для світового руху волонтерства. Для нашої країни тема волонтерства є ключовою у боротьбі за суверенітет та незалежність оскільки дозволяє закрити специфічні проблеми в забезпеченні війська. Потреба в полегшенні та швидкої координації таких ініціатив впливає з нагальної потреби військових. Цифровізація, прозорість та надійність у цифровому волонтерстві є одними з ключових та пріоритетних.

Перелік обговорюваних ідей

Під час початкового етапу проекту було розглянуто низку ідей, кожна з яких мала на меті вирішення актуальних проблем у сфері волонтерства, логістики та допомоги. Кожне рішення пропонувало унікальний функціонал – від координації евакуації до автоматизації доставки медикаментів чи краудфандингу. Після аналізу варіантів було обрано платформу "Волонтер+" як найбільш комплексне та актуальне рішення для реалізації. У таблиці 1 подано короткий опис усіх розглянутих ідей[6].

Таблиця 1. Перелік ідей для платформи координації допомоги

Ідея	Опис функціоналу
Brigade+	Формування локальних волонтерських бригад з можливістю делегування завдань та відстеження прогресу в реальному часі
Волонтер+	Єдина платформа для створення запитів, збору коштів, прозорого звітування через фото/відео та рейтингової системи учасників (обрана для реалізації)
MedSupply	Автоматизована система замовлення та доставки медичних препаратів і обладнання в кризові райони з інтеграцією з лікарнями та волонтерами
EvacHub	Координація евакуаційних заходів із мапою безпечних маршрутів та центрами збору, включно з модулем бронювання транспорту
ShelterNet	Платформа для управління житлом та тимчасовими притулками, з опціями бронювання місць, моніторингу доступності та зворотного зв'язку
CrowdFund+	Мультиканальна система краудфандингу з інтеграцією соцмереж, API для корпоративних донорів і розширеною аналітикою залучених коштів
SkillShareAid	Платформа для обміну волонтерськими навичками (медицина, будівництво, логістика) із механізмом верифікації кваліфікацій та сертифікатами для волонтерів
RapidResponse	Мобільний додаток для миттєвого оповіщення найближчих волонтерів про нові запити з можливістю GPSтрекінгу і планування маршруту

Порівняльний аналіз показав, що "Волонтер+" забезпечує найбільш комплексний набір функцій для створення запитів, збору коштів, прозорого звітування й рейтингової системи, тому саме ця платформа була обрана для подальшої реалізації та розвитку.

Аргументація вибору сфери та ідеї "Волонтер+"

– *ринкова ніша*: існуючі сервіси або фрагментарні (лише збір коштів або лише координація), або не мають повної інтеграції трьох груп користувачів; "Волонтер+" закриває цю прогалину ;

– *соціальний вплив*: залучення волонтерів та донорів до прозорих процесів підвищує довіру громадськості та ефективність гуманітарних ініціатив.

Огляд наявних на ринку ІТ-продуктів – конкурентів

Сервіс GoFundMe

Позиціонування. Глобальний краудфандинговий сервіс "для будь-яких цілей" – від особистих медичних витрат до гуманітарних ініціатив.

Ключові функції. Створення кампаній за 3 кроки; інтеграція із соцмережами; динамічний лічильник зібраних коштів; вбудовані платіжні шлюзи.

Цільова аудиторія. Фізичні особи та неприбуткові організації, що шукають масову підтримку.

Географія. Понад 20 країн; основний трафік – Північна Америка та Західна Європа.

Мовні локалізації. Англійська, французька, німецька, іспанська.

Модель монетизації. Комісія 2,9 % + \$0,30 з кожної транзакції.

Доступ/застосунки. Веб-латформа та офіційні додатки Android і iOS; ≈1,2 млн завантажень на місяць (AppMagic) [5].

Трафік. ≈25 млн відвідувань/міс (SimilarWeb) [5].

Соцмережі й блог. Активні FB, IG, X; регулярний корпоративний блог.

Переваги. Масова пізнаваність, мінімальний поріг входу, зручний UX (див. рис. 2).

Недоліки. Відсутні інструменти "запит-волонтер"; немає фокусу на військових потребах; комісія знижує ефективність зборів. [5]

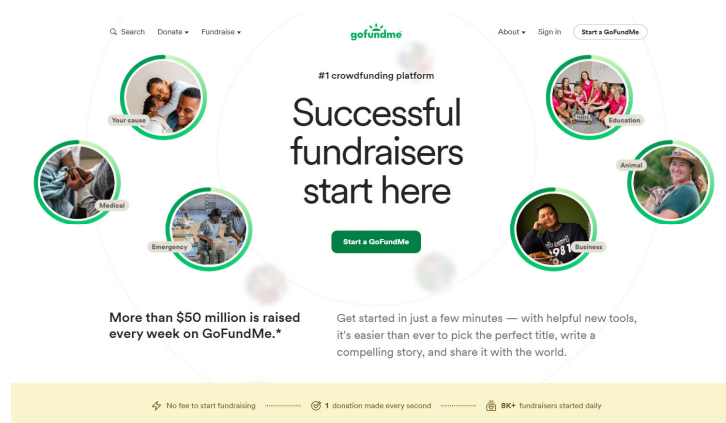


Рис. 2. GoFundMe

Організація Team Rubicon USA

Позиціонування. Неприбуткова організація, що мобілізує ветеранів для реагування на стихійні лиха та гуманітарні кризи.

Ключові функції. Формування спеціалізованих польових команд; цифровий облік місій; afteraction звітність.

Цільова аудиторія. Ветерани Збройних сил США; партнерські НКО; постраждалі громади[6].

Географія. Територія Сполучених Штатів (іноді – міжнародні місії).

Мовні локалізації. Англійська.

Модель монетизації. Фонди, гранти, приватні пожертви; користувачі не сплачують комісій.

Доступ/застосунки. Лише веб-портал (див. рис. 3).

Трафік. $\approx 0,5$ млн відвідувань/міс (SimilarWeb) [5].

Соцмережі й блог. FB, IG, X; детальні репортажі з місій у блозі.

Переваги. Високий рівень професіоналізму; офлайн-модуль "команди на місці".

Недоліки. Географічно обмежена платформа[8].

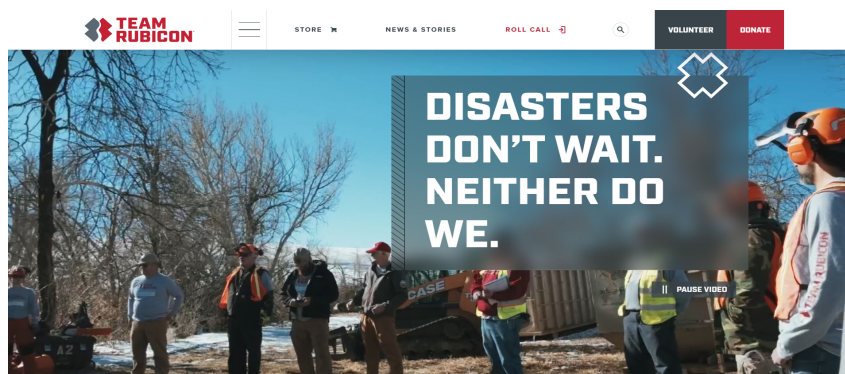


Рис. 3. Team Rubicon USA

Організація Razom for Ukraine

Позиціонування. Благодійний хаб для фінансування гуманітарних та військово-медичних проєктів в Україні.

Ключові функції. Таргетовані збори; фото/відеозвіти; партнерські програми з фондами.

Цільова аудиторія. Українська діаспора, міжнародні донори, громадські об'єднання.

Географія. Міжнародна, з фокусом на Україну.

Мовні локалізації. Українська, англійська.

Модель монетизації. Неприбуткова; адміністративні витрати покриваються окремими грантами.

Доступ / застосунки. Веб-платформа.

Трафік. $\approx 0,3$ млн відвідувань/міс (SimilarWeb) [5].

Соцмережі й блог. FB, IG, X; активний блогдайджест зі звітами.

Переваги. Вузька спеціалізація; максимальна прозорість витрат.

Недоліки. Немає прямого каналу "запит-волонтер" (див. рис. 4).

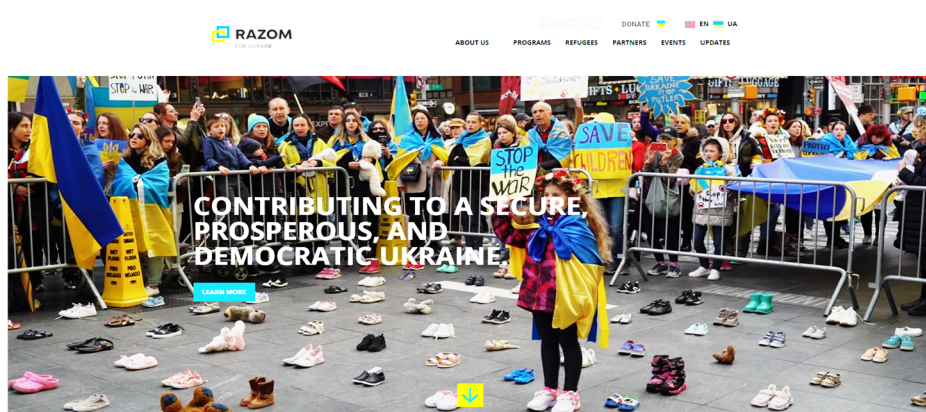


Рис. 4. Razom for Ukraine

Мапа продукту "Волонтер+":

Проблема. Військові та цивільні підрозділи, які працюють у зоні бойових дій, стикаються з трьома критичними викликами:

- відсутність єдиної онлайн-платформи, де можна оперативно сформулювати потребу та отримати відгук;
- низька прозорість фінансових потоків, що демотивує донорів [1];
- розрізнена комунікація між самими волонтерами, яка призводить до дублювання зусиль [3].

"Волонтер+" вирішує ці виклики, надаючи одну веб-платформу із повним циклом "запит → збір → звіт".

Сегменти клієнтів. Визначаються за наступним критерієм:

- військовослужбовці – створюють запити, відстежують виконання;
- волонтери – обирають завдання за географією/кваліфікацією, кооперуються в чатах, звітують;
- донори – фізичні особи й бізнес, що фінансово підтримують запити та стежать за фотозвітами;
- унікальна ціннісна пропозиція.

"Волонтер+" – передова українська інформаційно-краудфандингова платформа, створена на основі ключових принципів цифрового волонтерства: "запит → збір → звіт", прозорості та відкритості звітності. Система забезпечує прямий зв'язок між військовими та волонтерами – запити надходять безпосередньо від військовослужбовців, що мінімізує бюрократію, підвищує оперативність реагування та гарантує актуальність потреб. Завдяки цьому платформа сприяє ефективному розподілу ресурсів, зміцнює довіру донорів і формує нову культуру цифрової взаємодії в умовах війни. [2,8]

Рішення які пропонує платформа:

- для військових – уніфікована форма "створити запит" із шаблонами найбільш затребуваних категорій та автоматичною валідацією основних полів під кожену категорію;
- для донорів – можливість переглядати сформовані волонтерами звітів після успішного закриття зборів разом з інформацією про фінансові транзакції;
- для волонтерів – чати комунікації як з реальними користувачами (в тому числі волонтерів), так і з спеціально натренованими AI-асистентами;

Різноманітні канали фінансової підтримки військових

- фінансування за рахунок державних програм та грантів;
- добровільна підтримка від користувачів – регулярна підтримка у вигляді щомісячних внесків або у форматі одноразових транзакцій;
- брендowana продукція – частина коштів з продажу йде на підтримку платформи;
- ліцензування платформи для використання в інших країнах.

Структура витрат

Для запуску та ефективного функціонування платформи "Волонтер+" передбачено комплекс витрат, що охоплює всі ключові етапи створення цифрового продукту. Перш за все – це розробка, яка включає попередні дослідження, розробку структури сервісу, UX/UI-дизайн інтерфейсу, безпосереднє програмування, а також етап тестування і фінального запуску платформи. Успішне виконання цих завдань закладає основу для зручного, стабільного й масштабованого сервісу.

Окремий пріоритет – безпека. Витрати в цій категорії охоплюють реалізацію захисту даних користувачів, впровадження сучасних методів шифрування, проведення

незалежного аудиту кібербезпеки, а також забезпечення відповідності стандартам через отримання відповідних сертифікатів. Для гарантованої надійності також передбачено безпечний хостинг із постійною технічною підтримкою.

Людський ресурс – ще одна важлива стаття бюджету. У команду входять розробники, проєктний менеджер, дизайнер, спеціалісти з комунікацій (PR/SMM), технічна підтримка та координатор волонтерського сегмента. Кожен з них відповідає за окремий напрям реалізації та підтримки платформи.

Адміністративні витрати включають оплату юридичних та бухгалтерських послуг, реєстрацію, отримання необхідних дозволів, ведення документообігу. Ці витрати забезпечують легальність і прозорість діяльності платформи відповідно до вимог чинного законодавства.

Маркетингова стратегія вимагає фінансування для просування платформи серед цільової аудиторії. До витрат належать реклама в соціальних мережах і медіа, створення промовідео, друкованої продукції та організація PR-кампаній для залучення нових користувачів і донорів.

Після запуску також необхідні витрати на підтримку проєкту: це регулярна оплата хостингу, домену, оновлення системи, сервісна підтримка користувачів і резервні кошти на випадок непередбачених ситуацій. Усі ці складові дозволяють забезпечити безперебійну роботу платформи в довгостроковій перспективі.

Ключові метрики


Платформа "Волонтер+" оцінює свою ефективність через низку ключових метрик. Серед основних – MAU (місячна активна аудиторія), що показує рівень залучення військових, волонтерів і донорів. Важливою є також швидкість реакції: фіксується час від моменту публікації запиту до першої відповіді волонтера.

Прозорість контролюється через відсоток зборів, за якими волонтери надали звіти, та загальну суму зібраних коштів щомісяця. Для аналізу маркетингової ефективності використовуються такі показники, як Net Promoter Score, CPC, CPA, CPL і середній донат. Це дозволяє не лише вимірювати результати, а й оперативного вдосконалювати платформу.

Прихована перевага полягає в її комплексній технічній архітектурі, яка готова до горизонтального й вертикального масштабування та вже включає інтеграцію з AI-моделями (див. рис. 5).

ВОЛОНТЕР+

Увійти
Створити профіль



Волонтер
**Васильченко
Володимир
Іванович**

- 5 років на Волонтер+
- Виконано 1212 запитів
- Опубліковано 2322 звіти

Збори

Тепловізор для розвідки вночі

Техніка 93-тя окрема механізована бригада "Скіфія"

Збір на тепловізор для розвідки 3-го батальйону. Хлопці щодня виходять на позиції вночі, і бачити першого — означає вилітати.

Зібрано	Ціль
25 350грн	58 000грн

Підтримати

Дрон Mavic 3 для коригування вогню

Дрони 54-та окрема артилерійська бригада ім. Сірка

Зібрано	Ціль
71 000грн	119 000грн

Підтримати

Аптечки для бійців на "нулі"

Мідрасамети 72-га окрема механізована бригада "Сн"

Збираємо на 20 індивідуальних аптечок для хлопців, які щодня залітали на передову. У кожен — турникети, ізраїльські бандажі, протиголови.

Зібрано	Збір закрито
71 460грн	

Переглянути звіт

Генератор для польового шпиталю

Дрони Для 3 ОШБ

У польовому медпункті не завжди є електрика. Генератор забезпечить стабільну роботу обладнання під час евакуації та операції.

Зібрано	Збір закрито
11 467грн	

Переглянути звіт

Рис. 5. Дашборд з актуальними зборами

Завдяки такій стратегії "Волонтер+" може оперативно реагувати на збільшення навантаження й швидко впроваджувати нові аналітичні та автоматизовані сервіси, що підвищує ефективність підтримки актуальних зборів і якість прийняття рішень.

Бізнес модель

Платформа працює за моделлю "Impact First", без обов'язкових комісій чи платного доступу. Єдиним джерелом операційного доходу є добровільні пожертви від користувачів, які хочуть підтримати розвиток платформи. Донори можуть окремо додати суму на підтримку сервісу, не урізаючи основний збір.

Для масштабування інфраструктури передбачено грантову підтримку (наприклад, USAID Digital Impact Track, Emergency Support Fund) [6], яка не враховується в операційній P&L, а використовується виключно для розвитку технічної бази та безпеки. Такий підхід дозволяє зберігати максимальну прозорість і довіру аудиторії, роблячи сервіс справді громадським та незалежним.

Маркетингова стратегія

Платформа "Волонтер+" просуватиметься як прозоре, технологічне та суспільно важливе рішення для ефективної взаємодії між донорами, волонтерами та військовими. Основна увага буде зосереджена на створенні довіри, впізнаваності та постійній присутності в інформаційному просторі.

Цільова аудиторія – це активні донори, волонтери, благодійні фонди, українська діаспора та корпоративні меценати. Стратегія передбачає активну присутність у соціальних мережах, насамперед Instagram, Facebook, TikTok, де публікуватимуться реальні історії допомоги, відеозвіти та заклики до участі. Акцент робиться на візуальні матеріали, бо саме вони найкраще передають емоційну складову підтримки.

Партнерство з медіа, волонтерськими об'єднаннями та лідерами думок посилить охоплення. Важливу роль відіграватимуть амбасадори проєкту – відомі люди або організації, які публічно підтримають платформу. Також передбачено впровадження системи довіри: користувачі бачитимуть історію активності волонтерів і зборів, а це формуватиме репутацію на основі реальних дій.

Технічна частина просування передбачає інтеграцію з сервісами аналітики та реклами, що дозволить відстежувати трафік, конверсії, залучення. Крім того, регулярно публікуватиметься відкрита статистика – кількість зборів, їхній статус, географія допомоги – це створює додаткову цінність і викликає довіру.

Основна мета стратегії – сформувати образ "Волонтер+" як сучасної, чесною та ефективною платформи, де кожна гривня і кожне зусилля мають підтверджений результат.

SWOT аналіз

Сильні сторони:

- висока актуальність через потребу у прозорій інформації про військові формування, керівництво та стан забезпечення;
- можливість створити унікальну рейтингову систему, яка стимулюватиме покращення умов та конкуренції серед військових частин;

Слабкі сторони:

- складність отримання достовірної інформації від військових та державних структур;
- потреба у високому рівні безпеки для захисту критично важливих даних;

Ризики:

- стати жертвами ворожого ПІСО;
- конкуренція з подібними платформами, якщо такі з'являться;

Можливості:

- потенційне партнерство з Міністерством Оборони України для офіційної підтримки;
- розширення платформи для підтримки ветеранів та їхніх родин.

Огляд використаних технологій та обґрунтування

Backend – Spring Boot/Data/Hibernate. Обрано за стабільність, готові модулі безпеки (OAuth 2.0, JWT) та велику кількість opensource плагінів. Це зменшує timetofix для CVE та спрощує сертифікацію [1]. Також Spring Boot є стандартом в розробці ентерпрайз рівня додатків для міжнародних фінансових організацій. Hibernate використовується як ORM фреймворк для підключенної СКБД.

Реляційна система керування базами даних – MariaDB MariaDB пріоритезує увагу на тому, щоб не порушувати зворотну сумісність для своїх користувачів. Оновлення старих версій MySQL навіть до найновіших версій MariaDB підтримується оновленням на місці. Сервер MariaDB забезпечує режим сумісності синтаксису з OracleDB що корисно для інтеграцій з міжнародними банківськими системами таких як Eagle Investment Systems[13].

Frontend – React + TypeScript. React обрано для розробки "Волонтер+" завдяки його швидкодії, масштабованості та зручності в роботі з динамічним контентом. Компонентна структура дозволяє створювати повторно використовувані елементи (запити, звіти, статуси зборів), що спрощує підтримку. Бібліотека має велику екосистему (Redux, React Router, Axios) та активну спільноту, що прискорює розробку. Крім того, React сумісний із React Native – це відкриває шлях до мобільної версії в майбутньому.

DevOps – Docker[28]. Проєкт складається з кількох сервісів, які запускаються разом за допомогою docker-compose.yml. Основні компоненти:

- Frontend (Vite) – інтерфейс користувача, з live reload;
- Backend (Spring Boot) – основна логіка, підключається до бази, Redis, AI-моделі й трасування;
- MariaDB[13] – база даних для зберігання даних;
- Redis + RedisInsight – кеш та інтерфейс для його моніторингу;
- Ollama – локальний сервер LLM для AI-функцій;
- OpenTelemetry / Jaeger – збір і візуалізація трасування запитів.

Безпека – RateLimit / Spring Security[25]. У нашому проєкті використовується Spring Security для автентифікації користувачів за допомогою JWT, що забезпечує безпечний доступ до API. Для запобігання зловживанням та забезпечення справедливого використання ресурсів впроваджено обмеження частоти запитів (Rate Limiting) для кожного користувача. Це означає, що кожен користувач має власне обмеження на кількість запитів, які він може зробити за певний проміжок часу, що підвищує стабільність та безпеку системи

AI – Spring AI. Spring AI – це проєкт від Spring, який спрощує інтеграцію LLM (великих мовних моделей), таких як OpenAI, Ollama, HuggingFace, Azure OpenAI, у Spring Boot застосунки. Це частина Spring екосистеми (як Spring Data, Spring Security[25]), але для роботи з AI. На платформі використані такі провайдери великих мовних моделей як: OpenAI та Ollama що надають повний функціонал використання AI моделей в корпоративних системах (RAG, Tool calling).

Cache – Redis. Redis (REmote DIctionary Server) – це сховище даних у пам'яті (in-memory key-value store) з дуже високою швидкістю доступу. Підтримує різні типи структур даних: рядки, множини, списки, хеші, потоки, біти тощо. Було використано

саме для кешування, в якості векторного сховища даних для вбудованих моделей та як брокер повідомлень для веб сокету.

На рисунку – інтерфейс RedisInsight, під'єднаний до бази Redis (redis:6379, db0), із списком ключів типу JSON (з префіксами redis-ollama-prefix... та redis-open-ai-prefix...). Ліва панель показує назви й розміри ключів, права – деталі вибраного ключа: об'єкт із полями embedding, source (ім'я документа) та content (зразок українського тексту з описом ДСТУ для хлібобулочних і молочних виробів) (див. рис. 6).

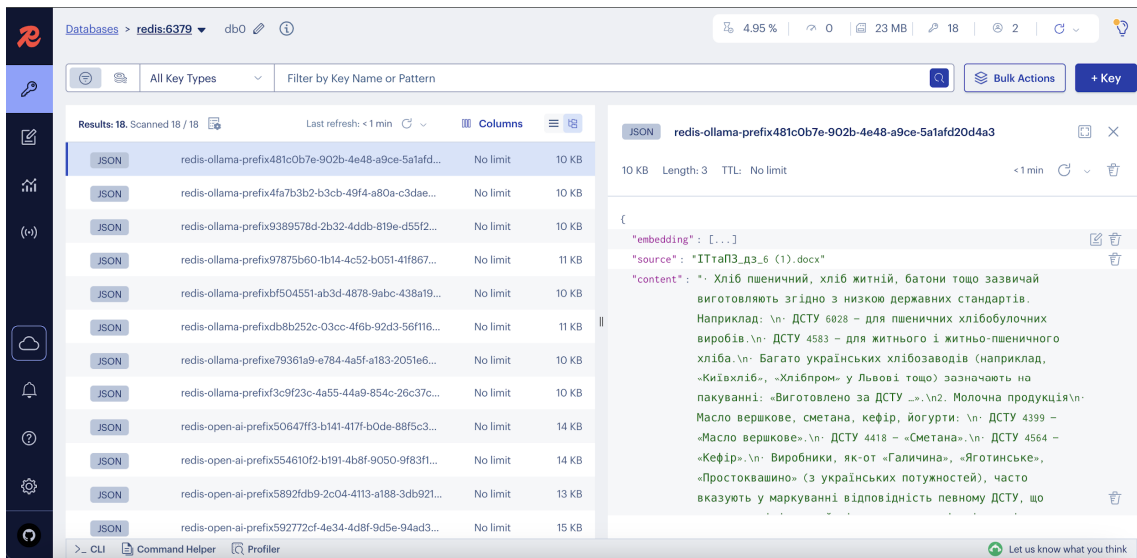


Рис. 6. Редіс дашборд

RedisInsight значно полегшує відлагодження Redis Vector Store та оперативний доступ до збережених ембедингів і пов'язаного контенту.

Веб-сокет – Spring WebSocket. Це частина Spring Framework та універсальний інтерфейс, який дозволяє легко реалізувати WebSocket-сервер у Spring Boot застосунку з підтримкою STOMP (протокол обміну повідомленнями поверх WebSocket) обробниками повідомлень (@MessageMapping) темами (channels) і користувачькими сесіями разом з додатковими засобами аутентифікації.

Аналітика – Jaeger/OpenTelemetry. Jaeger та OpenTelemetry використовуються для спостережуваності (observability), а саме – трасування запитів, моніторингу продуктивності та відлагодження в розподілених запитах таких як LLM виклики або Redis[26] запити на дані.

Для аналізу продуктивності та трасування запитів у нашому бекенд-сервісі використовується Jaeger UI. Нижче наведено головний дашборд, де зліва задаються фільтри (сервіс, операція, часовий інтервал), а праворуч – графік розподілу трейсів за часом виконання та список останніх запитів із деталями спенів. Це дозволяє швидко виявляти повільні виклики та досліджувати їхню внутрішню структуру (див. рисунок 7).

На рисунку представлено інтерфейс головного дашборду Jaeger UI, де в панелі зліва налаштовано фільтрацію трейсів за сервісом backend та часовим інтервалом. Праворуч відображено графік розподілу трейсів за часом виконання та їхній список із зазначенням кількості спенів, часу запуску та тривалості кожного запиту.

Для глибшого розуміння того, як проходить обробка HTTP-запиту /api-docs у бекенд-сервісі, виконується детальний аналіз конкретного трейсу в інтерфейсі Jaeger UI.

Нижче наведено розгортку спенів із їхнім власним ("Self") та сукупним ("Total") часом виконання (див. рис. 8).

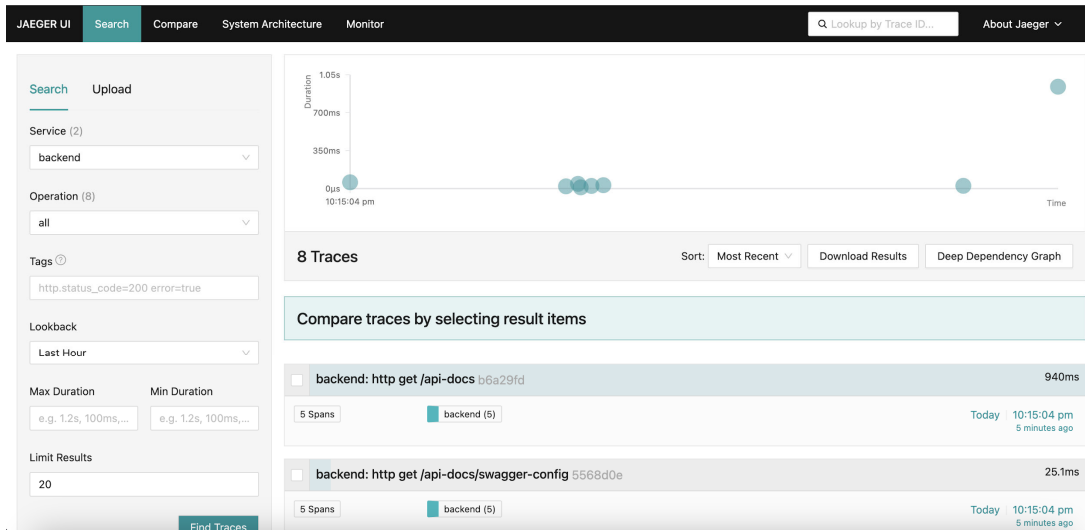


Рис. 7. OpenTelemetry/Jaeger основний дашборд

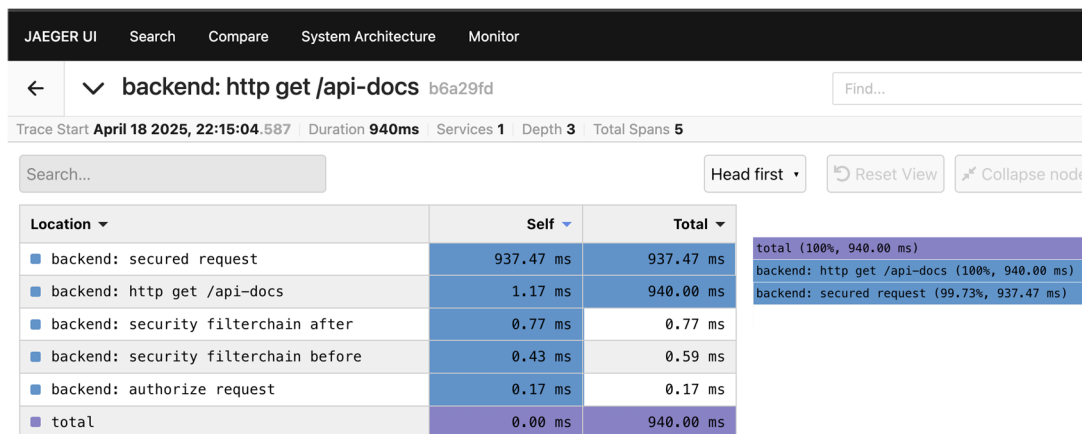


Рис. 8. OpenTelemetry/Jaeger аналіз API запиту

На даному прикладі видно, що найбільшу частину часу (937,47 мс) займає спен "secured request", тоді як фільтри безпеки та авторизація відпрацьовують за менш ніж 1 мс. Це дозволяє сконцентрувати оптимізацію саме на захищеному шарі обробки запитів.

У вступі викладено загальну мету дослідження – розробку ІТ-платформи "Волонтер+", яка об'єднає волонтерів, донорів та громадські організації для оперативного фінансування гуманітарних ініціатив. Обґрунтовано актуальність проекту з огляду на потреби кризових регіонів України та описано основні завдання, що стоять перед командою – від централізації запитів до забезпечення прозорості розподілу коштів.

Розділ "Пошук та валідація ідеї" детально описує процес аналізу потреб цільової аудиторії і формування концепції платформи. Спочатку проводилися бенчмарки й обговорення можливих напрямів, потім аргументовано обрано створення системи, яка покликана вирішити проблему розпорошеності волонтерських ініціатив. У цьому ж розділі наведено огляд конкурентів: платформи GoFundMe, волонтерської організації Team Rubicon USA та сервісу Razom for Ukraine, завдяки чому були визначені сильні та слабкі сторони існуючих рішень.

Далі представлено "Мапу продукту "Волонтер+" – описано ключові сегменти користувачів (волонтери, донори, організації), основні функції платформи (валідація запитів, автоматизований розподіл фінансів), різноманітні канали залучення коштів (краудфандинг, прями донати, гранти), структуру витрат (операційні, маркетингові, комісійні збори), а також ключові показники ефективності (кількість активних кампаній, обсяг зібраних коштів, швидкість реакції). Особливу увагу приділено прихованій перевазі – інтеграції з військовими та гуманітарними мережами, що підвищує рівень довіри до платформи.

У розділі "Бізнес-модель" викладено джерела доходу проєкту, котрі включають комісії з транзакцій та продаж преміум-функцій для організацій, окреслено маркетингову стратегію: SMM-кампанії, партнерські програми з НГО та таргетовану рекламу.

Нарешті, у SWOT-аналізі проєкту згруповано його сильні сторони (унікальна спрямованість на кризову підтримку, налагоджені зв'язки з військовими), слабкі – пов'язані з відсутністю впізнаваного бренду та обмеженими ресурсами на старті, ризики (регуляторні виклики та конкуренція з глобальними платформами) та можливості для зростання (вихід на міжнародні ринки та розширення сервісів, наприклад логістики чи юридичної підтримки).

У наступному розділі "Огляд використаних технологій та обґрунтування" описано, чому для реалізації "Волонтер+" було обрано сучасний стек: серверна частина на Java 17 із Spring Boot забезпечила швидку розробку, гнучке конфігурування REST- та WebSocket-інтерфейсів (STOMP/SockJS) для реального часу; Spring AI з Redis Vector Store дозволив інтегрувати семантичний пошук і зберігати ембеддинги в низьколатентному сховищі; Quartz Scheduler із Flyway гарантував надійне планування задач і контроль версій бази даних; OpenTelemetry і Jaeger – повну трасировку запитів для швидкого виявлення "вузьких місць"; фронтенд на React забезпечує інтерактивний UX, а Docker – стійке й масштабоване розгортання в хмарі. Кожен компонент обрано за критерієм продуктивності, простоти підтримки та можливості швидкого розширення функціоналу.

2. ОПИС ПРОГРАМНОГО ПРОДУКТУ "ВОЛОНТЕР+"

Призначення створення системи

"Волонтер+" – це платформа, що забезпечує прозору взаємодію між військовими, волонтерами та донорами. Її основна мета – спростити процес допомоги та зробити його максимально відкритим і зрозумілим для кожного учасника.

Платформа працює за логікою "запит → збір → звіт":

- військові підрозділи або окремі військовослужбовці публікують запити на необхідні речі чи обладнання;
- волонтери беруть ці запити в роботу, збирають кошти або речі та закривають запити власними силами;
- після виконання запиту волонтер самостійно публікує звіт із фото, чеками або коротким описом, що саме було передано.

Кожен запит має статус, що оновлюється в реальному часі: новий, у процесі, виконано, очікує звіту. Це дозволяє донорам бачити, на якому етапі збір і чи є ще потреба в участі.

Особливістю платформи є публічна історія діяльності волонтерів: будь-хто може переглянути, які запити брав у роботу конкретний волонтер, як звітував і наскільки оперативно виконував зобов'язання. Така відкритість формує довіру та стимулює відповідальність.

Також реалізовано систему рейтингу волонтерів – на основі кількості виконаних запитів, термінів виконання та якості звітів. Це допомагає донорам орієнтуватись, кому вони можуть довіряти найбільше.

"Волонтер+" створено як інструмент, що посилює горизонтальні зв'язки, зменшує бюрократію та фокусується на ефективній взаємодії між тими, хто просить про допомогу, і тими, хто її надає.

Вимоги до системи

Вимоги до системи в цілому:

- доступність 24/7. SLA не нижче 99,5 % (згідно з internal SLO);
- продуктивність. 95 відсотків часу відповіді API < 300 мс при 1 000 RPS;
- безпека. Захист OWASP Top10; двофакторна авторизація (email / TOTP);
- масштабованість. Горизонтальне розгортання контейнерів у AWS ECS (min 2 – max 10 tasks);
- доступність даних. Щоденне S3snapshotbackup + PointinTime Recovery (≤ 5 min RPO).

Вимоги до структури та функціонування

Для забезпечення надійної роботи платформи "Волонтер+" були визначені ключові вимоги до її структури та функціонування. Вони охоплюють як функціональні підсистеми, так і технології, що забезпечують їхню реалізацію. Особлива увага приділена безпеці, доступності, швидкодії та аналітиці. У таблиці 2 подано стислий опис кожної підсистеми, її призначення, використаних технологій та специфічних вимог.

Таблиця 2. Представлення підсистеми відповідно до вимог

Підсистема	Ключове призначення	Протокол / Технологія	Особливі вимоги
Адміністративна	керування користувачами, модерація запитів, фінансовий аудит	REST; Spring Boot Actuator	RBAC; режим readonly audit
Користувацька	UI для військових, волонтерів, донорів	React 18 + TSX;	WCAG 2.1 AA; LCP < 1,5 с
Платіжна	обробка donatетранзакцій	LiqPay API;	PCI DSS SAQA; webhooks retry
Сповіщення	Email	WebSocket/STOMP	≤ 30 с від тригера до доставки
Аналітика	збір KPI, формування звітів	Prometheus, Grafana	Dataretention 12 міс.

Збалансоване поєднання REST-інтерфейсів з Spring Boot Actuator для адміністрування, сучасного UI на базі React із суворим дотриманням WCAG[20] і вимог до швидкості завантаження, інтеграції з LiqPay[23] за стандартами PCI DSS, real-time сповіщень через WebSocket/STOMP та комплексної аналітики в Prometheus і Grafana забезпечує гнучкість, масштабованість і стійкість системи. Дотримання спеціальних вимог (RBAC, audit-режим, LCP < 1,5 с, retry-механізми для webhook, зберігання даних 12 міс.) гарантує безперебійну та безпечну експлуатацію "Волонтер+"[11].

Високорівнева архітектура системи демонструє основні компоненти платформи "Волонтер+" та їхню взаємодію на рівні сервісів, інтерфейсів і даних (див. рис. 9).

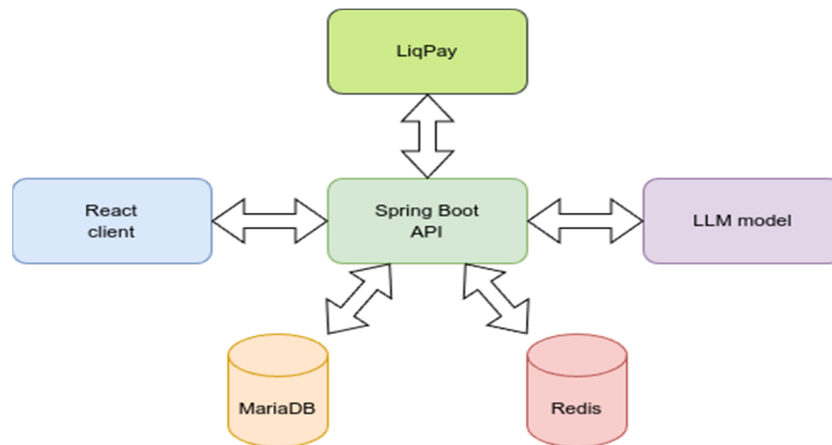


Рис. 9. Високорівнева архітектура системи

Така модульна структура сприяє гнучкому масштабуванню, високій доступності і простоті підтримки.

Незареєстрований користувач відображає основні сценарії взаємодії з платформою без проходження авторизації, зосереджуючись на доступі до публічної інформації та можливості початкового знайомства з функціоналом (див. рис. 10) [9].

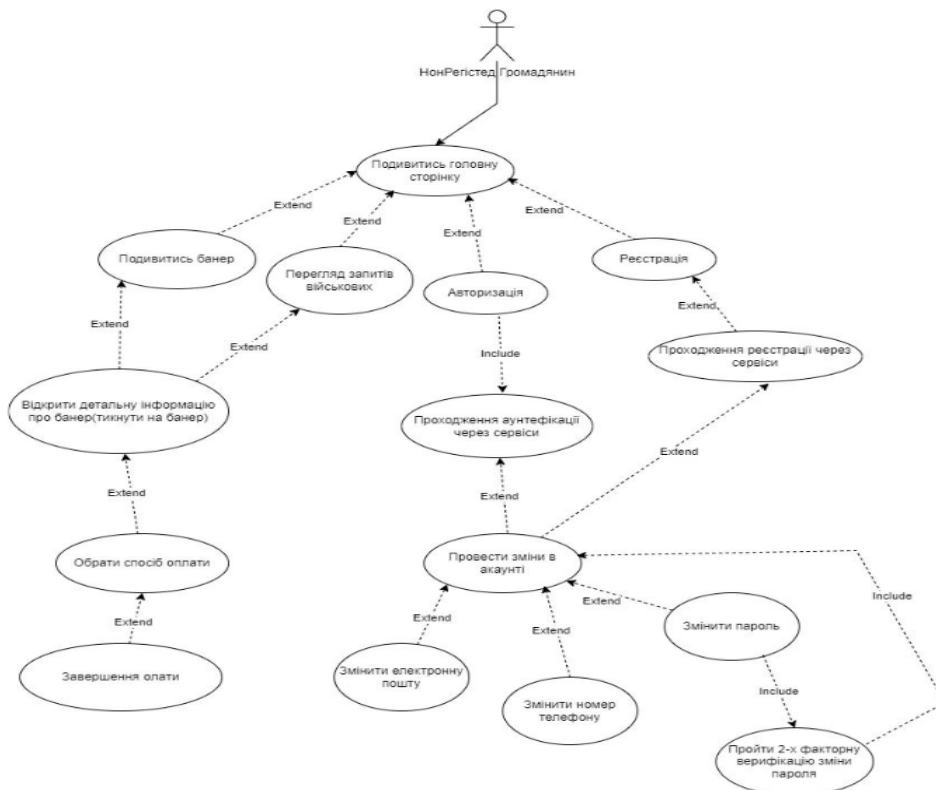


Рис. 10. Use-Case-діаграма "Незарєєстрований користувач"

У моделі "Нєрєєстрований громадянин" представлений як користувач без облікового запису, який одразу потрапляє на головну сторінку. Основний сценарій його взаємодії описано як "Подивитись головну сторінку": саме тут починається знайомство з платформою, відображаються ключові елементи UI та загальна навігація (рис. 10).

Окрім основного варіанта, до "Подивитись головну сторінку" прив'язані кілька розширень (extend), що демонструють додаткові можливості. Насамперед "Авторизація" дозволяє перейти до "Проходження аутентифікації через сервіси", яке включається в процес авторизації як внутрішня складова перевірки облікових даних через зовнішні провайдери. Аналогічно "Реєстрація" розгортається через підпроцес "Проходження реєстрації через сервіси", забезпечуючи створення нового облікового запису. Розширення "Перегляд запитів військових" дає змогу негайно ознайомитися з актуальними зверненнями навіть без реєстрації, а "Подивитись банер" веде до "Відкрити детальну інформацію про банер (тикнути на банер)", що у свою чергу включає вибір способу оплати ("Обрати спосіб оплати") та завершення операції ("Завершення оплати").

Завдяки такій багаторівневій побудові сценаріїв видно не лише можливості незареєстрованого користувача, а й порядок їхньої реалізації в системі. Включення й розширення чітко визначають, які дії завжди доступні, а які потребують додаткової взаємодії з сервісами аутентифікації, реєстрації чи платіжними шлюзами. Такий підхід забезпечує прозорість логіки, полегшує аналіз вимог і подальшу підтримку функціоналу.

Use-case діаграма "Зареєстрований користувач" показує, які дії доступні зареєстрованому громадянину на платформі, починаючи з основного сценарію "Подивитись головну сторінку" та включаючи низку додаткових можливостей (рис. 11).

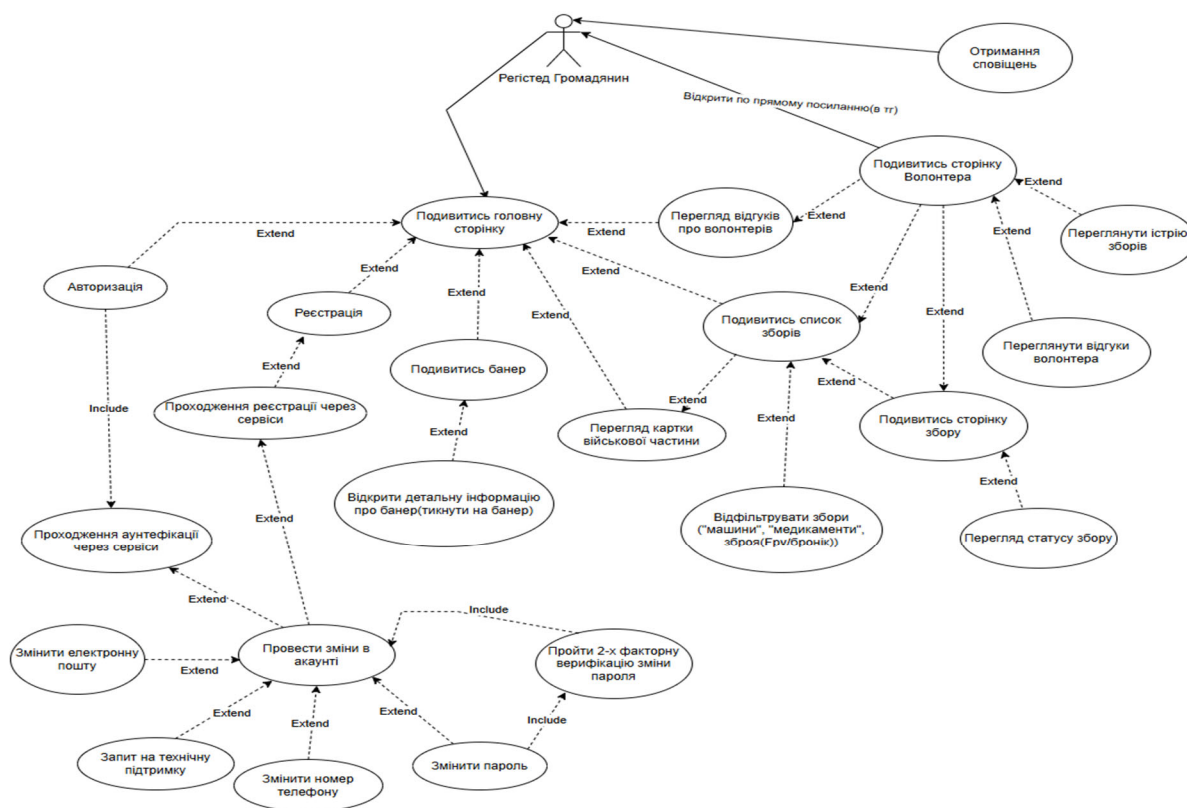


Рис. 11. Use-Case-діаграма "Зареєстрований користувач"

У ролі "Зареєстрований громадянин" користувач може не лише переглянути головну сторінку, а й за потреби розгорнути сценарій "Авторизація" через включений підпроцес "Проходження аутентифікації через сервіси" або, якщо ще не зареєстрований, звернутися до "Реєстрації" через підпроцес "Проходження реєстрації через сервіси". Додатково він

може переглянути банер із можливістю відкрити детальну інформацію та далі обрати спосіб оплати й завершити її, ознайомитися з відгуками про волонтерів, переглядати список зборів із включенням картки військової частини, фільтрувати ці збори за типами (машини, медикаменти тощо) і заходити на сторінку конкретного збору з відображенням його статусу. Крім того, зареєстрований користувач має змогу відвідати сторінку волонтера й подивитися відгуки про нього, переглянути історію зборів, отримувати сповіщення про важливі події та відкривати будь-яку сторінку за прямим посиланням – усе це чітко відображено у структурі варіантів використання системи.

Use-Case–діаграма роль "Волонтер" демонструє всі основні та розширені сценарії взаємодії користувача-волонтера з платформою (див. рисунок 12). Волонтер може налаштувати свій профіль волонтерської діяльності (зокрема надсилати повідомлення іншим волонтерам, звертатися по технічну підтримку, отримувати відгуки від військових і здобувати бейджі та досягнення), переглядати історію активності та статистику зборів (з можливістю редагування збору). Крім того, передбачено роботу із запитами (додавати коментарі й відгукуватися через включений сценарій "Відгук на запит"), створювати й інтегрувати збори в соціальні мережі, завантажувати фінансові звіти й медіа до зборів, формувати звіти та робити публічні пости. Доступні також функції перегляду списку донорів з можливістю отримувати сповіщення про статус звіту, запрошувати друзів до участі, користуватися AI-рекомендаціями щодо зборів, а також брати участь у спільнотах волонтерів і навчальних заходах (див. рис. 12).



Рис. 12. Use-Case – діаграма роль "Волонтер"

Чітка структуризація варіантів використання полегшує розуміння ролі волонтера, визначає послідовність дій і необхідні інтеграції з іншими сервісами. Це сприяє уніфікації вимог, покращує комунікацію між розробниками та зацікавленими сторонами та забезпечує гнучкість для подальшого розширення функціоналу.

Функціональні підсистеми. Вони формують каркас платформи "Волонтер+", розділяючи її на автономні модулі, кожен із яких відповідає за реалізацію певного бізнес-процесу: обробку запитів, управління кампаніями зі збору коштів, організацію комунікацій тощо. Такий поділ забезпечує прозорість архітектури, спрощує підтримку та масштабування системи.

Підсистема запитів являє собою основний інструмент взаємодії з платформою, який об'єднує можливості створення, редагування та видалення звернень користувачів, а також оперативного доступу до переліку актуальних запитів із розширеним пошуком і фільтрацією (див. таблицю 3).

Таблиця 3. Функції та задачі, що підлягають запитам

Функція	Задачі
Керування запитами	Створення нових запитів, редагування існуючих, видалення застарілих або помилкових
Перегляд запитів	Виведення списку актуальних запитів з можливістю пошуку та фільтрації
Перегляд деталей	Перегляд повної інформації про конкретний запит, включаючи опис, фото тощо

Внаслідок чіткого розподілу на функціональні блоки керування запитами, їх перегляду та детального ознайомлення з описом кожного запиту, команда забезпечує прозорість процесу обробки звернень і швидке реагування на потреби користувачів. Такий підхід мінімізує ризик втрати важливої інформації, підвищує ефективність координації та створює сприятливі умови для своєчасного задоволення запитів.

Підсистема зборів об'єднує ключові можливості створення, управління та завершення кампаній зі збору коштів, а також ведення звітності й візуалізації прогресу збору (див. таблицю 4).

Таблиця 4. Функції та задачі, що підлягають зборам

Функція	Задачі
Керування зборами	Створення зборів для запитів, зміна їхнього статусу (активний, завершений)
Звітність за зборами	Додавання фінансових звітів, фото, підтверджень доставки
Відображення прогресу	Показ відсотка зібраної суми, статусів виконання запиту

Завдяки чіткому розподілу на блоки керування зборами, формування фінансових і візуальних звітів та відображення відсотка зібраних коштів команда може ефективно планувати, контролювати та адаптувати кампанії під поточні потреби. Оперативне оновлення статусів зборів, своєчасне додавання фінансових підтверджень і фото забезпечує прозорість процесу для волонтерів і донорів, зміцнює довіру до платформи та сприяє досягненню поставлених цілей із максимальною відповідальністю.

Чат-підсистема забезпечує миттєвий обмін текстовими повідомленнями між волонтерами та військовими, гарантує доступ до повної історії листування для збереження контексту та домовленостей і автоматичне сповіщення про нові повідомлення чи зміни в статусах та запитах (див. таблицю 5).

Таблиця 5. Функції та задачі, що підлягають чат-системі

Функція	Задачі
Обмін повідомленнями	Надсилання та отримання текстових повідомлень між волонтерами та військовими
Перегляд історії	Доступ до всієї історії листування для збереження контексту та домовленостей
Сповіщення	Автоматичне сповіщення про нові повідомлення, зміни в статусах або запитах

Через те, що інтегрованої чат-системі команда та користувачі отримують безперебійну комунікацію, що сприяє оперативному прийняттю рішень, збереженню важливої інформації та підвищенню загальної ефективності взаємодії. Автоматичні сповіщення й легкий доступ до історії листування мінімізують ризик втрати даних і підтримують прозорість процесів, зміцнюючи довіру користувачів до платформи.

Системні вимоги

Веб-платформа "Волонтер+" розроблена як легкий, зручний і доступний інтерфейс, який не потребує встановлення додаткового програмного забезпечення. Для стабільної роботи достатньо сучасного браузера та базового інтернет-з'єднання. Платформа коректно функціонує в останніх стабільних версіях найбільш популярних веб-браузерів, таких як Google Chrome, Mozilla Firefox, Safari та Microsoft Edge. Це дозволяє більшості користувачів отримати повноцінний досвід роботи незалежно від операційної системи, якою вони користуються – як на Windows, так і на macOS [18,19].

З технічного боку, для нормального відображення елементів інтерфейсу рекомендується використовувати пристрої з роздільною здатністю не нижче 1280×720 пікселів. Хоча мобільна адаптація наразі не реалізована, інтерфейс залишається частково читабельним на планшетах та великих телефонах. Основна взаємодія з платформою відбувається через JavaScript, тому його підтримка та активний стан у браузері є обов'язковими. Платформа також використовує сучасні стандарти HTML5 і CSS3 для стилізації, а тому старі версії браузерів можуть не підтримувати всі візуальні елементи коректно.

Щодо підключення, достатньо навіть повільного, але стабільного інтернету – на рівні 1 Мбіт/с, аби завантажити інтерфейс, переглянути запити, надіслати повідомлення або звіт. Завдяки кешуванню та оптимізації запитів, більшість даних оновлюється без повної перезавантаження сторінки.

Таким чином, платформа не має високих вимог до клієнтського обладнання, але передбачає базовий рівень сучасності техніки користувача та дотримання стандартів безпеки браузера, включаючи підтримку HTTPS.

Інформаційна база/логічна структура

Використана СКБД. MariaDB – це сучасна реляційна база даних, яка виникла як відгалуження від MySQL після того, як останню придбала Oracle. Головна ідея MariaDB – зберегти повну відкритість коду та уникнути ризиків комерційної залежності від великого вендора. Саме тому вона стала вибором багатьох компаній, які шукають стабільну, продуктивну та юридично прозору альтернативу MySQL.

MariaDB повністю сумісна з MySQL, тож перехід або початок роботи з нею не створює труднощів для розробників, особливо у середовищі Spring Boot. Вона підтримує ті самі драйвери, синтаксис SQL, інтеграцію з ORM-фреймворками на кшталт Hibernate чи Spring Data JPA. Завдяки цьому можна легко будувати застосунки без додаткових конфігураційних складностей (див. рисунки 13, 14, 15).

Ще однією важливою причиною вибору MariaDB є її продуктивність. Вона оптимізована для високонавантажених систем, підтримує різні рушії зберігання даних, дозволяє масштабування і має чудову підтримку реплікації та кластеризації. Наприклад, Galera Cluster дозволяє розгорнути multi-master конфігурації для забезпечення високої доступності.

Також MariaDB добре підходить для контейнеризованих середовищ, таких як Docker[28] або Kubernetes. Це робить її привабливою для сучасних DevOps-процесів. Інструменти для моніторингу, бекупу та автоматизації роботи з базою легко інтегруються в CI/CD-пайплайни[13].

Опис таблиць

Таблиця QRTZ_CALEDNARS. Призначення: зберігає календарі під час налаштування графіків виконання тригерів.

Поля:

- SCHED_NAME; varchar(120), NOT NULL; унікальне ім'я календаря;
 - CALENDAR_NAME; varchar(190), NOT NULL; унікальне ім'я календаря;
 - CALENDAR; blob, NOT NULL; серіалізований об'єкт org.quartz.Calendar;
- PK: (SCHED_NAME, CALENDAR_NAME)

Таблиця QRTZ_FIRED_TRIGGERS. Призначення: лог запусків тригерів; використовується для відстеження історії виконання.

Поля:

- ENTRY_ID (varchar(95), NOT NULL) – унікальний запис запуску;
- TRIGGER_NAME, TRIGGER_GROUP (varchar(190)) – зв'язок з записом у QRTZ_TRIGGERS;
- INSTANCE_NAME (varchar(190)) – ім'я вузла кластеру, що спрацював;
- FIRED_TIME (bigint(13), NOT NULL) – UNIX timestamp фактичного запуску;
- SCHED_TIME (bigint(13), NOT NULL) – запланований UNIX timestamp;
- PRIORITY (int, NOT NULL) – пріоритет виконання;
- STATE (varchar(16), NOT NULL) – стан (e.g., ACQUIRED, EXECUTING);
- JOB_NAME, JOB_GROUP (varchar(190)) – пов'язаний job із QRTZ_JOB_DETAILS;
- IS_NONCONCURRENT, REQUESTS_RECOVERY (varchar(1)) – прапорці поведінки;

PK: (SCHED_NAME, ENTRY_ID)

Індекси:

- IDX_QRTZ_FT_INST_JOB_REQ_RCVRY on (SCHED_NAME, INSTANCE_NAME, REQUESTS_RECOVERY)
- індекси по JOB_NAME, TRIGGER_NAME, TRIGGER_GROUP та ін.

Таблиця QRTZ_JOB_DETAILS. Призначення: зберігає метадані всіх job-класів, доступних для запуску.

Поля:

- JOB_NAME, JOB_GROUP, SCHED_NAME – ідентифікатори job;
- DESCRIPTION (varchar(250)) – текстовий опис;
- JOB_CLASS_NAME (varchar(250), NOT NULL) – повний шлях Java-класу, що реалізує Job;

- IS_DURABLE, IS_NONCONCURRENT, IS_UPDATE_DATA, REQUESTS_RECOVERY (varchar(1), NOT NULL) – поведінкові прапорці;
- JOB_DATA (blob) – серіалізований JobDataMap;
- PK: (SCHED_NAME, JOB_NAME, JOB_GROUP)
- Індекси: по JOB_GROUP та REQUESTS_RECOVERY.

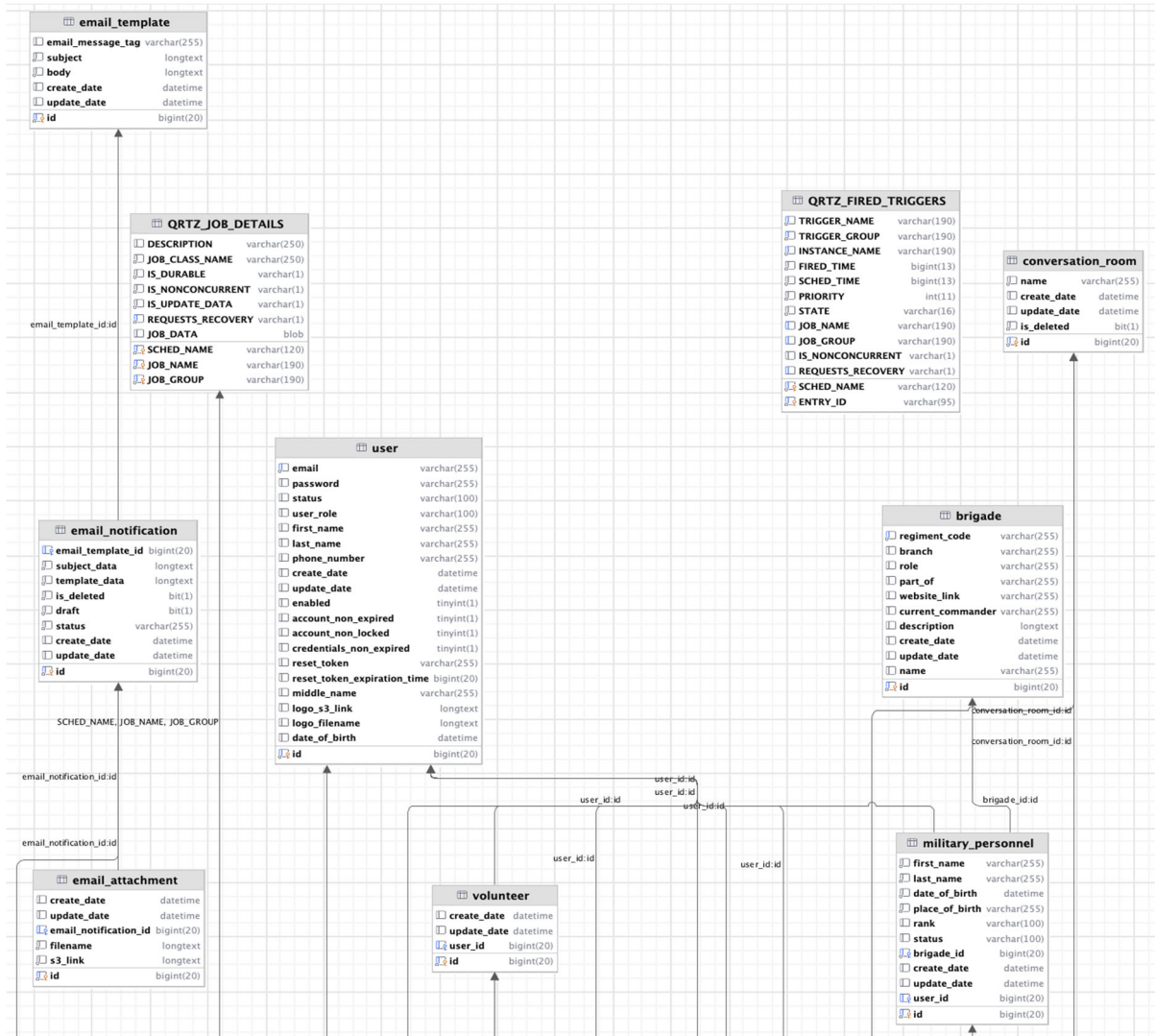


Рис. 13. DB schema діаграма платформи Волонтер+

Таблиця QRTZ_LOCKS. Призначення: забезпечує механізм блокування ресурсів у кластері Quartz.

Поля:

- LOCK_NAME (varchar(40)) – ім'я блокування;
- SCHED_NAME – контекст планувальника;
- PK: (SCHED_NAME, LOCK_NAME)

Таблиця QRTZ_PAUSED_TRIGGER_GRP. Призначення: зберігає групи тригерів, поставлені на паузу.

Поля:

- TRIGGER_GROUP (varchar(190)) – ім'я групи;

- SCHED_NAME;
PK: (SCHED_NAME, TRIGGER_GROUP)

Таблиця QRTZ_SCHEDULER_STATE. Призначення: відстеження "heartbeat" вузлів кластеру Quartz.

Поля:
- INSTANCE_NAME – унікальний ідентифікатор вузла;
- LAST_CHECKIN_TIME, CHECKIN_INTERVAL (bigint(13)) – часові мітки;
PK: (SCHED_NAME, INSTANCE_NAME)

Таблиця QRTZ_TRIGGERS. Призначення: основна таблиця опису тригерів та їхнього стану.

Поля:
- TRIGGER_NAME, TRIGGER_GROUP, SCHED_NAME – ідентифікатори;
- JOB_NAME, JOB_GROUP – зв'язок з QRTZ_JOB_DETAILS;
- DESCRIPTION (varchar(250)), NEXT_FIRE_TIME, PREV_FIRE_TIME (bigint(13));
PRIORITY (int); TRIGGER_STATE (varchar(16)); TRIGGER_TYPE (varchar(8));
START_TIME; END_TIME (bigint(13)); CALENDAR_NAME (varchar(190));
MISFIRE_INSTR (smallint(2)); JOB_DATA (blob);
PK: (SCHED_NAME, TRIGGER_NAME, TRIGGER_GROUP)
FK: на QRTZ_JOB_DETAILS (JOB_NAME, JOB_GROUP).
Індекси: NEXT_FIRE_TIME, TRIGGER_STATE, триггерні групи.

Таблиця QRTZ_CRON_TRIGGERS. Призначення: CRON-розклад для тригерів.

Поля:
- CRON_EXPRESSION (varchar(120), NOT NULL);
- TIME_ZONE_ID (varchar(80));
PK & FK: успадковується від QRTZ_TRIGGERS.

Таблиця QRTZ_SIMPLE_TRIGGERS. Призначення: прості інтервальні тригери.

Поля:
- REPEAT_COUNT (bigint(7), NOT NULL);
- REPEAT_INTERVAL (bigint(12), NOT NULL);
- TIMES_TRIGGERED (bigint(10), NOT NULL);
PK & FK: як у QRTZ_TRIGGERS.

Таблиця QRTZ_BLOB_TRIGGERS. Призначення: специфічні тригери з BLOB-даними.

Поля: BLOB_DATA (blob);
PK & FK: у QRTZ_TRIGGERS.

Таблиця: QRTZ_SIMPROP_TRIGGERS. Призначення: тригери з властивостями: строки, числа, булеві.

Поля:
- STR_PROP_1..3 (varchar(512));
- INT_PROP_1..2 (int);
- LONG_PROP_1..2 (bigint);

- DEC_PROP_1..2 (decimal(13,4));
 - BOOL_PROP_1..2 (varchar(1));
- PK & FK: як у QRTZ_TRIGGERS.

Таблиця add_request. Призначення: облік запитів на створення або оновлення ресурсів.

Поля:

- id (bigint, PK, AUTO_INCREMENT);
- request_id (longtext, NOT NULL) – зовнішній UUID;
- executed (bit, NOT NULL) – статус виконання (0/1);
- create_date, update_date (datetime) – часові мітки;
- regiment_code (varchar(255), NOT NULL) – код бригади.

Таблиця ai_chat_messages. Призначення: збереження історії повідомлень між користувачем та AI.

Поля:

- id (bigint, PK, AI);
- conversation_id (varchar(255), NOT NULL) – ідентифікатор діалогу;
- message_data (text) – JSON або текст повідомлення;
- create_date, update_date (datetime);

Індекси: idx_conversation_id на conversation_id.

Таблиця brigade. Призначення: довідник військових формувань.

Поля: id, regiment_code (varchar(255), UNIQUE), name, branch, role, part_of, current_commander, website_link, description, create_date, update_date.

Таблиця brigade_codes. Призначення: список унікальних кодів для бригаад.

Поля: id, code (varchar(255), UNIQUE), create_date, update_date.

Таблиця conversation_room. Призначення: чат-кімнати для WebSocket комунікацій.

Поля: id, name, create_date, update_date, is_deleted (bit).

Таблиця conversation_room_user. Призначення: зв'язок користувачів та кімнат. Поля:

conversation_room_id, user_id – FK на відповідні таблиці.

Таблиця email_template. Призначення: зберігання шаблонів електронних листів. Поля: id, email_message_tag, subject (longtext), body (longtext), create_date, update_date.

Таблиця email_notification. Призначення: конкретні відправлені або створені за шаблоном листи. Поля: id, email_template_id (FK → email_template), subject_data, template_data, is_deleted, draft, status, create_date, update_date.

Таблиця email_recipient. Призначення: отримувачі повідомлень. Поля: id, email_notification_id (FK), email_address, full_name, to_recipient, cc, bcc, sent, sent_date, create_date, update_date.

Таблиця email_attachment. Призначення: вкладення у повідомленнях. Поля: id, email_notification_id (FK), filename, s3_link, create_date, update_date.

Таблиця flyway_schema_history. Призначення: журнал застосованих міграцій Flyway. Поля: installed_rank, version, description, type, script, checksum, installed_by, installed_on, execution_time, success.

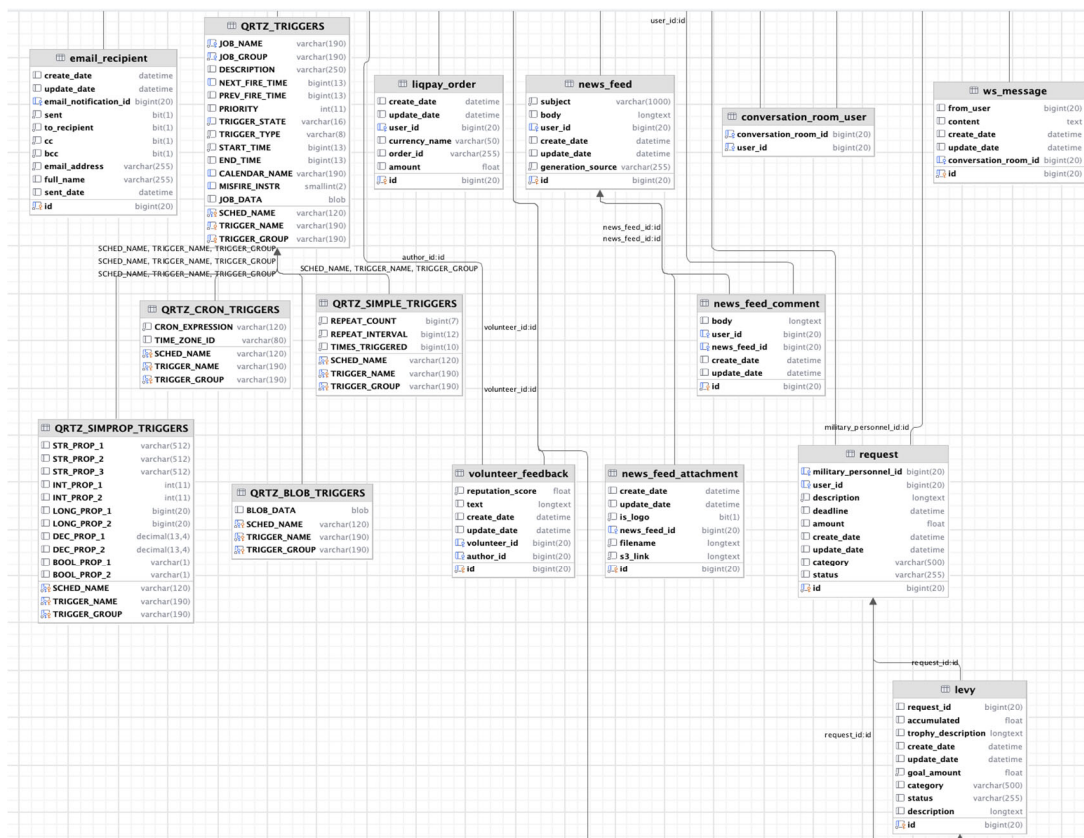


Рис. 14. DB schema діаграма платформи Волонтер+

Таблиця levy. Призначення: кампанії збору фінансів або ресурсів. Поля: id, request_id, accumulated, trophy_description, goal_amount, category, status, description, create_date, update_date.

Таблиця report. Призначення: звіти за кампаніями. Поля: id, levy_id (FK → levy), data (longtext), is_analyzed (bit), create_date, update_date.

Таблиця attachment. Призначення: файли-додатки до звітів.

Поля: id, report_id (FK → report), filepath, filename, create_date, update_date

Таблиця user. Призначення: профілі користувачів системи. Поля: id, email (UNIQUE), password, status, user_role, first_name, last_name, middle_name, phone_number, date_of_birth, create_date, update_date, enabled, account_non_expired, account_non_locked, credentials_non_expired, reset_token, reset_token_expiration_time, logo_filename, logo_s3_link.

Таблиця `military_personnel`. Призначення: дані про військовослужбовців. Поля: `id`, `first_name`, `last_name`, `date_of_birth`, `place_of_birth`, `rank`, `status`, `brigade_id` (FK → `brigade`), `user_id` (FK → `user`), `create_date`, `update_date`.

Таблиця `news_feed`. Призначення: новинна стрічка. Поля: `id`, `subject`, `body`, `user_id` (FK → `user`), `generation_source`, `create_date`, `update_date`.

Таблиця `news_feed_comment`. Призначення: коментарі до новин. Поля: `id`, `body`, `user_id` (FK → `user`), `news_feed_id` (FK → `news_feed`), `create_date`, `update_date`.

Таблиця `news_feed_attachment`. Призначення: медіа-вкладення у новинах. Поля: `id`, `news_feed_id` (FK → `news_feed`), `is_logo`, `filename`, `s3_link`, `create_date`, `update_date`.

Таблиця `request`. Призначення: запити на допомогу або ресурси. Поля: `id`, `military_personnel_id` (FK → `military_personnel`), `user_id` (FK → `user`), `description`, `deadline`, `amount`, `category`, `status`, `create_date`, `update_date`.

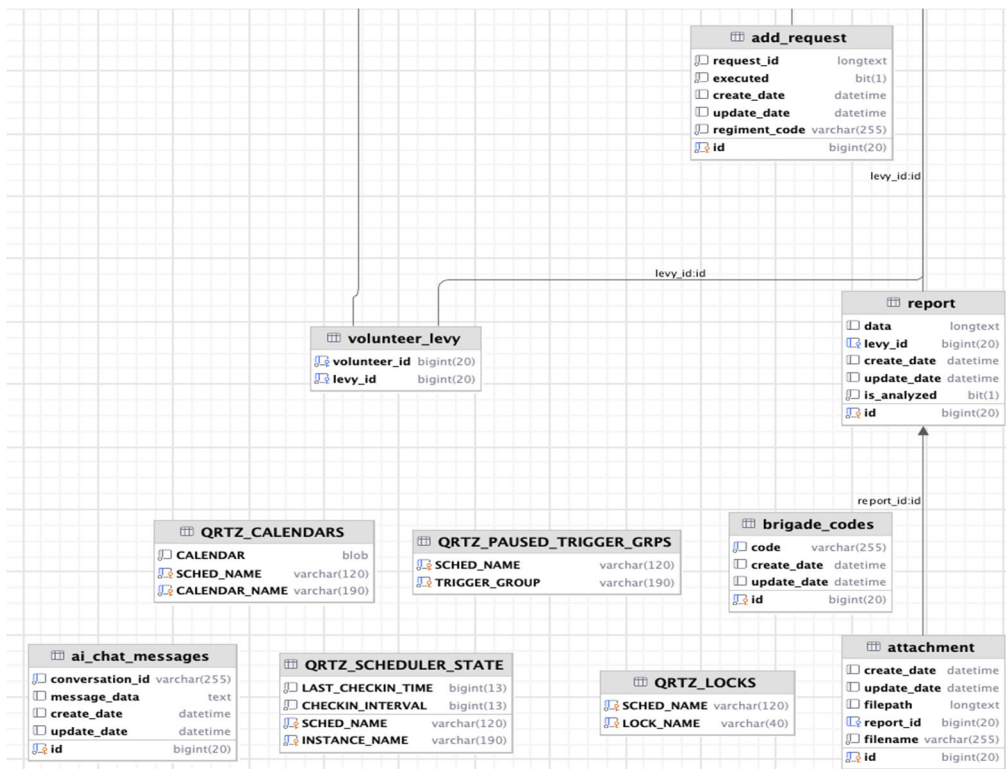


Рис. 15. DB schema діаграма платформи Волонтер+

Таблиця `volunteer`. Призначення: профіль волонтера. Поля: `id`, `user_id` (FK → `user`), `create_date`, `update_date`.

Таблиця `volunteer_feedback`. Призначення: відгуки про волонтерів. Поля: `id`, `volunteer_id` (FK → `volunteer`), `author_id` (FK → `user`), `reputation_score`, `text`, `create_date`, `update_date`.

Таблиця `volunteer_levy`. Призначення: зв'язок волонтерів зі зборами. Поля: `volunteer_id` (FK → `volunteer`), `levy_id` (FK → `levy`).

Таблиця `ws_message`. Призначення: повідомлення WebSocket чатів. Поля: `id`, `from_user` (FK → `user`), `content`, `conversation_room_id` (FK → `conversation_room`), `create_date`, `update_date`.

UI/UX дизайн та практичне застосування принципів

Закон Фіттса (*Fitts's Law*) [30]

Суть: Час, необхідний для досягнення цілі, залежить від її розміру та відстані до неї.

Застосування: Великі кнопки "Підтримати", "Створити запит" і "Додати звіт" – розміщені в зонах, які легко натиснути з першого разу.

Функція: Спрощення найважливіших дій користувача – донат, створення запиту, оновлення статусу (див. рисунок 16).

Вплив: Зменшує кількість хибних натискань, пришвидшує взаємодію, підвищує ефективність платформи.



Рис. 16. Приклад картки збору

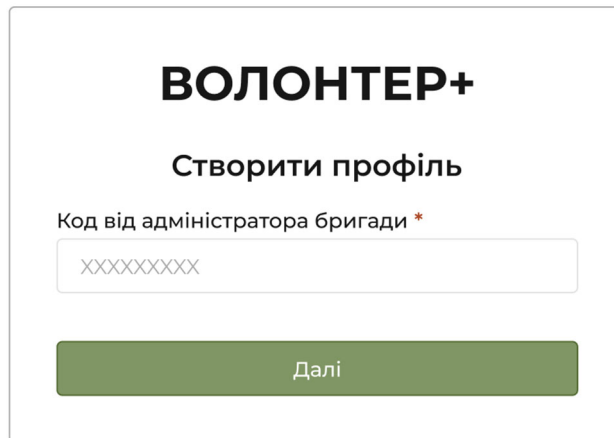
Закон Гік-аймана (*Hick's Law*) [31]

Суть: Чим більше варіантів – тим більше часу користувач витрачає на вибір.

Застосування: При реєстрації військовий спочатку бачить лише форму із кодом активації, а потім вже форму з іншою інформацією, як зображено на рисунку 16 та 17.

Функція: Зниження когнітивного навантаження.

Вплив: Користувач швидше ухвалює рішення та не губиться серед надлишку варіантів.



ВОЛОНТЕР+

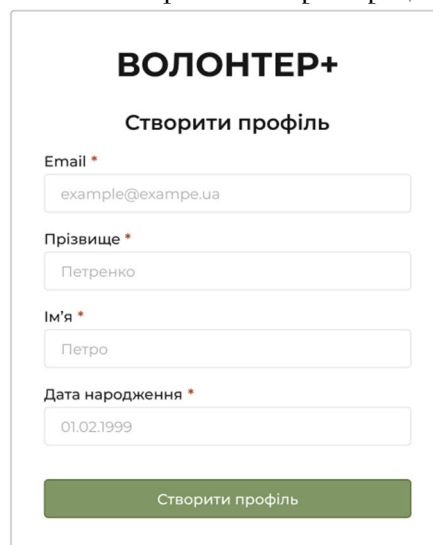
Створити профіль

Код від адміністратора бригади *

XXXXXXXXXX

Далі

Рис. 17. Перший етап реєстрації



ВОЛОНТЕР+

Створити профіль

Email *

example@example.ua

Прізвище *

Петренко

Ім'я *

Петро

Дата народження *

01.02.1999

Створити профіль

Рис. 18. Другий етап реєстрації

Закон Міллера (Miller's Law) [32]

Суть: Людина здатна утримувати в пам'яті лише 7 ± 2 одиниці інформації одночасно[12].

Застосування: У таблиці відображається лише частина записів. Інші доступні через пагінацію, як зображено на рис. 18.

Функція: Запобігання перевантаженню екрана.

Код	Статус	Дата створення	Дата активації	Email члена бригади
*****	Нове	12:40 12.01.2025	-	-
2323232323	Відкликане	12:40 12.01.2025	-	-
2323232323	Прийняте	12:40 12.01.2025	12:40 12.01.2025	petrenko@gmail.com
2323232323	Прийняте	12:40 12.01.2025	12:40 12.01.2025	petrenko@gmail.com
2323232323	Прийняте	12:40 12.01.2025	12:40 12.01.2025	petrenko@gmail.com
2323232323	Прийняте	12:40 12.01.2025	12:40 12.01.2025	petrenko@gmail.com

Рис. 19. Таблиця запрошень "Волонтер+"

Вплив: Користувач легше орієнтується.

Закон Якоба (Jakob's Law). [33]

Суть: Користувачі очікують, що ваш продукт працюватиме так само, як інші знайомі сервіси.

Застосування: У чаті використано стандартну структуру: аватар, бульбашки повідомлень, поле введення з кнопкою надсилання (див. рис. 20).

Функція: Мінімізація кривої навчання, максимальна передбачуваність дій.

Вплив: Користувач відчуває впевненість у роботі з платформою з першого використання.

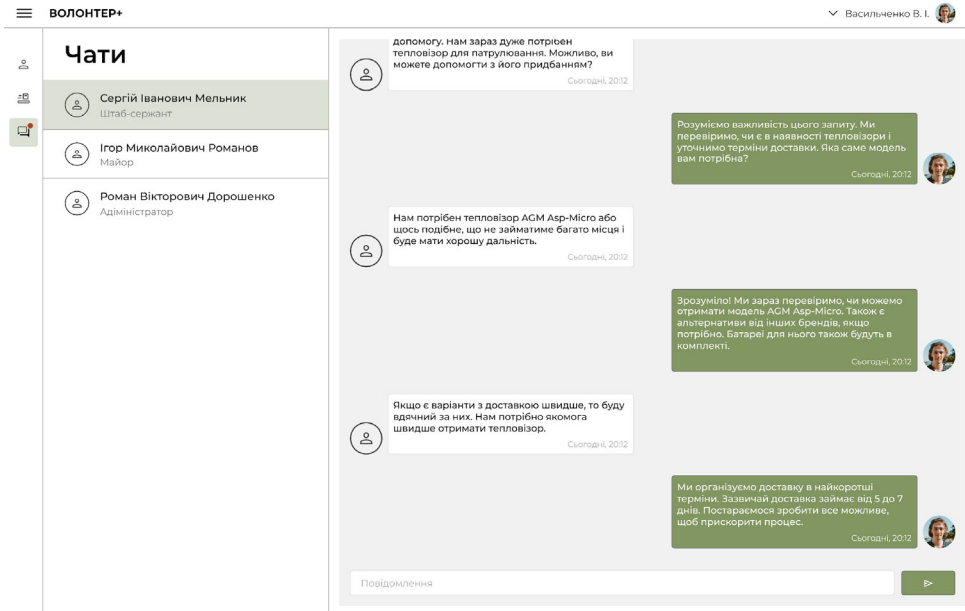


Рис. 20. Сторінка чатів, що відповідає загально прийнятим стандартам

Кольорова палітра

Кольорова палітра – це одна з ключових складових візуального оформлення, яка впливає на сприйняття користувачами. Вибір кольорів має бути зваженим і відповідати цілям, аудиторії та загальному стилю сайту (див. рисунок 21).

За основу було взяти два кольори: оливково зелений та сірий.

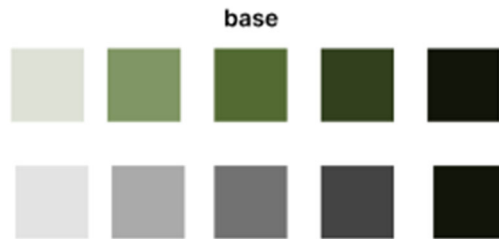


Рис. 21. Палітра кольорів Волонтер+

Кольорова палітра для системи збору коштів на потреби військових має бути стриманою, викликати довіру, серйозність і надійність. Розглянемо детальніше кожен колір (див. рисунок 22):

- #556B2F (оливково-зелений): Символізує військову тематику, стійкість і впевненість. Асоціюється з природою та спокоєм, що допоможе викликати у користувачів відчуття стабільності та підтримки.

- #727272 (сірий): Універсальний і нейтральний колір. Символізує професіоналізм, серйозність і функціональність, що важливо для проєкту з високим рівнем довіри.



Рис. 22. Допоміжна палітра кольорів

Детальніше про кольори:

- #DD5726 (помилка (error): Значення: Символізує небезпеку, проблему або критичну помилку. Цей колір інстинктивно привертає увагу та асоціюється із зупинкою чи застереженням. Приклад використання: Відображення невдалих дій, помилок у формах, повідомлень про збій системи.

- #F3B24A (попередження (warning): Значення: Викликає відчуття уваги та застереження, але менш критичне, ніж червоний. Асоціюється з необхідністю прийняття дій. Приклад використання: Інформація про важливі, але не критичні повідомлення, наприклад, попередження щодо завершення терміну дії кампанії або необхідності оновлення даних.

- #A4D92A (успіх (success): Значення: Передає позитивні емоції, завершення або підтвердження дії. Цей колір асоціюється з правильним виконанням завдань чи підтвердженням операцій. Приклад використання: Повідомлення про успішне пожертвування, виконання транзакцій, підтвердження дій.

Шрифт

Шрифт є важливою частиною дизайну, адже він не лише забезпечує читабельність тексту, але й створює певні асоціації, впливаючи на сприйняття проєкту. Для системи збору коштів на потреби військових вибір шрифту має підкреслювати серйозність, надійність та сучасність, викликаючи довіру у користувачів. Шрифт **Montserrat** чудово відповідає цим вимогам (див. рис. 23).

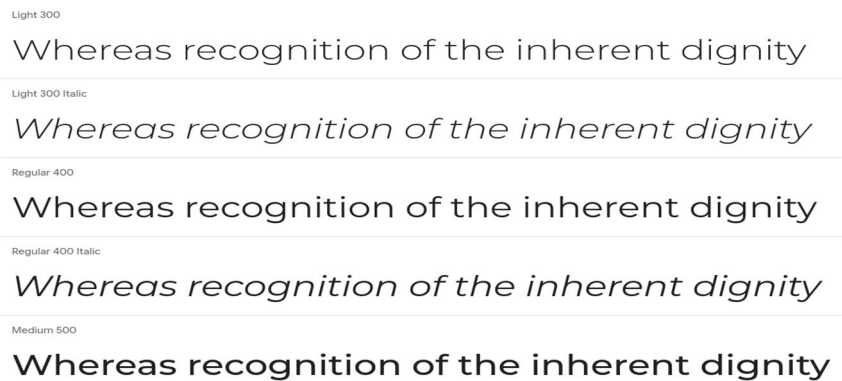


Рис. 23. Палітра шрифтів

Чому Montserrat підходить для нашої системи:

- Montserrat має чисті геометричні форми та чіткі лінії, що робить текст легким для сприйняття на будь-яких пристроях. У нашій системі, де важливо швидко донести інформацію про кампанії чи дії, це забезпечує комфорт користувачам.

- Цей шрифт поєднує сучасний стиль із елегантною простотою. Він створює враження професійного підходу, що особливо важливо для платформи, яка оперує фінансами. Montserrat допомагає викликати довіру до системи та донести, що кошти підуть на важливу і відповідальну справу.

- Montserrat підтримує кирилицю, що забезпечує якісне відображення українських текстів. Це робить його ідеальним для нашої платформи, орієнтованої на локальних користувачів.

- Цей шрифт був створений як данина сучасному міському середовищу. У контексті нашої системи він асоціюється із згуртованістю суспільства, мобілізацією громад для спільної мети підтримки військових.

3. ІМПЛЕМЕНТАЦІЯ ПРОДУКТУ/СИСТЕМИ ТА ТЕСТУВАННЯ

Загальна архітектура

3.1.1 Фронтенд (Presentation Layer)

Компоненти

UI компоненти написані на React. Відповідають за відображення даних та обробку подій користувача (кнопки, форми, навігація).

Логіка стану (State Management)

Redux – це інструмент для зберігання та керування глобальним станом додатку. Він дозволяє зручно працювати з даними, ділитися ними між компонентами й реагувати на зміни.

HTTP запити до бекенду

Fetch API для комунікації з REST API. Реалізовані сервіси для авторизації (отримання токенів), CRUD операцій і обробки помилок мережі.

3.1.2 Бекенд (Application Layer)

Контролери (Controllers):

- точки входу HTTP запитів (Spring MVC @RestController);
- приймають запит, валідують вхідні дані та делегують обробку в сервіси;
- повертають клієнту DTO або HTTP коди.

Сервіси (Services):

- бізнес логіка: об'єднують операції над даними, реалізують правила доменної області;

- оркестрація: викликають репозиторії, перевіряють права доступу, запускають кешування та транзакції;

- транзакційність: зазвичай позначаються `@Transactional` для атомарності групи операцій.

NFR (функціонал не функціональних вимог):

- безпека (Spring Security + OAuth2). Перехоплює запити й перевіряє аутентифікацію/авторизацію. Підтримка OAuth2 провайдерів (Google, Facebook, власний authсервер);

- кешування (Spring Cache – Redis). Прості анотації (`@Cacheable`, `@CacheEvict`) для збереження часто запитуваних результатів (наприклад, довідкові дані). Підключається до Redis[26] або іншого кешу для швидкого читання.

Персистентність (Data Layer):

Репозиторії (Repositories)

- Spring Data JPA / Hibernate: автоматичне створення CRUD методів за допомогою інтерфейсів;

- використовують JPA запити (методи, `@Query`, `Specification`) для складніших вибірок.

БД (PostgreSQL/MariaDB). Гнучкість Spring Data дозволяє приєднатися до будь якої з них (див. рисунок 24):

- реляційна СКБД для зберігання сутностей;

- схема із таблицями, індексами, зовнішніми ключами.

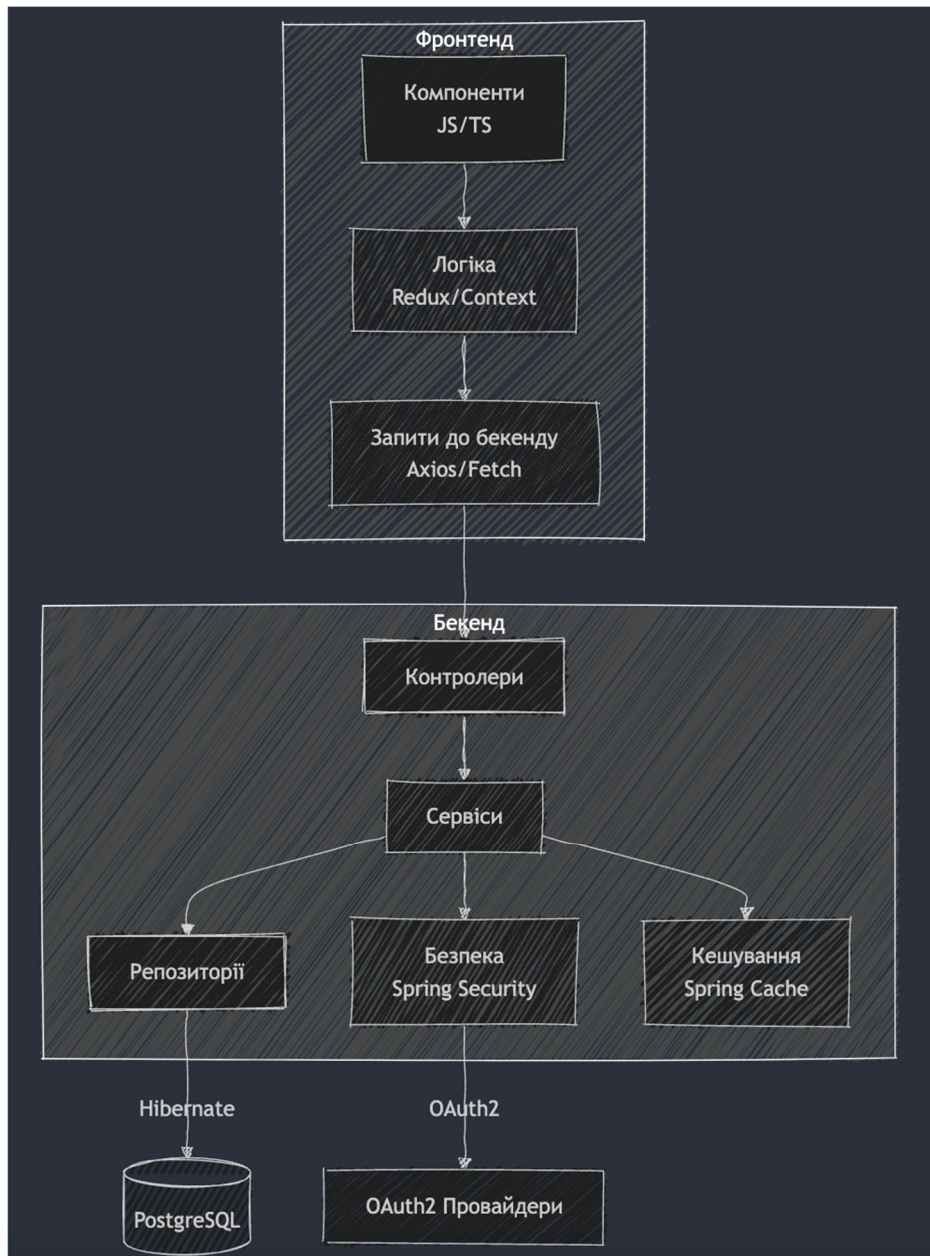


Рис. 24. Загальна архітектура

Послідовність запиту:

- користувач взаємодіє з UI-компонентом;
- компонент звертається до Redux/Context, щоб отримати/змінити стан ;
- при потребі даних: викликає сервіс HTTP (Fetch) ;
- запит доходить до Spring Boot контролера ;
- контролер передає вхідні дані в потрібний @Service ;
- сервіс перевіряє права через Spring Security[25] ;
- якщо дані в кеші відсутні, сервіс звертається до репозиторіїв ;
- Hibernate виконує SQLзапити в MariaDB ;
- отримані дані кешуються (Spring Cache) і повертаються наверх ;
- контролер формує HTTPвідповідь (JSON) і відсилає клієнту ;

– фронтенд отримує відповідь та оновлює компоненти (див. рис. 25).

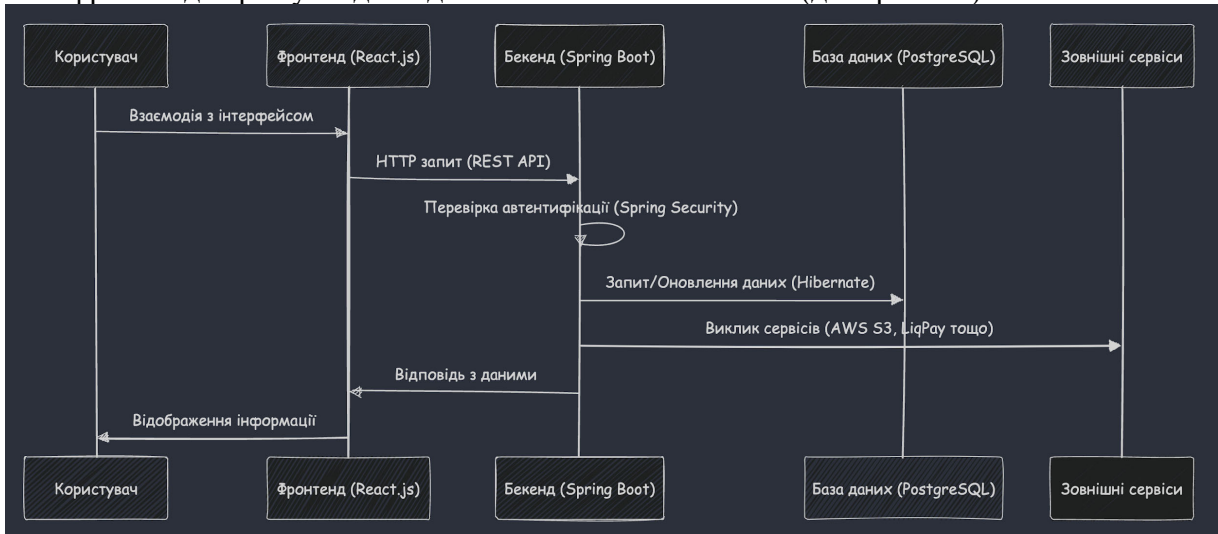


Рис. 25. Послідовність запиту в систему

Також реалізована система надсилання імейл повідомлення з різним змістом та вкладеннями (див. рис. 26, 27, 28).

Волонтер+: Maksym Khodakov, оплата успішно пройшла! ➔



Volunteer+ <volunteersystemservice@gmail.com>

кому мені ▾

Сума: 300

Дякуємо за вашу проактивність на нашій платформі.

З повагою,
Команда Волонтер+

Рис. 26. Імейл про оплату

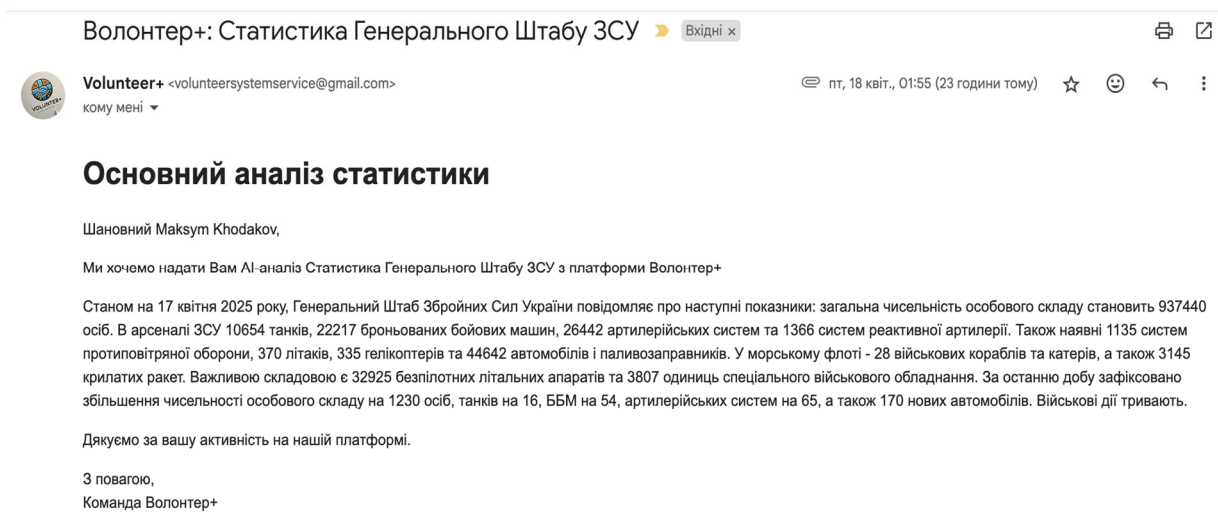


Рис. 27. Імейл AI аналіз статистики Генерального Штабу ЗСУ

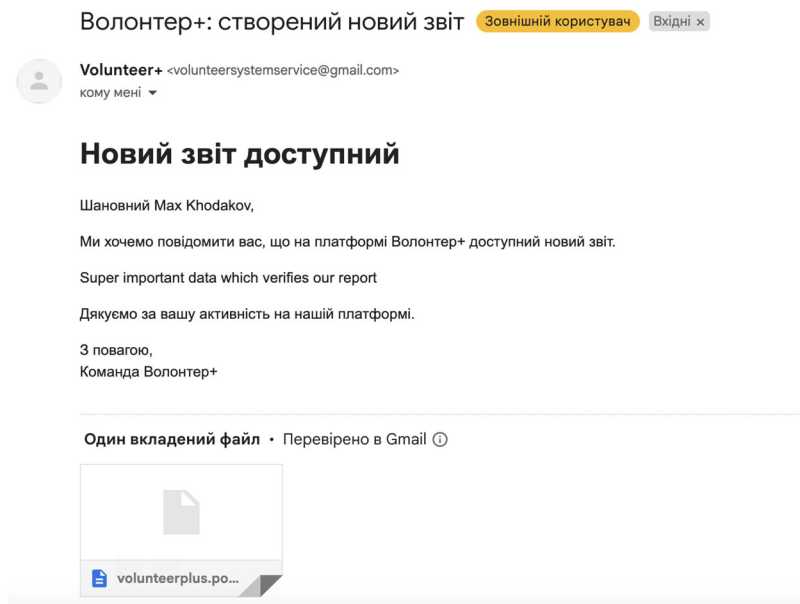


Рис. 28. Імейл створення нового звіту

Стратегія та середовище тестування

Для забезпечення стабільності та надійності серверної частини "Волонтер+" використовується модульне тестування з використанням фреймворку JUnit. Основна мета – перевірити, що всі ключові компоненти системи (сервіси, контролери, утиліти) працюють згідно з очікуваною логікою в різних сценаріях.

Тестування охоплює як прості окремі методи (наприклад, створення нового запиту або перевірка статусу збору), так і більш складні взаємодії між класами. Для ізоляції залежностей (наприклад, бази даних або зовнішніх API) застосовується мокінг із бібліотекою Mockito, що дозволяє зосередитися саме на логіці бізнес-процесів.

JUnit-тести інтегруються в пайплайн розробки та виконуються при кожній зміні коду, що дозволяє швидко виявляти помилки й не допускати їх у фінальну версію продукту. Такий підхід робить розробку безпечнішою, прискорює внесення змін і підвищує загальну якість платформи[14,16,17].

Модульні тести (див. рис. 29).

TESRUNNER: 125 total, 125 passed		1 m 8 s
Collapse Expand		
donorSeesRequestStatusInRealTime557()	passed	775 ms
donorSeesRequestStatusInRealTime907()	passed	631 ms
donorSeesRequestStatusInRealTime934()	passed	535 ms
ratingAffectsVolunteerRanking77()	passed	474 ms
updateRequestStatusToCompleted269()	passed	441 ms
filterRequestsByStatusCompleted58543()	passed	464 ms
requestTimeEstimationAlgorithm()	passed	767 ms
paginationInVolunteerRequestList()	passed	801 ms
reportIncludesAllRequiredFields325()	passed	561 ms
reportIncludesAllRequiredFields858()	passed	739 ms
filterRequestsByStatusCompleted585()	passed	131 ms
viewDetailedReportHistory()	passed	485 ms
adminRemovesInappropriateReport()	passed	259 ms
adminMarksRequestAsSpam446()	passed	484 ms
donorLeavesFeedbackForVolunteer()	passed	370 ms

Рис. 29. Модульні тести Волонтер+

4. ДОКУМЕНТАЦІЯ ПРОЄКТУ "ВОЛОНТЕР+"

Документація API

Для зручності роботи з backend частиною платформи "Волонтер+" використовується інтерактивна документація, згенерована за допомогою Swagger (OpenAPI). Цей інструмент дозволяє описати всі доступні REST-ендпоінти системи – включно з методами, параметрами запитів, типами даних і прикладами відповідей[17].

Swagger дає змогу не лише читати документацію, а й взаємодіяти з API прямо з браузера, що значно спрощує тестування для розробників, волонтерських інтеграторів та технічних партнерів. Це особливо корисно під час командної роботи або при відкритті API для зовнішніх сервісів (див. рис. 30).

Документація оновлюється автоматично при зміні контролерів, що зменшує ризик помилок та забезпечує актуальність. Завдяки Swagger усі сторони мають чітке уявлення про структуру запитів і відповідей, що сприяє швидшій інтеграції та прозорості роботи системи, як зображено на рис. 31.

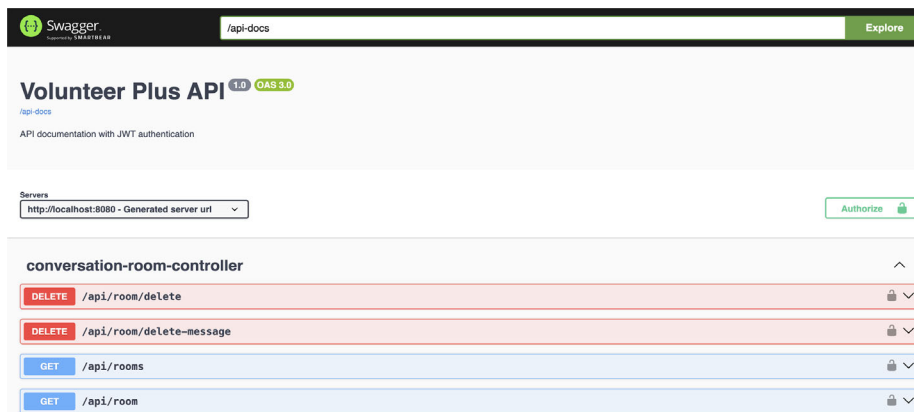


Рис. 30. Фрагмент Swagger документація API

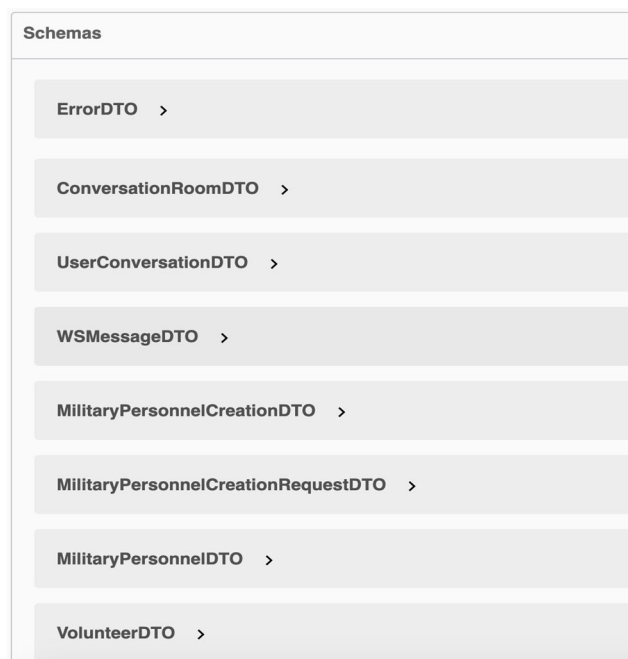


Рис. 31. фрагмент схем Swagger API

Зовнішня документація (для користувачів, партнерів, регуляторів)[29]

У кожній підсекції наведені кроки для п'яти ролей:

- Незареєстрований користувач;
- Супер адміністратор;
- Адміністратор бригади;
- Волонтер;
- Військовий.

Незареєстрований користувач

Підтримати збір (зробити пожертву без реєстрації):

- Перейдіть на головну сторінку платформи (див. рис. 32).

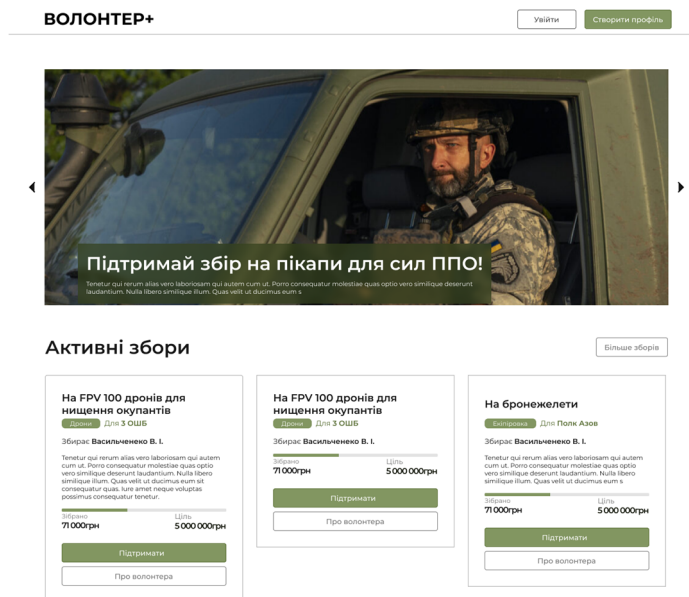


Рис. 32. Головна сторінка сайту

- Виберіть збір, який хочете підтримати, і натисніть "Підтримати". У формі введіть: ваш email (обов'язково), суму внеску в гривнях (див. рис. 33).

Підтримати збір

На FPV 100 дронів для нищення окупантів

Дрони Для З ОШБ

Збирає Васильченко В. І.

Зібрано 71 000грн Ціль 5 000 000грн

Tenetur qui rerum alias vero laboriosam qui autem cum ut. Porro consequatur molestiae quas optio vero similique deserunt laudantium. Nulla libero similique illum. Quas velit ut ducimus eum s

Ваш email *

example@example.ua

Сума вашого внеску *

₴ 0.00

Підтримати

Про волонтера

Рис. 33. Сторінка "Підтримати збір"

- Натисніть "Підтримати", після чого система перенаправить на сторінку оплати.
- Завершіть платіж через платіжну систему.

Супер Адміністратор

Додати бригаду:

- Увійдіть → "Бригади" → "Додати бригаду", відповідно до рис. 34.

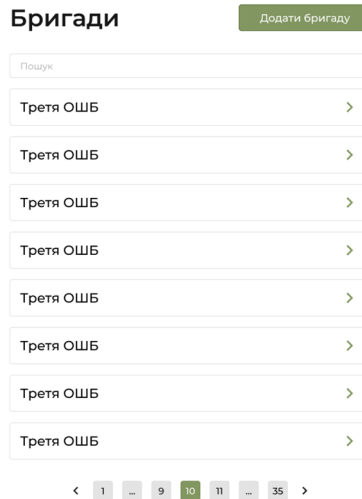


Рис. 34 – Список бригад

- Заповніть: назву бригади, email адміністратора, опис (див. рис. 35).

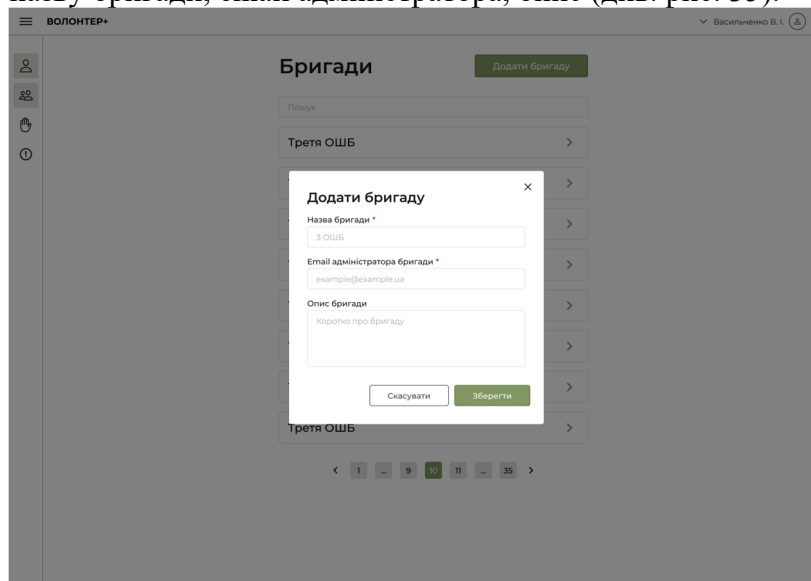


Рис. 35. Форма створення бригади

- Натисніть "Зберегти".

Адміністратор бригади

Додати військового:

- Увійдіть → "Моя бригада" → "Запрошення до бригади"
- Натисніть "Додати запрошення", введіть кількість кодів, відповідно до рис. 36.

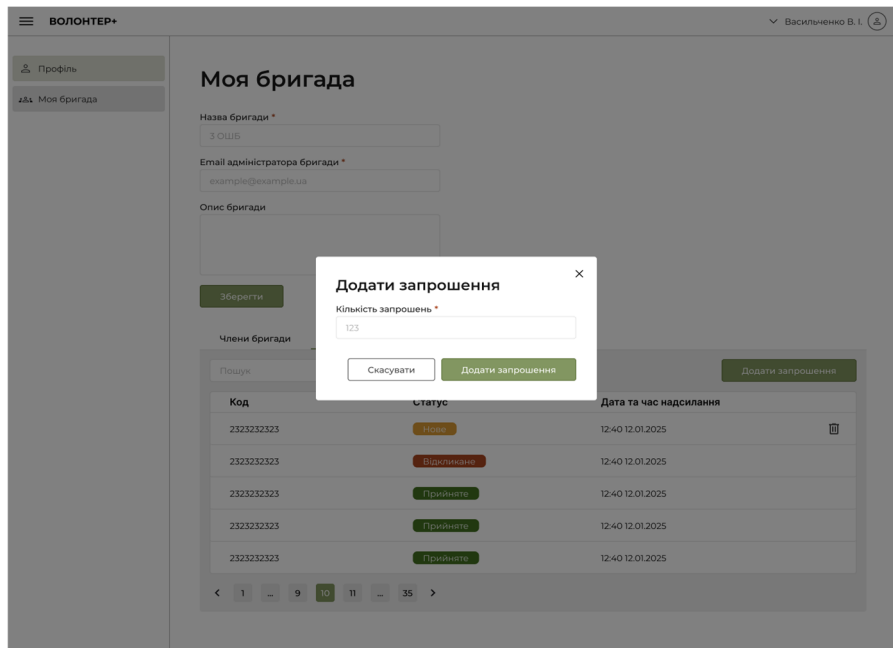


Рис. 36. Форма створення бригади

– Скопіюйте або завантажте PDF із кодами. Як вказано на рисунку 37.

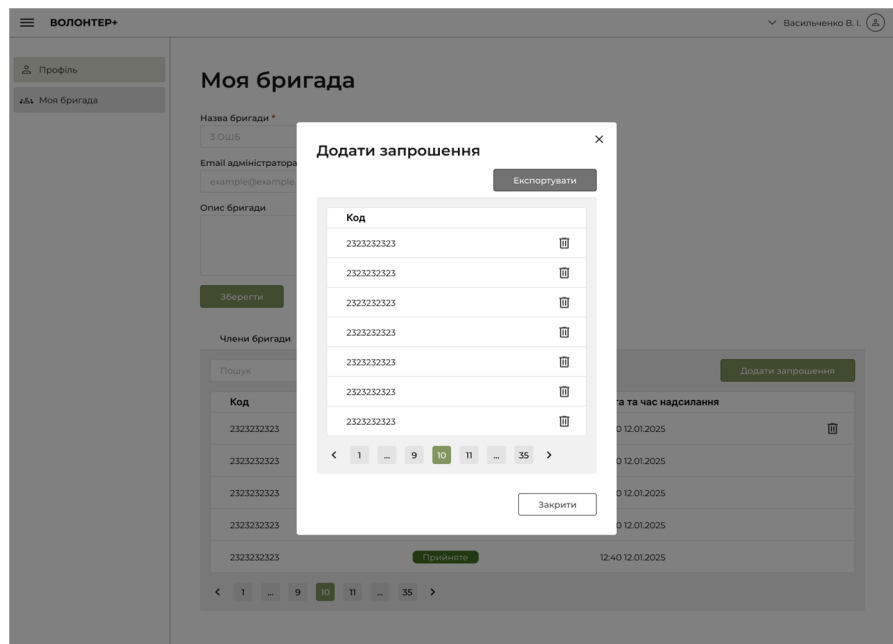


Рис. 37. Список згенерованих запрошень

Перегляд учасників:

- У вкладці "Члени бригади" переглядайте список.
- За потреби фільтруйте або вмикайте показ неактивних.

Редагувати бригаду:

- У вкладці "Моя бригада" змініть назву, опис чи email.
- Натисніть "Зберегти". Відповідно до рис. 38.

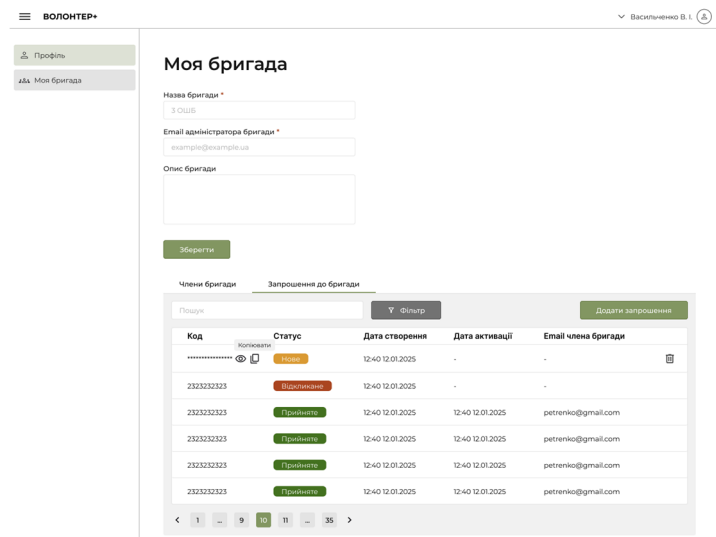


Рис. 38. Сторінка "Моя бригада"

Волонтер

Перегляд запитів:

- Перейдіть у "Запити" → "Доступні запити".
- Натисніть на запит, щоб переглянути (див. рис. 39).

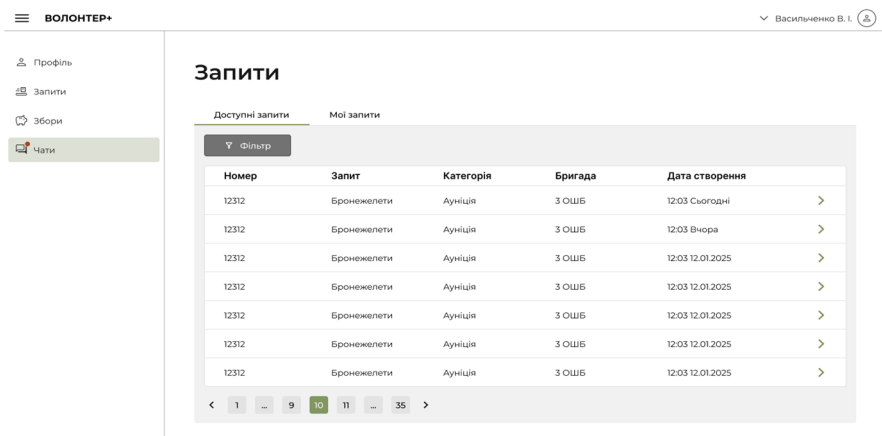


Рис. 39. Список запитів

Надіслати пропозицію:

- Відкрийте запит → натисніть "Запропонувати допомогу".
- Напишіть повідомлення → натисніть "Надіслати". Як показано на рис. 40.

Створити збір:

- "Збори" → "Додати збір".
- Заповніть: назву, категорію, суму, опис → натисніть "Зберегти" (див. рис. 41)

Військовослужбовець

Створити запит:

- "Запити" → "Додати запит".
- Заповніть заголовок, категорію, опис, крайній термін → "Зберегти". Як зображено на рис. 42.

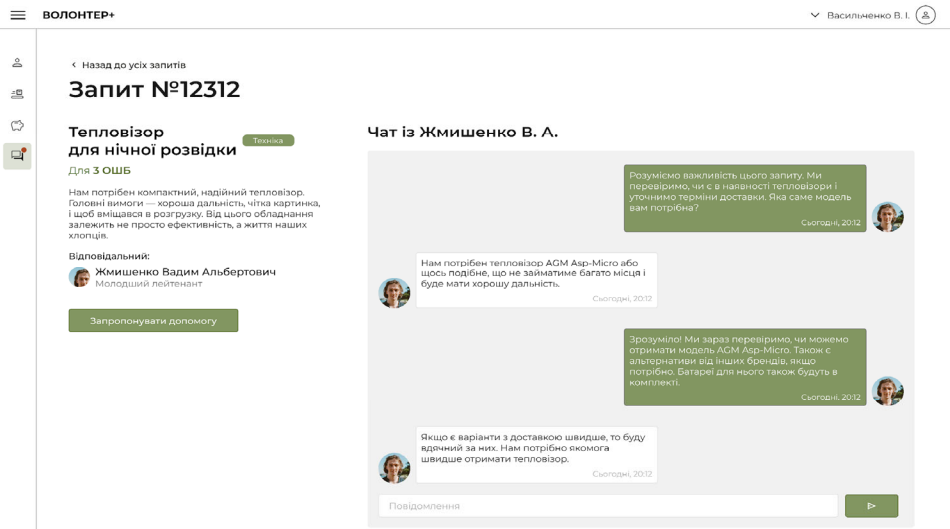


Рис. 40. Сторінка Запиту

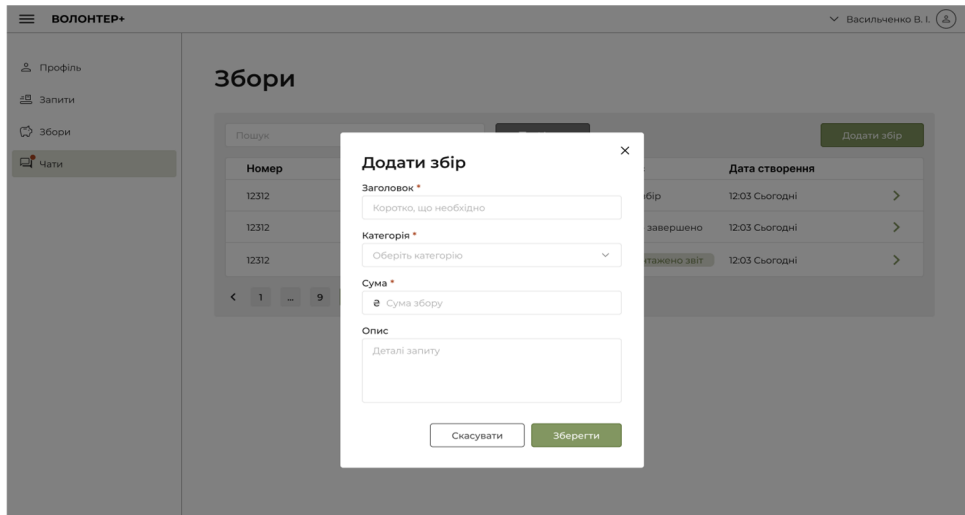


Рис. 41. Форма створення збору

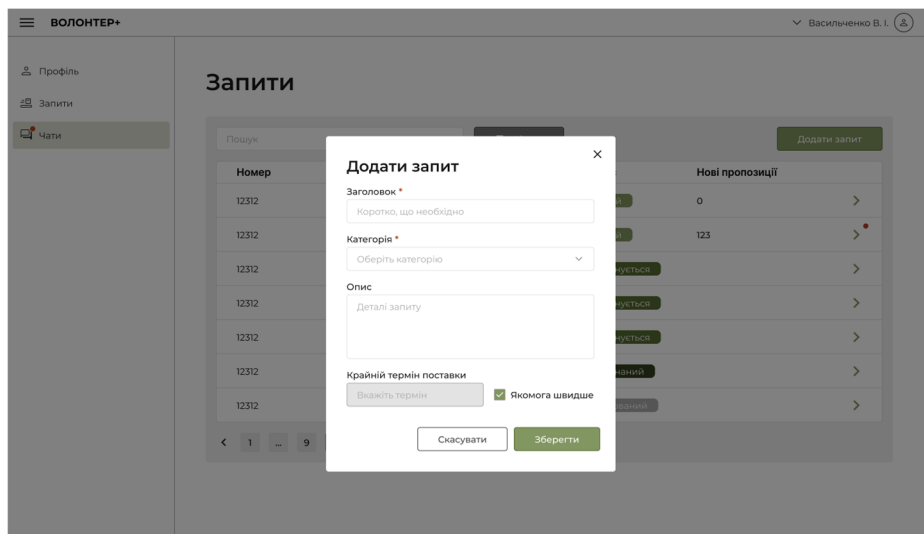


Рис. 42. Форма створення запиту

Перегляд і прийняття пропозицій:

- Відкрийте запит.
- Перегляньте список пропозицій.
- Натисніть "Прийняти", (див. рисунок 43).

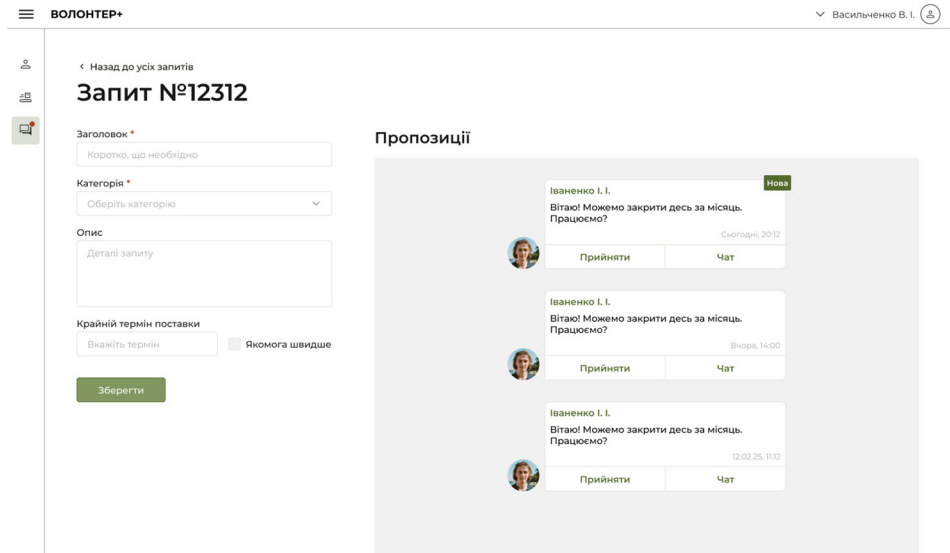


Рис. 43. Сторінка запити

Підтвердити виконання:

- Відкрийте запит після допомоги.
- Натисніть "Запит виконано".

Правове середовище та норми дотримання. Визначаються ключові вимоги до обробки персональних даних, забезпечення платіжної безпеки, захисту авторських прав, доступності та безпеки програмного забезпечення (див. таблицю 6).

Таблиця 6. Правове середовище та норми дотримання

Сфера	Норма / Рамка	Як виконується у "Волонтер+"
Персональні дані	GDPR 2016/679, Закон України № 2297VI (2010)	Збір – за згодою; право на видалення;
Платіжна безпека	PCI DSS v4.0 (оплата карткою)	Токенізація; карткові дані не зберігаємо.
Авторське право	Закон України "Про авторське право та суміжні права", ліцензія MIT (вихідний код)	Репозиторій з файлом LICENSE.
Доступність	WCAG 2.1 AA, Постанова КМУ № 106 (2023)[20]	Контраст $\geq 4.5:1$; HTML теги alt та arialabel.
Безпека ПО	OWASP Top10 2023 [11], ISO/IEC 27002:2022	CSP, XSS фільтри, 2FA

Забезпечення відповідності відбувається через впровадження чітких внутрішніх політик і процедур, що регламентують збирання персональних даних на умовах згоди з правом їх видалення, токенизацію карткових даних і відмову від їхнього зберігання, розміщення коду під ліцензією MIT у відкритому репозиторії з файлом LICENSE, а також застосування контрастних палітр із відповідними HTML-атрибути для

доступності. Додатково налаштовано механізми Content Security Policy[25], фільтри XSS і двофакторну автентифікацію, а процеси регулярно перевіряються відповідно до стандартів OWASP Top10 і ISO/IEC 27002[16]. Результати внутрішніх та зовнішніх аудитів документуються в реєстрі інцидентів і запитів на доступ до даних, а команда проходить системні навчання з питань безпеки й нормативних змін. Такий комплексний підхід мінімізує юридичні та фінансові ризики, підвищує прозорість процесів і зміцнює довіру користувачів "Волонтер+".

5. УПРАВЛІННЯ ПРОЄКТОМ "ВОЛОНТЕР+"

Команда та ролі. Кожен проєкт – це немов музичний твір, де кожен виконавець має свою партію: хтось веде мелодію ідей, хтось додає ритм коду, а хтось дбає про безпеку та якість звучання. Саме завдяки злагодженій грі різних ролей наростає гармонія командної роботи й народжується досконалий продукт (див. таблицю 7).

Таблиця 7. Ролі та обов'язки

Учасник	Роль у спринті	Основні обов'язки
Богдан Чебан	Product Owner / ScrumMaster[22]	Формує Backlog, проводить Refinement & Review, фасилітує ретроспективи, спілкування зі стейкхолдерами.
Денис Шевель	Front-end Developer	Розробка інтерфейсу користувача на базі React/Redux. SEO-оптимізація. Підтримка конфігурації клієнтського додатку
Артем Бариш	Backend Developer / Security Specialist	Spring Boot, модуль платежів LiqPay[23], Spring Security, OAuth2, JWT, аналітика по OWASP.
Максим Ходаков	Backend Developer / DB lead	Проектування DB схеми, Spring Boot, OpenAPI/Ollama, email-service[27], Redis[26], WebSocket.
Євгеній Сьомкін	QA	TestPlan, JUnit 5, перевірка OWASP Top10[10,13,14,16].

Після впорядкованого розподілу ролей і обов'язків кожен учасник набуває чіткої відповідальності за свій фрагмент роботи. Така прозорість сприяє оперативній комунікації, швидкому виявленню слабких місць і своєчасному усуненню ризиків. Об'єднуючи зусилля, команда рухається вперед із високою ефективністю й натхненням, перетворюючи кожен ідею на якісний результат.

Методологія та інструментарій. Управління проєктом нагадує тонке налаштування механізму: кожен елемент методології та кожен інструмент мають чітко вписуватися у загальну картину, забезпечуючи плавний рух і швидке реагування на зміни. Саме тому для нашого проєкту обрано гнучкі підходи та перевірені платформи (див. таблицю 8).

Обрані методології та інтеграції необхідних інструментів команда отримує можливість оперативно відстежувати прогрес, адаптуватися до нових вимог і підтримувати ефективну комунікацію. Такий підхід сприяє прозорості процесів,

зменшенню ризиків і підвищенню загальної продуктивності, дозволяючи зосередитися на створенні цінності для кінцевих користувачів.

Таблиця 8. Вибір методології

Аспект	Вибір	Обґрунтування
Процес	Scrum (2 тижневі спринти) [22,23]	Динамічні змінні вимоги благодійної сфери; потрібен швидкий зворотний зв'язок волонтерів.
Відстеження	Jira Software Cloud[24]	Jira обрана для управління проєктом та відстеження задач, оскільки вона дозволяє ефективно планувати, призначати завдання та контролювати їх виконання. Гнучкість налаштувань, зручний інтерфейс і можливість інтеграції з іншими інструментами забезпечують прозорість робочого процесу і полегшують співпрацю між командою[24].
Комунікації	Google Meet, Discord	Google Meet обрано для проведення онлайн-зустрічей та презентацій через стабільність відеозв'язку та інтеграцію з іншими інструментами Google, що забезпечує зручність планування та комунікації. Discord використовується для щоденної комунікації, обміну файлами та організації задач у реальному часі завдяки зручному розподілу на канали та можливості голосових чатів, що сприяє ефективному спілкуванню та швидкому прийняттю рішень.

Планування спринтів. Проводились узгодження щодо завдань в команді. У Sprint Backlog відображаються найважливіші завдання, їхні пріоритети й оцінки складності, що створює чітку карту руху на найближчий ітераційний цикл (див. рис. 44).

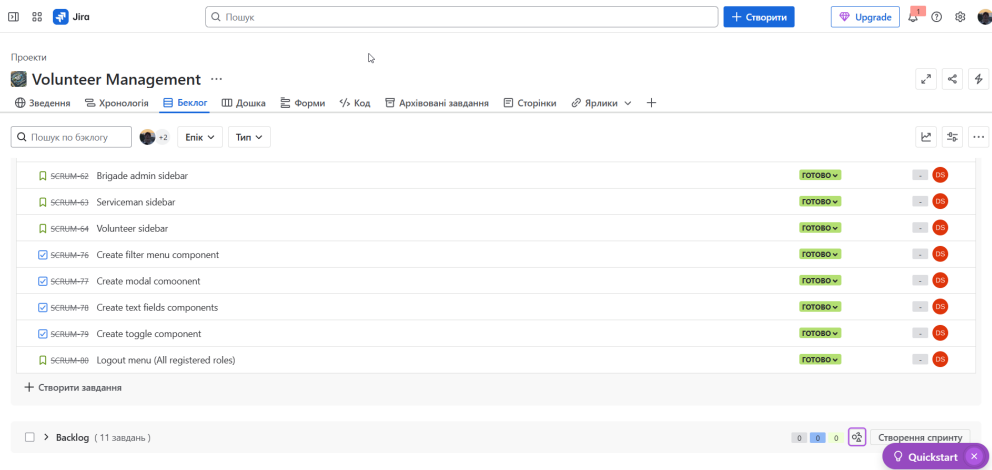


Рис. 44. Фрагмент Sprint Backlog з Jira

Ретельне формування Sprint Backlog дає змогу команді розпочати роботу з упевненістю, розуміючи, які результати очікуються наприкінці спринту. Такий структурований підхід спрощує моніторинг прогресу, сприяє своєчасному виявленню перешкод і забезпечує прозорість процесу для всіх учасників. Відтак команда може сфокусуватися на доставці максимальної цінності, еволюціонуючи разом із вимогами проєкту.

Burndown chart – це своєрідний пульс спринту, що відображає темп виконання завдань від старту до фінішу ітерації. Він наочно демонструє залишок роботи в кожний момент часу, дозволяючи команді оперативного помітити відставання або перевиконання плану (див. рис 45).

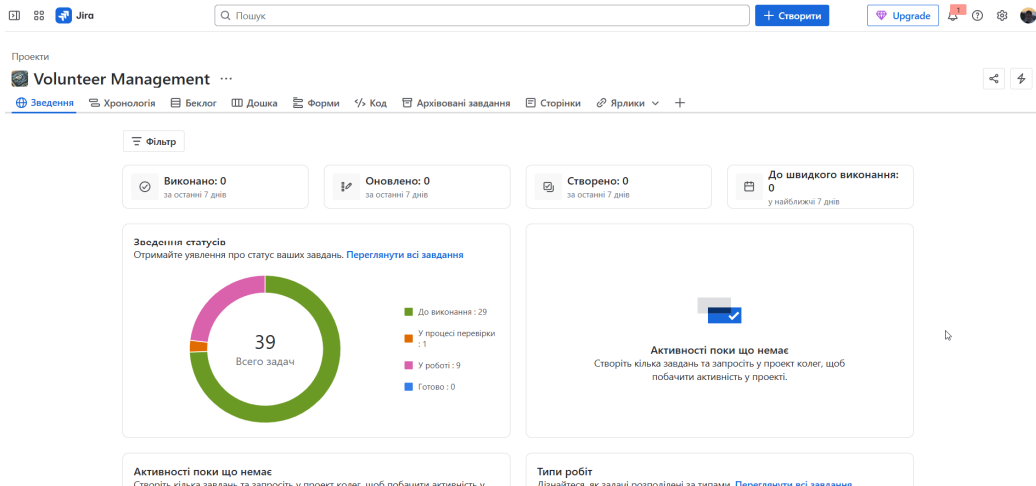


Рис. 45. Burndown chart.

Завдяки регулярному аналізу Burndown chart команда може вчасно реагувати на відхилення від запланованого графіка, коригувати стратегії та підтримувати сталий ритм роботи. Така прозорість процесу сприяє кращій координації, підвищенню ефективності й своєчасній доставці цінності користувачам.

Рефлексія команди. Своєрідне дзеркало, у якому відображаються не лише досягнення й невдачі, а й уроки, що ведуть уперед. Аналізуючи результати кожного спринту, ми вчимося працювати злагодженіше та оперативніше реагувати на виклики (див. таблицю 9).

Таблиця 9. Рефлексія по завданням

Що вдалося	Чому
0 blocking багів у Sprint 4	Збільшили кількість командних колів
Надмірні Story Points у Sprint 2 (+30 %)	Переоцінили швидкість створення Redux слайсів. Провели Story Splitting, ввели Planning Poker.
Запізнення з WCAG аудитом[20]	Не було експерта A11y.

Завдяки регулярному обміну враженнями та виявленню причин успіхів і помилок команда здобуває цінний досвід, який закладає фундамент для наступних спринтів. Така прозорість дозволяє оптимізувати процеси, вчасно коригувати підходи й підтримувати колективний розвиток, що в результаті підвищує якість продукту і згуртованість команди.

Чого навчилися (індивідуальна рефлексія)

- **Богдан Ч.** – освоїв Jira Automation[24].
- **Денис Ш.** – покращив навички в розробці за допомогою таких інструментів, як React, Redux і Jira, а також навчився ефективно працювати в команді та враховувати потреби користувачів для створення зручного інтерфейсу.
- **Артем Б.** – інтеграція LiqPay Server2Server та досвід роботи з HMACSHA1 signature[23].

– **Максим Х.** – Під час роботи над індивідуальним проєктом я покращив навички комплексного рефакторингу міграцій бази даних із використанням Flyway – упорядкував і оптимізував скрипти, що підвищило надійність та прозорість процесу оновлення схеми. Паралельно я навчився впровадити інтеграцію штучного інтелекту, що дозволило автоматизувати ключові бізнеспроцеси та значно розширити функціонал системи.

– **Євгеній С.** – автоматизував тестування використовуючи Junit, провів OWASP Top 10 аналіз застосування [13,14].

ВИСНОВКИ

У дослідженні було визначено три стратегічні цілі: створити єдину вебплатформу для координації військових, волонтерів і донорів, забезпечити прозорість фінансових потоків через інтеграцію з платіжним шлюзом і реалізувати двофакторну аутентифікацію з дотриманням вимог OWASP Top 10. Ці цілі було повністю виконано: розроблено SPA-застосунок на базі Spring і React, інтегровано LiqPay API v3 Checkout для надійної обробки платежів, застосовано JWT для авторизації та TOTP як другий фактор. У межах прикладних завдань проведено ґрунтовний аналіз існуючих рішень і побудовано SWOT-матрицю, результати якої лягли в основу вибору стратегії "web-only"; сформовано детальну схему Swagger API й описано API-контракти в форматі OpenAPI 3.1 (YAML доступні за адресою /swagger-ui/index.html); розроблено модуль платежів LiqPay з коректною обробкою статусів sandbox і success; додано email-сервіс на базі Google SMTP із асинхронною обробкою повідомлень; створено систему внутрішньої аналітики з використанням OpenTelemetry, Jaeger та Spring Boot Actuator. Розроблено систему внутрішньої аналітики, що дозволяє в реальному часі відстежувати телеметрію застосування, аналізувати показники продуктивності та оперативно реагувати на відхилення від норми. Досліджено ринок гуманітарних платформ, нормативно-правову базу (GDPR, WCAG 2.1, Регламент ЄС 2019/1150) та технічні ризики OWASP, що забезпечило комплексне розуміння потреб користувачів і вимог до безпеки й доступності.

У результаті отримано інформаційно-краудфандингову платформу "Волонтер+", яка забезпечує централізовану, прозору та безпечну взаємодію між військовими, волонтерами та донорами й готова до масштабування. Аргументація вибору сфери та ідеї "Волонтер+" полягає в тому, що існуючі сервіси або фрагментарні (лише збір коштів чи лише координація), або не охоплюють усі три групи користувачів, і наша платформа закриває цю прогалину, водночас залучаючи волонтерів і донорів до прозорих процесів, що значно підвищує довіру громадськості та ефективність гуманітарних ініціатив. Для подальшого розвитку рекомендовано впровадити модуль push-нотифікацій через Service Worker, додати підтримку локалізацій англійською та польською мовами й підвищити рівень доступності до стандарту WCAG AAA.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Іваненко І. Л. Аналіз систем координації гуманітарної допомоги. – Київ : NGO Research, 2023.

2. Smith A., Jones B. Benchmarking Volunteer Platforms. – Journal of Crisis Informatics, 2024.

3. Результати онлайн-опитування "Волонтер+" (n = 120). – Січень 2025. (дата звернення: 17.04.2025).

4. SWOT-аналіз ринку гуманітарних платформ. URL: <https://www.apteka.ua/article/13729> (дата звернення: 17.04.2025).

5. SimilarWeb Market Snapshot / AppMagic Dashboard (дані про трафік та завантаження, лютий 2025). URL: <https://www.similarweb.com> (дата звернення: 17.04.2025).

6. U.S. Embassy Kyiv Ukraine. URL: <https://ua.usembassy.gov>. (дата звернення: 17.04.2025)
7. Osterwalder A., Pigneur Y. Business Model Generation. – Hoboken : Wiley, 2010.
8. Meta Business Suite. Campaign Report "UA Charity". – Грудень 2024.
9. Fowler M. UML Distilled. – Boston : Addison-Wesley, 2018.
10. OWASP Foundation. OWASP Top 10 Web 2023. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 17.04.2025).
11. Krug S. Don't Make Me Think. – San Francisco : New Riders, 2014.
12. ISO 9241-210:2019. Ergonomics of human-system interaction. – Женева : ISO, 2019.
13. MariaDB Documentation. URL: <https://mariadb.org/documentation/> (дата звернення: 17.04.2025).
14. Gamma E., Beck K., et al. JUnit 5 User Guide. – JUnit Team, 2023. URL: <https://junit.org/junit5/docs/current/userguide/> (дата звернення: 17.04.2025).
15. Microsoft. Playwright Test Runner Docs. – Redmond, 2024. URL: <https://playwright.dev/docs/testrunner> (дата звернення: 17.04.2025).
16. OWASP Foundation. ZAP User Manual v2.14. – 2024. URL: <https://www.zaproxy.org/docs/> (дата звернення: 17.04.2025).
17. Apache Software Foundation. Apache JMeter User Manual v5.6. – 2023. URL: <https://jmeter.apache.org/usermanual/> (дата звернення: 17.04.2025).
18. ISO/IEC/IEEE 12207:2017. Systems and software engineering – Software lifecycle processes. – Женева : ISO/IEC, 2017.
19. ISO/IEC 26514:2022. Systems and software engineering – Documentation of software systems. – Женева : ISO/IEC, 2022.
20. W3C. Web Content Accessibility Guidelines (WCAG) 2.1. – 2018. URL: <https://www.w3.org/TR/WCAG21/> (дата звернення: 17.04.2025).
21. European Parliament. Regulation (EU) 2019/1150 on promoting fairness of online platforms. – Страсбург, 2019.
22. Schwaber K., Sutherland J. The Scrum Guide (2020 Edition). URL: <https://scrumguides.org/> (дата звернення: 17.04.2025).
23. LiqPay Developers. LiqPay API v3 Documentation. – 2024. URL: <https://www.liqpay.ua/documentation> (дата звернення: 17.04.2025).
24. Atlassian. Jira Software Sprint Report Field Guide. – Сідней, 2023. URL: <https://support.atlassian.com> (дата звернення: 17.04.2025).
25. Snyk Ltd. State of OpenSource Security 2024. URL: <https://snyk.io/opensourcesecurity2024> (дата звернення: 17.04.2025).
26. Redis Inc. Redis 6.2 Commands Reference. URL: <https://redis.io/commands/> (дата звернення: 17.04.2025).
27. SendGrid. Transactional Email API v3 Guide. – Денвер, 2024. URL: <https://docs.sendgrid.com/apireference/> (дата звернення: 17.04.2025).
28. Docker Ltd. Docker Engine Security WhitePaper. – 2023. URL: <https://docs.docker.com/engine/security/> (дата звернення: 17.04.2025).
29. Recharts Team. Recharts v2 Documentation. – 2024. URL: <https://recharts.org/enUS/> (дата звернення: 17.04.2025).
30. Fitts' Law. URL: <https://www.interaction-design.org/literature/topics/fitts-law?srsltid=AfmBOoqtKXujqSLol6o6gKSOLKfCI1-Is6U8UcVVhhsOL2NQP-Z4tYA9> (дата звернення: 17.04.2025).
31. Hick's Law. URL: <https://lawsofux.com/hicks-law/> (дата звернення: 17.04.2025).
32. Miller's Law. URL: <https://lawsofux.com/millers-law/> (дата звернення: 17.04.2025).
33. Jakob's Law. URL: <https://lawsofux.com/jakobs-law/> (дата звернення: 17.04.2025).

РОЗРОБКА СИСТЕМИ "BRIEFLY"

***Кирило БЕДРАК, Олександр ОСТРЕНКО, Матвій КОВАЛИК,
Яна МАЛАШКІНА, Юліана КАРПЦОВА, Анастасія МОЗГОВА,
Андрій ТОЛСТОПЯТИЙ, Владислав ШПАРАГОВСЬКИЙ***

Дана робота описує процес створення та роботи над системою "BRIEFLY" – інструменту для автоматичного підсумовування повідомлень, текстів, текстових документів і контенту з YouTube-відео. Рішення оптимізує роботу з великими обсягами інформації, підтримуючи українську мову й альтернативні платформу Discord. Розроблено вебзастосунок, що забезпечує сумаризацію текстів і мультимедійних джерел через API. У роботі висвітлено аналіз ринку, моделювання алгоритмів генеративного й екстрактивного підсумовування, створення API та оптимізацію сервісів для зменшення споживання пам'яті.

**ПІДСУМОВУВАННЯ ТЕКСТУ, ПІДСУМОВУВАННЯ PDF, ПІДСУМОВУВАННЯ
ВІДЕО YOUTUBE, ВЕБЗАСТОСУНОК, ГЕНЕРАТИВНІ АЛГОРИТМИ,
ЕКСТРАКТИВНІ АЛГОРИТМИ, РОЗРОБКА API**

DEVELOPMENT OF THE "BRIEFLY" SYSTEM

***Kyrylo BEDRAK, Oleksandr OSTRENKO, Matviy KOVALYK,
Yana MALASHKINA, Yuliana KARPTSOVA, Anastasiia MOZHOVA,
Andriy TOLSTOPIATY, Vladyslav SHPARANOVSKY***

This report describes the development and implementation of the "BRIEFLY" system – a tool for automatic summarisation of messages, texts, documents, and YouTube video content. The solution enhances the efficiency of processing large volumes of information, with support for the Ukrainian language and alternative platforms such as Discord. A web application was developed to enable text and multimedia summarisation through an API. The report outlines market analysis, the modeling of generative and extractive summarisation algorithms, API development, and service optimisation to reduce memory consumption.

**TEXT SUMMARIZATION, PDF SUMMARIZATION, YOUTUBE CONTENT
SUMMARIZATION, WEB APPLICATION, GENERATIVE ALGORITHMS,
EXTRACTIVE ALGORITHMS, API DEVELOPMENT**

ВСТУП

У сучасному світі, де обсяги інформації постійно зростають, особливо актуальною стає проблема ефективного опрацювання великих масивів даних у цифровому середовищі. Месенджери, які сьогодні є ключовими інструментами комунікації, створюють значний потік інформації, що ускладнює швидке сприйняття важливих даних. Проєкт "Briefly" спрямований на розробку рішення, яке допомагає користувачам автоматично підсумовувати повідомлення, тексти та статті, надаючи стислу й точну інформацію.

Актуальність розробки проєкту підкреслюється особливими обставинами локального ринку. Наприклад, ситуація із заборонаю популярного месенджера у багатьох українських університетах показала необхідність створення інструменту, що підтримує альтернативні платформи, такі як Discord. "Briefly" став відповіддю на цей виклик,

дозволяючи користувачам різних платформ отримувати якісні автоматичні підсумки. Особливий акцент на підтримці української мови вирізняє продукт серед інших, сприяючи вирішенню проблеми нестачі локалізованих рішень.

Розробка продукту базувалася на таких методах дослідження:

- аналіз існуючих рішень на ринку для підсумовування текстів і чатів;
- моделювання алгоритмів генеративного та екстрактивного підсумовування.

Метою проєкту є створення універсального інструменту для автоматичного підсумовування текстів, який дозволить оптимізувати роботу з інформацією для широкої аудиторії користувачів, особливо українських.

Основні завдання проєкту включають:

- аналіз і вибір оптимальних алгоритмів обробки тексту;
- розробку API для взаємодії з платформами;
- тестування й вдосконалення алгоритмів підсумовування тексту;
- розробка вебсайту.

Окрім прикладних завдань, проєкт має на меті професійний розвиток учасників команди. Реалізація "Briefly" дала змогу закріпити теоретичні знання, поглибити практичні навички з розробки API, обробки тексту та інтеграції систем, а також вивчити сучасні підходи до автоматизації обробки інформації. Це забезпечує високий рівень професійної підготовки розробників і сприяє підвищенню їхньої компетентності у сфері інформаційних технологій.

1. ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

Опис пошуку ідеї продукту

Сучасна цифрова екосистема характеризується стрімким зростанням обсягу текстового та медіаконтенту на різних платформах – у месенджерах, на YouTube, у документах та електронній пошті. В умовах постійного інформаційного навантаження користувачам стає дедалі складніше швидко орієнтуватися в отриманій інформації та виокремлювати ключові ідеї. Особливо це актуально в корпоративному середовищі, навчанні та особистому користуванні, де обсяги контенту щодня зростають.

Наша ідея – створити універсальну платформу "Briefly" для автоматичного підсумовування текстів, чатів і відео за допомогою генеративних моделей. Основна мета – забезпечити користувачів стислими та структурованими резюме контенту, зекономити час на обробку великих обсягів інформації та підвищити ефективність роботи з даними.

Під час брейншторму команда обговорила п'ять потенційних напрямків розвитку продукту:

- Сервіс для підсумовування чатів, документів і відео;
- Електронний універсальний мовний словник;
- Аналог Grammarly з підтримкою української мови;
- Сервіс для заміни англіцизмів у текстах;
- Додаток для розвитку навичок публічного мовлення.

Для кожної ідеї було проведено SWOT-аналіз (Рисунок 1), що включав оцінку сильних і слабких сторін, можливостей і загроз.

Збірник матеріалів за результатами IT-проекту міждисциплінарної інтеграції, 2024-2025 н.р.

Ідея	Сутність	Сильні сторони	Слабкі сторони	Можливості	Загрози
Ідея 1: Бот для підсумків чату	Розробка чат-бота або API, який автоматично створює стислий огляд повідомлень у чатах за визначений період.	- Актуальність для великих групових чатів у месенджерах. - Універсальність API для роботи з різними платформами (WhatsApp, Discord, Google Chat). - Можливість персоналізації (довжина, стиль підсумку). - Простота інтеграції у популярні месенджери.	- Конкуренція зі сторонніми генеративними ШІ (GPT). - Технічні обмеження на деяких платформах. - Складність монетизації.	- Попит на автоматизацію обробки тексту. - Додавання нових функцій, таких як аналіз настроїв чи автоматичний переклад. - Інтеграція з новими платформами.	- Конкуренція з великими технологічними компаніями (Google, OpenAI). - Проблеми конфіденційності та обробки даних користувачів.
Ідея 2: Електронний універсальний мовний словник	Створення централізованого словника, який об'єднує дані з різних джерел і надає глибокий лінгвістичний аналіз.	- Перший унікальний продукт для українського ринку. - Високий попит серед науковців, перекладачів, студентів. - Широка сфера застосування.	- Складність реалізації через різноманіття даних. - Високі витрати на лінгвістичні ресурси. - Конкуренція з існуючими рішеннями.	- Створення API для сторонніх сервісів. - Інтеграція з навчальними платформами.	- Низька точність даних через застарілість джерел. - Конкуренція з безкоштовними альтернативами (Google Translate).
Ідея 3: Аналог Grammarly	Сервіс для перевірки граматики, стилістики та орфографії текстів українською мовою.	- Великий попит на україномовні сервіси. - Актуальність для широкого сегменту користувачів (студенти, письменники, перекладачі).	- Складність реалізації через специфіку української мови. - Високі витрати на тренування моделей.	- Масштабування на інші мови. - Інтеграція з популярними текстовими редакторами.	- Конкуренція з великими компаніями (Grammarly). - Висока вартість обчислень.
Ідея 4: Сервіс для заміни англіцизмів у текстах	Інструмент для пошуку англіцизмів у текстах та їх заміни на відповідники українською мовою.	- Попит серед україномовних користувачів. - Простота реалізації.	- Обмежений функціонал. - Проблеми з заміною незамінних англіцизмів.	- Інтеграція з перевіряючими текстами, як-от «український Grammarly».	- Низька ефективність через обмеженість словника англіцизмів. - Негативна реакція користувачів на заміну усталених термінів.
Ідея 5: Додаток для розвитку навичок публічного мовлення	Програма для аналізу мовлення користувача (слова-паразити, паузи) і вдосконалення його публічних виступів.	- Великий попит серед тих, хто прагне розвивати особисті навички. - Перспективи масштабування.	- Високі вимоги до обробки аудіо. - Складність збору якісних даних.	- Інтеграція з Zoom чи Microsoft Teams. - Розширення на інші ринки.	- Конкуренція з англійськими застосунками. - Проблеми обробки української мови через діалекти й суржик.

Рис. 1. SWOT-аналіз

Аргументація вибору сфери та ідеї розроблюваного продукту. Команда дійшла висновку, що ідея створення мультиплатформного сервісу для підсумовування текстового та медіаконтенту є найбільш перспективною для реалізації. Вона має значні конкурентні переваги завдяки широкому спектру застосування, високому попиту на ринку та гнучким можливостям масштабування як для особистого використання, так і для бізнесу. Наступним кроком стало створення детального плану розробки системи "Briefly", формування вимог до функціоналу та тестування перших прототипів.

Вибір сфери. Сфера автоматизації обробки тексту була обрана через її високу актуальність та швидкий розвиток. Зростаюча кількість текстової інформації, особливо в чатах і месенджерах, створює попит на інструменти, які допомагають користувачам ефективно взаємодіяти з великими обсягами даних.

Ключові аргументи:

- попит: Більшість користувачів стикаються з проблемою перевантаженості інформацією в групових чатах. Автоматизація підсумовування дозволяє зекономити час;
- технологічний потенціал: Сучасні методи обробки природної мови (NLP) дають змогу створювати інструменти, які можуть надавати якісні, персоналізовані підсумки текстів;
- ринок: Україномовний сегмент має значний потенціал, оскільки багато існуючих рішень зосереджені на глобальному ринку, ігноруючи локальні потреби.

Вибір ідеї. Ідея створення бота для підсумків чату має значні переваги, які виділяють її серед інших обговорюваних варіантів.

Сильні сторони:

- інноваційність: Відсутність подібних рішень у сегменті українських користувачів;
- гнучкість: Можливість використання продукту як чат-бота чи API для інтеграції у сторонні платформи;

– якість: Використання сучасних генеративних моделей, таких як GPT від OpenAI[1] або Llama[2] у поєднанні з екстрактивними методами, які представлені у статті Gulati та співавт.[3] запропоновано поєднання TextRank і BM25+ для покращення якості екстрактивного резюмування;

– підтримка локального ринку: Підтримка української мови відповідає запиту місцевих користувачів;

– масштабованість: Розширення функціоналу (аналіз настроїв, переклад тексту) дає можливість виходу на глобальний ринок.

Можливості:

– зростання ринку: Інструменти обробки тексту стають важливими в бізнесі, освіті та особистому користуванні;

– розширення: Продукт може бути адаптований для обробки текстів із відео (наприклад, підсумки YouTube) чи інтеграції з CRM-системами для автоматизації бізнес-процесів;

– партнерства: Співпраця з платформами, такими як Discord чи Slack, дозволить залучити більше користувачів.

Обґрунтування вибору. Порівняно з іншими розглянутими ідеями (електронний словник, аналог Grammarly, сервіс заміни англіцизмів, застосунок для публічного мовлення), ідея створення платформи для підсумовування контенту має такі переваги:

– практичність: Вона вирішує актуальну проблему перевантаження інформацією як у чатах, так і в медіа-контенті, надаючи користувачам швидкий доступ до основних тем;

– технічна здійсненність: Завдяки розвитку генеративних моделей і технологій обробки природної мови (NLP), реалізація продукту є цілком можливою навіть із обмеженими ресурсами;

– ринкові можливості: Відсутність комплексних рішень із сумаризації для локального ринку створює вікно можливостей для успішного виходу і подальшого масштабування платформи.

Виклики та загрози. Хоча ідея має значний потенціал, слід враховувати ризики:

– залежність від зовнішніх API: Використання платних сервісів для NLP може бути дорогим і нестабільним;

– конкуренція: Великі компанії (Google, Microsoft) можуть розвивати аналогічні функції;

– конфіденційність: Обробка текстів із приватних чатів викликає питання безпеки даних;

– технічні ризики: труднощі з інтеграцією, потенційні обмеження з боку месенджерів на використання ботів;

– ризик недооцінки обсягу роботи: недооцінка складності процесу обробки природної мови українською мовою, особливо з урахуванням сленгу та скорочень у чатах;

– ризик конфіденційності: можливі побоювання користувачів щодо збереження конфіденційності повідомлень;

Висновком було те, що бот для підсумків чату – це продукт, який:

– відповідає реальним потребам ринку;

– використовує сучасні технології NLP;

– має значний потенціал для розширення та монетизації.

Це рішення може стати першим якісним інструментом обробки тексту для україномовного сегменту, а також знайти застосування у глобальних масштабах.

Огляд наявних на ринку ІТ-продуктів – конкурентів

Огляд конкурентів був проведений командою для аналізу наявних рішень на ринку, їх функціоналу та можливостей. У процесі роботи було досліджено різні платформи, зокрема месенджери, веб-сервіси та рішення для автоматизації текстового аналізу. Це дослідження допомогло визначити ключові особливості продуктів, які мають попит, та сформулювати основу для розробки власного інноваційного рішення.

З метою оцінки наявних популярних рішень у популярних месенджерах, командою здійснено пошук найпопулярніших телеграм-ботів, які забезпечують функціонал автоматичного підсумовування текстів у чатах. Аналіз охопив як україномовні, так і міжнародні рішення. А також проведено SWOT-аналіз обраних ботів із врахуванням кількості користувачів, якості роботи та моделі монетизації. Це дозволило виділити сильні сторони, виявити недоліки та зрозуміти можливості для покращення функціоналу.

Summarizer BOT[4] – приклад продукту, який не доступний для незареєстрованих користувачів, потребує спеціального пароля з веб версії. Результати аналізу представлені в таблиці 1.

Таблиця 1. Аналіз Summarizer BOT

Мова / мови	англійська
Сильні сторони	невідомі
Слабкі сторони	потребує пароль з веб версії
Якість роботи	невідомо
К-ть користувачів	невідомо
Монетизація	відсутня

Суммаризатор від Synapse[5] – бот, створений комерційною компанією Synapse, яка займається реалізацією технологічних проєктів. Сервіс потребує оплати, тому недоступний для звичайного користувача, який орієнтується на безплатні боти. Бот потребує введення часового поясу, що викликає питання про приватність. Результати аналізу представлені в таблиці 2.

Таблиця 2. Аналіз Суммаризатора

Мова / мови	російська
Сильні сторони	Добре розроблений функціонал: додавання в групу, вибір періоду, за який потрібно підсумувати повідомлення, вибір кількох груп. Добре побудована взаємодія з користувачем
Слабкі сторони	потребує оплати
Якість роботи	висока
К-ть користувачів	невідомо
Монетизація	Оформлення підписки на місяць дає можливість отримувати підсумки з усіх груп, до яких додано бота, протягом 1 місяця. Підписка на місяць коштує 100 Telegram Stars

БіЧ Summarizer[6] – доступний для усіх користувачів без підписки і надає доволі якісні підсумки повідомлень. Результати аналізу представлені в таблиці 3.

Таблиця 3. Аналіз БіЧ Summarizer

Мова / мови	Стандартна мова – англійська, але підтримує українську. Підтримує різні мови, адже використовує API генеративної мовної моделі (можливо, ChatGPT) для створення підсумків; при повторній генерації новий підсумок трохи відрізняється від попереднього
Сильні сторони	<ul style="list-style-type: none"> - Не потребує додавання в групу, адже може робити підсумки пересланих йому повідомлень. - Робить підсумки для 100 останніх повідомлень. - Проводить докладний аналіз посилань, наявних у повідомленнях. Може аналізувати тексти, розміщені за цими посиланнями.
Слабкі сторони	<ul style="list-style-type: none"> - Галюцинації мовної моделі. Іноді видає словосполучення із лексичними та граматичними помилками або слідами суржику, наприклад: "ця веб-сторінка знай знакомить з бібліотекою spaCy". - Відсутність конкретності. Підсумки є надто розмитими та абстрактними. - Відсутність можливості вибирати кількість повідомлень (працює лише для 100 останніх повідомлень) - Функція "саммарі з сарказмом" не працює, хоча наявна у боті та видима для користувача
Якість роботи	висока
К-ть користувачів	невідома
Монетизація	немає

Для тестування якості, боту було відправлено 100 повідомлень із чату 17 гуртожитку КНУ. Результат роботи показано на рис. 2.

<ul style="list-style-type: none"> - Обговорення ремонту в гуртожитку • Проблеми з оплатою за послуги • Бронювання пралок • Взаємодія з сусідами в гуртожитку • Вивчення корейської мови • Спільні заходи та ігри в гуртожитку • Досвід проживання в кімнатах гуртожитку • Стан речей (прання, забір речей) • Досвід догляду за котами • Виклик до спільних покупок або допомоги <p>Друге генерування:</p> <ul style="list-style-type: none"> - Враження про ремонти в гуртожитку • Взаємодія між мешканцями щодо прання та бронювання пралок • Обговорення про тварин, зокрема котів • Питання щодо навчання корейської мови • Ситуації з платежами та банківськими транзакціями • Обговорення соціальних активностей в гуртожитку • Враження про умови життя в гуртожитку та порівняння з іншими будівлями

Рис. 2. Результат роботи БіЧ Summarizer

SB summarizer[7] – це бот, призначений для аналізу великих текстів та статей. Може аналізувати тексти, написані будь-якою мовою, але результати надає виключно англійською. Результати аналізу представлені в таблиці 4.

Таблиця 4. Аналіз Summarizer

Мова / мови	англійська
Сильні сторони	Виконує докладний аналіз тексту, який надсилає користувач
Слабкі сторони	<ul style="list-style-type: none"> - Подає підсумок лише англійською мовою, хоча може аналізувати український текст. Як і всі попередні боти, використовує API генеративної мовної моделі. Орієнтований на англомовного користувача - Підсумки є надто розгорнутими, і підходять скоріше для аналізу тексту, а не для отримання основної інформації
Якість роботи	висока
К-ть користувачів	невідома
Монетизація	немає

На рис. 3 текст статті у публіцистичному стилі на історичну тематику, яку було взято для перевірки роботи.

Текст:
 Salvête, amici! Сьогодні ми з вами поговоримо про улюблений колір імператорів — пурпур!

Якщо ви знаєте значення фрази «вбратися в пурпур», то для вас не буде неочікуваністю той факт, що пішла вона від римлян. У часи, коли всі барвники виготовлялися з натуральних продуктів та матеріалів, одним з найдорожчих кольорів був так званий тірський пурпур. Його отримували з родини морських черевоногих молюсків під назвою Мурексові. Центром виготовлення якийсь період був Тір, що у Фінікії (наразі територія Лівану), звідси цей барвник і отримав свою назву, хоча в історії він більше відомий як «імператорський» чи «царський» пурпур. Чому саме - ми зараз розберемо. 😊

👉 Найлегше припустити, що імператори просто носили найдорожче, що могли знайти, в принципі це не є правдою. Проте, як завжди буває з Римською Імперією, тут присутні деякі витоки з Республіки. У часи, коли Сенат ще мав суттєві повноваження, тірський пурпур використовували для створення церемоніальних тог. Вони ніколи не були пофарбовані повністю (єдине виключення тоги та туніки для триумфу), натомість лише мали широку фіолетову стрічку як окантовку, ця тога praetexta була призначена для курульних магистратів у своїх офіційних функціях, вільнонароджених хлопчиків та деяких вільнонароджених дівчат до повноліття (як знак захисту законом) і для деяких жерців, включаючи понтифіків, авгурів, епулонів та арвальських братів.

🏰 З настанням періоду Принципату пурпурові тоги, що використовувалися для триумфів, перекочують у повсякденний гардероб імператора, так само як повноваження Сенату перекочують до його рук. Згадки про наш багрянний колір можна знайти навіть у законах про розкіш, де регулювався той факт, що ніхто, окрім імператора, не може носити пурпур.

За часів Візантії використання тірського пурпуру чи багрянцю все сильніше укорінювалося в імператорський церемоніал. Чи говорить вам про щось фраза «народжений у пурпурі»? Для підданих Константинополя починаючи з X століття цей вираз означав доволі багато, оскільки був підтвердженням приналежності когось до імператорської родини шляхом прямого наслідування. Отже, як це було. Лев VI, чи як ми його ще знаємо Лев Мудрий, четвертий представник Македонської династії, ввів новий титул до і так грандіозно великого офіціозу Константинопольської ієрархії. Цей титул звучав як — Багрянородний, чи Порфірородний, тобто буквально народжений у пурпурі. Він надавався імператорським дітям, які народилися під час правління їх батька, коли він «носив пурпур», та обов'язково від законної дружини. До такого більш метафоричного значення ще додається той факт, що діти правлячого імператора народжувалися в спеціальній порфіровій (багрянній) кімнаті у Великому палаці Константинополя.

Рис. 3. Приклад тексту для тестування SB Summarizer

Результат роботи бота Summarizer представлений на рис. 4.

Central Thesis or Objective:
 The text discusses the historical significance and cultural evolution of the color purple, specifically Tyrian purple, and its association with imperial power and aristocracy in ancient Rome and Byzantium.

Three Pillar Insights:

1. **Symbol of Power and Privilege:** Tyrian purple, derived from marine mollusks called Murex, was a rare and expensive dye in ancient times, symbolizing power and authority. It was used by the Romans for ceremonial togas and became intimately associated with high-ranking officials and the imperial family.
2. **Transition from Republican to Imperial Rome:** The use of Tyrian purple transitioned from being a symbol of magisterial and religious office in the Roman Republic to a daily emblem of imperial authority during the Principate, with laws restricting the wearing of purple to the emperor alone.
3. **Byzantine Imperial Birthright:** In Byzantium, the phrase "born in the purple" implied a legitimate succession to the throne, as imperial children born during their father's reign were entitled "Porphyrogenitos" or "born in the purple," a practice instituted by Emperor Leo VI.

Author's Stance & Underpinnings:
 The author takes an informative stance, presenting historical facts and anecdotes about Tyrian purple's evolution in symbolism from Republican Rome to Byzantine times. The author outlines the color's significance through its restricted use by specific social classes and notes the development of legislative and ceremonial practices reinforcing purple as the color of the elite.

People, Countries, and Companies Mentioned:

- Tyrian purple is named after Tyre in Phoenicia, now part of Lebanon.
- Roman Senate and magistrates, highlighting their role in the Republic.
- Emperor Leo VI (Leo the Wise) of the Macedonian dynasty, significant for his contribution to Byzantine imperial customs.
- Murex mollusks, the source of Tyrian purple dye.

Reflective Critique:
 The text effectively illustrates the historical importance and prestige associated with Tyrian purple. It presents a clear narrative of how the color's use developed over time, reflecting changing political structures. However, the author assumes a certain level of prior knowledge, as the text references Roman and Byzantine history without extensive background information, which may limit comprehension for some readers. Additionally, the text presents the adaptive

Рис. 4. Результат роботи Summarizer (частина тексту підсумку).

Результатом опрацювання тексту є докладний аналіз, який відображає основну тему, головні тези, думку автора, згадки країн та людей, а також надає коротку критику тексту.

Крім наведених ботів для підсумовування повідомлень та текстів, було знайдено також інші – але вони не відповідали на запити або були орієнтовані лише на російських користувачів (не відповідали сегменту клієнтів, на який орієнтується наша розробка).

У процесі дослідження також було проведено аналіз існуючих рішень (рис. 5), що забезпечують автоматичне підсумовування текстів, чатів та статей у вебсередовищі. Аналіз охопив сервіси, які використовуються для обробки великих обсягів інформації, автоматизації аналізу контенту та полегшення доступу до ключових даних.

Аналіз існуючих рішень для сумаризації текстів і чатів показав, що на ринку переважають інструменти, орієнтовані на англomовну аудиторію, з обмеженою

Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції, 2024-2025 н.р.

підтримкою інших мов, зокрема української. Більшість сервісів фокусуються на екстрактивній сумаризації, тоді як рішення з абстрактивною сумаризацією є дорогими та рідко доступними. Існує також дефіцит інструментів, здатних ефективно працювати з чатами, а не лише зі статичними текстами чи статтями.

Це створює значні можливості для розвитку нашого продукту, який буде орієнтований на українську аудиторію, забезпечуючи якісну мовну підтримку, багатоплатформність і доступність завдяки універсальному API. Таким чином, наш продукт може заповнити важливу нішу на ринку, задовольнивши як потреби локальної аудиторії, так і вимоги до сучасних багатofункціональних рішень.

1	Назва рішення	Тип контенту	Тип сумаризації	Технологія / Модель	Мова підтримки	Вартість	Особливі функції	Плюси	Недоліки
2	Resומר	Текст, статті	Екстрактивна	NLP	Багатомовний	Безкоштовно з обмеженнями, платний Pro	Швидке виділення ключових моментів з великих текстів	Підтримка кількох мов, простий інтерфейс	Лише екстрактивна сумаризація, обмеження у безкоштовній версії
3	SMMRY	Текст, статті, URL	Екстрактивна	Власний алгоритм	Англійська	Безкоштовно	Простий інструмент для швидкого скорочення тексту	Швидкий і зручний у використанні	Лише англійська мова, дуже базова сумаризація
4	QuillBot Summarizer	Текст, статті	Екстрактивна та абстрактивна	GPT-3	Англійська	Безкоштовно з обмеженнями, платний Pro	Можливість налаштування глибини сумаризації	Потужна сумаризація, інтеграція з іншими інструментами QuillBot	Лише англійська мова, обмеження на кількість тексту у безкоштовній версії
5	TLDR This	Текст, статті, URL	Екстрактивна	NLP, ML	Англійська	Безкоштовно з обмеженнями, платний Pro	Сумаризація веб-сторінок за URL, інтеграція з Chrome	Легкий у використанні, ефективний для новинних статей	Лише англійська мова, відсутність абстрактивної сумаризації
6	SummarizeBot	Текст, статті, чати	Екстрактивна та абстрактивна	GPT, TextRank, BERT	Англійська та інші	Платна, з безкоштовним пробним періодом	Підтримка різних типів контенту (тексти, аудіо, відео)	Підтримка багатьох платформ, гнучкість у налаштуваннях	Відсутність підтримки української мови, висока вартість
7	TLDR Bot for Slack	Чати (Slack)	Екстрактивна	NLP	Англійська	Платна з безкоштовним періодом	Автоматична сумаризація чатів у Slack	Зручний для командного використання в Slack	Підтримує лише Slack, відсутність підтримки інших платформ
8	ChatGPT	Текст, чати	Абстрактивна	GPT-4	Багатомовний	Платна (залежить від API виклику)	Потужна абстрактивна сумаризація, інтерактивність	Висока якість результатів, багатомовність	Висока вартість, може бути перевантажений сервіс, відсутність спеціальних функцій для чатів
9	Paper Digest	Наукові статті	Екстрактивна та абстрактивна	NLP, ML	Англійська	Платна з безкоштовним періодом	Автоматична сумаризація наукових статей	Спеціалізований для наукових текстів	Висока вартість, обмеження у використанні для інших типів тексту
10	UpSum	Веб-статті, новини	Екстрактивна	NLP	Англійська	Безкоштовно	Швидка сумаризація новинних статей	Легкий у використанні, фокус на новини	Лише англійська мова, відсутність підтримки інших типів контенту
11									

Рис. 5. Аналіз існуючих рішень

Мапа продукту

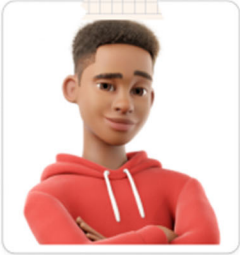
У межах роботи над проєктом у Figma була розроблена мапа продукту "Briefly" (рис. 6), яка чітко візуалізує ключові аспекти проєкту, його функціональні можливості, унікальні переваги та основні сценарії використання. Мапа включає опис ключових функцій продукту, таких як підсумовування чатів, персоналізація підсумків, інтеграція з популярними платформами та розширені можливості аналізу.

Також були визначені унікальні аспекти продукту, серед яких якісна підтримка української мови, впровадження універсального API та багатоплатформність. Окрім цього, мапа демонструє сценарії взаємодії кінцевого користувача з ботом через інтерфейс мобільної версії, що забезпечує зручність використання.

Також було розроблено Lean Canvas (рис. 8), який деталізує бізнес-модель, включаючи проблеми, ключові метрики, джерела доходів і ціннісну пропозицію для українських користувачів. Ця мапа є основою для подальшого тестування продукту, допомагаючи забезпечити його зручність і ефективність для широкої аудиторії.



Рис. 6. Мапа продукту "Briefly"

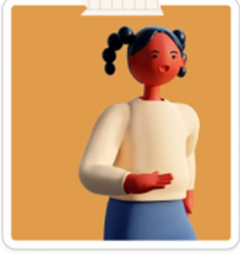


Майкл
User

- Вік: 24
- Роль: студент 2-го курсу ФКНК
- Київ

Біо

Майкл часто потопає в повідомленнях навчальних чатів: важливу інформацію з лекцій та проєктів важко знайти серед жартів і флуду. Briefly дозволяє Майклу швидко отримувати суть розмови, виділяючи тільки ключові моменти, що зекономить його час і допоможе не пропустити важливе.



Софія
User

- Вік: 17
- Роль: Абітурієнт 2025
- Львів

Біо

Софія готується до вступу і перебуває в численних чатах, де обговорюють підготовку до НМТ. Важлива інформація губиться серед непотрібних обговорень. Завдяки Briefly Софія може зосередитися лише на ключових порадах і новинах, економлячи час та зусилля.

Рис. 7. Приклади користувачів системи "Briefly"

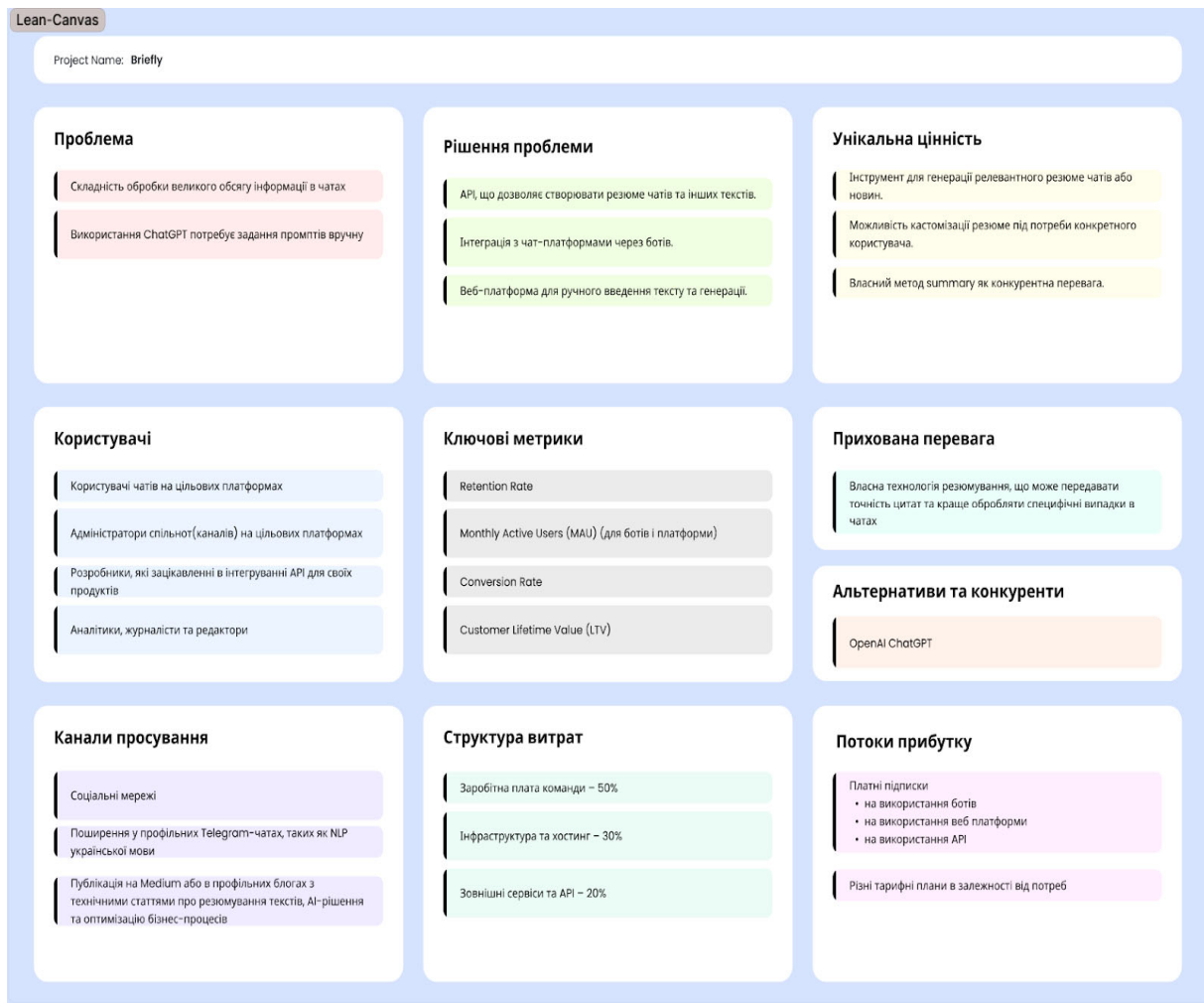


Рис. 8. Lean Canvas проекту "Briefly"

Система "Briefly" вирішує проблему надмірного обсягу інформації, яка часто виникає у великих чатах та каналах месенджерів. Основними проблемами, що постають перед користувачами, є неможливість охопити всі обговорення, потреба у збереженні часу та уникнення інформаційного перевантаження. За допомогою автоматизованих підсумків, які генерує бот, користувачі можуть швидко отримувати основні тези та висновки дискусій.

Сегментами клієнтів є активні користувачі месенджерів, які беруть участь у великих чатах або каналах із великою кількістю інформації, адміністратори чатів та каналів, які потребують автоматизованого дайджесту для своїх спільнот, а також користувачі, які мають обмежений час і хочуть залишатися в курсі основних подій.

Унікальна ціннісна пропозиція "Briefly" полягає у можливості швидкого створення підсумків обговорень, персоналізації стилю та обсягу інформації відповідно до потреб користувачів, інтеграції з популярними платформами (WhatsApp, Discord, Viber, Google Chat) та підтримці української й англійської мов.

Рішенням, яке пропонує "Briefly", є автоматичне генерування підсумків обговорень у чатах та каналах на основі заданих параметрів (період, стиль, обсяг інформації). Також бот надає функції створення дайджестів новин та обговорень із каналів, інтеграцію з різними платформами та підтримку багатомовності.

Канали, які будуть використовуватися для поширення продукту (рис. 9), включають соціальні мережі, платформи обміну месенджерами, співпрацю з лідерами думок та створення власного контенту каналах, пов'язаних з автоматизацією та месенджерами.

1	Канал	Опис
2	Соціальні мережі	Telegram: Офіційний канал Briefly для новин, інструкцій і комунікації. Facebook: Таргетована реклама для адміністраторів груп і активних користувачів. Twitter: Публікація оновлень про функції бота, взаємодія з користувачами. Instagram: Відео-демонстрації функціоналу та інтерактивний контент. LinkedIn: Охоплення бізнес-спільноти, адміністраторів корпоративних чатів.
3	Месенджери	WhatsApp: Демонстрація бота у тестових групах для зворотного зв'язку. Discord: Тестування функцій у великих серверах.
4	Тематичні платформи та форуми	Product Hunt: Розміщення Briefly як інноваційного продукту. Reddit: Інформація у спільнотах r/TelegramBots і r/Productivity.
5	Інфлюенс-маркетинг	Співпраця з адміністраторами Telegram-каналів і мікроінфлюенсерами у сфері технологій.
6	Контент-маркетинг	Публікація кейсів, гайдів і порівнянь із конкурентами. Розміщення статей на Ain.ua, MC.today.
7	Реклама	Google Ads: Контекстна реклама за запитами "бот для підсумків чатів", "автоматизація месенджерів". YouTube: Відеореклама з демонстрацією функцій.
8	Спільноти користувачів	Розміщення інформації про бота у групах, пов'язаних із автоматизацією та продуктивністю.
9	Електронна пошта	Розсилка рекомендацій, прикладів використання та оновлень функціоналу бота.
10	Офлайн-канали	Участь у конференціях, таких як IT Arena, демонстрація продукту на заходах.

Рис. 9. Канали використання для "Briefly"

Структура витрат включає розробку та підтримку бота, витрати на інфраструктуру, маркетингові кампанії, співпрацю з інфлюенсерами, створення контенту, а також забезпечення підтримки клієнтів.

Ключовими метриками для оцінювання ефективності є кількість активних користувачів на день (DAU), кількість запитів на створення підсумків та рівень конверсії у преміум-підписку.

Сегментами клієнтів є активні користувачі месенджерів, які беруть участь у великих чатах або каналах із великою кількістю інформації. Це, зокрема, користувачі студентських каналів, які шукають актуальні новини про університетське життя, розклад, оголошення та навчальні матеріали. Також це користувачі каналів новин кожного міста країни, які хочуть залишатися в курсі локальних подій. Особливий інтерес до продукту можуть мати підписники каналів про стипендії та можливості стажування, які потребують оперативної інформації щодо дедлайнів та умов участі. Крім того, до сегмента належать користувачі освітніх каналів, таких як групи підготовки до НМТ, які шукають структуровану інформацію та доступ до матеріалів. Адміністратори таких каналів та чатів також є важливою аудиторією, оскільки їм потрібні інструменти для автоматизованого дайджесту основних обговорень, що дозволяє ефективно взаємодіяти зі спільнотою. Окрім того, продукт буде корисним для користувачів, які мають обмежений час, але прагнуть залишатися в курсі основних подій та дискусій.

Унікальність "Briefly" полягає в його адаптивності та здатності відповідати викликам сучасного ринку. Ситуація із заборонаю Telegram у багатьох українських університетах стала катализатором для розвитку бота, спрямованого на підтримку альтернативних платформ, таких як WhatsApp, Discord, Viber та Google Chat. Це дозволило зробити продукт універсальним і доступним для ширшої аудиторії.

Особливий акцент на якісній підтримці української мови забезпечує точність і природність підсумків, що робить "Briefly" Bot унікальним рішенням для українських користувачів. Бот не лише вирішує проблему інформаційного перевантаження, але й сприяє популяризації альтернативних месенджерів серед локальної аудиторії, демонструючи, як інноваційні рішення можуть змінювати спосіб комунікації.

Потенційний функціонал в майбутньому:

- інтеграція з іншими месенджерами: додавання підтримки нових платформ;
- аналіз тону повідомлень: визначення настрою чату (оптимістичний, нейтральний, негативний);
- summary статей, контенту, відео на YouTube;
- підтримка спеціальних інтерфейсів для адміністраторів: можливість задавати налаштування для каналів та груп.

Бізнес-модель

Джерелами доходів є модель freemium (рисунок 10), яка передбачає безкоштовний доступ до основного функціоналу, а також преміум-підписку з розширеними можливостями, такими як персоналізовані стилі підсумків, частіші оновлення та інтеграція зі спеціальними платформами.

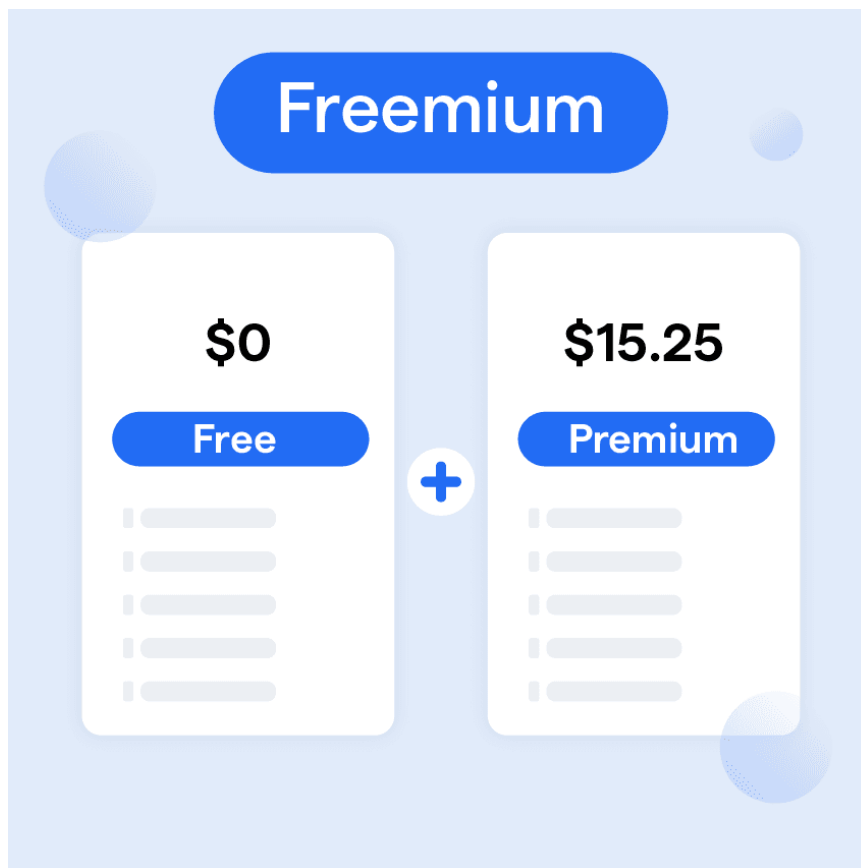


Рис. 10. Модель freemium

Сильні і слабкі сторони розроблюваного продукту

Аналіз сильних і слабких сторін "Briefly" дозволяє оцінити його конкурентні переваги та визначити напрями вдосконалення.

Сильні сторони.

- відсутність на ринку багатьох аналогічних рішень, зокрема для українського сегменту;
- використання різних методів підсумовування (extractive і generative) для покращення якості summary.
- проєкт передбачає використання API, що дозволяє легко інтегрувати рішення у різні системи;
- можливість розширення на інші платформи (YouTube, веб-застосунки), що збільшує потенційну аудиторію.
- використання generative моделей (GPT, Llama) та ефективних extractive методів (BM25, TF-IDF);
- використання передових технологій та моделей обробки тексту підвищує якість продукту.
- гнучкі налаштування (довжина summary, стиль підсумку), що дає користувачам можливість адаптувати продукт під власні потреби;
- підтримка різних мов, зокрема української, що є важливою для місцевого ринку.

Слабкі сторони:

- використання generative моделей, таких як ChatGPT або Llama, може бути дорогим у випадку великої кількості запитів;
- можливі обмеження щодо безкоштовного використання сторонніх API, що підвищує вартість розробки і обслуговування.
- продукт сильно залежить від якості текстових даних (чатів або контенту), особливо в неформальних чатах, де є багато помилок, суржику або специфічної лексики;
- складнощі з обробкою української мови через обмеженість мовних моделей та інструментів для цієї мови.
- команда може мати недостатній досвід у розробці таких проєктів, зокрема в обробці текстів (NLP), що може призвести до затримок у розробці;
- можливий брак ресурсів для масштабування інфраструктури проєкту (сервери, обчислювальні потужності).
- велика кількість неструктурованих даних у чатах (смайли, матюки, помилки) може ускладнити процес аналізу і зменшити якість підсумків.

Можливості:

- збільшення кількості текстової інформації, яку важко обробляти вручну, створює попит на автоматизовані рішення для створення summary;
- розвиток ринку інструментів обробки тексту, особливо в контексті бізнес-комунікацій і аналізу чатів.
- можливість додати нові фічі, такі як summary відео з YouTube, автоматичне створення тайм-кодів до відео, інтеграція з іншими платформами (Slack, Discord);
- можливість інтеграції з CRM-системами для автоматизації бізнес-процесів.
- підвищення інтересу до українського контенту та українськомовних застосунків відкриває нові можливості для залучення користувачів;
- відсутність якісних рішень для обробки тексту українською мовою створює нішу, яку можна зайняти.
- можливість співпраці з популярними платформами, такими як YouTube або Telegram, для просування продукту;
- інтеграція з сервісами аналітики та моніторингу (наприклад, Google Analytics) для покращення функціоналу.

Ризики:

- наявність подібних інструментів (телеграм-ботів для summary, чат-ботів з функцією summary), що можуть запропонувати аналогічний функціонал;
- конкуренція зі сторони великих компаній, які мають більше ресурсів та кращу інфраструктуру (Google, Microsoft).
- загроза блокування Telegram у деяких країнах через політичні чи безпекові причини може вплинути на доступність продукту для користувачів;
- залежність від однієї платформи (Telegram) може обмежити ріст проєкту.
- обробка текстів з приватних чатів може викликати питання щодо конфіденційності та захисту даних;
- ризик порушення політик конфіденційності Telegram та інших платформ, що може призвести до штрафів або блокування доступу до API.
- використання сторонніх API для generative моделей (ChatGPT, Llama) може призвести до непередбачених затримок або перебоїв у роботі сервісу;
- зміна політик або цінової політики сторонніх сервісів може зробити продукт менш рентабельним.

Огляд використаних технологій

У таблиці 5 вказані технології, які були використані для розробки продукту "Briefly".

Таблиця 5. Технології розробки продукту "Briefly"

Технологія	Причини вибору
Мова програмування: Python	<ul style="list-style-type: none"> - висока популярність і велика кількість бібліотек, які спрощують реалізацію NLP-завдань; - широка спільнота підтримки; - легкість в освоєнні, що дозволяє швидко адаптувати нових членів команди.
Фреймворк FastAPI[8]	<ul style="list-style-type: none"> - підтримка асинхронного програмування, що забезпечує високу продуктивність; - зручність у створенні документації API завдяки інтеграції OpenAPI; - простота у розгортанні та підтримці API-сервісів.
Бібліотека для обробки тексту: SpaCy[9]	<ul style="list-style-type: none"> - висока продуктивність і підтримка сучасних NLP-моделей; - можливість інтеграції з локальними моделями для обробки тексту; - простота у роботі з великими корпусами тексту.
Хмарна платформа: AWS	<ul style="list-style-type: none"> - надійна платформа для розгортання API та баз даних; - широкий вибір послуг для масштабування проєкту.
База даних: PostgreSQL	<ul style="list-style-type: none"> - надійність і потужна підтримка реляційної моделі даних; - простота роботи з великими обсягами текстових даних; - підтримка складних запитів і розширень, таких як Full Text Search, що важливо для роботи з текстами.

2. ОПИС ПРОГРАМНОЇ СИСТЕМИ

Призначення і цілі створення системи

Призначення системи. Система "Briefly" створюється для автоматизації процесу підсумовування текстового та медіаконтенту з різних джерел, таких як чати, документи й відео. Її основне призначення полягає у спрощенні роботи користувачів із великими обсягами інформації та наданні стислих, точних і персоналізованих резюме обговорень, доступних через вебзастосунок і ботів (WhatsApp, Discord, Slack тощо), що дозволяє економити час і підвищувати ефективність роботи.

Цілі створення системи. Система "Briefly" створюється з метою: забезпечення збору, обробки та аналізу інформації з чат-платформ, документів (PDF, DOCX, TXT) і YouTube-відео, що дає змогу користувачам швидко отримувати підсумкову інформацію; автоматизації підсумовування контенту за допомогою сучасних методів NLP, зокрема генеративного та екстрактивного підсумовування з підтримкою кількох мов; можливості персоналізації результатів підсумовування за параметрами, які задає користувач, такими як стиль, обсяг і період; оптимізації витрат на обробку шляхом тонкого налаштування prompt-ланцюжків, моделей та алгоритмів; створення мікросервісного API для інтеграції функцій у зовнішні продукти й автоматизації бізнес-процесів; запровадження гнучкої монетизації у вигляді підписок з лімітами використання.

Також система призначена для: використання як інструменту аналітики з виявлення ключових тем, настроїв і трендів у чатах та документах; формування корпоративних і освітніх дайджестів важливих повідомлень та даних.

Вимоги до системи

Вимоги до системи в цілому. Система "Briefly" має забезпечувати автоматизоване підсумовування текстового й медіаконтенту (чати, документи, відео) через веб-інтерфейс, чат-боти та публічний API, залишаючись зручною для кінцевих користувачів і сумісною з бізнес-процесами.

- надійність і стійкість до великих навантажень;
- реалізація сервісів на різних платформах – вебзастосунок, чат-боти та REST API;
- підтримка багатомовності (українська, англійська);
- інтеграція з API сторонніх платформ;
- можливість персоналізації підсумків за параметрами (обсяг, стиль, період);
- висока якість обробки текстів;
- система підписок і лімітів запитів із вбудованим білінгом.

Вимоги до структури та функціонування системи. Система "Briefly" має будуватися за мікросервісною архітектурою, що спрощує масштабування, оновлення та підтримку. Основні вимоги:

- підтримка двох режимів підсумовування – екстрактивного та генеративного, з можливістю легко додавати нові моделі;
- безпека даних, шифрування в транзиті й у статичному вигляді, рольова авторизація (OAuth 2.0 / JWT);
- інтуїтивний інтерфейс для взаємодії з користувачами та адміністраторами;
- наявність зручної системи логування та моніторингу запитів;
- масштабованість для обробки великої кількості запитів у реальному часі;

– кешування результатів для зниження вартості та часу відповіді при повторних запитах.

Вимоги до функцій, які виконуються системою. В таблиці 6 подано перелік функцій та задач, що підлягають автоматизації у підсистемі "Briefly" Web-App.

Таблиця 6. Перелік функцій, задач що підлягають автоматизації

Функція	Задача
Авторизація/реєстрація	Вхід через e-mail або OAuth.
	Створення облікового запису.
Завантаження контенту	Додавання документів (PDF, DOCX, TXT).
	Введення посилань на YouTube відео.
Використання підсумків	Отримання автоматично згенерованих підсумків.
	Перегляд історії створених підсумків.
	Вибір мови, стилю, обсягу, періоду резюме.
Керування підпискою	Перегляд тарифу, оплата, зміна плану.

В таблиці 7 подано перелік функцій та задач, що підлягають автоматизації у підсистемах чат ботів Briefly.

Таблиця 7. Перелік функцій, задач що підлягають автоматизації

Функція (Команда)	Задача
/start	Реєстрація користувача або підключення чату.
/summarize	Підсумок останніх N повідомлень чату.
/yt summarize <link>	Генерація резюме відео за URL.
/pdf summarize	Підсумок надісланого PDF-файлу.
/settings	Конфігурація періоду, мови, формату дайджесту.
/subscription	Керування підпискою.

Системні вимоги

Браузери: сайт повинен коректно відображатися у сучасних версіях браузерів: Google Chrome (версія 80 і вище), Mozilla Firefox (версія 75 і вище), Microsoft Edge (версія 85 і вище), Safari (версія 13 і вище). На некоректні дії користувача (введення невірних даних, пропуск обов'язкових полів) система повинна генерувати інформативні повідомлення про помилки українською мовою з дотриманням загального дизайну. Операційна система: система повинна підтримувати розгортання на: Linux (Ubuntu 20.04 і вище, CentOS 8), Windows Server 2019, MacOS 10.15 і вище (для локального тестування).

Вимоги до вебзастосунку

У цьому розділі описано основні закони та принципи UX-дизайну, які були враховані при розробці вебзастосунку. Кожен з цих принципів був застосований на певних етапах взаємодії користувача з продуктом для забезпечення інтуїтивно зрозумілого та ефективного досвіду.

Перш за все, важливим аспектом є чітка візуальна ієрархія, що дозволяє користувачу швидко орієнтуватися на головній сторінці (рис. 11). Тут розташовано короткий опис сервісу, який демонструє основні можливості продукту, а також кнопку "Get started free", що сприяє залученню користувача до активної взаємодії.

На сторінці YTVideo AI (Рисунок 12) користувач може ввести посилання на відео з YouTube, після чого система обробляє запит і генерує короткий підсумок відео. Додатково з'являються теми з часовими мітками, що полегшує навігацію по відео. Це

дозволяє користувачам швидко отримати важливу інформацію без необхідності переглядати все відео. Тут застосовуються принципи ефективного використання простору та зручної навігації.

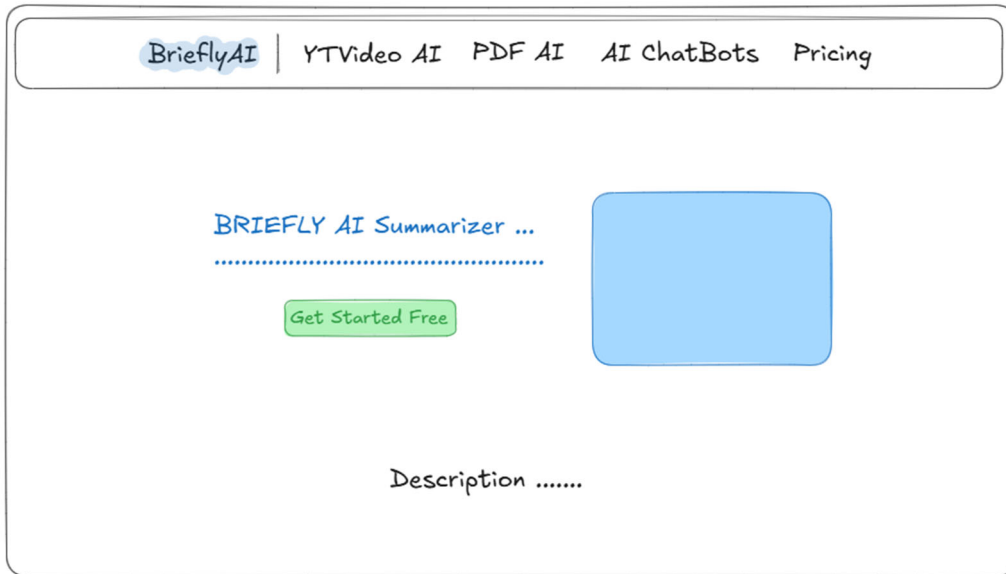


Рис. 11. Макет дизайну головної сторінки

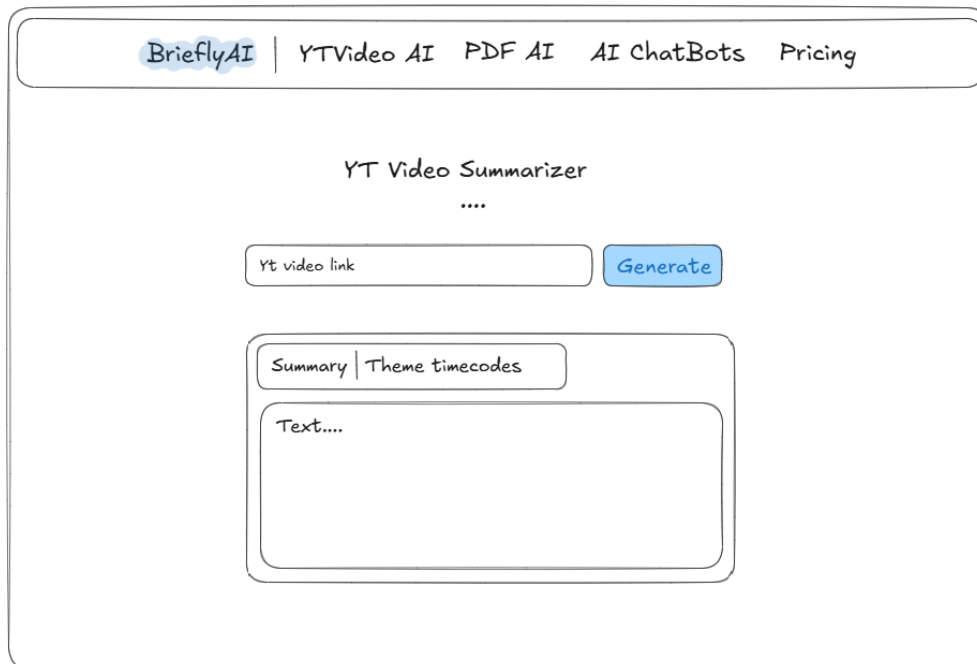


Рис. 12. Макет дизайну сторінки YTVideo AI

Сторінка PDF AI (рис. 13) забезпечує можливість генерації та перегляду PDF-документів за допомогою штучного інтелекту. Для цієї функції було важливо реалізувати принципи мінімалізму, аби не перевантажувати користувача зайвими елементами на сторінці.

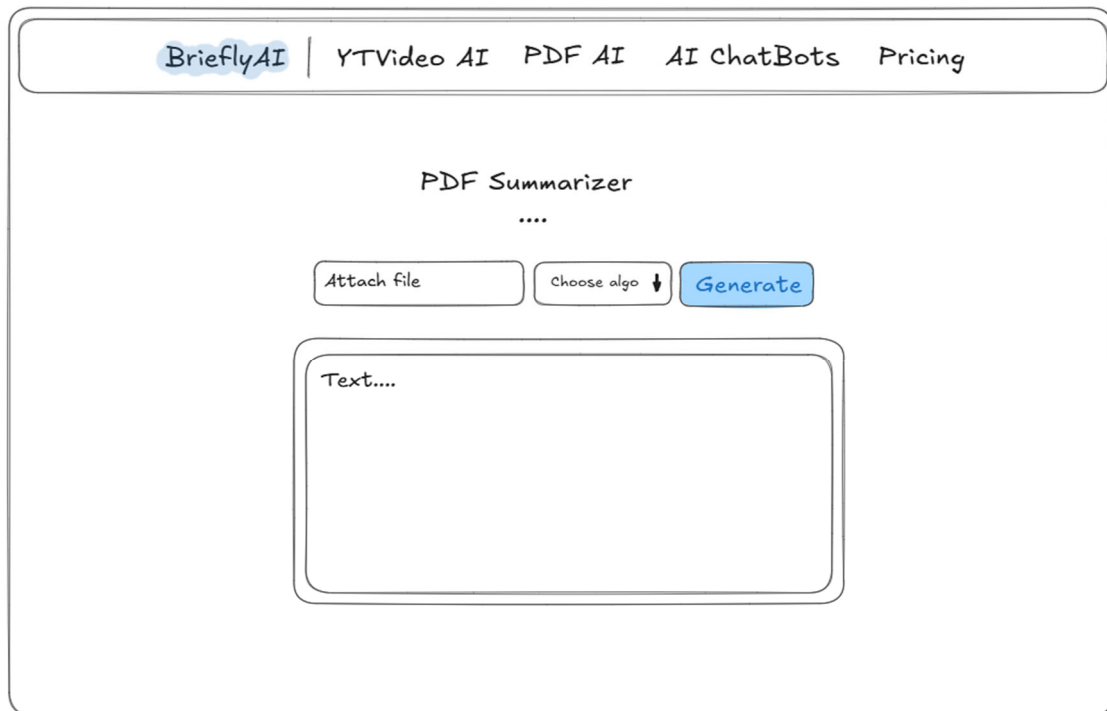


Рис. 13. Макет дизайну сторінки PDF AI Page

Сторінка AI Chat Bots містить інструкції щодо підключення чат-ботів до чатів, а також посилання на чат-ботів на різних платформах. Тут використовуються принципи чіткої структури та доступності інформації, що спрощує процес налаштування для користувача.

Загалом, застосування цих принципів дозволяє створити зручний та інтуїтивно зрозумілий інтерфейс, що позитивно впливає на поведінку користувача і сприяє досягненню цілей при взаємодії з продуктом.

UI/UX-дизайн продукту

У рамках розробки продукту значна увага була приділена UI/UX-дизайну, адже саме він формує перше враження користувача про сервіс та впливає на зручність його використання. Для створення прототипу та візуального оформлення (рис. 14) було обрано інструмент Figma[10], що дозволив ефективно спроектувати макети, протестувати логіку взаємодії з інтерфейсом і забезпечити візуальну цілісність усіх сторінок.

Основні принципи дизайну:

- user-friendly – всі елементи інтерфейсу максимально інтуїтивні, навіть для користувача без технічної підготовки;
- мінімалізм – прості форми, чиста композиція та обмежена палітра кольорів забезпечують комфортне сприйняття;
- візуальна ієрархія – заголовки, кнопки та ключові блоки виділено через контраст, розмір та розміщення;
- адаптивність – макети враховують відображення на різних типах пристроїв;
- єдність стилю – всі сторінки побудовані в одному візуальному стилі, що підтримує впізнаваність бренду.

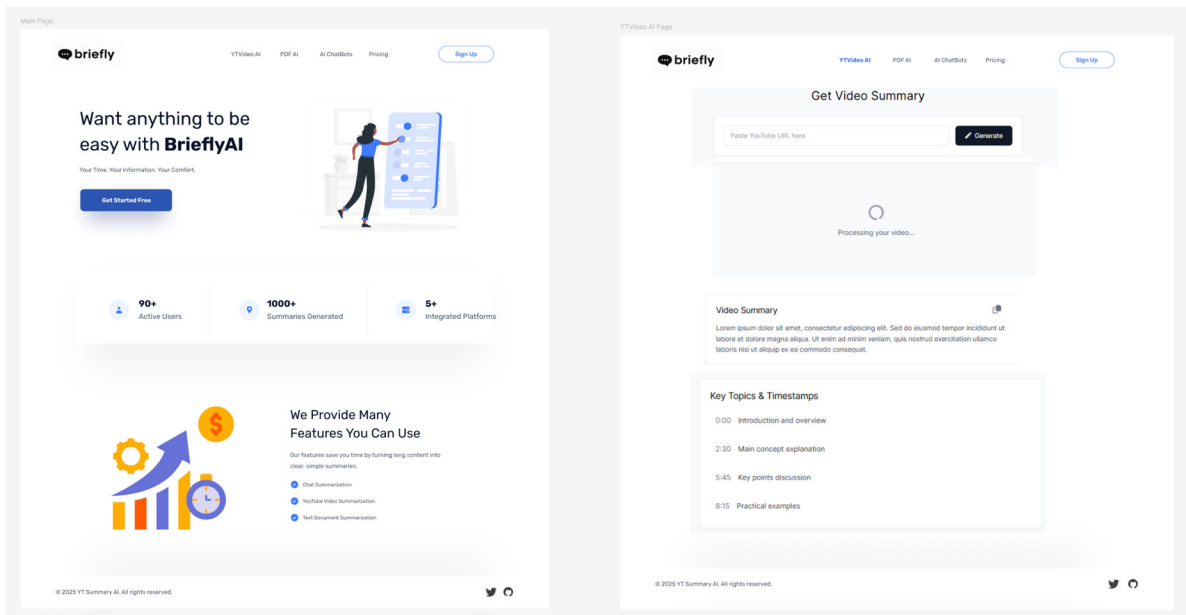


Рис. 14. Макети сторінок вебзастосунку у Figma

Створено п'ять основних сторінок:

- Main Page: дизайн орієнтований на перше знайомство користувача з платформою. Основні елементи: короткий опис, ілюстрація, іконки, СТА-кнопка "Почати зараз". Використано асиметричне розташування елементів для динаміки та візуального інтересу;
- YTVideo AI Page: у центрі – поле введення посилання. Далі йде виведення результатів (summary і теми з таймкодами). Усі блоки мають чітку сітку, що забезпечує легкість читання. Порожні стани оформлені з поясненням, що очікується від користувача;
- PDF AI Page: оформлення витримане у строгому стилі, з акцентом на зручність завантаження файлу. Застосовано прогресивне розкриття інформації: спочатку – завантаження, після обробки – вивід результату. Це дозволяє не перевантажувати інтерфейс зайвими елементами;
- AI Chat Bots Page: візуальна структура побудована навколо інструкції. Використано нумеровані блоки, піктограми платформ та лінки. Такий підхід забезпечує покрокове сприйняття інформації;
- Pricing Page: таблиця тарифів оформлена з урахуванням UX-принципів: чітке порівняння, кнопки "Обрати", акценти на найпопулярніший тариф. Колірні маркери допомагають візуально відрізнити блоки.

Візуальні елементи, кольорова палітра та шрифт описані в Таблиці 8.

Таблиця 8. Візуальні елементи

Основні кольори	Білий, сірий, темно-синій, акценти – помаранчевий та синій для кнопок та іконок
Шрифти	Сучасний шрифт для збереження чистоти і читабельності
Іконографіка	Використовувалась для підсилення змісту, не дублюючи текст
Ілюстрації	У мінімалістичному стилі, доповнюють зміст і додають емоційності.

Реалізація фронтенду та бекенду

Розробка вебзастосунку здійснювалась із чітким розділенням клієнтської (фронтенд) та серверної (бекенд) частин. Такий підхід дозволив забезпечити гнучкість у розробці, простоту масштабування та зручність обслуговування системи. Загальна структура описана в Таблиці 9.

Таблиця 9. Структура вебзастосунку

Компонент	Технології та інструменти
Фронтенд	<ul style="list-style-type: none"> - мова: JavaScript; - фреймворк: React[11]; - HTTP-запити: Fetch API.
Бекенд	<ul style="list-style-type: none"> - мова: Python; - фреймворк: FastAPI; - база даних: PostgreSQL; - інтеграції: OpenAI API, YouTube Transcript API, SpaCy.

Фронтенд відповідає за зовнішній вигляд і взаємодію з користувачем. Уся структура інтерфейсу була реалізована відповідно до дизайн-макетів, створених у Figma. Було створено п'ять основних сторінок: головна сторінка (Main), сторінка обробки YouTube-відео (YTVideo AI), сторінка аналізу PDF-документів (PDF AI), сторінка з інструкціями щодо підключення чат-ботів (AI ChatBots) та сторінка з тарифами (Pricing). Також реалізована авторизація користувачів, а навігація між сторінками відбувається через маршрутизацію. Важливою частиною роботи стало налаштування валідації форм введення, щоб уникати помилок при надсиланні запитів. Фронтенд надсилає дані до серверу через REST API та обробляє відповіді у зручному для користувача форматі.

Бекенд забезпечує логіку обробки всіх запитів, що надходять із фронтенду. Він відповідає за авторизацію та реєстрацію користувачів, а також за інтеграцію зі сторонніми сервісами, такими як OpenAI та YouTube API. Саме серверна частина виконує основні обчислення: створення текстових summary, виділення ключових тем з відео, обробку PDF-файлів тощо. Крім того, за потреби, реалізоване збереження історії запитів користувача.

Взаємодія між клієнтською та серверною частинами побудована на обміні даними у форматі JSON. Фронтенд надсилає запит, сервер його обробляє, і надсилає назад відповідь. Була також реалізована базова система обробки помилок, щоб користувач отримував повідомлення у випадку технічних збоїв чи некоректного введення.

3. ІМПЛЕМЕНТАЦІЯ СИСТЕМИ

Загальна структура системи

Система "Briefly" побудована на основі мікросервісної архітектури, яка забезпечує модульність, масштабованість та зручність підтримки. Основні компоненти системи:

- API (FastAPI): Відповідає за обробку запитів користувачів та передачу їх до служб підсумовування (Summary Services);
- summary services: Сервіси, що реалізують генеративне та екстрактивне підсумовування текстів;
- база даних (PostgreSQL): Зберігає конфігурації користувачів, логи та історію запитів;
- інтеграційні модулі: Підключення до платформ, таких як Slack, Discord, та інших.

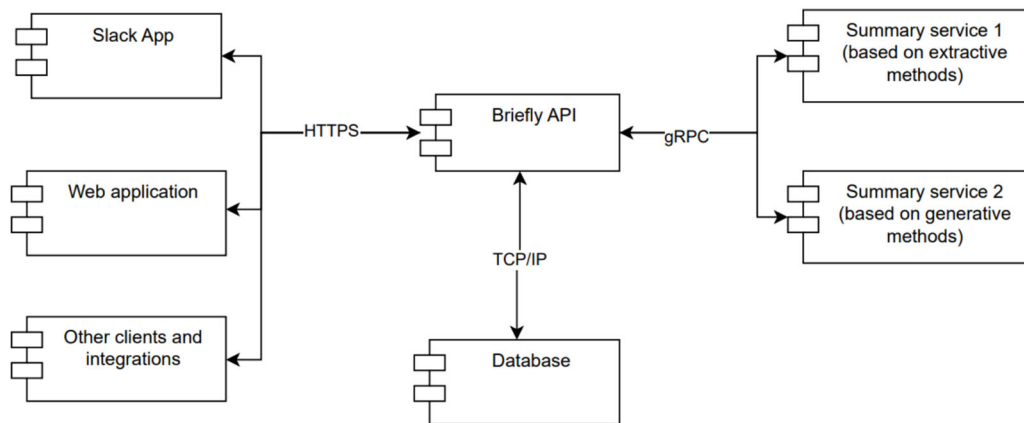


Рис. 15. Діаграма компонентів для архітектури системи

Обґрунтування вибору архітектури

Розділення на мікросервіси дозволяє масштабувати компоненти залежно від навантаження. Наприклад, якщо кількість NLP-запитів зростає, можна збільшити кількість екземплярів NLP-модуля. Асинхронність FastAPI забезпечує швидку обробку запитів. Обчислювально складні задачі (NLP) виконуються в окремих сервісах.

Оскільки сервіси ізольовані, відмова одного компонента (наприклад, бази даних) не призводить до повного збою системи. Логування всіх запитів і помилок дозволяє швидко діагностувати проблеми. Така система є гнучкою, забезпечує легку інтеграцію нових функцій завдяки модульній структурі.

API

API є основним інтерфейсом для взаємодії з системою. Воно дозволяє користувачам надсилати запити на підсумовування текстів через інтегровані платформи або напряму.

API зараз містить п'ять основних POST-ендпоінтів. /summarize приймає довільний текст і повертає універсальний дайджест. Для чатів є два маршрути: /summarize/chat/extractive, який добирає ключові речення з останніх повідомлень, і /summarize/chat/generative, що формує зв'язний генеративний огляд тієї самої вибірки. Відео YouTube обробляються через /api/v1/yt/summary: достатньо передати URL ролика, щоб отримати стислий переказ. Нарешті, /api/v1/summarize/pdf приймає PDF-документ (або DOCX/TXT) у multipart-запиті й повертає його підсумок разом із переліком ключових понять.

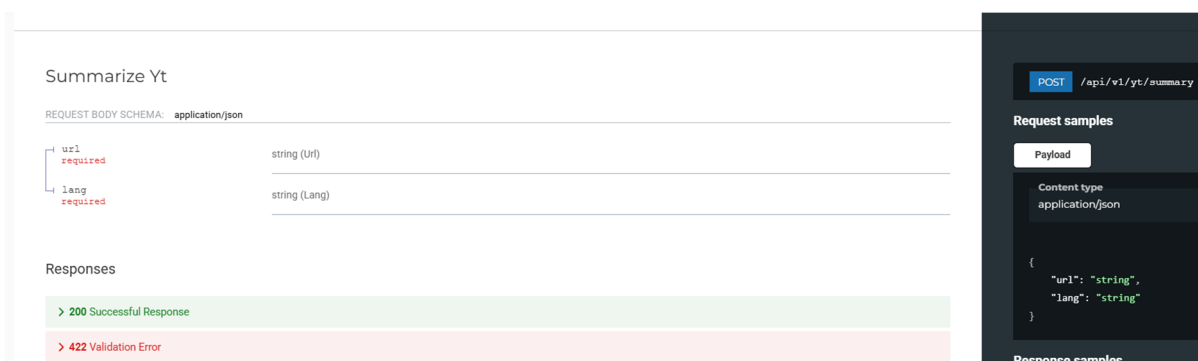


Рис. 16. Частина API документації з основним методом

Екстрактивний Summary сервіс

Даний сервіс реалізує одразу два механізми - резюмування структурованих текстів (на прикладі статей різної тематики) та неструктурованих (діалоги з соціальних мереж та месенджерів).

Загалом алгоритми екстрактивного резюмування відносяться до частотних алгоритмів. Оцінка "важливості" кожного речення напряму залежить від кількості "популярних" слів, з яких воно складається. Реалізації обох підходів подібні за виключенням однієї особливості.

Алгоритм резюмування структурованих текстів описаний у статті складається з чотирьох етапів – лематизація, формування матриці подібності речень та підсумкове вилучення речень. Алгоритм резюмування неструктурованих текстів включає додатковий етап - кластеризація речень. Даний етап має місце між лематизацією та формуванням матриці подібності. Його мета – об'єднати речення зі схожим змістом в кластери і таким чином структурувати текст.

Лематизація сленгу, просторіччя та обценної лексики. Було проаналізовано якість лематизації неформальної лексики (сленгу, просторіччя, обценної лексики) на прикладі україномовних текстів із датасетів повідомлень у Twitter. Для цього використовувалася наявна NLP-модель на базі spaCy, з якою було виявлено типову проблему: слова неформального вжитку часто залишаються без змін, адже не розпізнаються моделлю як відомі словоформи.

Для автоматизованого виявлення таких слів була реалізована функція, що ідентифікує лексеми, які не змінилися після лематизації, та відсіює незмінні частини мови (інфінітиви, називний відмінок, незмінні слова). Далі було розроблено порівняльний підхід з використанням двох альтернативних інструментів: бібліотек stanza і rymorphuz. За результатами тестування rymorphuz було обрано як основний інструмент дообробки, оскільки він дозволяє лематизувати окремі слова без контексту, потребує менше ресурсів і забезпечує порівняну точність.

Фінальна реалізація включає функцію, яка передає до аналізатора rymorphuz слова, не лематизовані spaCy. Якщо слово відсутнє в словнику rymorphuz, аналізатор генерує список можливих лем із вірогідностями. Вибирається найімовірніша лема, після чого слово додається в основний bag of words. Для підвищення точності обробки враховується морфологічна інформація, отримана від spaCy – наприклад, рід або частина мови – що дозволяє відсіювати невідповідні варіанти лем і покращує якість лематизації дієслів, іменників і прикметників.

Обробка скорочень та абrevіатур. Для виявлення скорочень і абrevіатур у текстах було використано ті самі датасети, що й у попередньому завданні. Спеціальна функція знаходила слова з крапкою в кінці та токени, позначені моделлю як абrevіатури (Abbr=Yes).

Під час аналізу з'ясувалося, що точність виявлення абrevіатур невисока: модель часто помилково позначала як абrevіатури вигуки, частки, текстові емоджі, іншомовні слова тощо. Для фільтрації хибнопозитивних результатів були застосовані правила, які враховували регістр, пунктуацію, іноземні символи та ін.

Було розроблено рішення для розшифрування скорочень: слова, розпізнані як абrevіатури, оброблялися окремим пайплайном. Якщо таке слово є в словнику скорочень, воно замінювалося на повну лематизовану форму, інакше залишалося без змін. Було створено два словники:

– скорочення мір та кількостей (год, км, Мб), які розшифровуються лише за наявності числівника перед ними;

– звичайні скорочення (фб → фейсбук, зп → зарплата), що не позначають величини. Особливий випадок – "(с)" для позначення цитати, який обробляється окремо.

Цей підхід дозволив розрізняти омонімічні скорочення (наприклад, мб → мегабайти або мб → "може бути") та зменшити кількість помилок. Надалі словники планується розширювати.

Кластеризація. Задача резюмування статей відрізняється від резюмування діалогів. Даний етап необхідний для об'єднання схожих за змістом повідомлень в кластери. Для кластеризації було використано алгоритм TextTiling, який використовує модифікацію TF-IDF задля обчислення подібності повідомлень. В якості межі схожості було встановлено 0.4; тобто якщо два повідомлення мають схожість 40% (в контексті TF-IDF), то ці повідомлення будуть об'єднані в єдиний кластер. Далі ця ж процедура рекурсивно об'єднує всі новостворені кластери. В кінці отримуємо список кластерів, що містять повідомлення, які можна віднести до якогось одного предмета обговорення.

Формування матриці подібності для речень\кластерів. Суть даного етапу полягає в тому, щоб за допомогою функції подібності оцінити "схожість" кожної пари кластерів і відповідні результати зберегти у вигляді матриці. В запропонованому рішенні використовується комплексний підхід – комбінування функцій ранжирування BM25+ та TextRank. Згідно яким будуються дві матриці подібності, потім ті матриці нормалізуються, а їх сума і є кінцевою матрицею подібності.

Оцінка речень для неструктурованого тексту. Даний етап є актуальним тільки для резюмування неструктурованих текстів, де наявна кластеризація речень. В даному підході важливо вибрати не найважливіші кластери, а саме найважливіші речення цих кластерів. Для цього, згідно з алгоритмом ClusterRank, вводиться поняття центроїду – набір тих слів в кластері, які несуть найбільшу інформативність. Після чого кожне окреме речення відповідного кластера оцінюється з точки зору "близькості" до центроїду. Таким чином найбільш інформативні речення найбільш популярних кластерів отримують вищі оцінки.

Оцінка речень для структурованого тексту. В даному випадку ми реалізуємо класичну оцінку речень згідно TextRank. Даний алгоритм є похідним від Page Ranking Algorithm, тільки в якості структури для оцінювання використовується зважений граф, де кожна вершина – окреме речення, а вага ребра між вершинами – подібність цього речення до сусіднього за матрицею подібності.

Підсумкове вилучення речень для неструктурованого тексту. Вилучення речень в кінцеве резюме проводиться таким чином, щоб максимально зменшити надмірність. Тому перед додаванням речення в резюме, згідно алгоритму ClusterRank, рахується коефіцієнт надмірності. Якщо значення коефіцієнта більше певного порога, то вважається, що тематика даного речення вже достатньо розкрита в існуючому резюме, тому його ми пропускаємо. Якщо ж речення має достатній рівень унікальності, то воно додається до кінцевого резюме. Ідея такого підходу полягає в тому, щоб уникнути обговорення однієї і тієї ж ідеї по колу, а надавати пріоритет іншим темам, щоб кінцеве зведення максимально широко охоплювало всю бесіду, а не концентрувалось на окремій тематичі.

Підсумкове вилучення речень для структурованого тексту.

В даному підході все простіше – допускається, що структурований текст не має проблеми надмірності, тому в ньому просто сортуються речення по спаданню важливості та вибираються найважливіші.

Інтегровані платформи

Система інтегрується з платформами, такими як Discord, Slack, та іншими, для забезпечення зручності використання та автоматизації запитів. Інтеграція системи з такими платформами включає моменти, які забезпечують безперебійну взаємодію, адаптивність до інших платформ і потенціал для розширення функціональності.

І Slack, і Discord вимагають безпечних методів автентифікації викликів API. Зазвичай для цього використовуються механізми на основі токенів. Наприклад, Slack використовує Bot Token в рамках OAuth, в той час як Discord вимагає Bot Token, керований через портал розробника. Належне поводження з цими токенами забезпечує безпечний зв'язок, контроль доступу та відповідність специфічним для платформи інструкціям.

Інтеграція з Slack вимагає механізму для обробки вхідних подій, таких як повідомлення або команди, через API подій Slack. Це передбачає прослуховування POST-запитів у визначених endpoint, таких як події перевірки URL-адреси під час налаштування або певні команди, такі як `"/summ"`. Розбір цих запитів і запуск відповідних дій бекенда є основним завданням.

Discord працює через події в реальному часі на основі WebSocket. Сюди входять події життєвого циклу, такі як `on_ready()` для підтвердження готовності бота або команди типу `"/summarize"`. Вони виконуються за допомогою структурованих обробників команд, які реагують на вхідні дані користувача і викликають робочі процеси бекенда.

Обидві платформи задають пошук останніх повідомлень, що є критично важливим для таких операцій, як підсумовування. У Slack для отримання повідомлень з певного каналу використовується кінцева точка API `conversations.history`. У Discord функція `channel.history()` надає подібний доступ, дозволяючи ботам витягувати дані для обробки. Такі механізми пошуку стандартизують вхідні дані, що дозволяє проводити подальшу обробку, наприклад, аналіз тону розмови, узагальнення або трансформацію контенту.

Ключовою особливістю інтеграції Slack і Discord є можливість взаємодії із зовнішніми API. Наприклад, узагальнений контент, згенерований зовнішнім сервісом, динамічно витягується і відправляється назад на вихідну платформу. Це передбачає створення структурованих запитів до API, обробку корисного навантаження відповіді та забезпечення належної серіалізації даних.

Після завершення зовнішньої і внутрішньої обробки інтеграція гарантує, що відповідь, наприклад, узагальнений текст або повідомлення про помилку, буде передано відповідному користувачеві. У Slack endpoint API `chat.postMessage` використовується для надсилання відповідей. У Discord метод `ctx.send()` для передачі повідомлень безпосередньо на сервері.

Модульна структура цих інтеграцій дозволяє легко реплікувати їх на додаткові платформи. Стандартизація структур даних (наприклад, для повідомлень або сумування) гарантує, що вхідні та вихідні дані можуть легко адаптуватися до різних платформ. Абстрагуючись від викликів API, специфічних для конкретної платформи, розробники можуть в подальшому інтегрувати нові служби обміну повідомленнями з мінімальними змінами. Використання фреймворків, таких як FastAPI або централізовані API, ще більше підтримує цю адаптивність. Файли YAML забезпечують спосіб керування конфігураціями, специфічними для платформи (наприклад, токенами, кінцевими точками, тригерами команд), що дозволяє швидко впровадити на платформу.

Архітектура інтеграції за своєю суттю є розширюваною, що дозволяє розробникам впроваджувати нові команди або робочі процеси, такі як розширена аналітика або автоматизація взаємодії, розширювати функціональність, підключаючи додаткові API (наприклад, машинне навчання, модерація контенту), сприяти крос-платформній

взаємодії, дозволяючи ботам Slack і Discord обмінюватися даними або виконувати спільні операції.

Інтеграція з Slack і Discord є комплексним підходом для поєднання зручних інтерфейсів з потужними внутрішніми процесами. Зосереджуючись на безпечній автентифікації, ефективній обробці подій та модульній архітектурі, ці інтеграції не лише забезпечують надійну функціональність, але й створюють можливості для розширення на нові сервіси.

Генеративний Summary сервіс

Генеративний сервіс підсумовування, що є ядром системи "Briefly", призначений для перетворення великих масивів уже підготовленого тексту – транскриптів відео, документів або витягів з чатів – на зрозумілий стислий або розгорнутий дайджест. Робота сервісу починається з перевірки вхідних даних і параметрів, які задає користувач: мова, бажаний обсяг, стиль та максимальний час очікування відповіді. Далі формується промпт-ланцюжок: до системних інструкцій додаються вимоги користувача, а довжина контексту контролюється, щоб не перевищити ліміт токенів. Згенерований промпт надсилається у LLM (GPT-4o за замовчуванням). Після одержання відповіді сервіс виконує пост-обробку: виправляє форматування, узгоджує стиль і перевіряє дати, числа та назви.

Архітектурно сервіс реалізовано як безстанна gRPC-служба, що масштабується горизонтально, сам сервіс не спирається на жодні сторонні контексти або сховище стану й обробляє підсумок виключно на основі даних, що надходять у запиті.

Результати роботи системи та тестування

Рис. 17 ілюструє сторінку сервісу YouTube Summary. Після введення посилання система автоматично відтворює відео й під ним формує два блоки результатів. Перший – "Video Summary" – подає стислий переказ змісту ролика, другий – "Key Topics & Timestamps" – групує ключові теми з точними таймкодами, що дає змогу перейти безпосередньо до потрібного фрагмента. Таким чином користувач отримує швидкий огляд основних ідей без потреби переглядати відео повністю.

Рис. 18 демонструє окрему сторінку сервісу Document Summary. Користувач завантажує PDF, DOCX або TXT, після чого система генерує три секції: "Summary" зі зв'язним переказом документа, "Highlights" із маркер-тезами для швидкого ознайомлення та "Keywords" із релевантними ключовими словами. Це забезпечує миттєвий огляд суті тексту та спрощує тематичний пошук і класифікацію матеріалів.

Було протестовано продуктивність сервісу генерації підсумків відео з YouTube – зокрема залежність часу відповіді від тривалості відео. Для цього було зібрано набір відео різної тривалості (від 4 до 131 хвилини), до кожного з яких надсилався запит на генерацію summary через API. Вимірювався фактичний час обробки запиту (ResponseTime), тобто затримка між надсиланням і отриманням підсумку. Результати тестування представлені на рис. 19.

На етапі тестування був створений скрипт, який підтверджує коректну роботу системи відповідно до поставлених завдань. Цей крок відбувався після кожного додавання нового функціоналу для перевірки написаного коду. На рисунку 19 зображені результати тестування для моделей ChatSummaryRequest та Message.

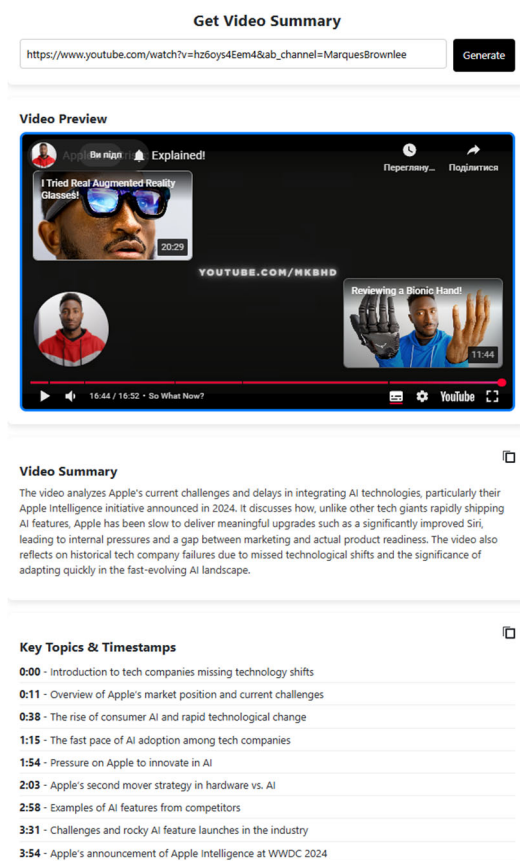


Рис. 17. Результат роботи сервісу YouTube Summary після обробки відео

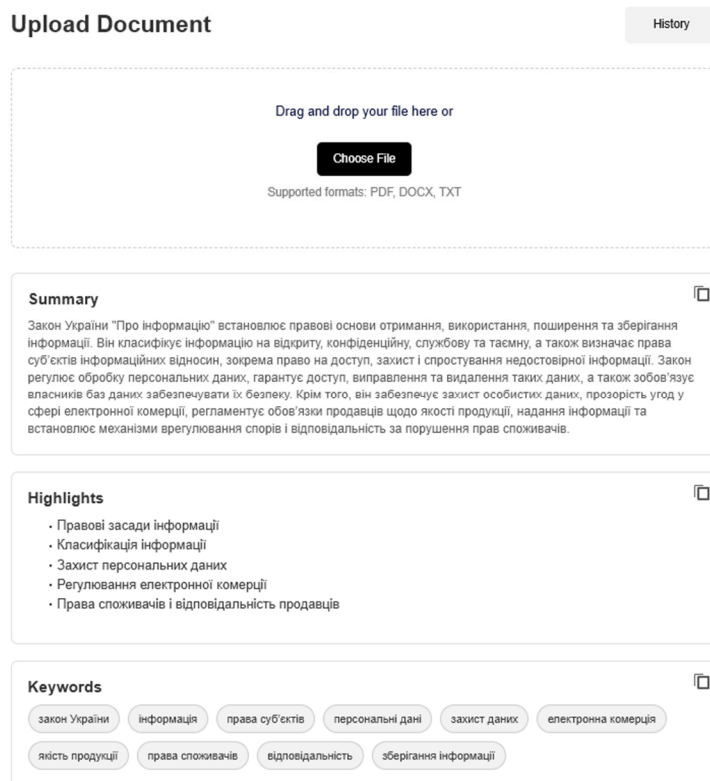


Рис. 18. Результат роботи сервісу Document Summary після завантаження файлу

Таблиця (Посилання Тривалість відео (хв) Час генерації (сек)):		
url	duration	ResponseTime
https://www.youtube.com/watch?v=fBWStmXMnUM	53	7.64
https://www.youtube.com/watch?v=A0syzRowDjM	4	4.44
https://www.youtube.com/watch?v=07zhc63-hFE	72	8.02
https://www.youtube.com/watch?v=noee78gkxgw	38	6.71
https://www.youtube.com/watch?v=DcWqzZ3I2cY	131	12.22
https://www.youtube.com/watch?v=atAEWcqbxCA	23	7.72
https://www.youtube.com/watch?v=9lcrbnack0E	56	12.39
https://www.youtube.com/watch?v=ygAqYC8JOQI	112	12.77
https://www.youtube.com/watch?v=V18vAJ42R80	102	11.49
https://www.youtube.com/watch?v=pgC-b_TnzXI	46	6.47
https://www.youtube.com/watch?v=oVjsB7Zn5CE	30	8.43
https://www.youtube.com/watch?v=gw1clfiGOHk	26	7.33
https://www.youtube.com/watch?v=3bRnNl-ANko	116	12.65
https://www.youtube.com/watch?v=c07f9mbaX8E	13	5.87
https://www.youtube.com/watch?v=1HwukHhZx_c	62	12.06
https://www.youtube.com/watch?v=vC5cHjcg5g	57	8.57
https://www.youtube.com/watch?v=xWowuHMvpzU	5	4.05

Рис. 19. Результати тестування швидкості генерації відповіді

```
tests.py::test_create_chat_summary_request PASSED [ 20%]
tests.py::test_chat_summary_request_serialization PASSED [ 40%]
tests.py::test_invalid_chat_summary_request PASSED [ 60%]
tests.py::test_message_instantiation PASSED [ 80%]
tests.py::test_message_initialization PASSED [100%]
```

Рис 20. Результати тестування двох моделей

Тест `test_create_chat_summary_request` перевіряє, чи може модель `ChatSummaryRequest` бути успішно створена з коректними даними деякого повідомлення. Тест гарантує, що властивості отриманого об'єкта моделі точно відповідають вхідним значенням, таким чином перевіряючи правильність процесу ініціалізації. Готується набір словників коректних повідомлень, кожен з яких містить атрибути `sender`, `timestamp` та `text`. Кожен словник перетворюється в об'єкт `Message` шляхом передачі даних конструктору. Отримані об'єкти використовуються для створення об'єкта класу `ChatSummaryRequest`. Для цього вказуються додаткові параметри - метод сумування (`summ_method`) і тип відповіді (`summ_resp_type`). Виконується перевірка таких властивостей: `summ_method` та `summ_resp_type` відповідають заданим вхідним значенням, `chat` та список об'єктів `Message` містять коректну кількість повідомлень. Властивості (`sender`, `timestamp` і `text`) кожного об'єкта `Message` у списку `chat` відповідають вхідним даним.

Тест `test_chat_summary_request_serialization` гарантує, що модель `ChatSummaryRequest` може бути точно перетворена у представлення словника. Об'єкт `ChatSummaryRequest` інстанціюється з використанням набору об'єктів `Message` та відповідних значень атрибутів `umm_method` та `umm_resp_type`. Викликається метод моделі `dict()` для генерації словникового представлення об'єкта. Виконується перевірка таких властивостей: словник містить точні значення для полів `umm_method` та `umm_resp_type`, поле `chat` у словнику містить усі повідомлення з відповідними

властивостями (sender, timestamp, text), які коректно серіалізовано, кількість повідомлень у полі chat відповідає заданій.

Тест `test_invalid_chat_summary_request` перевіряє, чи модель `ChatSummaryRequest` виконує правила перевірки даних. Зокрема, він гарантує, що спроба інстанціювати модель з недійсними даними, наприклад, зі словником повідомлень, в якому відсутні необхідні атрибути, призведе до відповідної помилки. Готується некоректний словник повідомлень, у якому відсутнє обов'язкове поле (text). Некоректний словник повідомлень використовується для спроби створення об'єкта `Message`, а потім об'єкта `ChatSummaryRequest`. Тест підтверджує, що виникає `ValidationError`, сигналізуючи про те, що дані не відповідають очікуваній структурі.

Тест `test_message_instantiation` перевіряє, що модель `Message` може бути коректно інстанційована, коли їй надано повні та коректні дані. Він гарантує, що атрибути моделі належним чином зберігають вхідні дані. Складається словник, що містить коректні атрибути `Message` (sender, timestamp, text). Словник передається конструктору для створення об'єкту `Message`. У твердженнях перевіряється відповідність атрибутів sender, timestamp і text результуючого об'єкта вхідним значенням.

Тест `test_message_initialization` тестує процес ініціалізації об'єкта класу `Message`, зокрема обробку поля timestamp. Він гарантує, що наданий рядок мітки часу правильно відформатовано та збережено в моделі. Об'єкт класу `Message` інстанціюється з використанням допустимих вхідних даних, включаючи timestamp в очікуваному форматі дата-час. Поле timestamp результуючого об'єкта перевіряється на коректність форматування у вигляді рядка за шаблоном. Твердження перевіряють, що поля sender і text точно рівні заданим значенням.

На рис. 21 зображений результат тестування, яке всебічно перевіряє функціональність класів `Message`, `Chat` і `ChatSummary`. Вони забезпечують точність процесів попередньої обробки тексту, управління повідомленнями, серіалізації даних чату та сумування. Перевіряючи як окремі методи, так і взаємодію між компонентами, ці тести роблять значний внесок у загальну надійність і коректність системи.

<code>test_exct.py::test_clean_text</code>	PASSED	[12%]
<code>test_exct.py::test_split_sentences</code>	PASSED	[25%]
<code>test_exct.py::test_generate_bag_of_words</code>	PASSED	[37%]
<code>test_exct.py::test_add_message</code>	PASSED	[50%]
<code>test_exct.py::test_sort_messages</code>	PASSED	[62%]
<code>test_exct.py::test_get_json</code>	PASSED	[75%]
<code>test_exct.py::test_clusters_init</code>	PASSED	[87%]
<code>test_exct.py::test_get_summary_by_ClusterRank</code>	PASSED	[100%]

Рис. 21. Результати перевірки взаємодії `Message`, `Chat` і `ChatSummary`

Тест `test_clean_text` гарантує, що клас `Message` може обробляти та очищати необроблений текстовий ввід, видаляючи зайві символи, посилання та надлишкові пробіли, зберігаючи при цьому сенс вмісту. Об'єкт `Message` створюється за допомогою зразка тексту, що містить недоліки, наприклад, кілька пробілів, спеціальні символи та URL-адресу. Потім текст обробляється належним чином. Отриманий очищений текст перевіряється на правильність: видалено зайві пробіли, видаляються посилання та зайві символи, зберігаючи лише вміст. Текст перетворюється в нижній регістр, це потрібно для узгодженості.

Тест `test_split_sentences` перевіряє здатність класу `Message` розділяти свій текстовий вміст на окремі речення за допомогою обробки природної мови (NLP). Поділ речень є

важливим етапом попередньої обробки для подальшого аналізу, наприклад, кластеризації або узагальнення. Об'єкт `Message` інстанціюється з текстом, що складається з кількох речень. Виконується метод `plit_sentences` для розбиття тексту на окремі речення. Тест підтверджує, що вихідний список містить всі речення у правильній послідовності, з відповідними розділовими знаками та регістром.

Тест `test_generate_bag_of_words` оцінює здатність класу `Message` генерувати набір слів. Модель спрощує текст до набору окремих слів для обчислювального аналізу. Створюється об'єкт `Message` зі зразком тексту. Викликається метод `generate_bag_of_words` для розділення слів з тексту. Отриманий список слів перевіряється на відповідність очікуваній токенизації, виключаючи розділові знаки та сторонні символи.

Тест `test_add_message` перевіряє функціональність класу `Chat` при додаванні нового повідомлення до його колекції. Він гарантує, що процес додавання оновлює внутрішню структуру даних і зберігає цілісність повідомлення. Ініціалізується об'єкт `Chat` з унікальним ідентифікатором чату та списком учасників. Нове повідомлення додається в об'єкт класу `Chat` за допомогою методу `add_message`. Тест підтверджує роботу: `messages` містить нове повідомлення, `sender`, `timestamp` та `text` доданого повідомлення відповідають вхідним значенням.

Тест `test_sort_messages` – перевіряє, чи може клас `Chat` сортувати свої повідомлення у хронологічному порядку на основі їхніх міток часу. Сортування необхідне для підтримки логічної послідовності в даних чату. Створюється об'єкт `Chat` і заповнюється декількома повідомленнями, доданими у нехронологічному порядку. Викликається метод `sort` для впорядкування повідомлень за міткою часу. Перевіряється, що у відсортованому списку повідомлення впорядковані у хронологічному порядку зі збереженням цілісності даних повідомлень.

Тест `test_get_json` оцінює здатність класу `Chat` конвертувати свої дані в JSON-сумісний формат. Серіалізація необхідна для зберігання, передачі даних та інтеграції з API. Створюється об'єкт `Chat` та заповнюється повідомленнями. Викликається метод `get_json` для генерації JSON-представлення чату. Отриманий JSON-об'єкт перевіряється на коректність: він містить правильний ідентифікатор чату та учасників, `messages` містить всі повідомлення чату з їх атрибутами.

На етапі тестування була проведена перевірка основних компонентів системи, таких як обробка тексту, управління повідомленнями, серіалізація даних і кластеризація. Тести продемонстрували відповідність результатів заданим вимогам і показали коректну взаємодію між компонентами. Зокрема, було протестовано очистку тексту, розбиття на речення, створення об'єктів із коректними атрибутами, сортування повідомлень, серіалізацію даних у формат JSON, а також кластеризацію та формування підсумків. Результати підтвердили надійність системи, гарантували точність роботи кожного методу й загальну узгодженість її функціонування.

4. ДОКУМЕНТАЦІЯ ПРОЄКТУ

Для ведення документації проєкту "Briefly" було обрано централізовану модель на основі інструментів Confluence та Jira, які дозволяють ефективно структурувати та зберігати всю необхідну інформацію. Інструменти забезпечують зручність роботи команди, швидкий доступ до матеріалів та інтеграцію з іншими сервісами, такими як Figma для візуалізації дизайну, а також Google Meet для організації регулярних зустрічей.

Комунікація

Owned by [Kyryl Bedrak](#) •••
 Last updated: Nov 13, 2024 • 1 min read • 👁 See how many people viewed this page

Планові зустрічі:

Briefly Proj Daily	Пн-Пт 16:00	@Kyryl Bedrak @Yana Malashkina @Olexandr @Matthew Kovalyk
Briefly Proj Catch-up	Сб/Нд/Пн 20:00	@Kyryl Bedrak @Yana Malashkina @Olexandr @Анастасія @Matthew Kovalyk @Юліана Карпцова @Denys Bondarchuk
Briefly Proj Demo	Вт/Ср 13:00	@Kyryl Bedrak @Yana Malashkina @Olexandr @Matthew Kovalyk Викладачі

🔗 Посилання на Google Meet: <https://meet.google.com/npn-bzht-uzz>

Рис. 22. План комунікації

У рамках роботи було створено кілька видів документів. До внутрішньої документації належать нотатки планових зустрічей, що фіксують основні рішення та прогрес у проєкті (рис. 23).

В аналітичній частині представлено SWOT-аналіз і Lean Canvas, які описують ключові переваги, ринкові можливості та бізнес-модель проєкту. У розділі планування зібрано дорожню карту та RACI-матрицю для визначення відповідальності членів команди (рис. 24).

Зовнішня документація включає документи, орієнтовані на користувачів, партнерів та наглядові органи. Для користувачів створено інструкцію, яка містить покроковий опис роботи з ботом, наприклад: запуск у Slack, вибір параметрів підсумовування, отримання стислих результатів. Інструкція супроводжується ілюстраціями з прикладами інтерфейсу.

Для партнерів підготовлено технічну документацію API, що описує доступні ендпоінти, формати запитів і сумісність із популярними платформами. Для наглядових органів розроблено політику конфіденційності, яка визначає обробку та зберігання даних користувачів, відповідно до стандартів безпеки.

Ця система документації дозволяє ефективно організувати процес розробки, забезпечити доступність матеріалів для всіх стейкхолдерів і створити зручний інтерфейс взаємодії із зовнішнім середовищем.

Правові норми. Правові документи, що регулюють діяльність продукту:

– Закон України "Про інформацію"[12], регулює доступ до інформації, її захист і обмеження.

– Закон України "Про захист персональних даних"[13], забезпечує права користувачів на захист, виправлення або видалення персональних даних.

Збірник матеріалів за результатами IT-проекту міждисциплінарної інтеграції, 2024-2025 н.р.

- Закон України "Про віртуальні активи"[14], встановлює статус цифрових активів у разі їх використання в рамках системи.
- Закон України "Про авторське право і суміжні права"[15], регулює права на використання алгоритмів і контенту, що генерується системою.
- Закон України "Про електронну комерцію"[16], легалізує електронні угоди, оплату та захист прав користувача.
- Закон України "Про захист прав споживачів"[17], гарантує право користувача на якісний сервіс, достовірну інформацію та захист у разі спорів.

📅 Дата - 04.10.2024

👤 Учасники

@Kyryl Bedrak @Yana Malashkina @Olexandr @Анастасія @Matthew Kovalyk @Юліана Карпцова @Denys Bondarchuk

📌 Мета

- Обговорити ідеї, попередньо запропоновані учасниками команди у чаті
- Обрати найперспективнішу із запропонованих ідей виходячи зі SWOT-аналізу

🗨️ Обговорення

Тема	Деталі
Обговорення ідей	Результати у вигляді SWOT-аналізу: SWOT ідеї (Розширена версія)
Розподіл ролей	
Технології реалізації	<ul style="list-style-type: none">• Python як основна мова розробки.• Використання бібліотек: Spacy, NLTK, можливо, TextRagik та BM25 для алгоритмів ранжування.
Зустрічі та комунікація	<ul style="list-style-type: none">• Створено чат для комунікації у месенджері• Буде створено акаунт Jira для планування завдань• Домовились зустрічатись раз в тиждень усією командою в Google Meet

✅ Завдання

- Лінгвісти нададуть список бібліотек та інструментів, з якими вже працювали.
- Кібернетики проаналізують статтю про Text Summarization, зроблять огляд та визначать можливі алгоритми.
- Написати в ком'юніті українських лінгвістів (наприклад, NLTK) для пошуку додаткових статей та рішень.
- Організувати робочий простір у Jira/Confluence та додати таски.
- Розпочати аналіз існуючих аналогів та ринку.

📌 Рішення

👉

Рис. 23. Нотатки онлайн зустрічей

Software Development / Дорожня карта
RACI матриця

Завдання	Керівник проекту	Інженери з розробки API	Месенджерів	Філологи та лінгвісти	Тестувальники	Фахівці з документації
Планування та координація проекту	A	I	I	I	I	C
Формування вимог та цілей	A	I	C	C	I	C
Оцінка ресурсів та визначення термінів	A	C	C	I	I	I
Визначення ключових показників ефективності (KPI)	A	C	I	I	C	I
Розробка архітектури та основного плану	A	R	C	C	I	I
Створення технічних специфікацій	A	R	C	I	I	I
Узгодження архітектури з командою	A	R	C	C	I	C
Визначення ключових алгоритмів обробки тексту	I	R	I	C	C	I
Розробка API та реалізація алгоритмів	I	R	I	C	I	I
Розробка бекенд-логіки для API	I	R	I	C	I	I
Розробка основних алгоритмів обробки тексту	I	R	I	C	C	I
Налаштування бази даних	C	R	I	I	I	I
Інтеграція з платформами месенджерів	I	C	R	I	I	I
Налаштування інтеграції з Telegram	I	C	R	I	C	I
Налаштування інтеграції з WhatsApp та Discord	I	C	R	I	C	I
Управління API-ключами та безпекою	C	C	R	I	I	I
Адаптація алгоритму для української мови	C	I	I	R	I	I
Лінгвістична адаптація алгоритму	I	I	I	R	C	I
Обробка неформальної мови та сленгу	I	I	I	R	C	I
Налаштування обробки стоп-слів	I	C	I	R	C	I
Тестування функціональності та якості	I	C	I	C	R	I
Тестування функціональності API	I	C	I	I	R	I
Тестування інтеграції з месенджерами	I	I	C	I	R	I
Оцінка якості саммаризації та коректності	I	C	I	C	R	I
Створення документації та навчання	C	I	I	I	C	R
Розробка документації для розробників	I	I	C	I	I	R
Підготовка документації для користувачів	I	I	I	I	C	R
Підготовка навчальних матеріалів	C	I	I	I	I	R
Маркетинг та підготовка до запуску	R	I	I	I	I	C
Підготовка матеріалів для маркетингової кампанії	A	I	I	C	I	C
Планування презентацій для користувачів	A	I	I	I	I	C
Розробка маркетингової стратегії	A	I	I	I	I	C

Рис. 24. RACI-матриця

5. УПРАВЛІННЯ ПРОЄКТОМ

Опис команди, ролей в команді

– Проєкт "Briefly" API реалізовано командою, яка складалася з фахівців різних напрямків, кожен з яких мав чітко визначену роль у процесі розробки:

– керівник проєкту – відповідав за загальне управління проєктом, планування завдань та координацію роботи команди. Його роль полягала в забезпеченні ефективної взаємодії між учасниками команди, контроль виконання завдань у встановлені терміни та прийняття стратегічних рішень щодо розробки продукту;

– scrum-мастер – координував роботу команди відповідно до методології Scrum. Його завданням було організовувати спринти, щоденні стендапи, ретроспективи та планування. Scrum-мастер слідкував за дотриманням Agile-практик, усуненням перешкод у роботі команди та підтримував культуру постійного вдосконалення процесів;

– інженери з розробки API – ця група фахівців займалася розробкою бекенду, зокрема реалізацією основних алгоритмів обробки тексту та логіки функціонування бота. Їхня робота включала створення структурованої архітектури API, впровадження

генеративних та екстрактивних методів саммаризації текстів, а також оптимізацію продуктивності системи;

- фахівці з інтеграції месенджерів – основним завданням цих фахівців було налаштування інтеграції "Briefly" API з платформою Discord та початок інтеграції з іншими месенджерами. Вони забезпечували коректну передачу даних між платформами та API, а також адаптацію продукту під специфіку кожного месенджера;

- філологи та лінгвісти – ці фахівці займалися адаптацією алгоритмів обробки тексту під українську мову, що включало роботу з неформальною мовою, діалектизмами та скороченнями. Їхня роль була ключовою для забезпечення високої якості саммаризації українською мовою, що є однією з унікальних переваг продукту;

- інженери з розробки бекенду – відповідали за серверну частину вебсайту, включаючи управління базами даних, обробку запитів та безпеку. Вони розробляли RESTful API, забезпечували стабільну роботу серверної логіки, а також здійснювали підтримку інтеграції з іншими компонентами системи, зокрема з інфраструктурою саммаризатора та месенджерів;

- інженер з розробки фронтенду – займався реалізацією користувацького інтерфейсу на основі дизайну, наданого дизайнером. Його завдання включали розробку адаптивного вебінтерфейсу, інтеграцію з API, забезпечення швидкості завантаження сторінок, а також оптимізацію взаємодії з користувачем. Особливу увагу він приділяв зручності та інтуїтивності інтерфейсу;

- дизайнери вебсайту – відповідали за розробку візуальної концепції продукту та створення макетів вебінтерфейсу. Їхня робота включала розробку UI/UX-дизайну, підбір кольорової гами, типографіки та елементів взаємодії, що забезпечують інтуїтивну та зручну навігацію для користувачів. Дизайнери тісно співпрацювали із фронтенд-розробниками для реалізації свого бачення у готовому продукті;

- аналітик і фахівець з документації – відповідав за формування вимог до системи, роботу з мапою продукту, структурування функціоналу та фінальне оформлення звітної документації. Його завдання включали опис логіки роботи системи, API, баз даних і створення зрозумілих матеріалів для команди та користувачів;

- тестувальники – відповідали за перевірку функціональності API, інтеграції бота та якості саммаризації текстів. Вони проводили ручне та автоматизоване тестування, ідентифікували помилки та перевіряли, як продукт відповідає вимогам і очікуванням користувачів.

Розробка відбувалася за методологією Scrum, що дозволило команді працювати ітеративно, постійно вдосконалюючи функціонал. На кожному етапі всі члени команди взаємодіяли через планові зустрічі та координаційні наради, результатом яких став чіткий план дій та подальша перевірка реалізованих завдань.

Завдяки чітко розподіленим ролям у команді та використанню сучасних інструментів для управління процесами Jira та Confluence, вдалося забезпечити ефективну комунікацію, якість розробки та відповідність кінцевого продукту початковим цілям.

Опис обраної методології, інструментарію, обґрунтування вибору.

Команда обрала модель Scrum через її гнучкість та орієнтованість на швидку адаптацію до змінних вимог. Цей підхід дозволяє регулярно переглядати пріоритети, ефективно планувати завдання та зосереджуватися на досягненні конкретних результатів.

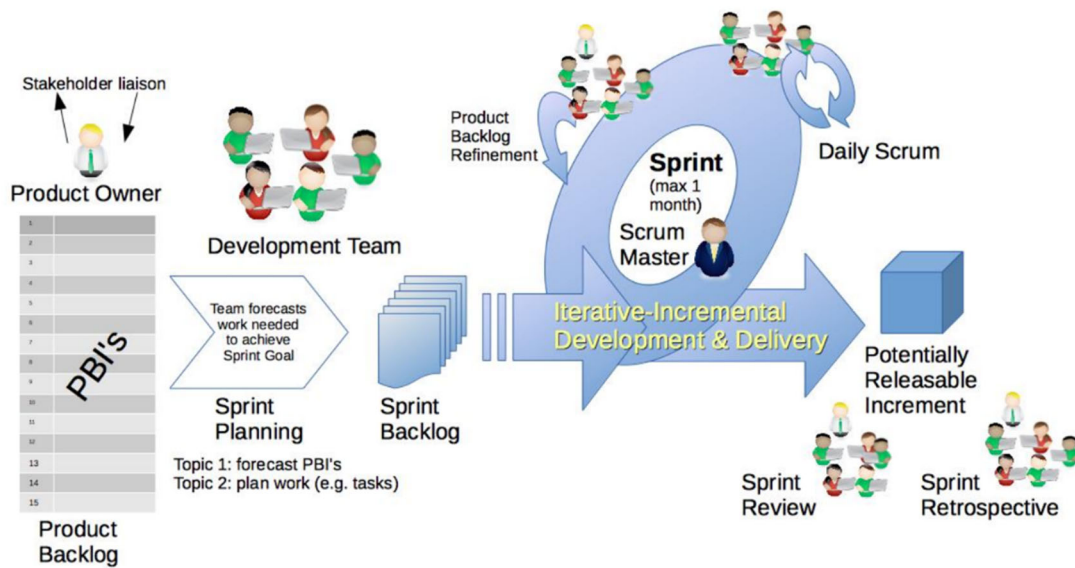


Рис. 25. методологія Scrum

Головною перевагою Scrum є можливість отримувати робочий продукт після кожного спринту, що дозволяє швидко тестувати функціонал і реагувати на зворотний зв'язок від користувачів. Це особливо важливо для проєкту "Briefly", де потрібна висока гнучкість у реагуванні на потреби українських користувачів та інтеграції з різними платформами.

Вибір Scrum вплинув на організацію та структуру ведення документації в проєкті. Scrum передбачає створення документації на кожному етапі розробки, включаючи:

- планування беклогу продукту та кожного спринту;
- ведення нотаток зустрічей для синхронізації команди;
- документування результатів спринтів, що дає змогу аналізувати прогрес і вдосконалювати процеси;
- постійне оновлення та фіксація завдань у інструментах Jira та Confluence.

Scrum створив умови для прозорого та послідовного ведення документації, яка є важливою як для внутрішньої роботи команди, так і для зовнішньої комунікації з партнерами та користувачами. Таким чином, обраний підхід сприяв організованості та актуальності всієї проєктної документації.

Демонстрація планування спринтів

У процесі розробки "Briefly" API ключову роль відіграло планування спринтів, яке забезпечувало структурований ітеративний підхід до реалізації задач. У рамках кожного спринту завдання визначалися, пріоритизувалися та розподілялися серед членів команди відповідно до їхніх ролей і компетенцій.

На Рисунку представлено загальний таймлайн проєкту, який демонструє розподіл завдань на етапи розробки. Кожен спринт був спрямований на досягнення конкретних цілей, таких як реалізація окремих функцій API, інтеграція з платформами або тестування.

Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції, 2024-2025 н.р.

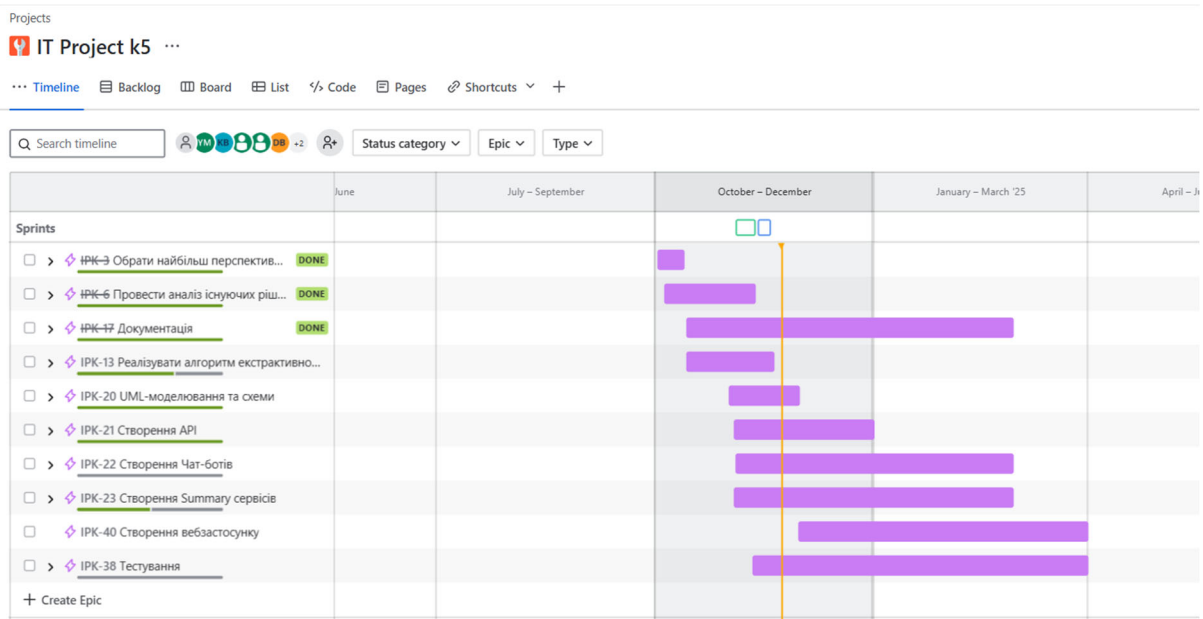


Рис. 26. Діаграма Ганта

Розподіл завдань у спринтах проводився з використанням RACI-матриці, яка забезпечувала чіткий розподіл обов'язків:

- відповідальні (A) особи, які приймали остаточні рішення щодо виконання задач;
- виконавці (R): члени команди, які виконували задачи;
- консультанти (C): експерти, що надавали необхідну інформацію;
- інформовані (I): учасники, яких тримали в курсі прогресу.

На Рисунках 27 та 28 наведені приклади спринтів та завдань.

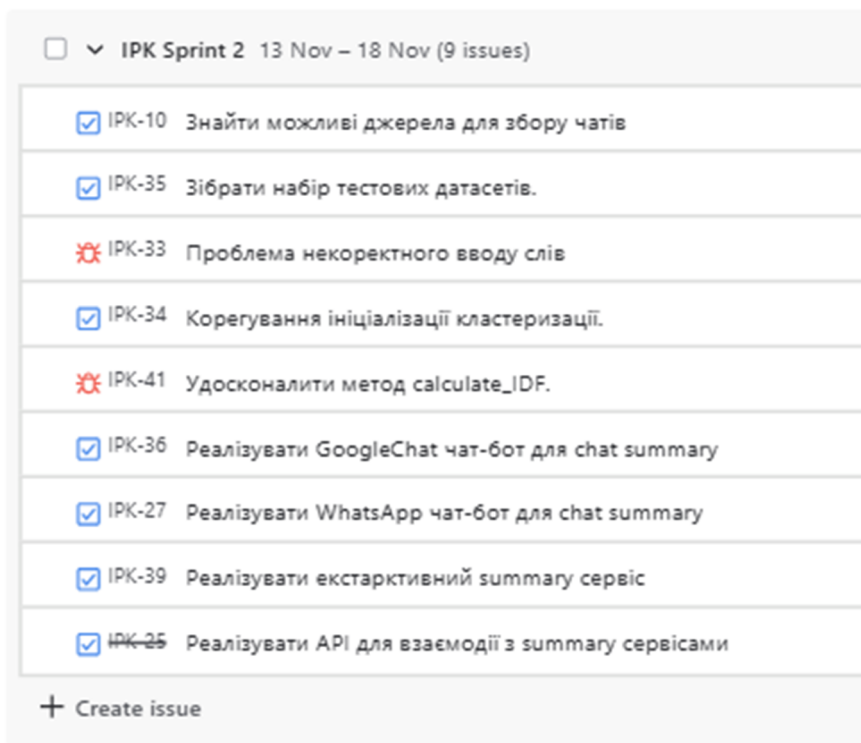


Рис. 27. План спринта

IPK-13

Реалізувати алгоритм екстрактивного сумаризатора

+ Add

Description
Add a description...

Child issues Order by ▾ ⋮ +

66% Done

<input type="checkbox"/>	IPK-33	Проблема некоректного вводу слів	▬ -		TO DO ▾
<input checked="" type="checkbox"/>	IPK-34	Корегування ініціалізації кластеризації.	▬ -		TO DO ▾
<input type="checkbox"/>	IPK-41	Удосконалити метод calculate_IDF.	▬ -		TO DO ▾
<input checked="" type="checkbox"/>	IPK-14	Попередня обробка тексту чатів	▬ -		DONE ▾
<input checked="" type="checkbox"/>	IPK-15	Формування матриці схожості	▬ -		DONE ▾
<input checked="" type="checkbox"/>	IPK-16	Ранжування речень та формування summary	▬ -		DONE ▾
<input type="checkbox"/>	IPK-30	Після лематизації залишаються слова, що починаються з великої літери	▬ -		DONE ▾
<input type="checkbox"/>	IPK-31	Попередня обробка тексту (preprocessing)	▬ -		DONE ▾
<input type="checkbox"/>	IPK-32	Неоднозначність з розбиттям на речення.	▬ -		DONE ▾

Рис. 28. Приклади завдань

Кожне завдання було розподілено між учасниками відповідно до їхніх спеціалізацій. Інженери з API займалися реалізацією алгоритмів, фахівці з інтеграції – налаштуванням платформ, а тестувальники перевіряли якість реалізованих функцій.

Особливу увагу приділяли гнучкості планування. Завдання могли коригуватися залежно від прогресу спринту чи змін у пріоритетах проєкту. Це дозволяло команді швидко реагувати на нові виклики та адаптувати роботу до змінних вимог.

Планування спринтів у такій структурі забезпечило ефективну координацію команди, прозорість виконання задач і досягнення результатів у встановлені терміни.

ВИСНОВКИ

У ході роботи над проєктом було виконано низку завдань, визначених на етапі планування та відображених у відповідних розділах звіту. В результаті було реалізовано систему "Briefly", яка забезпечує автоматичне підсумовування як текстових, так і медіа-даних із різних джерел. Було створено вебзастосунок і API, що дозволяють генерувати структуровані резюме чатів, документів та відео, а також Discord-бот у рамках інтеграції з месенджерами. Основний функціонал MVP охоплює генеративне й екстрактивне підсумовування, підтримку параметрів користувача, таймкодів для відео, багатомовність і інтеграцію з платформами типу YouTube.

У процесі реалізації команда поглибила знання в галузі обробки природної мови (NLP), застосувавши алгоритми BM25, TF-IDF, а також генеративні моделі на базі GPT для створення узагальнень тексту й тем за таймкодами. Значну увагу було приділено оптимізації швидкодії API та підвищенню точності результатів, зокрема шляхом впровадження паралельної обробки, структуризації відповідей і тестування моделей на реальних даних.

Команда також здобула досвід організації розробки за принципами Scrum – були сформовані спринти, backlog, налагоджено ефективну взаємодію через Jira і Confluence. Це дозволило систематизувати роботу, фіксувати результати і своєчасно вносити необхідні коригування.

Загалом, реалізована версія Briefly повністю відповідає меті MVP і демонструє високий рівень технічної готовності, актуальність і масштабованість продукту, що створює передумови для його подальшого розвитку та комерційного використання.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. OpenAI – офіційна платформа. URL: <https://platform.openai.com> (дата звернення: 23.04.2025).
2. LLaMA – модель штучного інтелекту. URL: <https://ai.meta.com/llama> (дата звернення: 23.04.2025).
3. Gulati, V.; Kumar, D.; Popescu, D.E.; Hemanth, J.D. Extractive Article Summarization Using Integrated TextRank and BM25+ Algorithm // Electronics. – 2023. – 12(372). – 17 с. URL: <https://doi.org/10.3390/electronics12020372/> (дата звернення: 23.04.2025).
4. Summarizer BOT. URL: https://t.me/shrtdb_bot (дата звернення: 23.04.2025).
5. Суммаризатор від Synapse. Telegram. URL: <https://t.me/summarisatorBot> (дата звернення: 23.04.2025).
6. БіЧ Summarizer. URL: https://t.me/bich_summary_bot (дата звернення: 23.04.2025).
7. SB Summarizer. URL: https://t.me/SB_Summarizer_Bot (дата звернення: 23.04.2025).
8. Документація FastAPI. FastAPI. URL: <https://fastapi.tiangolo.com/> (дата звернення: 23.04.2025).
9. Документація SpaCy. URL: <https://spacy.io> (дата звернення: 23.04.2025).
10. Інструмент дизайну Figma. URL: <https://www.figma.com> (дата звернення: 23.04.2025).
11. Бібліотека React. URL: <https://reactjs.org> (дата звернення: 23.04.2025).
12. Про інформацію : Закон України від 02.10.1992 № 2657-XII : станом на 15 лист. 2024 р. URL: <https://zakon.rada.gov.ua/laws/show/2657-12#Text> (дата звернення: 23.04.2025)
13. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI : станом на 18 січ. 2025 р. URL: <https://zakon.rada.gov.ua/laws/show/2297-17#Text> (дата звернення: 23.04.2025).
14. Про віртуальні активи : Закон України від 17.02.2022 № 2074-IX : станом на 15 лист. 2024 р. URL: <https://zakon.rada.gov.ua/laws/show/2074-20#Text> (дата звернення: 23.04.2025).
15. Про авторське право і суміжні права : Закон України від 23.12.1993 № 3792-XII : станом на 1 січ. 2023 р. URL: <https://zakon.rada.gov.ua/laws/show/3792-12#Text> (дата звернення: 23.04.2025).
16. Про електронну комерцію : Закон України від 03.09.2015 № 675-VIII : станом на 1 січ. 2024 р. URL: <https://zakon.rada.gov.ua/laws/show/675-19#Text> (дата звернення: 23.04.2025).
17. Про захист прав споживачів : Закон України від 12.05.1991 № 1023-XII : станом на 24 груд. 2024 р. URL: <https://zakon.rada.gov.ua/laws/show/1023-12#Text> (дата звернення: 23.04.2025).

ВЕБ-ЗАСТОСУНОК "ЗРУЧНОРОЗКЛАД"

*Денис АСТАФ'ЄВ, Сергій ХАЛІК, Костянтин ДЕМЕНТЬЄВ, Арсеній САГАЛЯНОВ,
Олександр ПИЩАНЕЦЬКИЙ, Діана СУШКО, Євгенія БАЖМАЄВА*

Дана робота описує процес розробки системи для автоматизації складання та управління навчальними розкладами. Проект спрямований на полегшення процесу організації навчального процесу для студентів, викладачів та адміністрації навчальних закладів. Система дозволяє генерувати розклад автоматично, редагувати його вручну та надавати зручний інтерфейс для перегляду. Розробка базується на сучасних технологіях, таких як React, NestJS та MySQL.

РОЗКЛАД, АВТОМАТИЗАЦІЯ, НАВЧАЛЬНИЙ ПРОЦЕС, WEB-ДОДАТОК, REACT, NESTJS

WEB APPLICATION "ZRUCHNOROZKLAD"

*Denys ASTAFIEV, Serhiy KHALIK, Kostiantyn DEMENTIEV, Arsenii SAHALIANOV,
Oleksandr PISHCHANETSKIY, Diana SUSHKO, Yevheniia BAZHMAEVA*

This work describes the development process of a system for automating the creation and management of educational schedules. The project aims to simplify the organization of the educational process for students, teachers, and administrative staff. The system allows for automatic schedule generation, manual editing, and provides a user-friendly interface for viewing. The development is based on modern technologies such as React, NestJS and MySQL.

SCHEDULE, AUTOMATION, EDUCATIONAL PROCESS, WEB APPLICATION, REACT, NESTJS

ВСТУП

Постановка задачі. Створення розкладу занять завжди було складною задачею, яка вимагала врахування багатьох факторів: наявності аудиторій, завантаженості викладачів, особливостей навчальних груп тощо. Традиційний спосіб планування вручну забирає багато часу і часто призводить до помилок, особливо в великих університетах. Наш проект "ЗручноРозклад" запропонував автоматизоване рішення цієї проблеми, розробивши зручний інструмент для генерації, редагування та перегляду розкладу з мінімальними зусиллями з боку користувачів.

Актуальність теми. Ефективне планування навчального процесу завжди було ключовим елементом успішної роботи навчальних закладів. Невдало складений розклад часто призводив до накладок, плутанини серед студентів і викладачів, а також знижував загальну продуктивність. Існуючі рішення виявились або занадто складними, або обмеженими у функціоналі. Саме тому ми реалізували інтуїтивно зрозумілу та автоматизовану систему, яка ефективно вирішує ці проблеми.

Методи дослідження. Для створення оптимального рішення ми провели детальне вивчення потреб кінцевих користувачів – студентів, викладачів і адміністрації. Проаналізували існуючі аналоги, виявивши їхні переваги та недоліки. На основі

отриманих даних розробили прототип системи, який протестували в реальних умовах. Це дозволило нам доопрацювати інтерфейс та алгоритми, забезпечивши високу зручність та ефективність роботи з системою.

Мета та завдання роботи. Основною метою проєкту було спрощення процесів створення та управління розкладом занять. Для її досягнення ми розробили зручний веб-інтерфейс, реалізували ефективний алгоритм автоматичної генерації розкладу та забезпечили можливість експорту даних у зручних форматах. Особливу увагу приділили тестуванню системи, що дозволило гарантувати її стабільну та безпомилкову роботу. В результаті було створено інструмент, який істотно полегшив організацію навчального процесу.

1. ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

Опис пошуку ідеї продукту

Обрана сфера для розробки продукту – це *EdTech* (освітні технології). Ця сфера активно розвивається, оскільки з кожним роком зростає попит на інноваційні рішення в освіті, що дозволяють полегшити навчальний процес як для студентів, так і для викладачів. Платформи, які оптимізують процеси навчання, управління часом і створення розкладів, набувають особливої популярності в умовах цифровізації освіти.

Додатково були розглянуті ідеї, що містять *HRTech* – технології, спрямовані на автоматизацію та вдосконалення процесів управління людськими ресурсами та *GIS* (геоінформаційні системи) – системи для збору, аналізу та відображення географічних даних.

Ідеї, які обговорювались командою:

- *EdTech* – додаток для навчального розкладу: Платформа, що дозволяє студентам переглядати, редагувати та генерувати розклад занять з автоматичними нагадуваннями;
- *HRTech* – фріланс біржа для студентів: Платформа, яка дозволяє студентам знаходити тимчасові фріланс-проєкти для додаткового заробітку та розвитку кар'єрних навичок;
- *GIS* – локації інтересу для студентів: Додаток, що використовує географічні інформаційні системи для відображення найближчих навчальних ресурсів, кафе, бібліотек і інших корисних локацій для студентів.

Аргументація вибору сфери та ідеї

Сфера *EdTech* була обрана через високий попит на технології, що покращують освітній процес. Особливо актуальною є потреба в інструментах для управління часом, що дозволяють студентам ефективно організувати свої навчальні графіки. Серед запропонованих ідей, додаток для навчального розкладу був вибраний завдяки своїй універсальності та прямій користі для основної цільової аудиторії. Він дозволяє автоматизувати створення розкладу, що значно полегшує роботу адміністрації навчальних закладів і підвищує зручність для користувачів.

Ідеї, пов'язані з *HRTech* і *GIS*, також є актуальними, але вони менш специфічні для поточних потреб користувачів, оскільки ринок фрілансу вже насичений, а застосування геоінформаційних систем для студентів має обмежене практичне застосування порівняно з автоматизацією процесу планування розкладу.

Таким чином, ідея створення додатку для навчального розкладу виглядає найбільш перспективною, оскільки вона вирішує нагальну проблему і має значний потенціал для розвитку в майбутньому.

Огляд наявних на ринку ІТ-продуктів – конкурентів

Система розкладу *UNITIME*. Аналіз даного продукту подано в таблиці 1.

Таблиця 1. Аналіз системи розкладу UNITIME

Критерій оцінки	Дані системи UNITIME
Назва продукту та посилання	UniTime – https://www.unitime.org
Позиціонування продукту	Веб застосунок для освітніх установ, який надає комплексні рішення для складання розкладів занять, іспитів, студентських секцій та управління подіями
Ключові функції	Складання розкладів занять і іспитів
	Автоматизація змін у розкладі з мінімальними порушеннями
	Управління студентськими групами і секціями
	Інтерактивний інтерфейс для редагування розкладу в реальному часі
Цільова аудиторія продукту	Освітні установи, включаючи університети та коледжі, що потребують автоматизації процесу складання розкладу
Географія	Використовується в університетах по всьому світу, включаючи США, Європу, Азію та інші регіони
Мовні локалізації	Англійська
Модель монетизації	Open-source
Вартість доступу до продукту	Безкоштовно
Тип застосунку	Веб-платформа
Переваги продукту	Відкрите програмне забезпечення з можливістю кастомізації
	Підтримка складних алгоритмів для оптимізації розкладів
	Підходить для великих установ із тисячами студентів
Недоліки продукту	Відсутність мобільного застосунку
	Складність налаштування для нових користувачів

Загальна оцінка продукту – потужний інструмент для складання освітніх розкладів, що пропонує високий рівень автоматизації, але потребує вдосконалення для мобільних користувачів і простоти налаштування [1].

Освітня система "ЄДИНА ШКОЛА". В таблиці 2 приведено огляд основних метрик продукту.

Загальна оцінка продукту – потужний інструмент для складання освітніх розкладів, що пропонує високий рівень автоматизації, але потребує вдосконалення для мобільних користувачів і простоти налаштування [2].

Таблиця 2. Аналіз освітньої системи "ЄДИНА ШКОЛА"

Критерій оцінки	Дані системи "ЄДИНА ШКОЛА"
Назва продукту та посилання	Платформа "Єдина школа" – https://eschool-ua.com
Позиціонування продукту	Інформаційно-комунікаційна автоматизована система для підтримки освітнього процесу. Призначена для електронного ведення журналів, щоденників, аналізу відвідуваності, оцінок та управління освітнім процесом
Ключові функції	Електронний журнал і щоденник
	Аналітика і звітність для закладів освіти
	Інструменти для дистанційного навчання
	Управління навчальними модулями, тестами і домашніми завданнями
Цільова аудиторія продукту	Заклади середньої освіти, включаючи адміністрацію, педагогів, учнів та їхніх батьків
Географія	Україна. Розроблена для національного використання в закладах середньої освіти

Збірник матеріалів за результатами IT-проєкту міждисциплінарної інтеграції, 2024-2025 н.р.

Мовні локалізації	Українська мова. Інші локалізації не зазначені
Модель монетизації	
Вартість доступу до продукту	Безкоштовно (на час воєнного стану)
Тип застосунку	Веб-платформа, мобільні додатки
Переваги продукту	Комплексне вирішення для автоматизації освітніх процесів
	Зручне використання для вчителів, учнів і батьків
	Підтримка дистанційного навчання та аналітичні модулі
Недоліки продукту	Залежність від стабільного інтернет-з'єднання
	Обмежений рівень інтеграції засобів доступу

Мапа продукту. Мапа продукту подана у вигляді Lean Canvas на рис. 1.

Проблема	Розв'язок	Унікальна цінність	Неринкова перевага	Цільова аудиторія
Учасники навчального процесу використовують неефективні ресурси керування розкладом	Зручний застосунок для управління розкладом з простим інтерфейсом і зручними функціями	Допомога працівникам ЗВО у генерації розкладу за введеними даними	- Налаштування під кожного окремого користувача - Унікальний функціонал з простим і доступним інтерфейсом	Учасники навчального процесу: - Студенти - Викладачі - Працівники навчальних закладів
Альтернативи	Ключові метрики - Кількість користувачів - Кількість годин, проведених у застосунку - Кількість навчальних закладів, що використовують застосунок	Високорівнева концепція	Канали - Рекомендації користувачів - Співпраця з ЗВО - Реклама у соц. мережах	Перші користувачі
- Розклад Нова Школа - Єдина Школа - Unitime - USched		Застосунок, що допомагатиме у вирішенні будь-яких проблем що стосуються планування навчального процесу.		Студенти і викладачі ФКНН КНУ, що зацікавлені у простому управлінні розкладом навчального процесу
Структура витрат		Джерела доходу		
- Створення сайту - Реклама у соц. мережах - Підтримка контактів з закладами освіти - Оренда домену		- Донати від користувачів - Допомога від освітніх організацій/закладів освіти - Рекламні банери		

Рис. 1. Lean Canvas.

Сильні і слабкі сторони розроблюваного продукту

Сильними сторонами продукту є зручний доступ до розкладу через веб-сторінку, а також оперативне оновлення даних і сповіщення про зміни. Однак існує й слабкий бік – наразі відсутній мобільний додаток, що може обмежувати користувачів. Серед потенційних ризиків варто відзначити загрози безпеці даних і можливі технічні збої, які тимчасово порушують доступ до сервісу. Разом з тим, система має значні можливості для розвитку: розширення функціоналу (наприклад, додавання аналітики або запису на консультації), а також партнерства з університетами для впровадження продукту в їхній навчальний процес.

Огляд використаних технологій

Для розробки проєкту "ЗручноРозклад" обрано сучасні та ефективні технології, які відповідають вимогам розробки веб-застосунку, оптимізованого під мобільні пристрої:

- **фронтенд.** Наш проєкт використовує *React* – популярна JavaScript-бібліотека для створення інтерфейсів користувача і *MUI* – бібліотека компонентів для React, яка реалізує принципи дизайну Material Design [3-4];

- **axios.** В проєкті застосовувались *Axios* – це бібліотека для роботи з HTTP-запитами та *Orval Codegen* – інструмент для автоматичної генерації клієнтських запитів на основі OpenAPI-специфікацій [5-6];

- **бекенд.** Код написаний на *NestJS* – прогресивному фреймворку для створення бекенд-застосунків на Node.js; тестування проводилося за допомогою *Jest* – фреймворку для JavaScript і TypeScript що пропонує широкий функціонал для модульного та інтеграційного тестування; а для управління базами даних використовувалась *MySQL* [7-9].

Також ми додатково адаптували дизайн для зручного перегляду на мобільних пристроях.

2. ОПИС СИСТЕМИ

Призначення і цілі створення системи. "ЗручноРозклад" був створений аби забезпечити користувача методами ефективного створення навчального розкладу.

Система розроблялась таким чином, аби сприяти:

- організації навчального процесу, надаючи студентам і викладачам зручний інструмент для доступу до розкладу;
- автоматизації процесу створення розкладу, спрощуючи виконання відповідних задач для адміністрації навчального закладу і зменшення помилок;
- оптимізації часу, надаючи швидкий доступ до актуальної інформації для всіх учасників навчального процесу;
- гнучкого керування і можливості редагування розкладу відповідно до змін у потребах.

Вимоги до системи. Для сприяння злагоженому процесу розробки продукту, необхідно визначити низку вимог, яким цей продукт має відповідати. Для зручності ми розподілимо їх на типи:

- загальні вимоги, наведені в таблиці 3;
- вимоги до структури та функціонування системи, що описані в таблиці 4;
- вимоги до функцій що виконуються системою, задачі яких вказані в таблицях 5 і 6 (для адміністратора і користувача відповідно).

Таблиця 3. Загальні вимоги

Вимога	Опис
Автоматизація роботи	Створення, редагування, перегляд та управління розкладом
Зручність доступу	Веб платформа з підтримкою мобільних пристроїв
Безпека даних	Авторизація та захист інформації
Інтеграція	Підключення до зовнішніх систем (навчальні портали тощо)

Таблиця 4. Вимоги до структури та функціонування системи

Підсистема	Функціонал
Адміністративна	Управління розкладом (пар, груп, викладачів)
	Генерація звітів
	Контроль доступу
Користувацька	Реєстрація/аутентифікація
	Перегляд розкладу (дати, групи, викладачі)
	Додавання побажань

Підсистема адміністратора – набір функцій, які може запускати адміністратор.

Таблиця 5. Перелік функцій, задач що підлягають автоматизації

Функція	Задача
Керування користувачами	Додавання, редагування та видалення інформації про користувачів
Робота з розкладом	Генерація розкладу, редагування пар, викладачів, аудиторій
Формування звітів	Візуалізація статистики та логів активності користувачів
Робота з побажаннями	Обробка запитів від викладачів щодо часу занять

Підсистема користувача – набір функцій, які може запускати звичайний користувач.

Таблиця 6. Перелік функцій, задач що підлягають автоматизації

Функція	Задача
Реєстрація	Введення інформації про себе
Перегляд розкладу	За датами, групами, викладачами
Подання побажань	Вибір бажаного часу занять
Оновлення інформації	Зміна даних профілю користувача

UML діаграми. Діаграма Use-Case, наведена на рис. 2, демонструє основний функціонал застосунку у зручному візуальному форматі, тоді як діаграма класів, зображена на рис. 3 дозволить нам зрозуміти внутрішню структуру.

Системні вимоги. Програма повинна коректно функціонувати у всіх сучасних веб-браузерах, включаючи *Google Chrome*, *Mozilla Firefox*, *Microsoft Edge*, *Safari* та *Opera*, на комп'ютерах та мобільних пристроях (Android, iOS). Інтерфейс адаптований до різних розмірів екрану і забезпечує зручне користування. Помилки введення даних або некоректні дії користувача візуально позначаються в інтерфейсі. Для роботи необхідний стабільний інтернет-зв'язок.

Опис інформаційної бази. Відповідні характеристики даних наведено у таблиці 7.

Таблиця 7. Опис інформаційної бази

Категорія	Характеристики і обмеження
Структура Excel файлу	Ім'я викладача: рядок 6, клітинка 4
	Дані про предмети: з рядка 19
	Стовпець 1: Назва предмета
	Стовпець 2: Курс (напр. "1 курс" → "1")
	Стовпець 7: Тип роботи ("лек.", "лаб.", "сем.", "практ.")
	Стовпець 13: Кількість годин
Умови завершення обробки	Обробка припиняється при слові "Всього" у назві предмета
Фільтрація даних	Пропускаються рядки зі словами "факультет" або "навчально-науковий інститут"
Продуктивність	Використання fs.writeFileSync блокує основний потік
	Не оптимізовано для великих файлів
Валідація даних	Відсутня детальна перевірка даних
	Помилки не логуються
Масштабованість	Не підтримує великі обсяги даних
	Обмежена кількість одночасних запитів
Кодування	Очікується UTF-8 (українські символи)
Залежності	Бібліотеки: ExcelJS, fs
	Необхідна сумісність з Node.js/NestJS
Ресурси	Не оптимізовано для обмеженої пам'яті/процесора
Конкурентність	Відсутня асинхронна обробка (блокування при багатокористувацькому доступі)
Логування	Фіксує лише успішне збереження (немає детальних логів помилок)

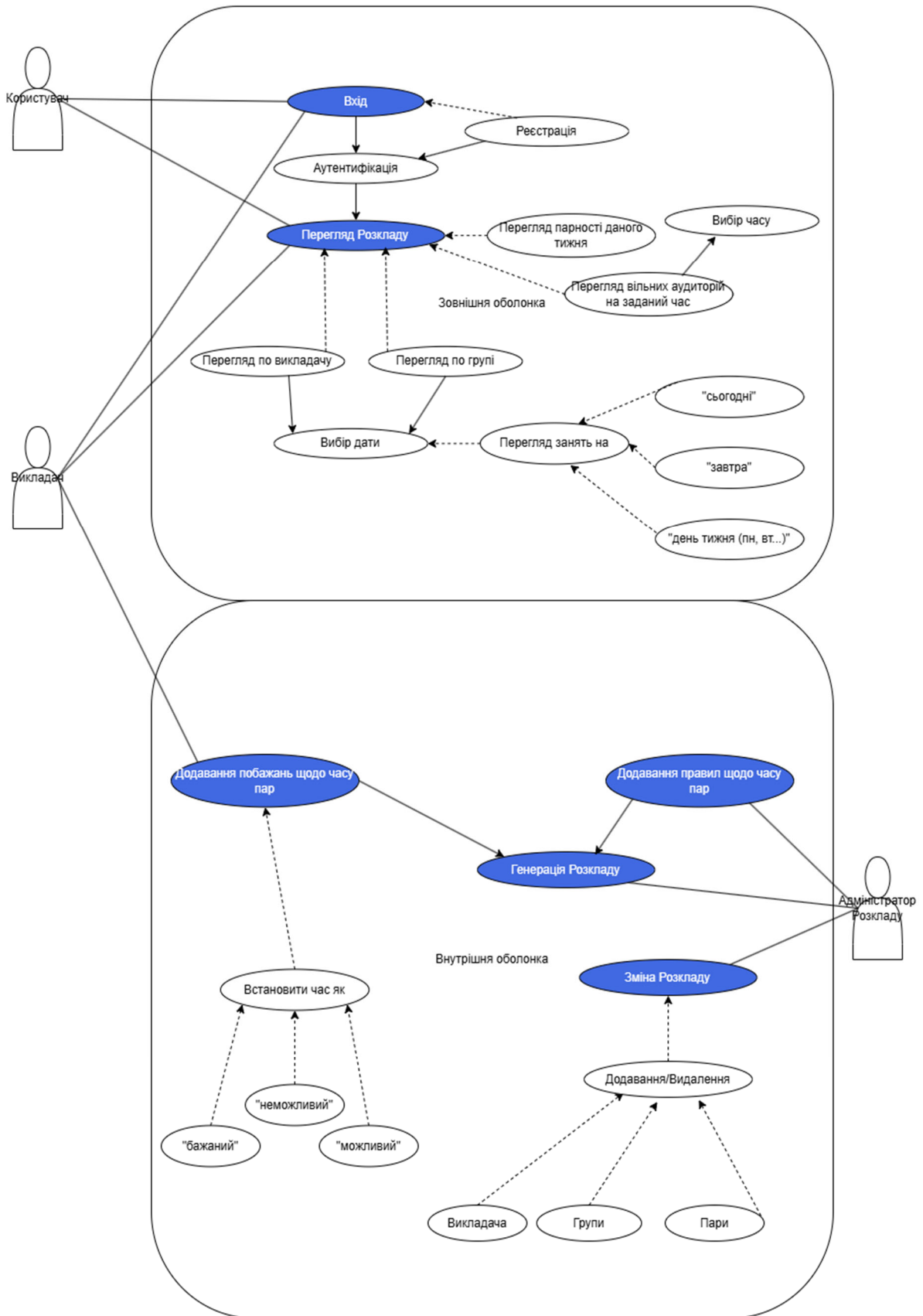


Рис. 2. Діаграма Use-Case

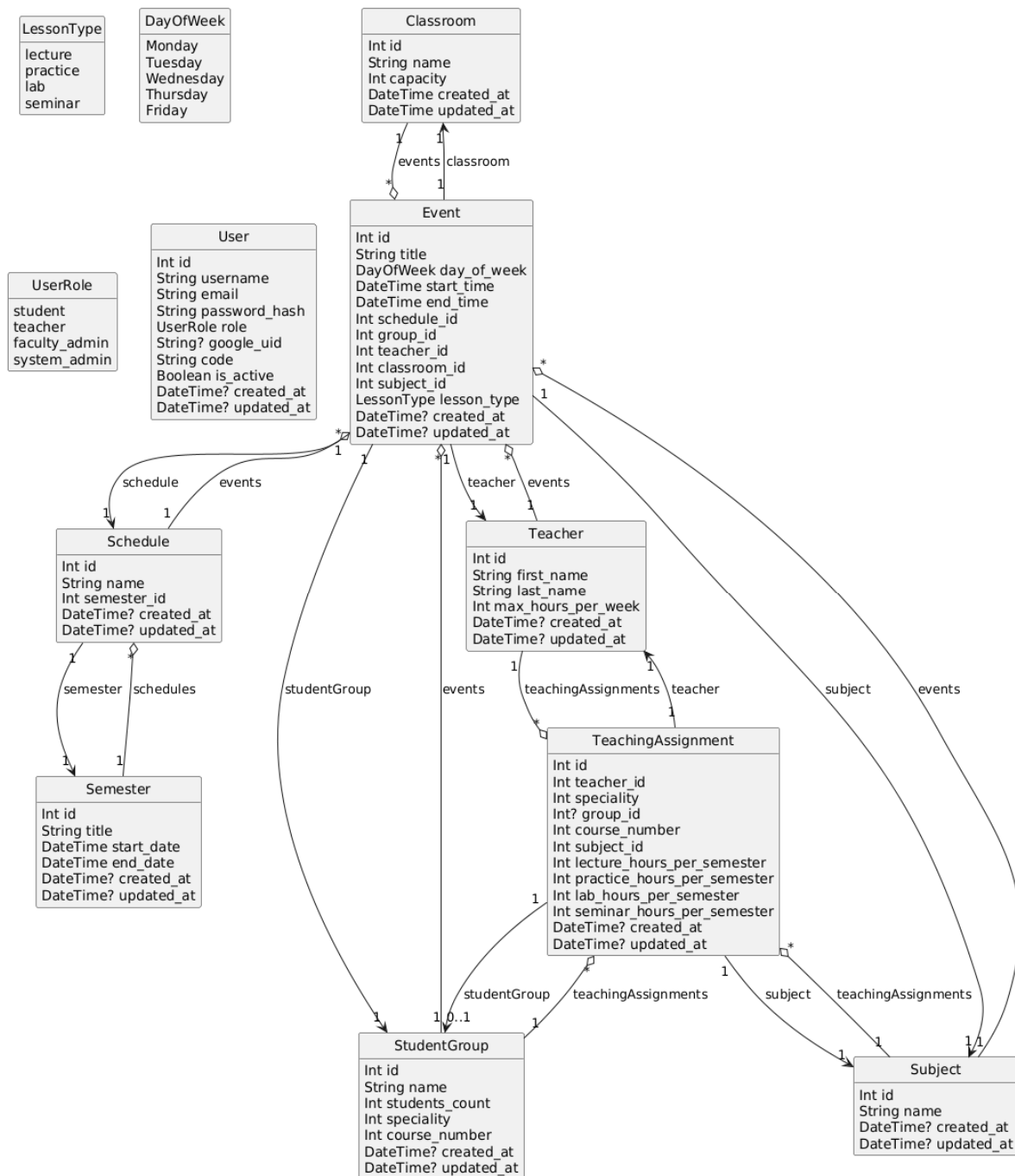


Рисунок 3 – діаграма класів

Логічна структура бази даних. У таблиці 8 описано структуру всіх елементів бази даних, що використовується в проєкті.

UI/UX дизайн. В роботі використовуються різні закони дизайну [10], зокрема:

- простота взаємодії (Закон Паркінсона) – у таблиці розкладу показано лише базову інформацію – предмети для кожного дня й уроку;
- принцип доступності елементів (Закон Фіттса) – основні дії розташовані вгорі сторінки у вигляді великих кнопок із кольоровим маркуванням. Їхній розмір і контрастність сприяють легкому знаходженню та використанню;

- послідовність (Consistency) – кнопки на верхній панелі мають єдиний стиль, що допомагає користувачеві орієнтуватися в інтерфейсі без необхідності переучування;
- когнітивне перевантаження (Закон Міллера) – інтерфейс не містить надлишкових елементів чи інформації, завдяки чому фокус користувача спрямований лише на виконання основних завдань.

Таблиця 8. Логічна структура бази даних

Сутність	Опис	Зв'язки
Classroom	Містить дані про аудиторії (назва, місткість, дати створення/оновлення)	1-N з Event
Event	Заняття: день тижня, час, посилання на розклад/групу/викладача/предмет	N-1 з Schedule, StudentGroup, Teacher, Classroom, Subject
Schedule	Розклад для конкретного семестру	1-N з Event; N-1 з Semester
Semester	Інформація про семестр (назва, дати початку/закінчення)	1-N з Schedule
StudentGroup	Дані про студентську групу (назва, курс, спеціальність, к-сть студентів)	N-1 з Event; 1-N з TeachingAssignment
Subject	Навчальний предмет (назва тощо)	1-N з Event та TeachingAssignment
Teacher	Викладачі (ПІБ, максимальна к-сть годин)	1-N з Event та TeachingAssignment
User	Аутентифікація та ролі (student, teacher)	Опційно зв'язаний з Teacher (для додаткової логіки)
TeachingAssignment	Призначення викладача на предмет (години, тип заняття, група/курс)	N-1 з Teacher і Subject; опційно з StudentGroup

3. ІМПЛЕМЕНТАЦІЯ ПРОДУКТУ/СИСТЕМИ ТА ТЕСТУВАННЯ

Архітектура застосунку складається з бази даних MySQL, backend-сервісу реалізованому за допомогою мови програмування TypeScript, а також фронтенд застосунку, який взаємодіє з бекенд сервісом через REST API [11]. На рис 4 зображена схема бази даних, використана в продукті.

Для взаємодії з фронтенд застосунком реалізовано набір REST API, які реалізують CRUD операції з аудиторіями, навчальними дисциплінами, групами та викладачами, а також створено API ендпоінт для виклику генерації розкладу.

Генерація розкладу

Додаток використовує генетичний алгоритм, щоб автоматично формувати розклад із дотриманням бізнес-правил:

Збір початкових даних. За допомогою функції `getInitialData()` завантажується вся необхідна інформація з бази даних (семестри, групи, предмети, викладачі, аудиторії, призначення).

Початкова перевірка. Переконається, що семестр існує, що достатньо аудиторій для розміщення всіх груп тощо. Перевірка `assignGroupsToTeachingAssignments()` коригує призначення (`TeachingAssignment`), якщо потрібна група ще не прив'язана або не існує.

Запуск генетичного алгоритму. Створюється початкова популяція можливих розкладів (`WeeklySchedule`). У межах `fitness function` враховуються обмеження: не перевищувати 4 пар на день, не дублювати аудиторію чи викладача на той самий час, не перевищувати максимальних годин викладача тощо.

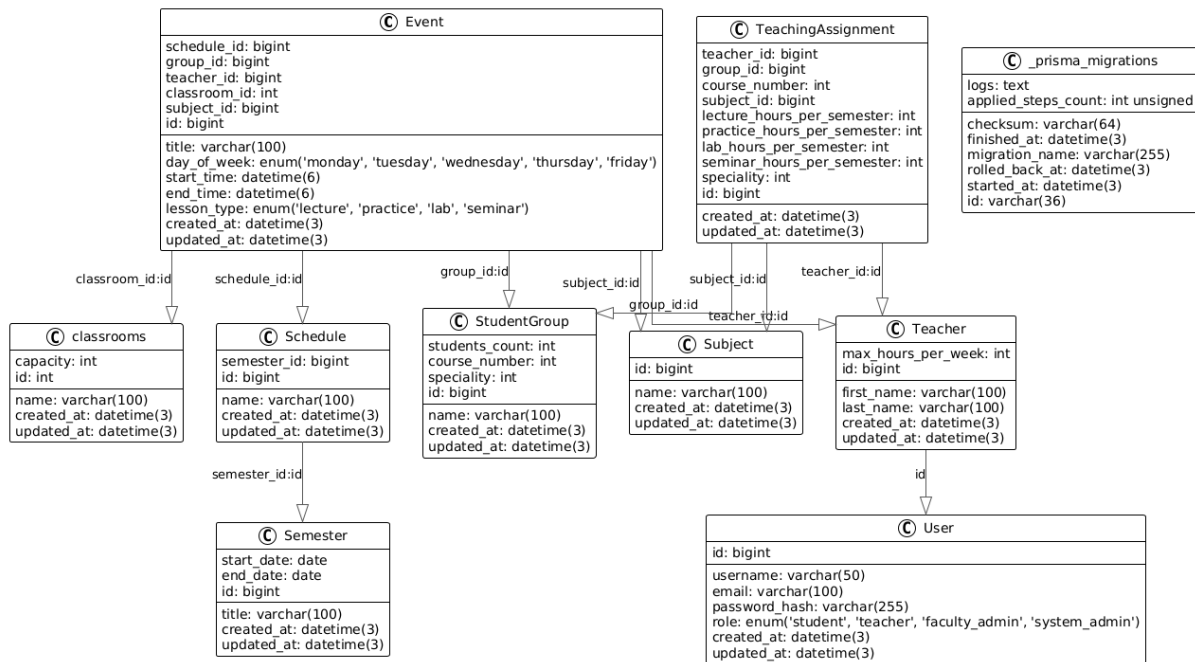


Рис. 4. UML схема бази даних.

Еволюція (відбір, схрещування, мутація). Поступово відбираються найкращі розклади, схрещуються та зазнають мутацій (наприклад, перестановка пар, додавання чи вилучення подій). Кожне покоління оцінюється з точки зору fitness. Чим менше конфліктів і краща відповідність годинам семестру – тим вище fitness.

Отримання найкращого тижневого розкладу. Після визначеної кількості поколінь береться найкращий індивідуум. Це і буде тижневий графік занять.

Розширення розкладу на весь семестр. Метод `expandWeeklyScheduleToSemester` генерує конкретні дати подій, використовуючи прив'язку дня тижня, з урахуванням початку й кінця семестру.

Збереження результатів. Створюється запис `Schedule` для даного семестру. За допомогою `createEventsForSchedule()` усі події (`Event`) додаються до бази даних (через `Prisma`). Кінцевий розклад повертається як `ScheduleDto`, придатний для відображення чи подальших CRUD-операцій.

Авторизація

В застосунку реалізована авторизація в особистий кабінет для зручного налаштування. Для додаткового захисту використовується мультифакторна автентифікація.

Авторизація за допомогою email-адреси та пароля. Вхід до системи за допомогою адреси електронної пошти разом із паролем є традиційним, але надійним способом ідентифікації користувача. Даний метод базується на створенні унікального облікового запису, до якого прикріплюється пароль. Важливо, що система ніколи не зберігає пароль у відкритому вигляді – замість цього використовується криптографічне хешування.

Під час реєстрації новий користувач проходить наступний ланцюжок дій:

- заповнює форму реєстрації, вводячи свою електронну адресу та придуманий пароль;
- система одразу ж обробляє пароль, застосовуючи криптографічний алгоритм хешування (у цьому випадку – `SHA-256`), і генерує незворотній хеш;

- отриманий хеш, разом з іншими даними, записується у відповідну таблицю в базі даних;

- на вказану електронну пошту надсилається лист з кодом активації;
- лише після введення цього коду користувачеві надається повноцінний доступ до облікового запису – обліковий запис активується.

Процедура входу виконується за схожим принципом:

- користувач вводить email та пароль у форму авторизації;
- введений пароль хешується знову, після чого порівнюється із раніше збереженим хешем у базі;

- якщо обидва значення збігаються, та обліковий запис активовано, система генерує унікальний JWT-токен – цифровий ключ, що підтверджує особу користувача впродовж сесії.

Авторизація через Google OAuth. Ще одним сучасним і зручним способом входу є автентифікація за допомогою стороннього сервісу – наприклад, облікового запису Google. Такий підхід реалізований на основі протоколу OAuth 2.0 і дозволяє користувачам увійти без введення пароля. Система в такому разі довіряє Google як сторонньому постачальнику ідентифікації [12].

Етапи автентифікації виглядають так:

- користувач натискає кнопку авторизації через Google на інтерфейсі;
- після вибору облікового запису, Google пересилає спеціальний токен – свідчення того, що користувач успішно пройшов перевірку;
- сервер отримує цей токен і звертається до Google, щоб підтвердити його дійсність;
- якщо користувач із таким email уже є в базі, генерується звичний JWT-токен, як у стандартному вході;
- якщо ж користувача ще не існує, система створює новий запис, використовуючи дані з Google-профілю.

Зберігання даних користувачів, що увійшли до системи. Уся інформація про зареєстрованих та авторизованих користувачів зберігається в базі даних, керованій через ORM Prisma [13].

Захист паролів через криптографічне хешування. Важливою складовою інформаційної безпеки є захист паролів. У системі не допускається зберігання паролів у явному вигляді. Замість цього кожен пароль проходить через алгоритм хешування SHA-256, реалізований через модуль crypto. Під час автентифікації введений користувачем пароль знову обробляється аналогічно і порівнюється із записом у базі. Таким чином, навіть у випадку витіку даних, зловмисники не матимуть доступу до реальних паролів.

Обробка одночасного використання різних способів входу. Іноді користувач спочатку реєструється через email, а згодом вирішує скористатися входом через Google. Щоб уникнути дублювання акаунтів, система передбачає перевірку: чи існує обліковий запис з тією ж email-адресою. Якщо так – до існуючого запису просто додається ідентифікатор google_uid.

Таким чином, користувач зберігає доступ до своїх даних незалежно від способу входу.

Система підтвердження та управління активністю акаунтів. Щоб гарантувати, що лише справжні користувачі можуть скористатися сервісом, запроваджено механізм підтвердження акаунтів. Кожен обліковий запис створюється у неактивному стані й активується тільки після верифікації електронної адреси.

Захист від входу неактивованих користувачів. Користувачі, які не завершили процедуру підтвердження, не допускаються до системи. Це реалізовано в методі login, який перевіряє поле is_active перед генерацією токена. У випадку невідповідності система повідомляє про помилку типу "Inactive account" або "Invalid credentials".

Взаємодія з Excel файлами

У рамках командного проєкту було реалізовано модуль, який дозволяє завантажувати Excel-файли через веб інтерфейс, зчитувати з них структуровані дані та зберігати ці дані у базі даних. Основна мета – автоматизація процесу додавання навчального навантаження викладачів на основі Excel-документів.

Короткий опис роботи. Функціонал включає реалізацію бекенду та фронтенду для:

- завантаження Excel-файлу користувачем через веб інтерфейс;
- обробки цього файлу на сервері за допомогою бібліотеки ExcelJS;
- зчитування з файлу даних про викладачів, предмети, курси та кількість годин за різними типами занять;
- збереження зчитаних даних у базу даних через відповідні сервіси;
- формування зручного JSON-подання зчитаних даних.

Реалізація бекенду. Виконаний на основі NestJS, бекенд містить:

Контролер (ExcelparserController)

● *POST /excelparser/upload* – Цей метод обробляє завантаження Excel-файлу. Після отримання файлу він передається в сервіс ExcelparserService, який займається парсингом;

● *GET /excelparser/teacher-data* – Повертає JSON-файл із збереженими даними (для подальшого експорту до бази даних);

● *GET /excelparser/upload* – Повертає HTML-сторінку з інтерфейсом для завантаження файлу.

Сервіс (ExcelparserService)

- завантажує Excel-файл у пам'ять за допомогою бібліотеки ExcelJS;
- зчитує прізвище та ім'я викладача із 6-го рядка, 4-ї колонки;
- починаючи з 18-го рядка, по черзі зчитуються предмети, курси, типи занять та відповідна кількість годин;
- також перевіряється наявність коду спеціальності або назви факультету.

Дані групуються та структуруються в об'єкти з полями:

```
{  
  "teacherName": "Ім'я викладача",  
  "subject": "Назва предмету",  
  "courseNumber": "Номер курсу",  
  "lec": 0,  
  "lab": 0,  
  "sem": 0,  
  "pract": 0,  
  "specialityCode": "123"  
}
```

Після завершення парсингу ці дані по черзі передаються в сервіси:

- *teachersService.importFromJson(...)*
- *subjectsService.importFromJson(...)*
- *teachingAssignmentsService.importFromJson(...)*

Кожен із цих сервісів зберігає відповідні об'єкти у базу даних.

Обробка особливих випадків. Якщо рядок містить назву факультету або спеціальності, він пропускається, але з нього може бути зчитано код спеціальності. Рядки, що містять "Всього", слугують маркером закінчення таблиці. Підтримується мапінг типів занять:

- *лек.* → *lec*

- *лаб.* → *lab*
- *сем.* → *sem*
- *практ.* → *pract*

Фронтенд. Інтерфейс функціоналу реалізований за допомогою React та бібліотеки Material UI (MUI) для створення сучасного та адаптивного інтерфейсу користувача. Основна мета інтерфейсу – надати зручний спосіб перегляду, редагування та імпорту даних з Excel-файлів щодо навчальної програми. Користувач має доступ до:

- інтерфейсу сторінки завантаження Excel (upload.html);
- інтерфейсу React-компонента AssignmentsComponent.

Компонент AssignmentsComponent відповідає за управління таблицею з інформацією про предмети, викладачів, курси та види занять (лекції, лаби, семінари, практики). Основні можливості:

Інтерфейс підтримує наступні CRUD-операції:

- додавання нового рядка – через модальне вікно, яке відкривається після натискання кнопки "Додати рядок".
- редагування існуючих записів – відкривається той самий діалог з попередньо заповненими полями.
- видалення – із підтвердженням користувача.
- перевірка правильності введених даних (валідація) – наприклад, курс має бути числом від 1 до 6.

Імпорт з Excel. Користувач може завантажити файл у форматі .xlsx або .xls. Парсинг Excel-файлів здійснюється за допомогою бібліотеки xlsx (XLSX.js). Дані перетворюються у масив об'єктів ProgramData та виводяться у таблицю.

Відображення таблиці. Інформація про навчальну програму відображається у вигляді таблиці з такими стовпцями: *Предмет, Викладач, Курс, Група, Кількість лекцій, Лабораторних, Семінарів, Практичних, Дії (редагування/видалення)*.

При відсутності даних виводиться повідомлення "Дані відсутні".

Використання Material UI. Компонент побудований з використанням наступних компонентів з бібліотеки MUI: *Button, TextField, Dialog, Typography, Table, TableCell, TableContainer, TableRow, Paper, IconButton, Іконки Add, Edit, Delete з @mui/icons-material*.

Це дозволяє створити інтуїтивно зрозумілий інтерфейс із підтримкою тем, адаптивності та доступності.

Стилізація (CSS). Для оформлення інтерфейсу використовується кастомна таблиця стилів, яка реалізує такі блоки:

- .schedule-page – головна сторінка з розкладом (layout);
- .sidebar – навігаційне меню;
- .schedule-content – основна частина з таблицею;
- .schedule-table – блок з таблицею та заголовком;
- .modal-box, .header, .pagination – стилізація модального вікна, кнопок та пагінації.

Адаптивність та UX. Інтерфейс адаптований до різних розмірів екранів та підтримує зручну взаємодію завдяки:

- відповідному розміщенню елементів (flex, Box, Grid);
- використанню діалогових вікон замість перенаправлень;
- мінімалізованому дизайну з інтуїтивними кнопками дій;
- перевірці даних перед збереженням.

Реалізований Excel-парсер значно спрощує імпорт навчального навантаження в систему. Він забезпечує автоматизацію обробки складних Excel-файлів, створення та

оновлення відповідних записів у базі даних та простий у використанні вебінтерфейс для завантаження файлів.

Цей функціонал може масштабуватись для обробки інших типів документів або форматів даних у майбутньому.

Модульне тестування

У цьому розділі описані модульні, інтеграційні та E2E тести, які були виконані для перевірки коректності роботи API, а також описані виконані функціональні автоматизовані тести інтерфейсу користувача (UI), реалізовані за допомогою фреймворку Playwright [15]. Тести покривають базові сценарії навігації, завантаження даних, коректності відображення сторінок та верифікації ключових елементів інтерфейсу. Основна мета – переконатися, що дані з API завантажуються коректно, а компоненти фронтенду працюють згідно очікувань.

Були розроблені наступні тести:

- отримання всіх груп: Тест перевіряє, чи правильно повертаються всі групи з бази даних. Результат: тест пройдено, всі групи отримані коректно;
- отримання групи за ID: Тест перевіряє, чи повертається помилка NotFoundException, якщо група з вказаним ID не знайдена. Результат: тест пройдено, виключення згенеровано очікувано;
- створення нової групи: Тест перевіряє коректність створення нової групи. Результат: тест пройдено, групу створено коректно;
- генерація початкового розкладу на тиждень: Перевіряється, чи функція generateRandomWeeklySchedule успішно створює розклад із подіями. Для кожної події перевіряються ключові властивості: назва, день тижня, часовий слот (що потрапляє у діапазон доступних слотів), група, викладач, предмет, аудиторія, та тип заняття;
- обчислення фітнес-функції: Перевіряється коректність функції calculateFitness. Для згенерованого розкладу обчислюється фітнес, і тест перевіряє, що результат є числом;
- робота генетичного алгоритму: Функція runGeneticAlgorithm тестується на коректне виконання. Очікується, що алгоритм повертає розклад із подіями, який має фітнес, обчислений згідно з фітнес-функцією;
- перевірка унікальності занять для викладачів: Перевіряється, чи кожен викладач проводить лише одне заняття одночасно. Використовується функція checkScheduleConflicts, яка перевіряє конфлікти розкладу для викладачів;
- перевірка унікальності занять для груп: Перевіряється, чи кожна група відвідує лише одне заняття одночасно. Тест аналогічний попередньому, але перевіряє конфлікти розкладу для студентських груп;
- перевірка унікальності використання аудиторій: Перевіряється, чи кожна аудиторія використовується лише для одного заняття одночасно. Знову використовується checkScheduleConflicts, але цього разу для перевірки аудиторій.

Тести збірки. Було розроблено E2E тести збірки дозволили перевірити роботу системи як єдиного цілого, включаючи модулі GroupsModule і SchedulesModule.

При прогоні E2E тестів для груп отримали такі результати:

- отримання списку груп через API – список груп повернуто коректно;
- створення нової групи через API – групу створено успішно;
- помилка при отриманні неіснуючої групи через API – результат: повернуто очікувану помилку 404.

А E2E тести для розкладів показали наступні відповіді:

- створення нового семестру через API – семестр створено успішно;
- генерація порожнього розкладу через API – порожній розклад згенеровано успішно;

● помилка при генерації розкладу для неіснуючого семестру через API – повернуто очікувану помилку 404.

Параметри тестування маршруту / (головної сторінки) наступні:

- назва тесту: [Basic] - / route ;
- мета: Перевірити, що після входу головна сторінка завантажується успішно і містить ключовий елемент – верхню панель (AppBar);
- очікуваний результат: Елемент [data-testid="appbar"] присутній на сторінці;
- результат: Тест пройдено успішно. Головна сторінка відображається, панель навігації доступна.

А параметри тестування сторінки груп /groups такі:

- назва тесту: [Groups Page] - [Data Loading] - Basic Test;
- мета: Перевірити завантаження сторінки груп, а також отримання даних з API;
- сценарій:
 - навігація на сторінку /groups;
 - очікування відповіді від API на запит groups;
 - перевірка статусу відповіді (200 або 304);
 - перевірка наявності першого рядка таблиці;
 - перевірка видимості верхньої панелі.
- результат: Тест пройдено. Дані груп успішно завантажуються, таблиця відображається, інтерфейс працює стабільно.

Тестування сторінки класів /classes відбувалось таким чином:

- назва тесту: [Classes Page] - [Data Loading] - Basic Test ;
- мета: Перевірити, що дані про класи коректно завантажуються з API;
- кроки:
 - після автентифікації перейти на сторінку /classes;
 - очікувати запит до API classrooms;
 - перевірити, що запит повертає статус 200 або 304;
 - перевірити видимість першого рядка таблиці.
- результат: Тест виконано успішно, сторінка завантажується, дані класів доступні.

Тестування сторінки дисциплін /disciplines показало наступні дані:

- назва тесту: [Disciplines Page] - [Data Loading] - Basic Test ;
- мета: Верифікація завантаження сторінки та успішного запиту до API subjects;
- результат: API відповідає з кодом 200 або 304. Дані успішно завантажуються, елементи таблиці доступні на сторінці.

Тестування сторінки викладачів /teachers дало такі результати:

- назва тесту: [Teachers Page] - [Data Loading] - Basic Test ;
- мета: Переконаватися, що сторінка викладачів відображається коректно, і дані з API (teachers) доступні;
- результат: Всі перевірки пройдені, таблиця з викладачами рендериться, дані отримані з API.

Артефакти тестування: відео та звіти. Для забезпечення повної прозорості процесу тестування та зручного аналізу результатів, усі тести виконуються з увімкненим записом відео. Це дозволяє у разі помилки або збоїв наочно побачити, як саме відбувався сценарій і на якому етапі виникла проблема.

В процесі були помічені наступні особливості відеофіксації:

- відео записується автоматично для кожного окремого тесту;
- кожен файл має унікальну назву, яка відповідає назві тесту, що значно спрощує навігацію серед записів;

• у разі помилки відео надає повну картину контексту проблеми (наприклад, затримка відповіді API, неправильний рендеринг, або помилки інтерфейсу).

Генерація тестових звітів. Після запуску всієї тестової сесії автоматично генерується структурований звіт у зручному HTML-форматі, що включає в себе:

- назви всіх виконаних тестів;
- статус кожного тесту (пройшов/не пройшов);
- посилання на відповідне відео;
- повідомлення про помилки, якщо такі були.

Звіт дозволяє швидко проаналізувати результати, навіть не переглядаючи самі відеозаписи, якщо всі тести пройдені успішно, як показано на рис. 5.

Project: chromium 13.04.2025, 15:23:10 Total time: 16.9s

▼ e2e/Classes/classes.spec.ts	
✓ [Classes Page] - [Data Loading] - Basic Test › [Data Loading] - Page loads and fetches data	8.3s
e2e/Classes/classes.spec.ts:16 ▶	
▼ e2e/Common/common.spec.ts	
✓ Basic Test › [Basic] - / route	1.3s
e2e/Common/common.spec.ts:17 ▶	
▼ e2e/Disciplines/disciplines.spec.ts	
✓ [Disciplines Page] - [Data Loading] - Basic Test › [Data Loading] - Page loads and fetches data	1.3s
e2e/Disciplines/disciplines.spec.ts:16 ▶	
▼ e2e/Groups/groups.spec.ts	
✓ [Groups Page] - [Data Loading] - Basic Test › [Data Loading] - Page loads and fetches data	1.4s
e2e/Groups/groups.spec.ts:16 ▶	
▼ e2e/Teachers/teachers.spec.ts	
✓ [Teachers Page] - [Data Loading] - Basic Test › [Data Loading] - Page loads and fetches data	1.4s
e2e/Teachers/teachers.spec.ts:16 ▶	

Рис. 5. Результат тестування у формі html звіту.

Усі тести було успішно пройдено.

4. ДОКУМЕНТАЦІЯ ПРОЄКТУ

Модель ведення документації та інструменти. Проєкт використовує гібридний підхід, поєднуючи переваги Agile методу і класичної технічної документації. Більшість документації введеться і оновлюється разом з розробкою продукту протягом його життєвого циклу. Для контролю змін ми застосовуємо систему версій.

Для створення і збереження документів використовувались вбудовані інструменти Jira, зокрема Confluence [16-17].

Внутрішня документація (для команди розробки). У внутрішню документацію входять:

• концептуальна модель системи – містить діаграму класів, яка відображає основні сутності системи та їх взаємозв'язки. Використовується при проєктуванні архітектури;

- use-case діаграма – описує всі можливі сценарії взаємодії користувачів із системою. Включає основні та альтернативні потоки подій;
 - вимоги до продукту – детальний документ, що містить функціональні та нефункціональні вимоги до системи. Включає технічні специфікації та обмеження;
 - тестові дані – документ, що містить інформацію про тестове середовище, набори тестових даних та сценарії тестування;
 - матриця відповідальностей (RACI) – визначає ролі та відповідальності членів команди за різні аспекти проекту.
- Зовнішня документація.** Під це визначення потрапляють:
- інструкції для користувачів – детальний посібник з використання системи, призначений для кінцевих користувачів (викладачів, студентів, адміністраторів);
 - API документація – технічний опис програмного інтерфейсу для розробників, які інтегрують систему з іншими сервісами;
 - презентація продукту – матеріали для демонстрації можливостей системи потенційним клієнтам та інвесторам;
 - статут проекту – офіційний документ, що визначає цілі, принципи та правила роботи над проектом;
 - Lean Canvas – бізнес-модель продукту, яка описує ключові аспекти проекту: проблему, рішення, метрики успіху тощо.

Інструкція користувача

При вході на основну сторінку (див. рис б), необхідно обрати семестр і групу аби відобразити збережений розклад.

Час	Понеділок	Вівторок	Середа	Четвер	П'ятниця
08:40 - 09:25	WEB-технології Антон Свістунов 205 - Лекція	Теорія програмування Олена Шишацька 204 - Практичне заняття	Системне програмування Тарас Панченко 306 - Лекція	Теорія програмування Олена Шишацька 306 - Лекція	
09:30 - 10:15	WEB-технології Антон Свістунов 205 - Лекція	Теорія програмування Олена Шишацька 204 - Практичне заняття	Системне програмування Тарас Панченко 306 - Лекція	Теорія програмування Олена Шишацька 306 - Лекція	
10:35 - 11:20	Системне програмування Андрій Шишацький 304 - Лабораторне заняття		Теорія програмування Андрій Криволап 308 - Практичне заняття		
11:25 - 12:10	Системне програмування Андрій Шишацький 304 - Лабораторне заняття		Теорія програмування Андрій Криволап 308 - Практичне заняття		
12:20 - 13:05	WEB-технології Антон Свістунов 305 - Семинар	Системне програмування Тарас Панченко 309 - Лекція			
13:10 - 13:55	WEB-технології Антон Свістунов 305 - Семинар	Системне програмування Тарас Панченко 309 - Лекція			
14:05 - 14:50					
14:55 - 15:40					

Рис. 6. Веб-сторінка з завантаженим розкладом

На цій сторінці можна переглянути розклад або згенерувати його за наявною інформацією. У меню зліва можна перейти на інші сторінку, що Вас цікавлять.

На кожній з веб-сторінок можна виконувати базові CRUD-дії:

- додавати інформацію;
- редагувати інформацію;
- видаляти інформацію.

У формі додавання/редагування (див. рис 7), достатньо ввести дані і натиснути кнопку "ОК"; або натиснути кнопку "відмінити", щоб вийти з форми не вносячи змін.

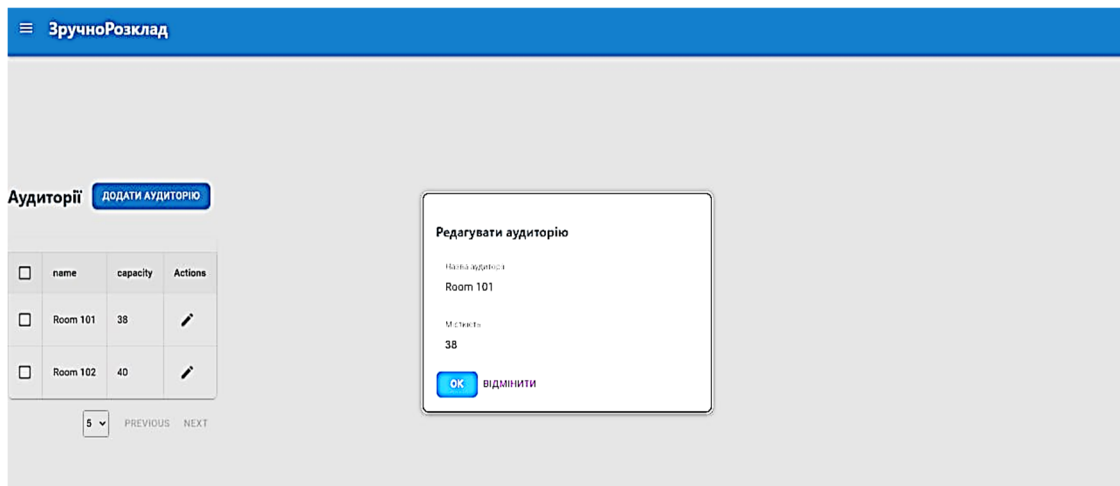


Рис. 7. Форма редагування даних

Також є можливість відредагувати побажання для викладачів у відповідному вікні, як показано на рис. 8.

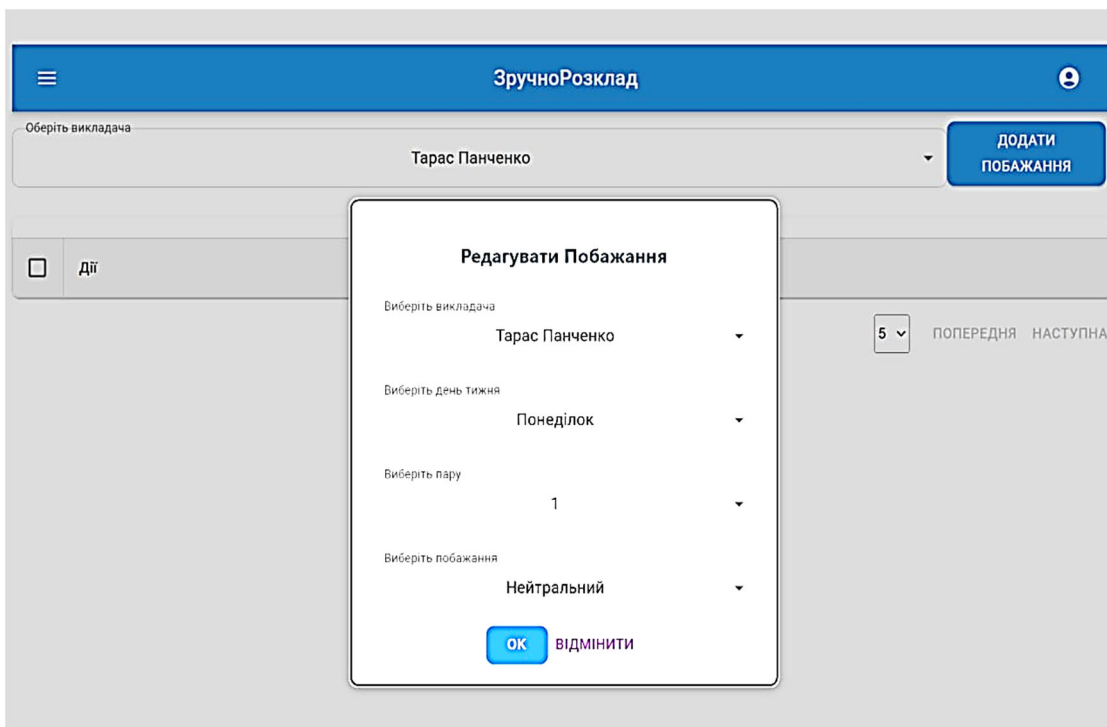


Рис. 8. Форма редагування побажань

Аби згенерувати розклад, необхідно виконати наступні кроки:

- перейдіть на сторінку "Навчальний план" і натисніть на кнопку "Завантажити з Excel" (див. рис 9);
- оберіть файл з навчальним планом для якого Ви бажаєте згенерувати розклад;
- перейдіть на сторінку "Групи" і додайте їх через заповнення відповідної форми (див. рис 10);
- повторіть дану процедуру з "Аудиторіями";
- за необхідності, додайте побажання викладачів (див. рис 11);
- перейдіть на сторінку "Розклад" (основна сторінка) і додайте/оберіть семестр для якого планується генерація розкладу;
- натисніть на кнопку "Розрахувати розклад" і після успішного генерування Ви можете обрати групу і переглянути створений для неї розклад!

ЗручноРозклад

Групи ДОДАТИ ГРУПУ

<input type="checkbox"/>	Назва групи	Кількість студентів	Курс	Спеціальність	Дії
<input type="checkbox"/>	ТТП-32	29	3	122	
<input type="checkbox"/>	К-14	35	1	122	
<input type="checkbox"/>	ТТП-41	29	4	122	
<input type="checkbox"/>	К-24	0	2	122	
<input type="checkbox"/>	Ф-21	33	2	35	
<input type="checkbox"/>	ІПС-11	32	1	121	
<input type="checkbox"/>	К-10	34	1	113	
<input type="checkbox"/>	Ф-11	36	1	35	
<input type="checkbox"/>	Ф-31	32	3	35	

5 ПОПЕРЕДНЯ НАСТУПНА

Рис. 9. Завантажений навчальний план

Дисципліна	Викладач	Курс	Група	Лекції	Лабораторні	Семінари	Практичні	Дії
Системне програмування	Тарас Панченко	3	ТТП-32	42	28	0	0	
Сучасні операційні системи	Тарас Панченко	1	К-14	26	10	0	0	
Методи паралельних обчислень	Анатолій Дорошенко	4	ТТП-41	42	0	0	0	
Логіка і Автоматизоване мислення/Logic and the Automated Deduction	Микола Нікітченко	2	К-24	14	0	12	0	
Математична логіка і теорія алгоритмів	Микола Нікітченко	2	Ф-21	30	0	14	0	
OK.05 Теорія і технологія розробки програмних систем (англійською мовою)	Микола Нікітченко	1	ІПС-11	10	0	0	6	
OK.05 Mathematical Foundations and Technology of Software Systems and Algorithms	Микола Нікітченко	1	К-10	16	0	0	0	
Математична логіка	Степан Шкільський	2	К-24	88	0	0	0	

Рис. 10. Вигляд сторінки "Групи"

<input type="checkbox"/>	День тижня	Пара	Побажання	Дії
<input type="checkbox"/>	Monday	1	Обов'язково вільний	

5 ПОПЕРЕДНЯ НАСТУПНА

Рис. 11. Вкладка "Побажань"

Вищевказані дії є універсальними і працюють як на комп'ютері так і на мобільному пристрої до тих пір, поки пристрій відповідає системним вимогам.

Правові норми

Проєкт "ЗручноРозклад" як програмний продукт для управління навчальним процесом університетів розроблений з урахуванням чинного законодавства України.

Закон України "Про вищу освіту". Відповідно до статей 10 та 32 Закону навчальні заклади мають автономію у питаннях організації освітнього процесу [18].

Університети самостійно визначають:

- структуру навчального року;
- формування розкладів занять;
- вибір інформаційних систем для підтримки навчального процесу.

Цей закон є фундаментальним для нашого проекту, оскільки він дозволяє давати рекомендації університетам стосовно графіку навчального процесу і який дійсно може бути впроваджений.

Закон України "Про захист персональних даних". Ми реалізували механізми захисту персональних даних відповідно до вимог статей 6 та 24 [19].

Система передбачає:

- авторизований доступ до даних;
- шифрування конфіденційної інформації;
- ведення журналів доступу;
- обробка даних здійснюється лише в межах навчального процесу.

Закон України "Про авторське право і суміжні права". Відповідно до статей 1, 6 та 24 [20]:

- програмний код та інтерфейс системи є об'єктом авторського права;
- розробники зберігають виключні майнові права на продукт;
- використання системи регулюється ліцензійною угодою.

5. УПРАВЛІННЯ ПРОЕКТОМ

Розробкою займалася команда із 7 студентів:

- **Денис Астаф'єв** – тимлідер, full-stack розробник;
- **Сергій Халік** – аналітик, фахівець з документації;
- **Костянтин Дементьєв** – бекенд розробник;
- **Арсеній Сагалянов** – фронтенд/бекенд розробник;
- **Олександр Піщанецький** – фронтенд розробник;
- **Діана Сушко** – тестувальник;
- **Євгенія Бажмаєва** – бекенд розробник.

Методологія. Нашою командою було обрано модель Scrum-Ban. Це гібридна модель, яка поєднує переваги Scrum та Kanban. Вона дозволяє поєднувати структуровані ітерації з гнучким управлінням потоками робіт. Ключові особливості включають візуалізацію задач на дошці, регулярні стендапи та адаптивне планування.

Ми обрали цю модель завдяки її гнучкості і легкості в використанні. Оскільки в нашій команді більшість людей не працювала над великими проектами раніше, вибір Scrum-Ban дозволив нам легко втягнутися в процес.

Інструментарій. Основні інструменти що використовувались:

- Jira – використовується для трекінгу задач, створення дорожніх карт та управління проектом;
- Google Meet – застосовується для проведення зустрічей [21];
- Confluence – служить централізованим сховищем для всієї документації;
- Draw.io – використовувався для розробки UI/UX макетів та прототипування [22].

Спринти проводилися щотижня, на них члени команди обговорювали виконані задачі і планували наступні відповідно до загального прогресу.

Рефлексія. Наша команда успішно реалізувала проект "ЗручноРозклад", незважаючи на низку викликів. Ось ключові моменти, які можна відзначити:

- позитивне – наша команда повністю виконала поставлений план. Усі необхідні кроки були виконані якісно;
- негативне – під час роботи над проєктом виникали розбіжності у навантаженні членів команди у певний момент часу. Рішення – ми активно допомагали одне одному аби пришвидшити процес;
- чому навчилися – найбільшим здобутком став важливий досвід роботи в команді. Група активно вивчала нові технології, досліджувала різноманітні аспекти створення і ведення проєкту.

ВИСНОВКИ

Виконання даного проєкту дозволило створити комплексну інформаційну систему для організації та управління навчальними розкладами, яка отримала назву "ЗручноРозклад". Протягом роботи було реалізовано всі основні етапи проєктування: від визначення вимог та створення юзкейсів до розробки користувацького інтерфейсу, тестування та підготовки документації. Система поєднує функціональність для адміністраторів і кінцевих користувачів, дозволяючи автоматизувати ключові процеси, такі як створення семестрів, додавання уроків та управління групами. Значний акцент було зроблено на забезпеченні інтуїтивного UX-дизайну, що спрощує взаємодію користувачів із продуктом.

Розробка системи дала змогу практично освоїти основи Scrum-Ban, навчила нас багатьом важливим аспектам, серед яких виділяється інтеграція теоретичних знань із практичною реалізацією. Ми розширили свої вміння у створенні документації, навчилися правильно структурувати вимоги до програмного забезпечення та коректно визначати функції системи.

Ще одним важливим результатом стало вивчення інструментів для командної розробки. Завдяки використанню Jira ми навчилися планувати та відстежувати виконання завдань, аналізувати прогрес і своєчасно коригувати плани. Scrum-Ban показав себе як ефективна методологія, яка дає змогу одночасно адаптуватися до змін і підтримувати стабільність у процесах.

Окрім технічних аспектів, ми прагнемо розвивати систему, орієнтуючись на зручність адміністраторів і кінцевих користувачів. Регулярне проведення опитувань і збору відгуків допоможе виявляти потреби користувачів і вчасно впроваджувати необхідні покращення. Ми також маємо намір запровадити автоматичне оновлення системи для швидкої адаптації до нових вимог і потреб.

Загалом, реалізація проєкту стала важливим етапом у нашому професійному зростанні. Вона допомогла нам не лише розвинути технічні вміння, а й зрозуміти важливість командної роботи, управління проєктами та орієнтації на потреби кінцевих користувачів. Ми сподіваємося, що система "ЗручноРозклад" стане корисним інструментом для навчальних закладів і дозволить значно спростити процес організації навчального процесу.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Система розкладу UNITIME. URL: <https://www.unitime.org/> (дата звернення: 20.04.2025).
2. Освітня система "ЄДИНА ШКОЛА". URL: <https://eschool-ua.com/#/> (дата звернення: 20.04.2025).

3. Бібліотека React. URL: <https://react.dev/> (дата звернення: 20.04.2025).
4. Бібліотека MUI. URL: <https://mui.com/> (дата звернення: 20.04.2025).
5. Бібліотека Axios. URL: <https://axios-http.com/> (дата звернення: 20.04.2025).
6. Orval Codegen. URL: <https://orval.dev/> (дата звернення: 20.04.2025).
7. Фреймворк NestJS. URL: <https://nestjs.com/> (дата звернення: 20.04.2025).
8. Фреймворк Jest. URL: <https://jestjs.io/ru/> (дата звернення: 20.04.2025).
9. База даних MySQL. URL: <https://www.mysql.com/> (дата звернення: 20.04.2025).
10. Sharoval Agency. ТОП-7 законів у UX-дизайні. Cases.media. 24.08.2022. URL: <https://cases.media/en/article/top-7-zakoniv-u-ux-dizaini> (дата звернення: 20.04.2025).
11. Redhat. What is Rest API?. redhat.com. 08.05.2020. URL: <https://www.redhat.com/en/topics/api/what-is-a-rest-api#:~:text=A%20REST%20API%20is%20an,should%20build%20a%20web%20API> (дата звернення: 20.04.2025).
12. Протокол Google OAuth. URL: <https://datatracker.ietf.org/doc/html/rfc6749> (дата звернення: 20.04.2025).
13. ORM Prisma. URL: <https://www.prisma.io/> (дата звернення: 20.04.2025).
14. Бібліотека ExcelJS. URL: <https://github.com/exceljs/exceljs> (дата звернення: 20.04.2025).
15. Фреймворк Playwright. URL: <https://playwright.dev/> (дата звернення: 20.04.2025).
16. Про вищу освіту: Закон України від 01.07.2014 № 1556-VII. Відомості Верховної Ради України. 2014. № 37–38, № 39. Ст. 2004. URL: <https://zakon.rada.gov.ua/laws/show/1556-18> (дата звернення: 20.04.2025).
17. Про захист персональних даних: Закон України від 01.06.2010 № 2297-VI. Відомості Верховної Ради України. 2010. № 34. Ст. 481. URL: <https://zakon.rada.gov.ua/laws/show/2297-17> (дата звернення: 20.04.2025).
18. Про авторське право і суміжні права: Закон України від 23.12.1993 № 3792-XII. Відомості Верховної Ради України. 1994. № 13. Ст. 64. URL: <https://zakon.rada.gov.ua/laws/show/3792-12> (дата звернення: 20.04.2025).
19. Інструмент менеджменту проєктів Jira. URL: <https://www.atlassian.com/software/jira> (дата звернення: 20.04.2025).
20. Інструмент менеджменту проєктів Confluence. URL: <https://www.atlassian.com/software/confluence> (дата звернення: 20.04.2025).
21. Google Meet. URL: <https://meet.google.com/landing> (дата звернення: 20.04.2025).
22. draw.io. URL: <https://app.diagrams.net/> (дата звернення: 20.04.2025).

МОБІЛЬНИЙ ЗАСТОСУНОК ДЛЯ СПРОЩЕННЯ ДОГЛЯДУ ЗА ДОМАШНІМИ ТВАРИНАМИ

*Єгор БАЧИНСЬКИЙ, Наталія ГРИЦИШИН, Аріна ЗАБРОДІНА,
Олександр КОРНІЄНКО, Катерина ЛИСЕНКО, Максим МАРКОВ,
Наталія СЕМЙОН, Антон ТІХОНОВ*

Ця робота описує процес розробки мобільного застосунку PetCare, що допомагає власникам домашніх тварин керувати інформацією про улюбленців: зберігати медичну історію, отримувати нагадування про щеплення та візити до ветеринара, створювати профіль тварини. Застосунок створено з акцентом на зручність, простоту інтерфейсу та турботу про тварин. У процесі реалізації було враховано потреби цільової аудиторії, використано сучасні технології та підходи до побудови інтуїтивно зрозумілих мобільних сервісів.

МОБІЛЬНИЙ ЗАСТОСУНОК, БАЗА ДАНИХ, REACT NATIVE, EXPO, API, UI/UX ДИЗАЙН, SWAGGER UI, ДОМАШНІ ТВАРИНИ.

MOBILE APPLICATION FOR SIMPLIFYING PET CARE

Yehor BACHYNSKYI, Nataliia HRYTSYSHYN, Arina ZABRODINA, Oleksandr KORNIIENKO, Kateryna LYSENKO, Maksym MARKOV, Nataliia SEMION, Anton TIKHONOV

This work describes the development process of the PetCare mobile application, which helps pet owners manage information about their animals: storing medical history, receiving reminders about vaccinations and vet visits, creating pet profiles. The application is designed with a focus on convenience, simplicity of the interface, and care for pets. The needs of the target audience were taken into account during the implementation, and modern technologies and approaches to building intuitive mobile services were used.

MOBILE APPLICATION, DATABASE, REACT NATIVE, EXPO, API, UI/UX DESIGN, SWAGGER UI, PETS.

ВСТУП

Актуальність роботи та підстави для її виконання. Сучасний світ орієнтується на технологічні інновації, які спрощують повсякденне життя людей. Одна з недостатньо розв'язаних проблем стосується власників домашніх тварин, які стикаються з потребою у зручних інструментах для догляду за своїми улюбленцями, моніторингу їхнього здоров'я та планування важливих заходів, наприклад, вакцинації чи візити до ветеринара. Розробка мобільного застосунку для власників тварин є актуальною, оскільки дозволяє об'єднати всі необхідні функції в одному рішенні, покращуючи якість догляду та заощаджуючи час користувачів.

Мета роботи. Метою цього проєкту є розробка мобільного застосунку, який допоможе власникам тварин у догляді за їхніми улюбленцями. Застосунок має забезпечувати:

- зручний доступ до профілів тварин;
- інтерфейс для введення та перегляду медичних даних;
- календар з нагадуваннями про важливі події (візити до ветеринара, вакцинації тощо).

Окрім досягнення прикладних цілей, розробка цього проєкту дозволить закріпити знання з програмування мобільних застосунків, поглибити навички роботи з базами даних, API та інструментами UI/UX-дизайну. Це сприятиме підвищенню професійної компетенції команди, знайомству з сучасними технологіями та їхнім ефективним застосуванням у реальних проєктах.

Методи дослідження. У рамках розробки було використано такі методи:

- аналіз ринку: дослідження рішень наявних на ринку та визначення їхніх сильних і слабких сторін;
- проєктування: створення архітектури застосунку з урахуванням потреб користувачів і технічних вимог;
- розробка: покрокова реалізація функціоналу із застосуванням C#, React Native, PostgreSQL, Swagger UI;
- тестування: перевірка роботи застосунку для забезпечення надійності й зручності використання.

1. ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

Вибір сфери для створення продукту. Під час обговорення сфери, у якій буде створено продукт, команда орієнтувалася на сучасні тенденції та реальні потреби користувачів. Особливу увагу було приділено повсякденним викликам, із якими стикаються люди, а також можливості запропонувати рішення, що спростить їхнє життя. Однією з таких сфер є догляд за домашніми тваринами, що набуває все більшої популярності в суспільстві.

Сфера догляду за домашніми тваринами була обрана з огляду на такі чинники:

- постійний ріст кількості власників домашніх тварин;
- відсутність універсального інструменту для управління інформацією про тварин і догляд за ними;
- популярність мобільних застосунків як способу вирішення рутинних завдань.

Перелік обговорюваних ідей. Під час пошуку ідеї продукту команда розглядала кілька можливих напрямків.

Застосунок для догляду за домашніми тваринами. Його основні функції мали б бути наступними:

- управління медичними даними, нагадування про вакцинації та візити до ветеринара;
- інтеграція з картами для пошуку клінік та інших важливих локацій;
- профілі тварин із можливістю збереження інформації про них.

Застосунок "Battle Royal" для проведення батлів між учнями та підготовки до тестів. Цей застосунок міг би бути орієнтований на учнів, які готуються до контрольних, тестів або зовнішнього незалежного оцінювання (ЗНО/НМТ). Формат Battle Royal дозволяє зробити навчання більш інтерактивним і захопливим завдяки змаганням у реальному часі. Основні функції:

- випадкове чи вибіркоче формування суперників;
- серії запитань з різних предметів із лічильником часу;
- система балів, досягнень і лідербордів;
- можливість створення приватних батлів між друзями чи однокласниками.

Застосунок "єСтудент+" для організації студентської спільноти. Ця ідея була спрямована на створення платформи для студентів, яка б поєднувала функції планування, комунікації та соціальної взаємодії. Основні функції:

- організація розкладу пар, подій та дедлайнів;
- групові чати для предметів або гуртків;
- можливість створення спільнот за інтересами (наприклад, спортивні, творчі або наукові клуби).

Такий застосунок допоміг би студентам краще організувати своє навчання і водночас створити міцнішу спільноту.

Застосунок для вивчення іноземних мов. Ідея цього застосунку полягала у створенні інструменту для вивчення іноземних мов з унікальним підходом до навчання. Основні функції:

- персоналізовані уроки залежно від рівня знань і мети користувача;
- інтерактивні вправи;
- ігрові елементи для закріплення матеріалу (мініігри, челенджі, бали та досягнення).

Аргументація вибору ідеї

Кожна з цих ідей мала свій потенціал і відповідала певній аудиторії, проте після обговорення команда вирішила обрати застосунок для власників домашніх тварин. Основними аргументами для цього вибору стали:

- актуальність потреби: багато власників тварин натрапляли на труднощі у плануванні догляду та збереженні медичної інформації, що створює попит на такий продукт;
- відсутність комплексного рішення: хоча на ринку існують подібні застосунки, більшість із них мають обмежений функціонал або складний інтерфейс;
- реалістичність реалізації: проєкт має чітко визначені межі, що дозволяє команді сконцентруватися на створенні MVP, не перевантажуючи продукт зайвими функціями;
- суспільна користь: застосунок сприятиме покращенню догляду за тваринами, підвищуючи їхній добробут і здоров'я.

Таким чином, ідея розробки мобільного застосунку для власників тварин об'єднала бажання створити корисний та технологічно досяжний продукт, що відповідатиме потребам сучасних користувачів.

Огляд наявних на ринку IT-продуктів - конкурентів

Застосунок для ветклінік та ветеринарів "JetVetPass" [1] - це електронний ветеринарний паспорт і органайзер для власників тварин. Застосунок надає змогу зберігати дані про тварину, історію візитів до ветеринара, планувати раціон і витрати, отримувати нагадування. Цільова аудиторія - власники домашніх тварин. Основна географія - Україна та країни СНД. JetVetPass підтримує українську, англійську та російську мови. Продукт є повністю безкоштовним, без варіантів платної підписки. Застосунок доступний для Android та iOS. Кількість завантажень застосунку на місяць - менше ніж 5 тис. JetVetPass має вебверсію з трафіком 20 167 візитів щомісяця [2]. Сторінок у соціальних мережах або блогу сервіс не має. Серед переваг: інтеграція з ветеринарними клініками та зручне зберігання медичних даних. Недоліки: технічні проблеми, обмежений функціонал.

Застосунок "MyPet" [3] - цифровий щоденник для власників тварин із широким функціоналом: облік медичних записів, щеплень, ліків, витрат, фотогалереї, а також моніторинг харчування та активності тварини. Основна аудиторія - власники тварин у США та Європі. Мовна локалізація - лише англійська. Модель монетизації - freemium: базовий функціонал безкоштовний, додаткові можливості доступні за підпискою.

Додаток доступний для Android та iOS. Кількість завантажень на місяць - менше ніж 5 тис. Платформа має вебверсію з трафіком 631 візит на місяць [2]. MyPet активний у соціальних мережах та веде блог. Переваги: зручний інтерфейс та широкий функціонал. Недоліки: відсутність локалізації для інших мов та відсутність інтеграції з клініками.

Програмне забезпечення PetCare by DaySmart Vet. PetCare - це частина платформи DaySmart Vet [4], яка надає власникам тварин доступ до медичних записів, функцію бронювання зустрічей з ветеринаром, оплату рахунків і перегляд документів. Продукт орієнтований на користувачів у США. Мовна локалізація - англійська. Модель монетизації - безкоштовний доступ для клієнтів клінік, які підключені до DaySmart Vet. Додаток доступний для Android та iOS. Щомісячна кількість завантажень - менше 5 тис. Вебверсія має високий трафік - 201 393 відвідування на місяць [2]. PetCare представлений у Facebook та LinkedIn, веде блог. Переваги: глибока інтеграція з ветеринарними клініками, простий інтерфейс, онлайн-запис. Недоліки: обмеження доступу лише для клієнтів підключених клінік, відсутність повноцінного функціоналу для інших користувачів.

Мапа продукту

Мапа продукту була розроблена за допомогою шаблону Lean Canvas (рис. 1 та 2)



Рис. 1. Lean Canvas продукту

Бізнес-модель

У рамках реалізації нашого мобільного застосунку використовується B2C-модель (Business-to-Consumer), яка передбачає безпосередню взаємодію з кінцевими споживачами - власниками домашніх тварин. Основна мета цієї моделі полягає в наданні корисного цифрового інструменту, що допоможе користувачам дбати про здоров'я та

добробут своїх улюбленців. Застосунок орієнтований на широку аудиторію людей, які мають домашніх тварин і потребують зручного способу для зберігання інформації про них, нагадувань про важливі події та організації догляду.

Обрана модель дозволяє налагодити ефективну комунікацію з кінцевими користувачами, отримувати зворотний зв'язок та адаптувати функціонал відповідно до їхніх потреб. Крім того, B2C-модель створює сприятливі умови для масштабування продукту, залучення нових користувачів через цифровий маркетинг, соціальні мережі та партнерські програми. Такий підхід сприяє швидкому зростанню користувацької бази та формуванню спільноти навколо продукту.



Рис. 2. Lean Canvas продукту

Сильні і слабкі сторони розроблюваного продукту. Для їхньої оцінки скористалися SWOT-аналізом.

Сильні сторони (Strengths):

- інтеграція ключових функцій: застосунок об'єднує профіль тварини, медичну картку, календар нагадувань, що забезпечує зручність для користувачів;
- інтуїтивно зрозумілий дизайн: акцент на простоті використання та доступності для широкої аудиторії;
- високий попит: зростання кількості власників домашніх тварин збільшує потребу у цифрових інструментах для догляду за ними;

- широка аудиторія: застосунок підходить для власників тварин різного віку, статі та рівня технічної грамотності;

- автоматизація рутинних процесів: нагадування та збереження даних зменшують час, витрачений на організацію догляду.

Слабкі сторони (Weaknesses):

- обмеження MVP: початковий функціонал може здатися недостатнім для деяких користувачів, які очікують більш розширених можливостей;

- конкуренція: на ринку вже існують схожі продукти, хоча й без комплексного підходу;

- ресурси для підтримки: необхідність регулярного оновлення бази даних (наприклад, ветеринарних клінік або аптек).

Можливості (Opportunities):

- розширення функціоналу: включення нових можливостей, наприклад, інтеграція з картами та ветклініками, статистика активності тощо;

- партнерства: співпраця з ветеринарними клініками, магазинами для тварин або страховими компаніями;

- розширення аудиторії: охоплення ринку в інших країнах або регіонах, враховуючи глобальну популярність домашніх тварин;

- мобільний маркетинг: активне просування через соціальні мережі, орієнтоване на молодь та молоді родини;

- гейміфікація: додавання ігрових елементів (наприклад, досягнень за регулярний догляд за твариною).

Загрози (Threats):

- конкуренція: на ринку можуть з'явитися сильні конкуренти з більш розширеним функціоналом або безкоштовними послугами;

- складність монетизації: знайти баланс між функціональністю безкоштовної версії та платного доступу;

- технічні складнощі: можливі проблеми з інтеграцією карт, API або підтримкою на різних платформах.

SWOT-аналіз демонструє, що мобільний застосунок для власників домашніх тварин має значний потенціал, однак його успішність залежатиме від правильної стратегії розвитку, врахування слабких сторін і активного реагування на зовнішні загрози.

Огляд використаних технологій. Для розробки мобільного застосунку для власників домашніх тварин було обрано низку сучасних технологій, які забезпечують ефективність, зручність розробки та гнучкість подальшого розвитку продукту.

C# (ASP.NET Core). C# [5] - це сучасна мова програмування, яку широко використовують для створення високопродуктивних застосунків і серверних компонентів. У поєднанні з ASP.NET Core [6] вона забезпечує можливість розробки масштабованого і надійного бекенду.

Обґрунтування вибору:

- простота інтеграції з іншими технологіями та сторонніми API;

- висока продуктивність та ефективність;

- велика спільнота розробників і доступна документація;

- підтримка кросплатформності.

React Native та Expo. React Native [7] - це бібліотека для розробки кросплатформених мобільних застосунків з використанням JavaScript та React. React Expo [8] - це інструментарій та сервісна платформа для React Native, яка спрощує процес розробки та тестування.

Обґрунтування вибору:

- кросплатформеність: Одночасна розробка для iOS та Android економить час та ресурси;

- швидкість розробки: Гаряче перезавантаження та широкий набір готових компонентів прискорюють процес;

- спільнота та підтримка: Велика спільнота забезпечує швидке вирішення проблем та доступ до численних бібліотек.

TypeScript. TypeScript [9] - це мова програмування, яка є надбудовою над JavaScript та додає статичну типізацію.

Обґрунтування вибору:

- надійність коду: Статична типізація допомагає виявляти помилки на етапі компіляції;

- масштабованість: Полегшує підтримку та розширення коду в міру зростання проєкту;

- інтеграція з інструментами: Покращує автодоповнення та рефакторинг коду.

HTML та CSS. HTML і CSS [10] - базові технології для створення структурованого та стильового оформлення інтерфейсу.

Обґрунтування вибору:

- необхідність уніфікованої розмітки для інтеграції з React;

- легкість налаштування зовнішнього вигляду;

- гнучкість для створення адаптивного дизайну, що зручно для різних пристроїв.

Swagger UI. Swagger UI [11] - це інструмент для документування RESTful API, який дозволяє автоматизувати процес тестування та розуміння API.

Обґрунтування вибору:

- полегшує взаємодію між розробниками фронтенду та бекенду;

- зручність у створенні інтерактивної документації для API;

- підвищує продуктивність розробки та тестування.

UI/UX-дизайн інструменти (Figma). Інструменти для створення дизайну інтерфейсу, зокрема Figma [12] допомагають на етапі прототипування та планування.

Причини вибору:

- візуалізація і тестування дизайну ще до початку розробки;

- можливість спільної роботи команди над прототипами;

- адаптивний підхід до різних платформ і розмірів екранів.

Axios. Axios [13] - це бібліотека для здійснення HTTP-запитів.

Обґрунтування вибору:

- зручність: Проста в використанні API;

- можливості: Підтримка перехоплення запитів/відповідей, скасування запитів;

- популярність: Широко використовується та має хорошу документацію.

React Query (@tanstack/react-query). React Query [14] - бібліотека для управління станом асинхронних даних та кешування.

Обґрунтування вибору:

- ефективність: Автоматичне кешування та оновлення даних;

- простота: Зменшує необхідність вручну писати код для управління станом запитів;

- поліпшення користувацького досвіду: Менше затримок при завантаженні даних.

React Hook Form та Zod. Це інструменти для управління формами та їх валідації [15].

Обґрунтування вибору:

- продуктивність: Мінімізує кількість перерендерів компонентів;

- зручність: Легке управління станом форми;

- валідація: Zod забезпечує декларативну, просту та зручну валідацію даних.

Tailwind CSS та NativeWind. Tailwind CSS [16] та NativeWind [17] - це інструменти для стилізації застосунку.

Обґрунтування вибору:

- швидкість розробки: Утилітарний підхід дозволяє швидко створювати адаптивні дизайни;

- консистентність: Забезпечує єдиний стиль через весь застосунок;

- гнучкість: Легко налаштовується під потреби проекту.

i18next та react-i18next. Бібліотеки [18] для інтернаціоналізації застосунку.

Обґрунтування вибору:

- масштабованість: Підтримка багатьох мов та динамічного перемикавання;

- простота інтеграції: Легка настройка з React;

- спільнота: Активно підтримується та оновлюється.

React Native Async Storage. React Native Async Storage [19] - бібліотека для зберігання даних на пристрої.

Обґрунтування вибору:

- стійкість даних: Зберігає дані між сесіями застосунку;

- зручність: Простий API для роботи з ключ-значеннями;

- сумісність: Працює як на iOS, так і на Android.

ESLint та Prettier. ESLint [20] та Prettier [21] - це інструменти для аналізу та форматування коду.

Обґрунтування вибору:

- якість коду: Виявлення потенційних помилок та недоліків;

- єдиний стиль: Забезпечує консистентний стиль написання коду;

- автоматизація: Можливість інтеграції з CI/CD для автоматичної перевірки.

Jest та Testing Library. Jest [22] та Testing Library [23] - інструменти для тестування додатку.

Обґрунтування вибору:

- надійність: Забезпечує написання детермінованих та ізольованих тестів;

- простота: Зрозумілий синтаксис та легка настройка;

- покриття коду: Допомогає виявляти недоліки та запобігати регресіям.

OpenAPI Generator CLI (@openapitools/openapi-generator-cli). OpenAPI Generator CLI [24] – це інструмент для автоматичної генерації клієнтського коду API на основі специфікацій OpenAPI.

Обґрунтування вибору:

- автоматизація: Зменшує ручну роботу з написання клієнтського коду для API, що економить час і знижує ймовірність помилок;

- синхронізація: Забезпечує, що клієнтський код завжди актуальний та відповідає останній версії API;

- стандартизація: Використання OpenAPI як стандарту опису API полегшує взаємодію між командами бекенду та фронтенду;

- гнучкість: Підтримує генерацію коду для різних мов та фреймворків, що може бути корисним у майбутньому;

- інтеграція: Легко інтегрується з існуючим процесом розробки через npm-скрипти, як-от команда generate-api.

2. ОПИС ПРОГРАМНОГО ПРОДУКТУ

Призначення і цілі створення системи

Мобільний застосунок створюється з метою надання власникам домашніх тварин зручного інструменту для управління інформацією про своїх улюбленців.

Система призначена для:

- збереження та обробки даних про домашніх тварин, приміром, медична картка, історія вакцинацій, прийоми у ветеринара;
- організації нагадувань про важливі події, включно з візитами до клініки, вакцинаціями та іншими аспектами догляду за твариною;
- відображення важливих локацій на карті, наприклад, ветеринарні клініки, магазини для тварин, тренувальні центри тощо.

Система також покликана допомогти власникам:

- ефективно організувати догляд за тваринами, мінімізуючи ризики пропуску важливих подій;
- мати всю необхідну інформацію під рукою в будь-який момент, наприклад, під час візиту до ветеринара;
- зекономити час завдяки автоматизованим нагадуванням та спрощеному доступу до необхідної інформації.

Основні цілі створення застосунку зображені на Use Case діаграмі на рис. 3.

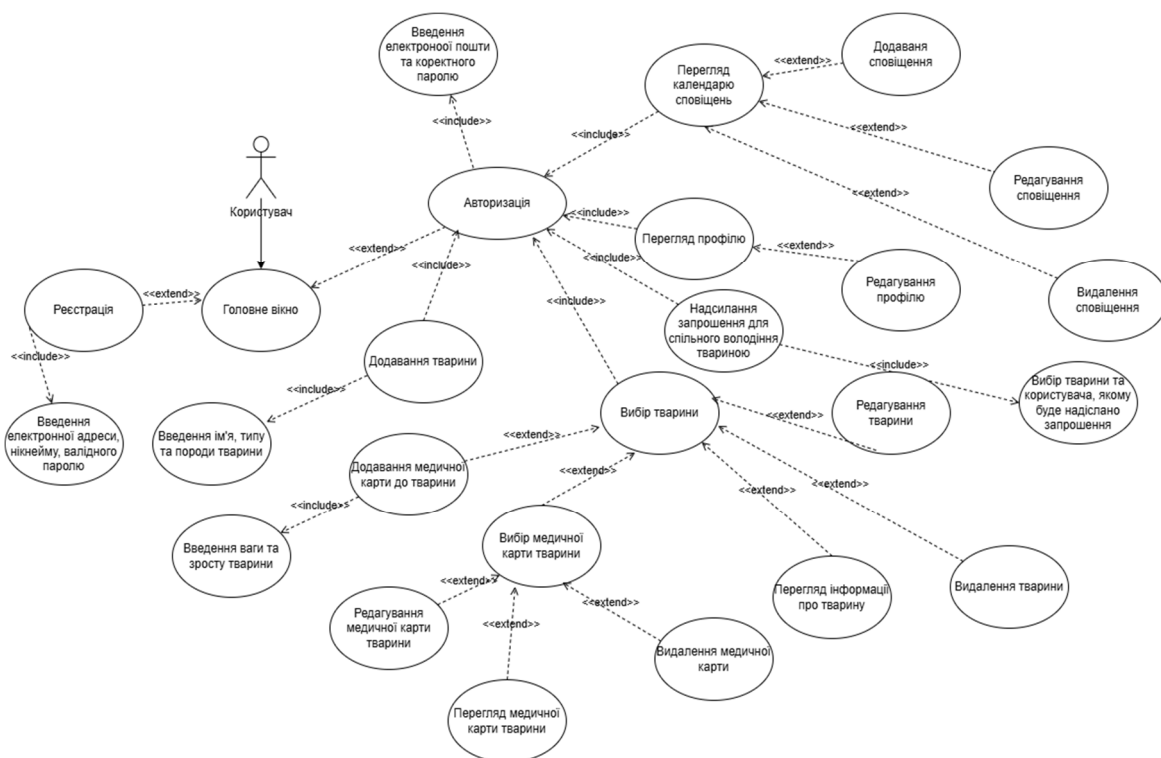


Рис. 3. Use Case діаграма застосунку

Застосунок створюється з метою:

- забезпечення збору, обробки та аналізу інформації про стан здоров'я тварини, історію вакцинацій та прийоми у ветеринара;
- автоматизації процесу планування догляду за твариною за допомогою календаря нагадувань;
- створення доступу до даних у зручному цифровому форматі, щоб власники могли використовувати цю інформацію в різних ситуаціях;
- підвищення рівня організації та якості догляду за тваринами.

Основні цілі створення системи мобільного застосунку для власників домашніх тварин:

- забезпечити зручне управління профілями тварин: надати можливість власникам швидко додавати та редагувати інформацію про своїх улюбленців, зберігаючи важливі дані, такі як ім'я, порода, вік та історія здоров'я;
- створити систему для збереження та перегляду медичних карток тварин: забезпечити власникам доступ до детальної інформації про вакцинації, прийоми у ветеринарів та інші важливі записи про здоров'я тварини;
- забезпечити автоматичні нагадування: реалізувати функцію нагадувань про важливі події;
- забезпечити інтуїтивно зрозумілий інтерфейс: створити користувацький інтерфейс, який дозволить легко та швидко отримувати доступ до необхідної інформації, редагувати дані та налаштовувати функції.

Вимоги до системи

Вимоги до системи в цілому

Система повинна забезпечити зручний і ефективний спосіб управління інформацією про домашніх тварин, їхній медичний стан, календар догляду та важливі локації для власників тварин. Застосунок має бути доступний на мобільних пристроях і повинен мати інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам швидко виконувати необхідні дії: додавання профілю тварини, перегляд медичних записів і налаштування нагадувань.

Основні вимоги до системи:

- простота у використанні: інтерфейс повинен бути зрозумілим навіть для користувачів з мінімальним досвідом у використанні мобільних застосунків;
- масштабованість: система повинна підтримувати можливість додавання нових функцій у майбутньому, таких як інтеграція з іншими сервісами або додаткові модулі для клінік.

Вимоги до структури та функціонування системи

Система мобільний застосунок для власників домашніх тварин "PET-Care" повинна містити кілька ключових функціональних підсистем для забезпечення ефективного управління та використання даних.

Адміністративна підсистема призначена для адміністрування системи та управління даними. Вона повинна містити функції:

- управління користувачами: можливість додавання нових користувачів, редагування їхніх даних;
- модерация та перегляд медичних карток: адміністратори можуть переглядати та редагувати записи про здоров'я тварин, додавати нові дані;
- налаштування системних функцій: налаштування параметрів нагадувань і інтерфейсу.

Підсистема користувача призначена для власників тварин, щоб мати доступ до всіх функцій мобільного застосунку. Вона повинна містити:

- створення та редагування профілю тварини: користувач може додавати нових тварин до свого профілю, редагувати інформацію (ім'я, порода, дата народження тощо);
- медична картка тварини: збереження даних про історію вакцинацій, результати аналізів, результати оглядів та інші медичні записи;

- календар нагадувань: користувач може додавати події, такі як графік харчування, вакцинації, візити до ветеринара, а також отримувати сповіщення про них;
- перегляд карт локацій: можливість переглядати та знаходити ветеринарні клініки, магазини для тварин та інші важливі локації.

Вимоги до функцій, які виконуються системою

У таблиці 1 та таблиці 2 наведено перелік функцій, які наявні у підсистемах адміністратора та користувача відповідно.

Таблиця 1. Перелік функцій, задач які наявні у підсистемі адміністратора

Функція	Задача
Керувати роботою користувачів системи	Редагування та видалення інформації про користувачів системи (власників тварин).
Робота з медичними картками тварин	Перегляд та редагування медичних записів тварин. Адміністратор може додавати нові медичні дані, переглядати історію вакцинацій, лікування та інші записи.
Управління функціями нагадувань	Налаштування параметрів нагадувань для користувачів, таких як вакцинація, візити до ветеринара та інші події.

Таблиця 2. Перелік функцій, задач, які наявні у підсистемі користувача

Функція	Задача
Реєстрація	Введення інформації про власника тварини: електронна адреса та пароль
Оновлення профілю користувача	Можливість редагування профілю, додавання нової тварини, зміна контактних даних.
Створення профілю тварини	Введення інформації про тварину: ім'я, порода, дата народження, стать, медичні записи та історія вакцинацій.
Можливість поділитися тваринкою	Можливість надіслати доступ до профілю тваринки іншому користувачу.
Система сповіщень	Сповіщення про доступ до інших тварин
Календар нагадувань	Додавання та перегляд подій у календарі (наприклад, вакцинація, візити до ветеринара). Отримання сповіщень про важливі події.
Перегляд локацій	Можливість переглядати на карті важливі локації: клініки, магазини для тварин та інші ресурси.
Взаємодія з іншими власниками тварин	Можливість спілкуватися або обмінюватися досвідом з іншими користувачами застосунку.

Системні вимоги

Мобільні платформи: мінімальна версія IOS 13.4 та Android 6.0

На довільні некоректні дії користувача, пов'язані з введенням некоректних даних, незаповненням обов'язкових полів введення в формах та інші, які можуть бути оброблені системою, генеруються відповідні повідомлення про помилки українською або англійською мовою, в межах загального дизайну застосунку.

Операційна система: IOS, Android

Джерелом даних для Системи повинна бути інформаційна система під управлінням СКБД PostgreSQL.

Опис організації інформаційної бази

Логічна структура бази даних. Під час розроблення застосунку використовується СКБД PostgreSQL. На рис. 4 зображена частина діаграми бази даних, яка відповідає за збереження профілю тварини, її медичної картки та запрошень.

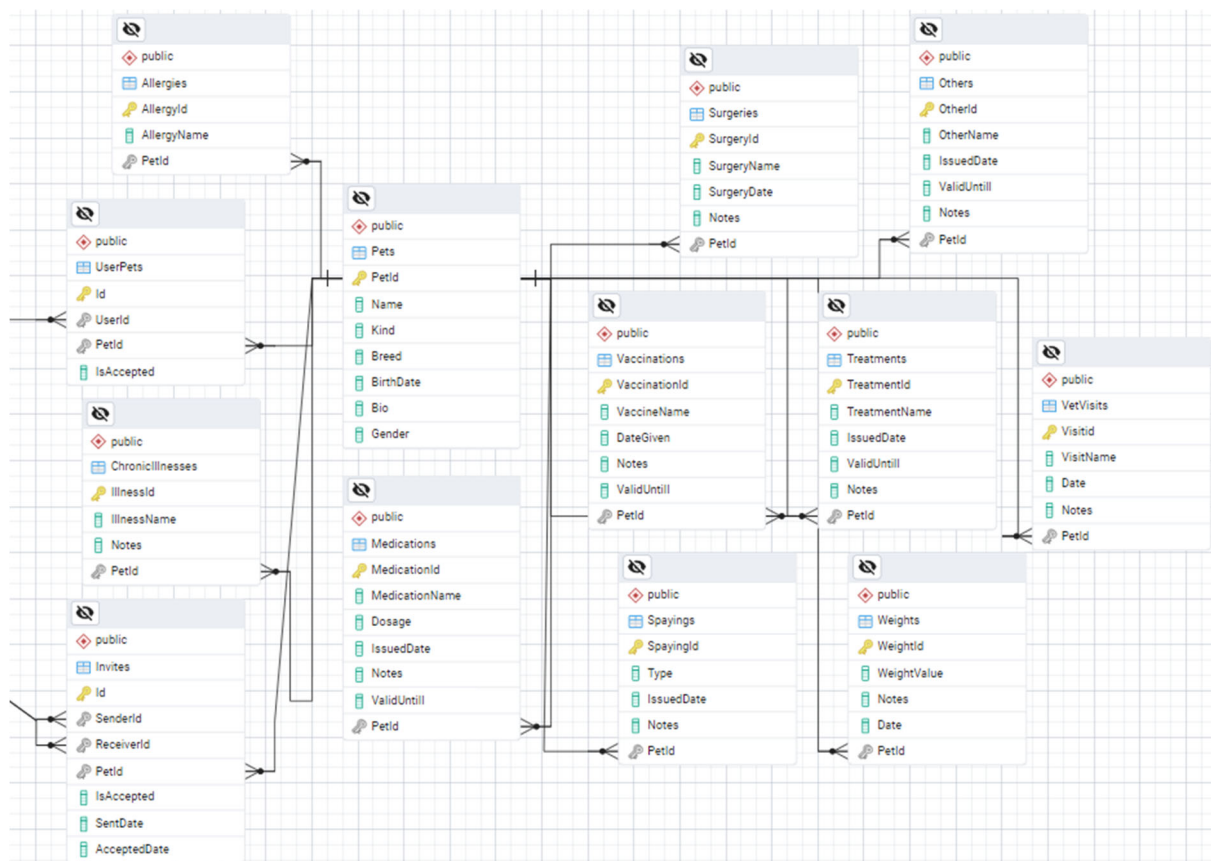


Рис. 4. Частина діаграми бази даних

У таблиці 3 наведений перелік таблиць у базі даних, які наявні на етапі проєктування бази даних.

Таблиця 3. Таблиці бази даних застосунку "PetCareApp"

№	Таблиця	Опис
1	Allergies	Містить інформацію про алергії
2	ChronicIllnesses	Містить інформацію про хронічні захворювання певної тварини
3	Medications	Містить інформацію про призначене лікування
4	Pets	Містить інформацію про тварину
5	Surgeries	Містить інформацію про операції певної тварини
6	Vaccinations	Містить інформацію про вакцинації певної тварини
7	MedRecords	Містить інформацію про медичну книжку певної тварини
8	UserPets	Містить інформацію про тварини певного користувача
9	AspNetUsers	Містить інформацію про користувачів системи

У таблицях 4-12 наведено перелік атрибутів таблиць бази даних проєкту.

Таблиця 4. Перелік атрибутів таблиці Allergies

Атрибут	Тип	Опис
AllergyId	integer	Ідентифікатор
AllergyName	varchar(50)	Назва алергії
MedRecordId	integer	Ідентифікатор медичного запису

Таблиця 5. Перелік атрибутів таблиці ChronicIllnesses

Атрибут	Тип	Опис
IllnessId	integer	Ідентифікатор
IllnessName	varchar(75)	Назва хвороби
MedRecordId	integer	Ідентифікатор медичного запису

Таблиця 6. Перелік атрибутів таблиці Medications

Атрибут	Тип	Опис
MedicationId	integer	Ідентифікатор
MedicationName	varchar(50)	Назва ліків
Dosage	double precision	Інформація про дозування певних ліків
MedRecordId	integer	Ідентифікатор медичного запису

Таблиця 7. Перелік атрибутів таблиці Pets

Атрибут	Тип	Опис
PetId	integer	Ідентифікатор
Name	varchar(100)	Ім'я тварини
Kind	varchar(50)	Тип тварини
Breed	varchar(50)	Порода тварини
BirthDate	date	Дата народження тварини
Bio	varchar(500)	Біографія тварини
MedRecordId	integer	Ідентифікатор медичного запису
Gender	varchar(10)	Стать тварини

Таблиця 8. Перелік атрибутів таблиці Surgeries

Атрибут	Тип	Опис
SurgeryId	integer	Ідентифікатор
SurgeryName	varchar(75)	Назва операції
SurgeryDate	timestamptz	Час проведення операції
MedRecordId	integer	Ідентифікатор медичного запису

Таблиця 9. Перелік атрибутів таблиці Vaccinations

Атрибут	Тип	Опис
VaccinationId	integer	Ідентифікатор
VaccineName	varchar(75)	Назва вакцини
DateGiven	timestamptz	Дата вакцинування
MedRecordId	integer	Ідентифікатор медичного запису

Таблиця 10. Перелік атрибутів таблиці MedRecords

Атрибут	Тип	Опис
MedRecordId	integer	Ідентифікатор
Weight	double precision	Вага тварини
Height	double precision	Ріст тварини

Таблиця 11. Перелік атрибутів таблиці UserPets

Атрибут	Тип	Опис
Id	integer	Ідентифікатор
UserId	text	Ідентифікатор користувача
PetId	integer	Ідентифікатор тварини
IsAccepted	boolean	Статус володіння твариною користувачем

Таблиця 12. Перелік атрибутів таблиці AspNetUsers

Атрибут	Тип	Опис
Id	integer	Ідентифікатор
RefreshToken	text	Токен користувача
RefreshTokenExpiryTime	timestampz	Дата закінчення строку дії токена
UserName	varchar(256)	Ім'я користувача
Email	varchar(256)	Електронна пошта користувача
Bio	text	Опис користувача
CreatedAt	timestampz	Дата створення профілю користувача
UpdatedAt	timestampz	Дата редагування профілю користувача

UI/UX-дизайн продукту

UX-дизайн (досвід користувача) базується на багатьох законах і принципах, які допомагають створювати зручні, інтуїтивно зрозумілі продукти, що відповідають очікуванням користувачів. Нижче наведено закони і принципи UX-дизайну, які були використані, та їхній вплив на поведінку користувача.

Закон Фіттса. Час, потрібний для досягнення цілі (наприклад, натискання кнопки), залежить від її розміру та відстані до неї [25].

Де використовується: На етапі розробки інтерфейсу кнопок, іконок чи інших інтерактивних елементів.

Функція: Зменшує час і зусилля для взаємодії.

Вплив на поведінку користувача: Сприяє швидшій та інтуїтивній взаємодії, що може збільшити конверсію.

Приклад: У мобільному застосунку при додаванні тваринки кнопка "Next" велика та розташована в нижній частині екрана, що дозволяє легко до неї дістатися великим пальцем (рис. 5).

Закон Гіка. Час, потрібний для ухвалення рішення, зростає із кількістю та складністю доступних варіантів [26].

Де використовується: у сповіщеннях із вибором "Прийняти" або "Відхилити"; у ситуаціях, де потрібно швидко ухвалити просте рішення.

Функція: полегшує ухвалення рішення внаслідок простоти; уникає когнітивного перевантаження, даючи лише чіткі варіанти.

Вплив на поведінку: спонукає користувача діяти негайно; забезпечує ефективну комунікацію між користувачами.

Приклад:

Сповіщення: "Користувач хоче поділитися тваринкою з вами. Прийняти чи Відхилити?" (рис. 6)

Принцип послідовності. Користувачі очікують, що елементи інтерфейсу будуть працювати передбачувано.

Де використовується: на всіх етапах дизайну продукту.

Функція: Забезпечує передбачуваність у взаємодії.

Вплив на поведінку користувача: Знижує кількість помилок і час, потрібний для освоєння продукту.

Приклад:

Кнопка "Назад" завжди розташована в одному місці на всіх екранах мобільного застосунку (рис. 7-9)

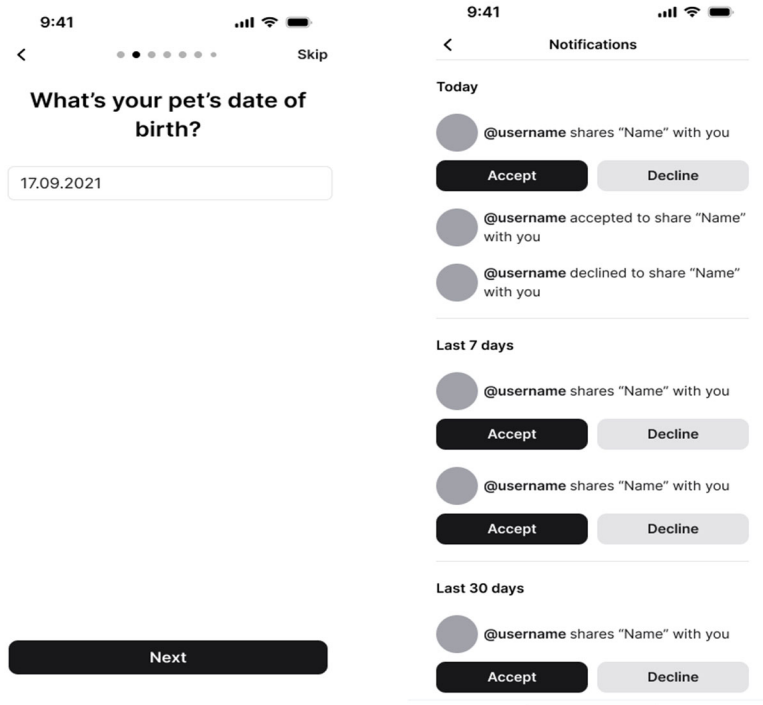


Рис. 5-6. Приклад застосування закону Фітса та закону Гіка

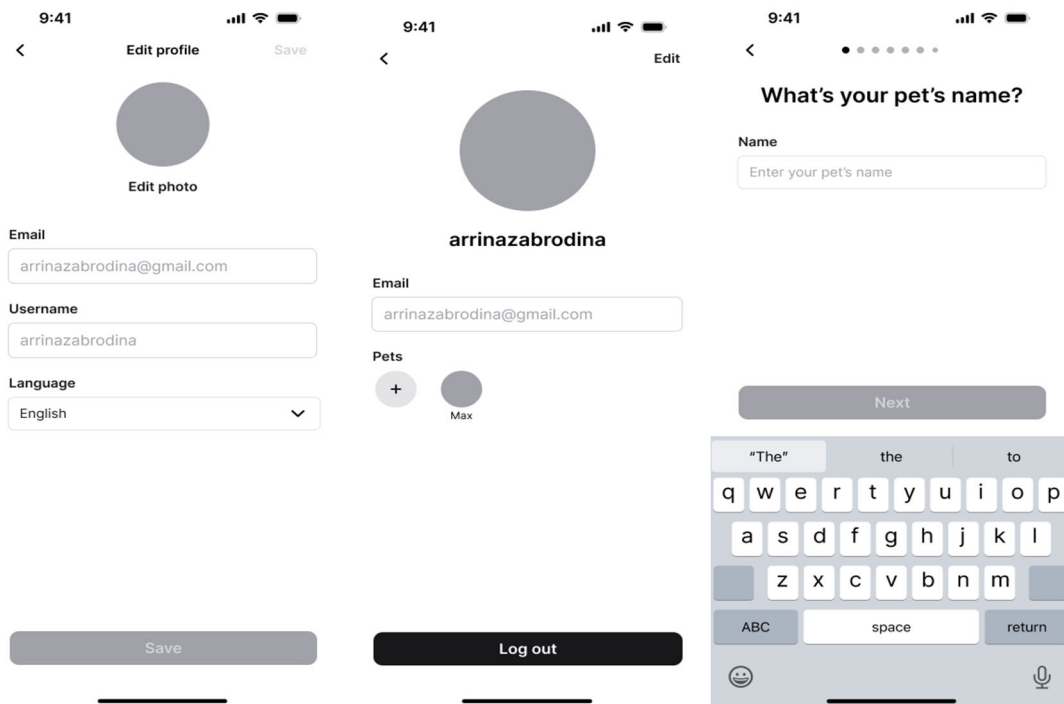


Рис. 7-9. Приклад застосування закону послідовності

Закон Міллера. Людина може одночасно утримувати в короткочасній пам'яті 5-9 елементів [27].

Де використовується: у розробці списків, меню або форм.

Функція: Полегшує сприйняття інформації.

Вплив на поведінку користувача: Знижує когнітивне навантаження, підвищуючи комфорт взаємодії.

Приклад:

Нижнє меню (tab bar) застосунку містить максимум 4 пункти, щоб не перевантажувати користувача (рис. 10).

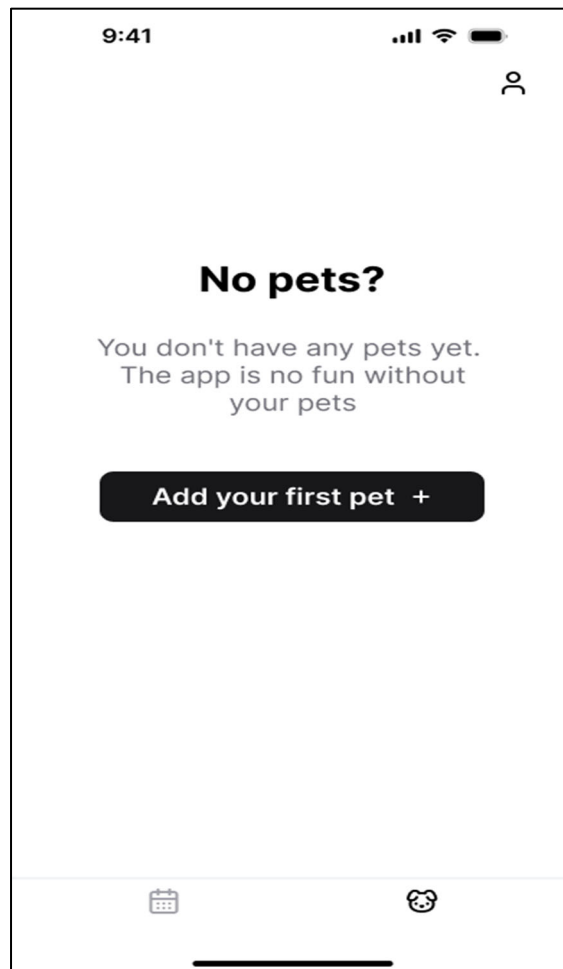


Рис. 10. Приклад застосування закону Міллера

Принцип Парето (80/20). 80% результатів досягаються завдяки 20% зусиль [28].

Де використовується: На етапі розставлення пріоритетів у функціональності продукту.

Функція: Фокусує увагу користувачів на ключових діях.

Вплив на поведінку користувача: Зменшує кількість непотрібних дій і покращує задоволення від використання продукту.

Приклад:

На екрані перегляду профілю тваринки акцент робиться на функціях "Редагувати" та "Поділитись" та кнопці "Медична картка".

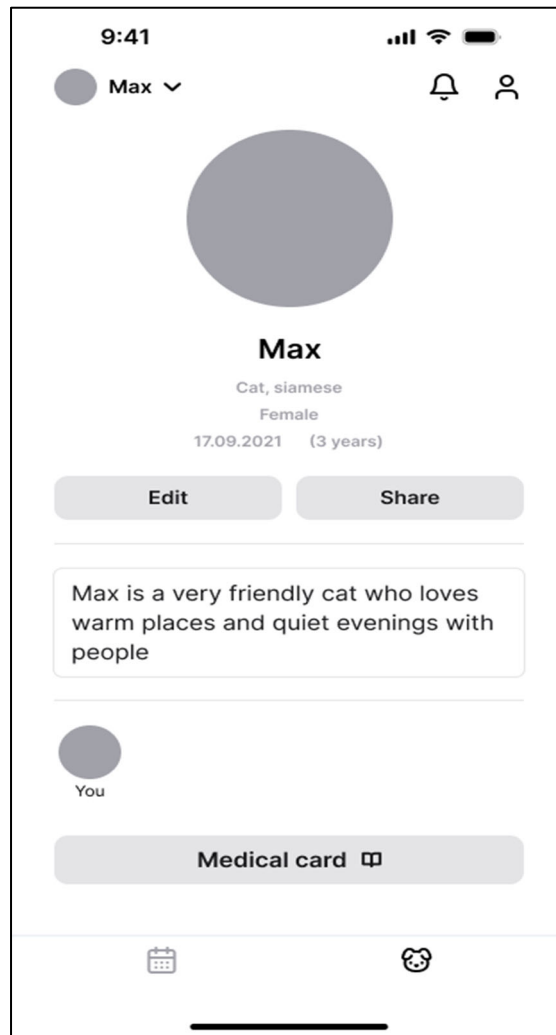


Рис. 11. Приклад застосування закону Парето

Використання цих законів дозволило створити зрозумілий та продуманий дизайн застосунку.

3. ІМПЛЕМЕНТАЦІЯ ПРОДУКТУ ТА ТЕСТУВАННЯ

Загальна архітектура системи. Опис архітектури продукту, основних компонентів та їхньої взаємодії:

- клієнтська частина: реалізована за допомогою React Native та Expo;
- серверна частина: створена з використанням ASP.NET Web API, яка обробляє запити клієнта, виконує бізнес-логіку та взаємодіє з базою даних;
- база даних: PostgreSQL для зберігання даних;
- протокол зв'язку: HTTP/HTTPS з використанням REST API для передачі даних між клієнтом та сервером.

Реалізація серверної частини. Основні компоненти ASP.NET API, налаштування маршрутизації та обробка запитів:

- маршрутизація: реалізована з використанням атрибутів контролерів;
- контролери: відповідають за обробку запитів та бізнес-логіку;

- сервіси: бізнес-логіка пов'язана з безпекою та автентифікацією, винесена в окремі сервіси.

Реалізація клієнтської частини. Опис компонентів React Native, використання API та обробки даних.

- екрани: кожен екран відповідає окремому маршруту;
- обробка запитів: використання бібліотеки axios та tanstack/react-query для виконання запитів до API;
- генерація типів: використовується @openapi-tools/openapi-generator-cli бібліотека

База даних. Для зберігання даних використовується об'єктно-реляційна СКБД PostgreSQL. Як платформа адміністрування використовується pgAdmin 4.

Інтеграція компонентів. Основна архітектура проєкту забезпечує взаємодію між клієнтом (мобільний застосунок React Native), сервером (ASP.NET Core API) та базою даних (PostgreSQL).

Ця інтеграція реалізується через REST API, забезпечуючи простий та уніфікований спосіб передачі даних.

Механізм роботи:

- користувач надсилає запит через мобільний застосунок. Наприклад, запит на логін або отримання тваринок;
- сервер обробляє запит та перевіряє облікові дані, взаємодіє з базою даних та формує відповідь;
- клієнт отримує відповідь. У відповідь клієнт отримує результат запиту (наприклад, дані користувача) або повідомлення про помилку.
- Інтеграція компонентів забезпечує ефективну роботу системи завдяки:
 - централізованій автентифікації через JWT;
 - обробці помилок як на сервері, так і на клієнтській частині;
 - ефективній взаємодії з базою даних через ORM (EF Core). Це створює надійний та зручний у використанні продукт.

DevOps

Pull Request Pipeline. На рис. 12 зображено створення Pull Request.

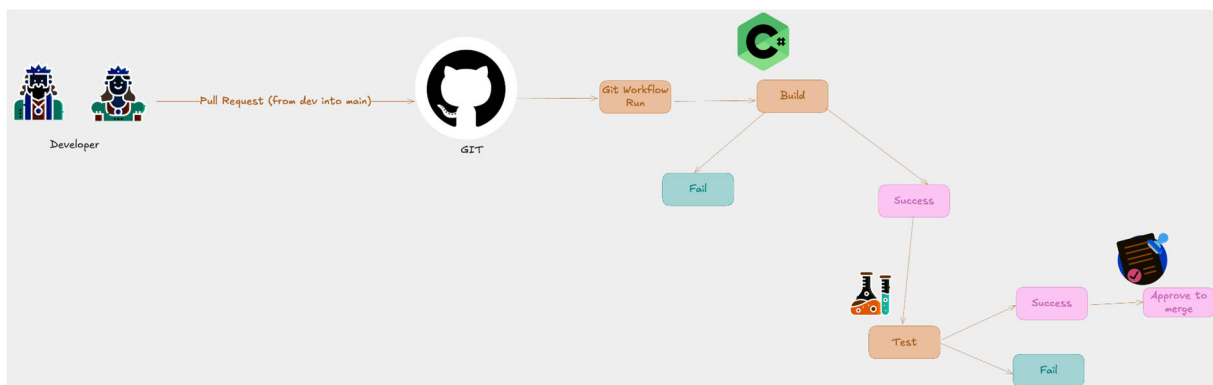


Рис. 12. Pull Request from dev into main flow

Процес створення Pull Request виглядав наступним чином.

Розробник вносить зміни в dev-гілку та створює Pull Request (PR) для злиття цих змін у main-гілку. PR запускає автоматизований процес у CI/CD пайплайні.

Тригер GitHub Workflow: як тільки PR подається, він активує GitHub Actions workflow.

Фаза збірки: Workflow спочатку ініціює процес build, що містить компіляцію коду.

- якщо збірка проходить успішно, процес переходить до фази тестування;
- якщо збірка не вдається, процес зупиняється, і розробник повинен усунути проблеми перед продовженням.

Фаза тестування:

- якщо крок збірки пройшов успішно, наступним етапом є запуск integration tests;
- якщо тести не пройдені, workflow зупиняється і розробник має виправити всі помилки перед повторним запуском пайплайну.

Затвердження для злиття: якщо обидві фази - Build та Tests - пройдені успішно, зміни стають доступними для затвердження командою.

Цей workflow гарантує, що до main-гілки зливаються лише якісний, добре протестований код, що підвищує стабільність та надійність коду.

Continuous Integration and Deployment. На рис. 13 зображена схема, яка описує CI/CD при злитті гілки Dev в гілку Main

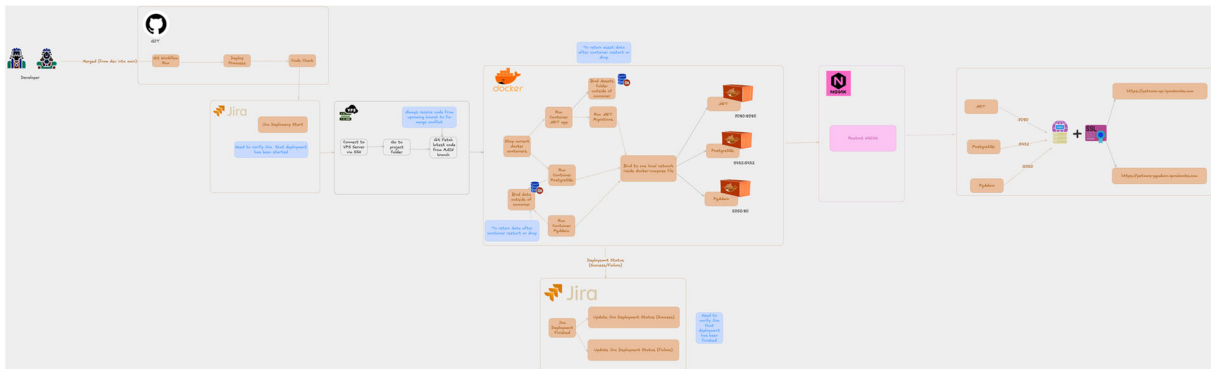


Рис. 13. Dev branch merged to Main branch (Deployment process)

Робочий процес розробників:

- розробники пушають і зливають зміни в код (з робочої гілки в main гілку) на GitHub;
- це запускає GitHub Actions для виконання CI/CD конвеєра.

Процес розгортання на VPS:

- CI/CD конвеєр підключається до сервера VPS через SSH;
- він переходить до папки проєкту та завантажує останній код з main гілки, вирішуючи конфлікти об'єднання, обираючи завжди зміни, які надходять.

Налаштування Docker:

- docker використовується для контейнеризації сервісів, включно з .NET-застосунок, базою даних PostgreSQL та pgAdmin;

- контейнери запускаються у локальній мережі, визначеній у файлі docker-compose, з прив'язкою кожного сервісу до певних портів:

.NET-додаток: порти 7080 - публічний і 8080 - локальний в контейнері;

PostgreSQL: порти 5432 - публічний і 5430 - локальний в контейнері;

pgAdmin: порти 5050 - публічний і 80 - локальний в контейнері.

- контейнери налаштовані так, щоб зберігати дані після перезапуску шляхом прив'язки даних поза контейнерами.

Nginx:

- Nginx використовується як зворотний проксі, обмежуючи зовнішній доступ;
- він маршрутизує трафік до відповідних сервісів, забезпечуючи безпечні з'єднання з SSL-сертифікатами;

Інтеграція з Jira:

- процес розгортання інтегрований з Jira, де відстежуються статуси розгортання (початок, успіх або помилка).

Точки доступу:

- після розгортання до програми можна отримати доступ через захищені URL-адреси з SSL-сертифікатами.

Pull Request Validation

На рис. 14 та 15 зображено повний цикл зливання коду в main гілку.

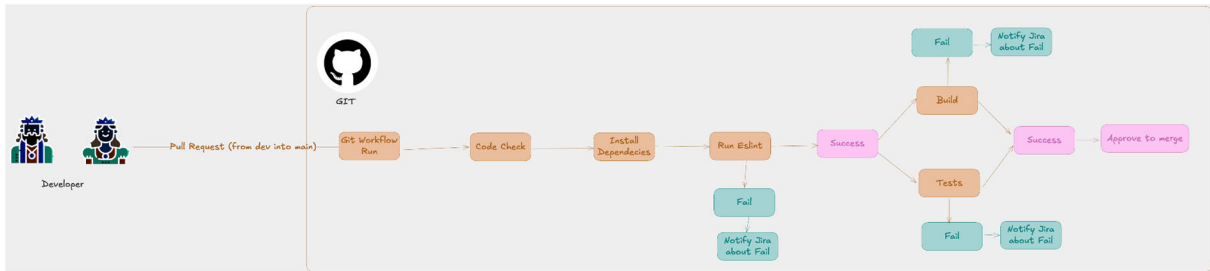


Рис. 14. FE Pull Request from dev to main

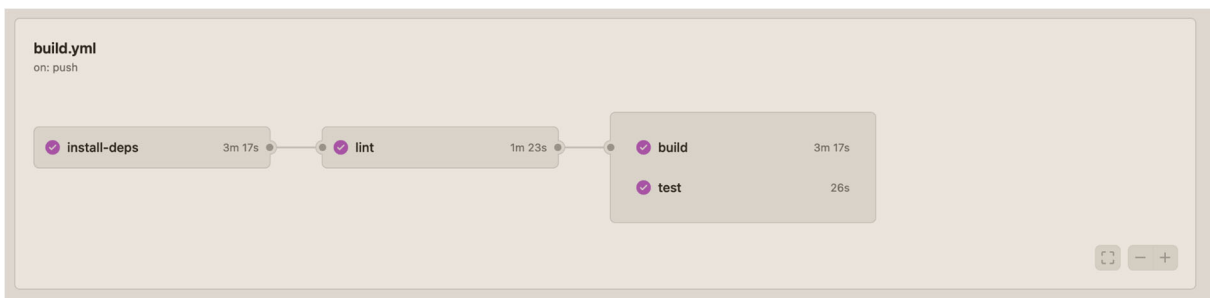


Рис. 15. Github приклад

Процес автоматизованої перевірки та затвердження коду під час створення Pull-запиту для злиття змін у основну гілку.

- розробник ініціює Pull-запит: Процес починається з того, що розробник створює pull request із гілки dev в основну гілку;
- запуск робочого процесу Git: Pull request активує робочий процес Git, що ініціює серію автоматизованих перевірок;
 - перевірка коду: Код перевіряється або аналізується на наявність проблем;
 - встановлення залежностей: Встановлюються необхідні для проєкту залежності;
 - запуск Eslint: Запускається Eslint, щоб переконатися, що код відповідає стилістичним вимогам і не містить синтаксичних помилок. Якщо Eslint не проходить, то надсилається сповіщення в Jira про помилку, і процес завершується тут. Якщо Eslint проходить успішно, то процес продовжується до наступних етапів;
 - збірка: Проєкт збирається для перевірки на наявність проблем зі збіркою;
 - Якщо збірка не проходить, то надсилається сповіщення в Jira про помилку; якщо збірка проходить успішно, то робочий процес переходить до етапу тестування;
 - тестування: Виконуються автоматизовані тести для перевірки функціональності та правильності коду. Якщо тести не проходять, то надсилається сповіщення в Jira про помилку; якщо тести проходять успішно, то процес переходить до наступного етапу;
 - успіх і затвердження: Якщо всі перевірки проходять успішно, pull request може бути затверджений для злиття в основну гілку.

Тестування

Модульне тестування. У межах функціональності модуля **TokenController** проведено модульне тестування з наступними тестами:

- **Refresh_ValidToken_ReturnsNewTokens**

Тест перевіряє, що при оновленні токена з валідним RefreshToken повертається новий AccessToken та RefreshToken.

Очікуваний результат: Успішне створення нових токенів, новий RefreshToken збережений для користувача;

- **Refresh_InvalidRefreshToken_ReturnsBadRequest**

Тест перевіряє, що при переданні недійсного RefreshToken сервер повертає помилку BadRequest.

Очікуваний результат: Помилка з кодом InvalidClientRequest;

- **Revoke_ValidUser_ReturnsNoContent**

Тест перевіряє, що для валідного користувача токен успішно анулюється, а оновлення у базі виконується правильно.

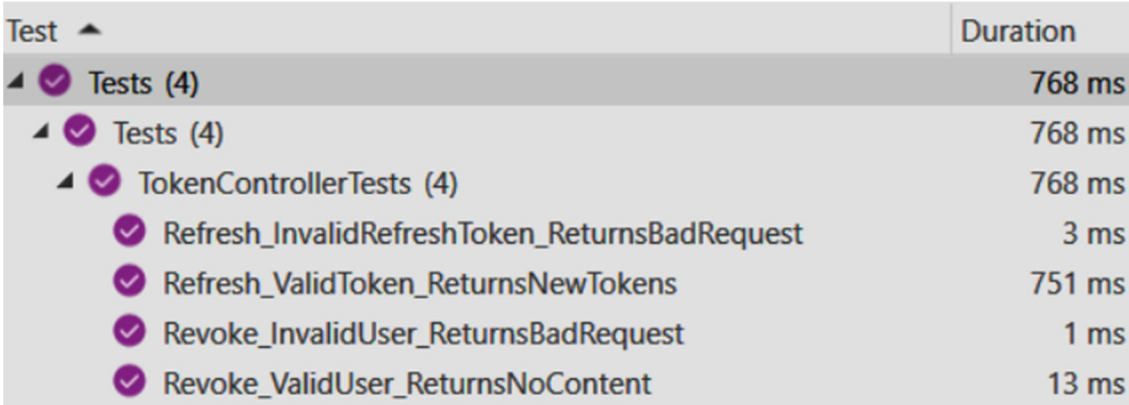
Очікуваний результат: StatusCode 204 та встановлення RefreshToken = null для користувача;

- **Revoke_InvalidUser_ReturnsBadRequest**

Тест перевіряє, що при спробі анулювання токена для невідомого користувача сервер повертає помилку BadRequest.

Очікуваний результат: Помилка з кодом UserNotFound.

На рис. 16 зображено результати модульного тестування TokenController



Test	Duration
Tests (4)	768 ms
Tests (4)	768 ms
TokenControllerTests (4)	768 ms
Refresh_InvalidRefreshToken_ReturnsBadRequest	3 ms
Refresh_ValidToken_ReturnsNewTokens	751 ms
Revoke_InvalidUser_ReturnsBadRequest	1 ms
Revoke_ValidUser_ReturnsNoContent	13 ms

Рис. 16. Результати модульного тестування TokenController

У межах функціональності модуля **PetPhotoController** проведено модульне тестування з наступними тестами:

- **UploadPetPhoto_ValidFile_ReturnsUrl**

Тест перевіряє, що при завантаженні валідного файлу повертається коректний URL для доступу до фото.

Очікуваний результат: Успішне створення шляху до фото, повертається коректна структура відповіді;

- **UploadPetPhoto_NoFile_ReturnsBadRequest**

Тест перевіряє, що якщо файл не переданий, повертається помилка BadRequest.

Очікуваний результат: Помилка з кодом NoFile;

- **UploadPetPhoto_FileTooLarge_ReturnsBadRequest**

Тест перевіряє, що якщо розмір файлу перевищує обмеження, повертається помилка BadRequest.

Очікуваний результат: Помилка з кодом TooLarge;

- **UploadPetPhoto_InvalidFileType_ReturnsBadRequest**

Тест перевіряє, що якщо тип файлу недійсний, сервер повертає помилку BadRequest.

Очікуваний результат: Помилка з кодом InvalidFileType;

- **GetPetPhoto_ValidPhoto_ReturnsFile**

Тест перевіряє, що при запиті на отримання існуючого фото повертається файл з коректним типом.

Очікуваний результат: Повернення файлу з типом image/jpeg;

- **GetPetPhoto_NoPhoto_ReturnsNotFound**

Тест перевіряє, що якщо фото не знайдено, сервер повертає помилку NotFound.

Очікуваний результат: Помилка з кодом ResourceNotFound;

- **DeletePetPhoto_ValidPhoto_ReturnsOk**

Тест перевіряє, що при видаленні існуючого фото сервер повертає статус ОК, а файл успішно видаляється.

Очікуваний результат: Успішне видалення файлу, повернення коду успіху;

- **DeletePetPhoto_NoPhoto_ReturnsNotFound**

Тест перевіряє, що якщо фото для видалення не існує, сервер повертає помилку NotFound.

Очікуваний результат: Помилка з кодом ResourceNotFound.

На рис. 17 зображено результати модульного тестування PetPhotoController.

Test	Duration
Tests (8)	573 ms
Tests (8)	573 ms
PetPhotoControllerTests (8)	573 ms
DeletePetPhoto_NoPhoto_ReturnsNotFound	< 1 ms
DeletePetPhoto_ValidPhoto_ReturnsOk	1 ms
GetPetPhoto_NoPhoto_ReturnsNotFound	< 1 ms
GetPetPhoto_ValidPhoto_ReturnsFile	551 ms
UploadPetPhoto_FileTooLarge_ReturnsBadRequest	8 ms
UploadPetPhoto_InvalidFileType_ReturnsBadRequest	1 ms
UploadPetPhoto_NoFile_ReturnsBadRequest	< 1 ms
UploadPetPhoto_ValidFile_ReturnsUrl	12 ms

Рис. 17. Результати модульного тестування PetPhotoController

У межах функціональності модуля **PetController** проведено модульне тестування з наступними тестами:

- **GetPetProfile_PetExistsAndIsOwner_ReturnsPet**

Тест перевіряє, що при наявності профілю у користувача з вірним зв'язком між користувачем і твариною, повертається інформація про тварину.

Очікуваний результат: Успішне повернення об'єкта Pet з відповідними даними;

- **GetPetProfile_PetNotExistsOrNotOwner_ReturnsNotFound**

Тест перевіряє, що якщо тварини не існує або користувач не є її власником, сервер повертає помилку NotFound.

Очікуваний результат: Помилка з кодом ResourceNotFound;

- **GetUserPets_UserHasPets_ReturnsListOfPets**

Тест перевіряє, що при наявності у користувача тварин, повертається список тварин користувача.

Очікуваний результат: Повернення списку тварин для користувача з відповідними даними;

- **AddPet_ValidPetData_ReturnsSuccessResponse**

Тест перевіряє, що при додаванні коректних даних для нової тварини повертається успішний результат.

Очікуваний результат: Повернення SuccessResponse з кодом PetCreated;

- **AddPet_UserNotFound_ReturnsNotFound**

Тест перевіряє, що якщо користувач не знайдений, то сервер повертає помилку NotFound.

Очікуваний результат: Помилка з кодом UserNotFound;

- **DeletePet_PetExistsAndIsOwner_ReturnsSuccessResponse**

Тест перевіряє, що для наявної тварини, яка належить користувачеві, виконується успішне видалення тварини.

Очікуваний результат: Повернення SuccessResponse з кодом PetDeleted;

- **UpdatePet_PetExistsAndIsOwner_ReturnsSuccessResponse**

Тест перевіряє, що при оновленні тварини, якщо вона належить користувачеві, змінюються дані тварини та повертається успішний результат.

Очікуваний результат: Повернення SuccessResponse з кодом PetUpdated.

На рис. 18 зображено результати модульного тестування PetController

Test	Duration
Tests (8)	3.8 sec
Tests (8)	3.8 sec
PetControllerTests (8)	3.8 sec
AddPet_UserNotFound_ReturnsNotFound	193 ms
AddPet_ValidPetData_ReturnsSuccessResponse	177 ms
Database_IsInMemory	160 ms
DeletePet_PetExistsAndIsOwner_ReturnsSuccessResponse	230 ms
GetPetProfile_PetExistsAndIsOwner_ReturnsPet	187 ms
GetPetProfile_PetNotExistsOrNotOwner_ReturnsNotFound	146 ms
GetUserPets_UserHasPets_ReturnsListOfPets	2.5 sec
UpdatePet_PetExistsAndIsOwner_ReturnsSuccessResponse	189 ms

Рис. 18. Результати модульного тестування PetController

У межах функціональності модуля **MedRecordController** проведено модульне тестування з наступними тестами:

- **AddAllergy_ValidData_ReturnsSuccessResponse**

Тест перевіряє, що при додаванні алергії з коректними даними для існуючої медичної картки, система повертає успішну відповідь.

Очікуваний результат: Успішне створення алергії, повідомлення про створення AllergyCreated;

- **AddAllergy_MedRecordNotFound_ReturnsErrorResponse**

Тест перевіряє, що при спробі додати алергію з медичним записом, що не існує, система повертає помилку з кодом MedRecordNotFound.

Очікуваний результат: Помилка з кодом MedRecordNotFound;

- **DeleteAllergy_MedRecordExists_ReturnsSuccessResponse**

Тест перевіряє, що при видаленні алергії для існуючого медичного запису система успішно виконує операцію.

Очікуваний результат: Успішне видалення алергії, повідомлення про видалення AllergyDeleted;

- **UpdateAllergy_MedRecordExists_ReturnsSuccessResponse**

Тест перевіряє, що при оновленні алергії для існуючого медичного запису система успішно виконуватиме операцію оновлення.

Очікуваний результат: Успішне оновлення алергії, повідомлення про оновлення AllergyUpdated;

- **AddChronicIllness_ValidData_ReturnsSuccessResponse**

Тест перевіряє, що при додаванні хронічного захворювання з коректними даними для існуючої медичної картки, система повертає успішну відповідь.

Очікуваний результат: Успішне створення хронічного захворювання, повідомлення про створення ChronicIllnessCreated;

- **AddChronicIllness_MedRecordNotFound_ReturnsErrorResponse**

Тест перевіряє, що при спробі додати хронічне захворювання з медичним записом, що не існує, система повертає помилку з кодом MedRecordNotFound.

Очікуваний результат: Помилка з кодом MedRecordNotFound;

- **DeleteChronicIllness_MedRecordExists_ReturnsSuccessResponse**

Тест перевіряє, що при видаленні хронічного захворювання для існуючого медичного запису система успішно виконує операцію.

Очікуваний результат: Успішне видалення хронічного захворювання, повідомлення про видалення ChronicIllnessDeleted;

- **UpdateChronicIllness_MedRecordExists_ReturnsSuccessResponse**

Тест перевіряє, що при оновленні хронічного захворювання для існуючого медичного запису система успішно виконуватиме операцію оновлення.

Очікуваний результат: Успішне оновлення хронічного захворювання, повідомлення про оновлення ChronicIllnessUpdated;

- **AddMedication_ValidData_ReturnsSuccessResponse**

Тест перевіряє, що при додаванні медикаменту з коректними даними для існуючої медичної картки, система повертає успішну відповідь.

Очікуваний результат: Успішне створення медикаменту, повідомлення про створення MedicationCreated;

- **AddMedication_MedRecordNotFound_ReturnsErrorResponse**

Тест перевіряє, що при спробі додати медикамент з медичним записом, що не існує, система повертає помилку з кодом MedRecordNotFound.

Очікуваний результат: Помилка з кодом MedRecordNotFound;

- **DeleteMedication_MedRecordExists_ReturnsSuccessResponse**

Тест перевіряє, що при видаленні медикаменту для існуючого медичного запису система успішно виконує операцію.

Очікуваний результат: Успішне видалення медикаменту, повідомлення про видалення MedicationDeleted;

- **UpdateMedication_MedRecordExists_ReturnsSuccessResponse**

Тест перевіряє, що при оновленні медикаменту для існуючого медичного запису система успішно виконуватиме операцію оновлення.

Очікуваний результат: Успішне оновлення медикаменту, повідомлення про оновлення MedicationUpdated;

- **AddSurgery_ValidData_ReturnsSuccessResponse**

Тест перевіряє, що при додаванні хірургії з коректними даними для існуючої медичної картки, система повертає успішну відповідь.

Очікуваний результат: Успішне створення хірургії, повідомлення про створення SurgeryCreated;

- **AddSurgery_MedRecordNotFound_ReturnsErrorResponse**

Тест перевіряє, що при спробі додати хірургію з медичним записом, що не існує, система повертає помилку з кодом MedRecordNotFound.

Очікуваний результат: Помилка з кодом MedRecordNotFound;

- **DeleteSurgery_MedRecordExists_ReturnsSuccessResponse**

Тест перевіряє, що при видаленні хірургії для існуючого медичного запису система успішно виконує операцію.

Очікуваний результат: Успішне видалення хірургії, повідомлення про видалення SurgeryDeleted;

- **UpdateSurgery_MedRecordExists_ReturnsSuccessResponse**

Тест перевіряє, що при оновленні хірургії для існуючого медичного запису система успішно виконуватиме операцію оновлення.

Очікуваний результат: Успішне оновлення хірургії, повідомлення про оновлення SurgeryUpdated;

- **AddVaccination_ValidData_ReturnsSuccessResponse**

Тест перевіряє, що при додаванні вакцинації з коректними даними для існуючої медичної картки, система повертає успішну відповідь.

Очікуваний результат: Успішне створення вакцинації, повідомлення про створення VaccinationCreated;

- **AddVaccination_MedRecordNotFound_ReturnsErrorResponse**

Тест перевіряє, що при спробі додати вакцинацію з медичним записом, що не існує, система повертає помилку з кодом MedRecordNotFound.

Очікуваний результат: Помилка з кодом MedRecordNotFound;

- **DeleteVaccination_MedRecordExists_ReturnsSuccessResponse**

Тест перевіряє, що при видаленні вакцинації для існуючого медичного запису система успішно виконує операцію.

Очікуваний результат: Успішне видалення вакцинації, повідомлення про видалення VaccinationDeleted;

- **UpdateVaccination_MedRecordExists_ReturnsSuccessResponse**

Тест перевіряє, що при оновленні вакцинації для існуючого медичного запису система успішно виконуватиме операцію оновлення;

- **GetMedRecords_ShouldReturn200Ok_WithListOfMedRecords**

Тест робить перевірку чи повертає запит список всіх медичних карток у нашій базі та чи існують вони взагалі;

- **GetMedRecord_ShouldReturn200Ok_WithSpecificMedRecord**

Тест перевіряє чи існує медична картка у користувача з заданим ідентифікатором, у разі існування повертає інформацію медичної картки цього користувача.

Очікуваний результат: Успішне оновлення вакцинації, повідомлення про оновлення VaccinationUpdated.

На рис. 19 зображено результати модульного тестування MedRecordController.

Test Name	Duration
MedRecordControllerTests (23)	3,6 c
AddAllergy_MedRecordNotFound_Re...	97 мс
AddAllergy_ValidData_ReturnsSucces...	90 мс
AddChronicIllness_MedRecordNotFo...	107 мс
AddChronicIllness_ValidData_Returns...	155 мс
AddMedication_MedRecordNotFoun...	122 мс
AddMedication_ValidData_ReturnsSu...	103 мс
AddSurgery_MedRecordNotFound_R...	87 мс
AddSurgery_ValidData_ReturnsSucce...	105 мс
AddVaccination_MedRecordNotFoun...	146 мс
AddVaccination_ValidData_ReturnsSu...	140 мс
Database_IsInMemory	96 мс
DeleteAllergy_MedRecordExists_Retu...	100 мс
DeleteChronicIllness_MedRecordExist...	94 мс
DeleteMedication_MedRecordExists_...	92 мс
DeleteSurgery_MedRecordExists_Ret...	88 мс
DeleteVaccination_MedRecordExists_...	101 мс
GetMedRecord_ShouldReturn200Ok_...	167 мс
GetMedRecords_ShouldReturn200Ok...	116 мс
UpdateAllergy_MedRecordExists_Ret...	125 мс
UpdateChronicIllness_MedRecordExi...	1,1 c
UpdateMedication_MedRecordExists...	121 мс
UpdateSurgery_MedRecordExists_Ret...	131 мс
UpdateVaccination_MedRecordExists...	88 мс

Рис. 19. Результати модульного тестування MedRecordController

У межах функціональності модуля *AvatarController* проведено модульне тестування з наступними тестами:

- **UploadAvatar_ValidFile_ReturnsUrl**

Тест перевіряє, що при завантаженні валідного файлу аватара для користувача система повертає URL для доступу до цього аватара.

Очікуваний результат: Повернення URL аватара у вигляді словника з відповідними полями;

- **UploadAvatar_NoFile_ReturnsBadRequest**

Тест перевіряє, що при спробі завантажити файл, який не передано, система повертає помилку BadRequest з відповідним кодом помилки.

Очікуваний результат: Помилка з кодом NoFile;

- **UploadAvatar_FileTooLarge_ReturnsBadRequest**

Тест перевіряє, що при завантаженні файлу, розмір якого перевищує максимальний ліміт, система повертає помилку BadRequest з відповідним кодом помилки.

Очікуваний результат: Помилка з кодом TooLarge;

- **UploadAvatar_InvalidFileType_ReturnsBadRequest**

Тест перевіряє, що при спробі завантажити файл з невірним типом (наприклад, PDF), система повертає помилку BadRequest з відповідним кодом помилки.

Очікуваний результат: Помилка з кодом InvalidFileType;

- **GetAvatar_ValidAvatar_ReturnsFile**

Тест перевіряє, що при наявності аватара для користувача, система повертає сам файл аватара у відповідь.

Очікуваний результат: Повернення файлу аватара з типом image/png;

- **GetAvatar_NoAvatar_ReturnsNotFound**

Тест перевіряє, що при відсутності аватара для користувача, система повертає помилку NotFound.

Очікуваний результат: Помилка з кодом ResourceNotFound;

- **GetAvatar_UserId_ValidAvatar_ReturnsFile**

Тест перевіряє, що при наявності аватара для користувача за вказаним ідентифікатором, система повертає сам файл аватара.

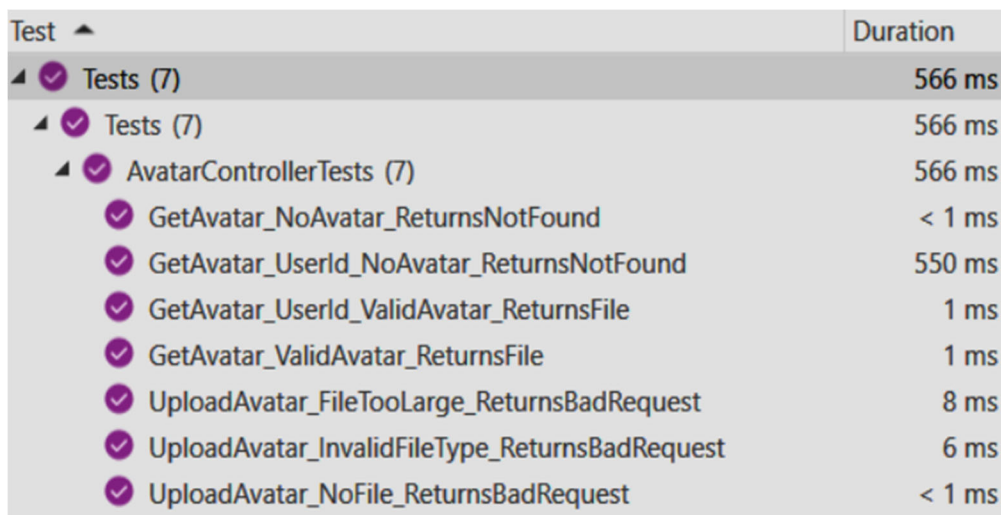
Очікуваний результат: Повернення файлу аватара з типом image/png;

- **GetAvatar_UserId_NoAvatar_ReturnsNotFound**

Тест перевіряє, що при відсутності аватара для користувача з вказаним ідентифікатором, система повертає помилку NotFound.

Очікуваний результат: Помилка з кодом ResourceNotFound.

На рисунку 20 зображено результати модульного тестування AvatarController



Test	Duration
Tests (7)	566 ms
Tests (7)	566 ms
AvatarControllerTests (7)	566 ms
GetAvatar_NoAvatar_ReturnsNotFound	< 1 ms
GetAvatar_UserId_NoAvatar_ReturnsNotFound	550 ms
GetAvatar_UserId_ValidAvatar_ReturnsFile	1 ms
GetAvatar_ValidAvatar_ReturnsFile	1 ms
UploadAvatar_FileTooLarge_ReturnsBadRequest	8 ms
UploadAvatar_InvalidFileType_ReturnsBadRequest	6 ms
UploadAvatar_NoFile_ReturnsBadRequest	< 1 ms

Рис. 20. Результати модульного тестування AvatarController

У межах функціональності модуля **AuthenticateController** проведено модульне тестування з наступними тестами:

- **Login_ValidCredentials_ReturnsJwtToken**

Тест перевіряє, що при наданні вірних даних для входу система повертає JWT токен.

Очікуваний результат: Повернення валідного токена у відповіді;

- **Login_InvalidCredentials_ReturnsBadRequest**

Тест перевіряє, що при наданні невірних даних для входу система повертає помилку BadRequest.

Очікуваний результат: Помилка з кодом BadRequest;

- **Register_ValidUser_ReturnsSuccess**

Тест перевіряє, що при реєстрації користувача з коректними даними система повертає успішну відповідь.

Очікуваний результат: Повернення успішної відповіді з повідомленням "Success";

- **Register_UserExists_ReturnsBadRequest**

Тест перевіряє, що якщо користувач з таким ім'ям уже існує, система повертає помилку BadRequest.

Очікуваний результат: Помилка з кодом BadRequest.

На рисунку 21 зображено результати модульного тестування AuthenticateController

Test	Duration
Tests (5)	3.4 sec
Tests (5)	3.4 sec
AuthenticateControllerTests (5)	3.4 sec
Database_IsInMemory	57 ms
Login_InvalidCredentials_ReturnsBadRequest	107 ms
Login_ValidCredentials_ReturnsJwtToken	2.7 sec
Register_UserExists_ReturnsBadRequest	182 ms
Register_ValidUser_ReturnsSuccess	323 ms

Рис. 21. Результати модульного тестування AuthenticateController

Перелік тестів збірки

Тест №1:

Послання модулів: UserController - контролер для обробки запитів до профілю користувача і UserManager<User> - для роботи з користувачами (отримання та оновлення користувача);

- ISanitizerService - для обробки даних користувача (санация введених даних);

- **GetProfile_ValidUser_ReturnsUserProfile**

Опис: Тестує взаємодію між контролером, менеджером користувачів та базою даних для повернення профілю користувача за заданим ID;

- **GetProfile_UserNotFound_ReturnsUnauthorized**

Опис: Тестує відповідь контролера у разі, якщо користувач не знайдений у базі даних;

- **UpdateProfile_ValidUpdate_ReturnsSuccess**

Опис: Тестує правильність оновлення профілю користувача, включаючи процес санітарії введених даних;

- **UpdateProfile_UserNotFound_ReturnsUnauthorized**

Опис: Тестує, чи правильно система обробляє запит на оновлення профілю для неіснуючого користувача;

- **UpdateProfile_UpdateFails_ReturnsBadRequest**

Опис: Тестує сценарій, коли оновлення користувача не вдається через помилки в базі даних або інші проблеми.

На рис. 22 зображено результати тестування.

Test	Duration
Tests (6)	2.3 sec
Tests (6)	2.3 sec
UserControllerTests (6)	2.3 sec
Database_IsInMemory	1.9 sec
GetProfile_UserNotFound_ReturnsUnauthorized	65 ms
GetProfile_ValidUser_ReturnsUserProfile	103 ms
UpdateProfile_UpdateFails_ReturnsBadRequest	52 ms
UpdateProfile_UserNotFound_ReturnsUnauthorized	51 ms
UpdateProfile_ValidUpdate_ReturnsSuccess	131 ms

Рис. 22. Результати тестування UserController

Тест №2:

Поєднання модулів: InviteController - контролер для обробки запитів щодо запрошень користувачів і ISocketService - для відправки сповіщень про зміни статусу запрошень;

- **GetUserInvitations_ValidRequest_ReturnsInvitations**

Опис: Тестує взаємодію між контролером та базою даних для отримання запрошень користувача. Перевіряється правильність повернення запрошень для існуючого користувача;

- **SendInvite_ValidRequest_ReturnsOk**

Опис: Тестує процес відправки запрошення між користувачем та отримувачем. Перевіряється, чи коректно система відправляє запрошення і повертає повідомлення про успіх;

- **SendInvite_UserNotFound_ReturnsNotFound**

Опис: Тестує, як система реагує на спробу відправити запрошення неіснуючому користувачу. Перевіряється, чи система повертає повідомлення про помилку з кодом "UserNotFound";

- **AcceptInvite_ValidInvite_ReturnsOk**

Опис: Тестує процес прийняття запрошення користувачем. Перевіряється, чи система правильно обробляє прийняття запрошення та повертає відповідь про успіх;

- **AcceptInvite_InviteNotFound_ReturnsNotFound**

Опис: Тестує обробку ситуації, коли користувач намагається прийняти запрошення, яке не існує. Перевіряється, чи система повертає повідомлення про помилку "InvitationNotFound".

На рис. 23 зображено результати тестування.

Test	Duration
Tests (6)	4.1 sec
Tests (6)	4.1 sec
InviteControllerTests (6)	4.1 sec
AcceptInvite_InviteNotFound_ReturnsNotFound	2.5 sec
AcceptInvite_ValidInvite_ReturnsOk	295 ms
Database_IsInMemory	296 ms
GetUserInvitations_ValidRequest_ReturnsInvitations	352 ms
SendInvite_UserNotFound_ReturnsNotFound	322 ms
SendInvite_ValidRequest_ReturnsOk	333 ms

Рис. 23. Результати тестування InviteController

Тестування інших параметрів

Тест безпеки 1: Перевірка захисту від XSS.

Назва тесту: UpdateProfile_ValidUpdate_ReturnsSuccess

Опис тесту: Перевірка на захист від XSS-атак при оновленні профілю користувача.

Зокрема, передача небезпечного введення в параметрі "Bio", яке містить HTML- або JavaScript-код, наприклад, `<script>alert('hack');</script>`.

Тест:

- створити об'єкт користувача з базовими даними (наприклад, старим значенням Bio);
- спробувати оновити значення Bio з потенційно небезпечним контентом (`<script>alert('hack');</script>`);
- використати сервіс для очищення введення від небезпечних елементів (SanitizerService);
- перевірити, чи контент було очищено від XSS-скриптів і чи було оновлено Bio користувача.

Результат:

Контент очищений SanitizerService. Строка Bio не містить шкідливого коду і виглядає наступним чином: "unsafe bio: ".

Код тесту наведено на рис. 24.

Результати тестування наведено на рис. 25.

```
[Fact]
public async Task UpdateProfile_ValidUpdate_ReturnsSuccess()
{
    var user = new User { Id = "123", UserName = "testuser", Email = "test@example.com", Bio = "Old bio" };
    _mockUserManager.Setup(um => um.FindByIdAsync("123")).ReturnsAsync(user);
    _mockUserManager.Setup(um => um.UpdateAsync(user)).ReturnsAsync(IdentityResult.Success);

    var updateModel = new UserUpdateModel { Bio = "unsafe bio: <script>" };
    updateModel.Bio = _sanitizerService.Sanitize(updateModel.Bio);

    var result = await _userController.UpdateProfile(updateModel);

    var okResult = Assert.IsType<OkObjectResult>(result);
    var successResponse = Assert.IsType<SuccessResponse>(okResult.Value);
    Assert.Equal(SuccessCodes.SuccessfulUpdate, successResponse.Message);
    Assert.Equal("unsafe bio: ", user.Bio);
}
```

Рис. 24. Код тесту для тестування сервісу санітизації тексту

Test	Duration
Tests (1)	131 ms
Tests (1)	131 ms
UserControllerTests (1)	131 ms
UpdateProfile_ValidUpdate_ReturnsSuccess	131 ms

Рис. 25. Результати тестування сервісу санітизації тексту

Тест безпеки 2: Перевірка доступу до GetProfile без авторизації.

Назва тесту: GetProfile_UserNotFound_ReturnsUnauthorized

Опис тесту: Перевірка доступу до методу GetProfile без наявності валідного токена доступу або автентифікації. У цьому тесті запит здійснюється без надання необхідних авторизаційних даних (наприклад, без токена або з недійсним).

Тест:

- зробити запит до методу GetProfile без авторизації користувача (не встановлений чи недійсний токен);
- перевірити, чи правильно обробляється відсутність авторизації.

Результат:

Метод повертає відповідь Unauthorized з кодом помилки ErrorCodes.UserNotFound, що вказує на відсутність прав доступу або автентифікації.

Код тесту наведено на рис. 26.

Результати тестування наведено на рис. 27.

```
[Fact]
public async Task GetProfile_UserNotFound_ReturnsUnauthorized()
{
    _mockUserManager.Setup(um => um.FindByIdAsync("123")).ReturnsAsync((User)null);

    var result = await _userController.GetProfile();

    var unauthorizedResult = Assert.IsType<UnauthorizedObjectResult>(result);
    var errorResponse = Assert.IsType<ErrorResponse>(unauthorizedResult.Value);
    Assert.Equal(ErrorCodes.UserNotFound, errorResponse.Message);
}
```

Рис. 26. Код тесту для тестування коректності прав доступу

Test	Duration
Tests (1)	65 ms
Tests (1)	65 ms
UserControllerTests (1)	65 ms
GetProfile_UserNotFound_ReturnsUnauthorized	65 ms

Рис. 27. Результати тестування коректності прав доступу

4. ДОКУМЕНТАЦІЯ ПРОЄКТУ

Обрана модель та інструменти ведення документації

Для ведення документації в рамках розробки мобільного застосунку для власників тварин використовується модель гнучкої документації, яка орієнтована на потреби проєкту та забезпечує максимальну зручність для користувачів та розробників. Документація розробляється з урахуванням принципів agile-методології, що дозволяє адаптувати документи до змінюваних вимог та швидко реагувати на зміни у процесі розробки.

Для ведення документації використовуються наступні інструменти:

- Jira [29] та Notion [30] – для створення текстових документів та структурованої інформації;
- GitHub [31] – для зберігання технічної документації, контролю версій коду та API документації;

- Swagger UI - для автоматичної генерації документації для API;
- Figma – для створення інтерфейсів користувача та документування UX/UI рішень;
- Confluence [32] – для зберігання проєктної документації, що містить опис бізнес-процесів, архітектури та етапів розробки.

Види створених документів. Були створені такі документи:

- звіт про огляд ринку та аналіз конкурентів;
- SWOT-аналіз;
- карта проєкту;
- статут проєкту;
- діаграма Ганта;
- матриця Ейзенхаузена;
- RACI-матриця ролей.

Зовнішня документація

Інструкція користувача. При відкритті застосунку користувач потрапляє на сторінку реєстрації нового профілю (рис. 28) або має можливість зайти з уже наявного (рис. 29).

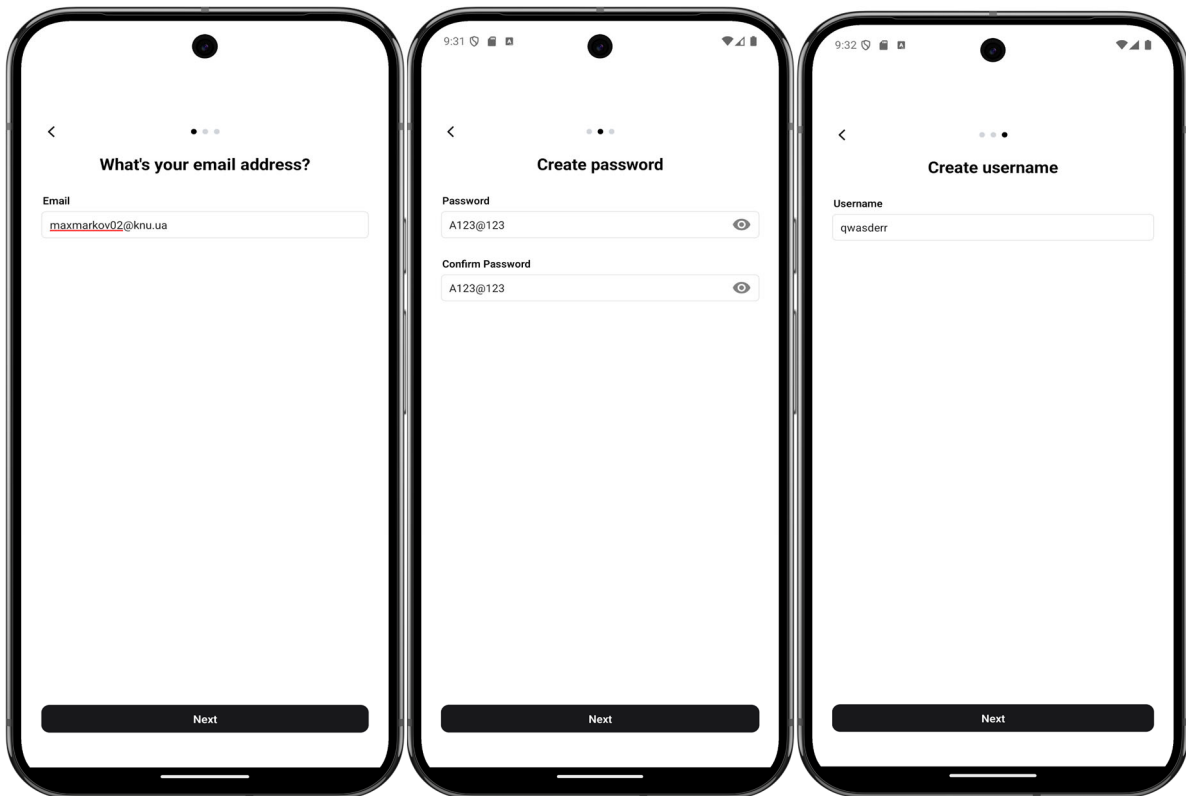


Рис. 28. Процес реєстрації

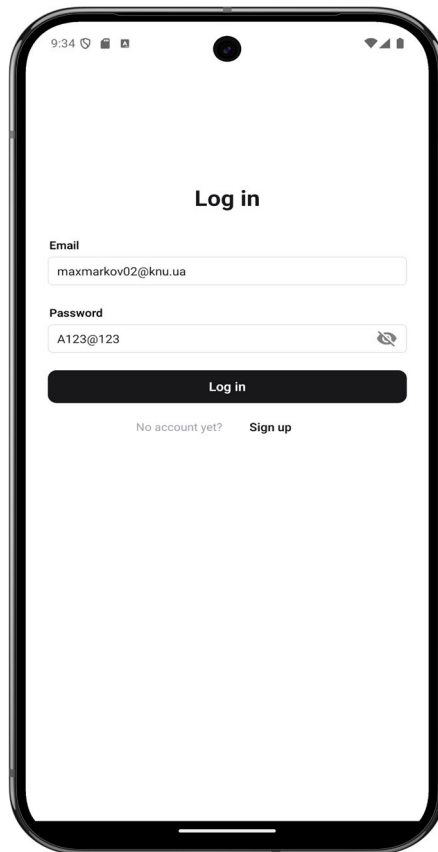


Рис. 29. Вхід у профіль користувача

Після входу користувач потрапляє на основну сторінку. Якщо він не має зареєстрованих тваринок, то йому пропонується екран з кнопкою "Add your first pet"; якщо користувач має тваринку, то йому відображається сторінка з посиланнями на профіль тваринки. Якщо перейти за кнопкою "Add your first pet", користувач опиняється на сторінці, де можна додати тваринку. Для цього потрібно по чергово заповнити усі обов'язкові поля (рис. 30-33)

Також користувач може додати фотографію та короткий опис свого улюбленця (рис. 34).

Профіль тваринки виглядає наступним чином (рис. 35)

Користувач може поділитися профілем тваринки з іншим користувачем. Для цього йому потрібно натиснути кнопку "Share" та ввести електронну пошту того, з ким хоче поділитися.

Іншому користувачу прийде сповіщення, де він може відхилити або прийняти запит.

Користувач, який надсилав запит, отримає сповіщення про схвалення або відмову запиту.

Користувач має можливість редагувати профіль тваринки.

У розділі "Редагувати профіль" користувач може змінити мову застосунку, обравши українську або англійську.

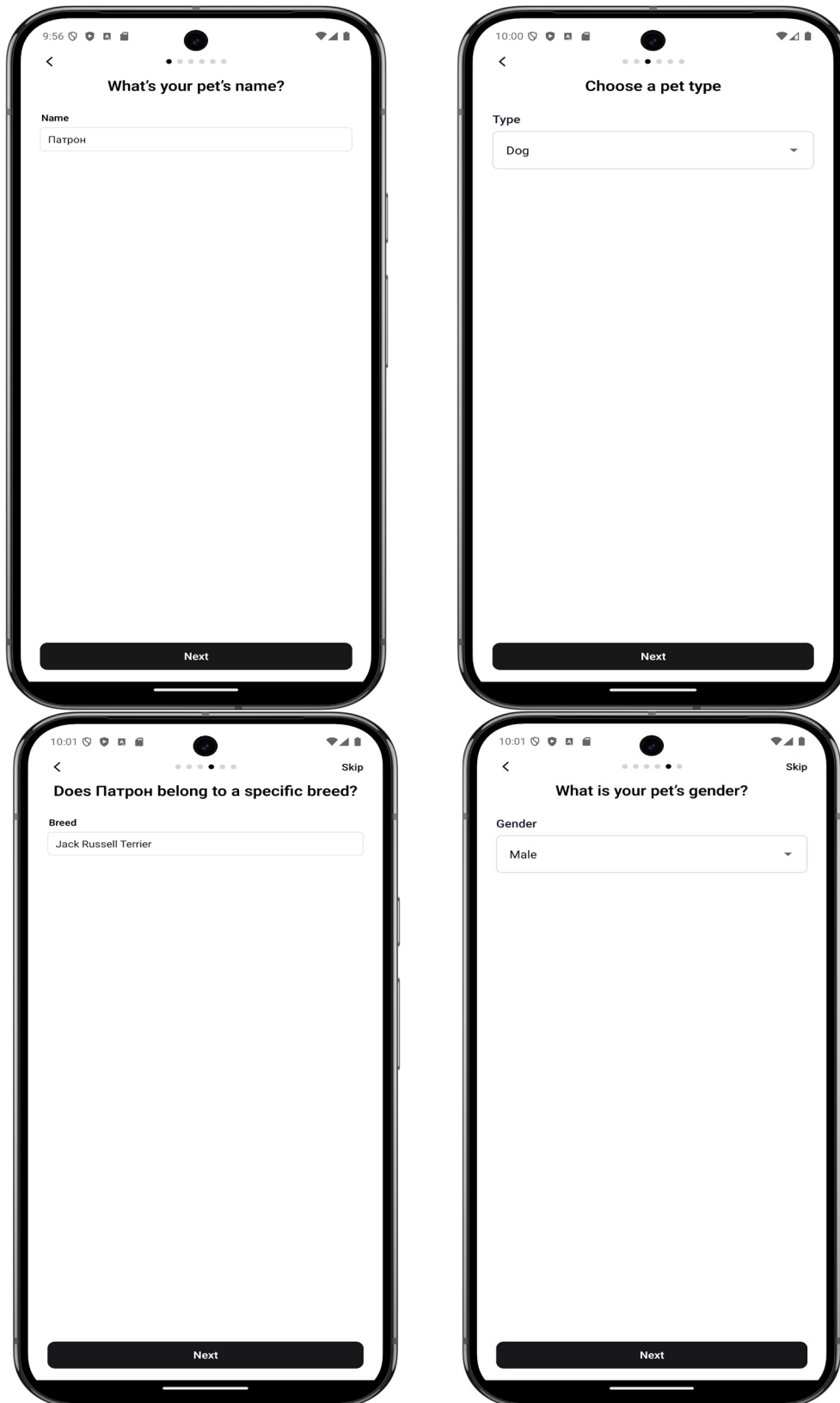


Рис. 30-33. Процес додавання тваринки

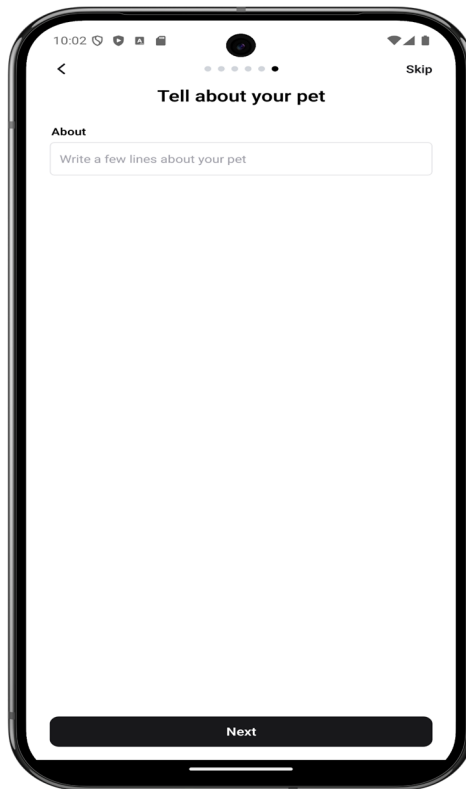


Рис. 34. Додавання опису тваринки

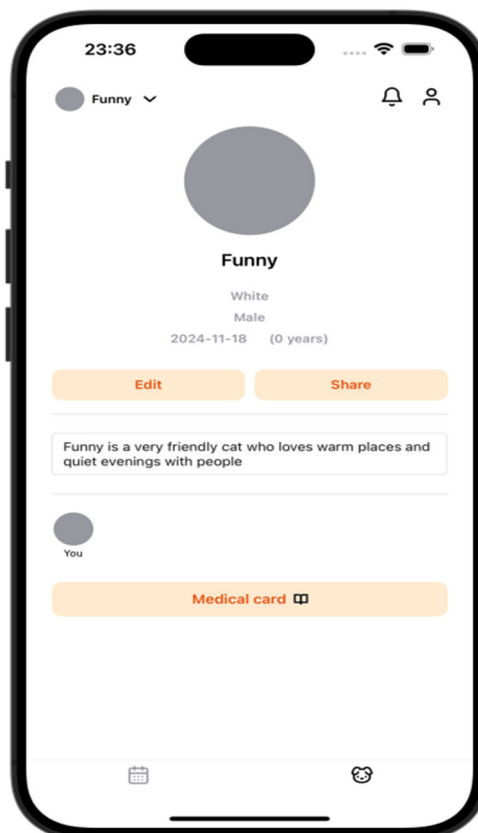


Рис. 35. Профіль тваринки

Правові норми, які регулюють відносини в сфері застосування програмного продукту.

На діяльність, пов'язану з використанням нашого мобільного застосунку для власників тварин, впливають декілька ключових законодавчих актів України. Вони встановлюють правові рамки у сфері захисту тварин, надання ветеринарних послуг, електронного врядування та обробки персональних даних.

"Закон України Про ідентифікацію та реєстрацію тварин" [33] регулює обов'язкову ідентифікацію та облік домашніх і сільськогосподарських тварин. Він визначає порядок реєстрації, обов'язки власників та встановлює вимоги до ведення реєстрів. Постанова Кабінету Міністрів України від 12 липня 2024 р. №811 [34] вносить зміни до нормативних актів щодо надання ветеринарних послуг і вдосконалює процедури реєстрації тварин. Вона забезпечує правову основу для цифровізації процесів у ветеринарній сфері. Постанова Верховної Ради України "Про прийняття за основу проекту Закону України про ветеринарну медицину та благополуччя тварин" [35] визначає основні принципи забезпечення добробуту тварин, етичного ставлення до них та кваліфікованого ветеринарного догляду. Закон України "Про особливості надання публічних (електронних публічних) послуг" [36] регулює процедури надання публічних електронних послуг, а також вимоги до їхньої безпеки, доступності та взаємодії з іншими державними системами. Закон України "Про захист персональних даних" визначає правила обробки, зберігання та захисту персональних даних користувачів [37].

5. УПРАВЛІННЯ ПРОЄКТОМ

Над проєктом працювала команда студентів, кожен відповідав за окремі ролі:

- Олександр Корнієнко - Team Lead, Frontend Developer, DevOps;
- Єгор Бачинський - Backend Developer;
- Наталія Грицишин - Project Manager, Frontend Developer;
- Аріна Забродіна - UI/UX Designer;
- Марков Максим - Tech Lead, Backend Developer;
- Наталія Семйон - Business Analyst, Docs;
- Антон Тіхонов - Backend Developer.

Наша команда вирішила працювати за методологією Scrum, адже вона забезпечує ефективну організацію процесу розробки та командної роботи. Переваги Scrum:

- чіткий розподіл ролей та відповідальності: Scrum передбачає ролі Product Owner, Scrum Master та команди розробників, що дозволяє кожному члену команди розуміти свої обов'язки та цілі. Це сприяє злагодженій роботі та швидкому досягненню результатів;

- ітераційний підхід: робота розподіляється на короткі спринти, що дозволяє регулярно переглядати й покращувати продукт, враховуючи зворотний зв'язок та зміни в умовах проєкту;

- постійне вдосконалення: ретроспективи після кожного спринту дають змогу аналізувати процеси, визначати проблемні місця й впроваджувати покращення, що підвищує ефективність роботи команди;

- фокус на потребах користувача: завдяки backlog'у продукту ми завжди мали змогу орієнтуватися на задачі з найвищим пріоритетом, зосереджуючись на функціоналі, важливому для користувачів;

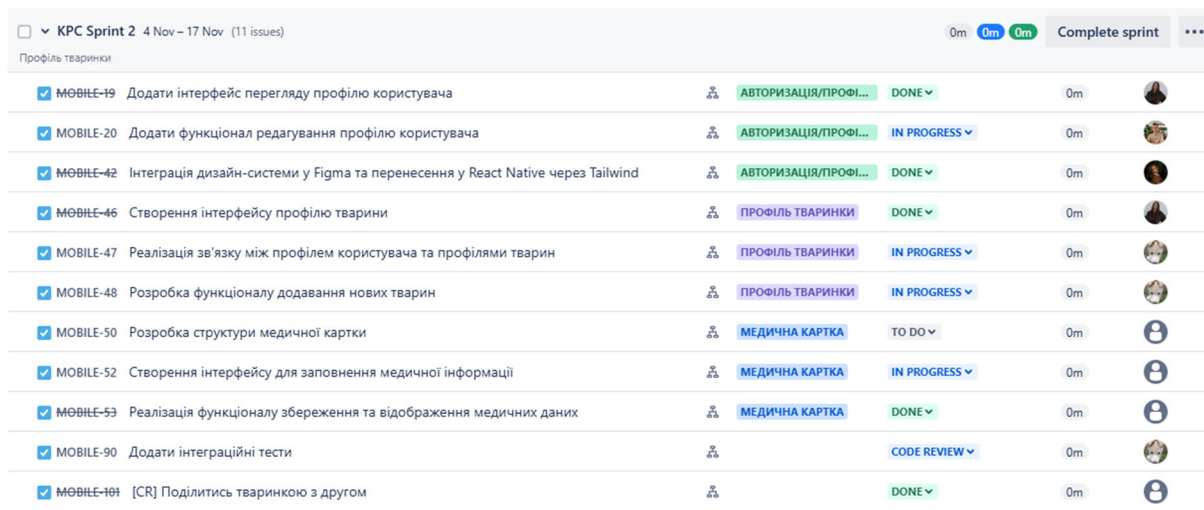
- регулярні Scrum-активності: Щоденні стендапи, планування та ретроспективи дозволяли команді відстежувати прогрес, оперативно розв'язувати проблеми та

допомагати одне одному в разі виникнення труднощів. Це забезпечило ефективну комунікацію та узгодженість дій.

Робота над проєктом була поділена на три спринти та розпочалася з нульового спринту, де був проведений аналіз потенційних користувачів та були визначені основні задачі застосунку.

Під час Sprint 1 команда розробила профіль користувача, під час другого спринту було реалізовано профіль тваринки та медичну картку, під час Sprint 3 було додано можливість ділитися тваринкою, налаштовано сповіщення та вдосконалено профіль користувача.

Планування спринтів велось в Jira. Приклад планування спринту показаний на рис. 36



Task ID	Task Description	Assignee	Status	Progress	Avatar
MOBILE-19	Додати інтерфейс перегляду профілю користувача	АВТОРИЗАЦІЯ/ПРОФІ...	DONE	0m	[Avatar]
MOBILE-20	Додати функціонал редагування профілю користувача	АВТОРИЗАЦІЯ/ПРОФІ...	IN PROGRESS	0m	[Avatar]
MOBILE-42	Інтеграція дизайн-системи у Figma та перенесення у React Native через Tailwind	АВТОРИЗАЦІЯ/ПРОФІ...	DONE	0m	[Avatar]
MOBILE-46	Створення інтерфейсу профілю тварини	ПРОФІЛЬ ТВАРИНКИ	DONE	0m	[Avatar]
MOBILE-47	Реалізація зв'язку між профілем користувача та профілями тварин	ПРОФІЛЬ ТВАРИНКИ	IN PROGRESS	0m	[Avatar]
MOBILE-48	Розробка функціоналу додавання нових тварин	ПРОФІЛЬ ТВАРИНКИ	IN PROGRESS	0m	[Avatar]
MOBILE-50	Розробка структури медичної картки	МЕДИЧНА КАРТКА	TO DO	0m	[Avatar]
MOBILE-52	Створення інтерфейсу для заповнення медичної інформації	МЕДИЧНА КАРТКА	IN PROGRESS	0m	[Avatar]
MOBILE-53	Реалізація функціоналу збереження та відображення медичних даних	МЕДИЧНА КАРТКА	DONE	0m	[Avatar]
MOBILE-90	Додати інтеграційні тести	CODE REVIEW		0m	[Avatar]
MOBILE-101	[CR] Поділитись тваринкою з другом		DONE	0m	[Avatar]

Рис. 36. Приклад планування спринту

У кінці кожного спринту команда збиралася на відеозустрічі, щоб підбити підсумки, провести ретроспективу та обговорити завдання наступного спринту.

У процесі роботи над проєктом наша команда отримала цінний досвід у розробці мобільних застосунків, співпраці в команді та використанні сучасних технологій.

З позитивних моментів виділимо наступні:

- розробка MVP: успішно реалізовано основний функціонал системи (створення профілю користувачів, тварин, медичні картки). Це стало можливим завдяки використанню гнучкої методології розробки, яка дозволила сфокусуватися на першочергових задачах;

- комунікація: регулярні зустрічі та обговорення прогресу допомагали розв'язувати проблеми швидко та знаходити оптимальні рішення;

- залученість кожного члена команди в розробці: кожен учасник проявив ініціативу, починаючи від обговорення ідеї застосунку й закінчуючи виконанням конкретних завдань. Розподіл завдань проводився з урахуванням інтересів і сильних сторін кожного члена команди. Це дозволило залучити всіх у ті аспекти проєкту, в яких вони найбільше зацікавлені або компетентні;

- підтримка та взаємодопомога: якщо хтось із членів команди стикнувся зі складнощами, решта охоче допомагали розібратися. Це створило позитивну атмосферу взаємопідтримки.

Однак не обійшлося і без труднощів. Однією з основних проблем стало ефективне планування навантаження на спринт, що було викликано браком досвіду.

Ми використали такі рішення для вирішення проблеми:

- збільшення комунікації: ми стали більше обговорювати кожну задачу під час планування спринтів, щоб краще зрозуміти обсяг роботи, необхідний для її виконання. Це допомогло краще оцінювати час і ресурси, необхідні для реалізації завдань;

- детальний розбір завдань: під час обговорення кожен учасник детально пояснював свій прогрес та виклики, з якими зіткнувся. Це дозволило команді швидко виявляти проблеми й оперативно їх вирішувати;

- зменшення обсягу завдань на спринт: замість того, щоб намагатися зробити все одразу, ми сфокусувалися на меншій кількості пріоритетних задач, що дозволило завершувати їх вчасно й із високою якістю.

Під час роботи над проектом ми здобули та покращили багато навичок, зокрема:

Технічні навички:

- розробка застосунків на C# та React Native: поглибили знання у цих технологіях, особливо у побудові RESTful API та їхній інтеграції;

- проектування баз даних: здобули досвід у створенні структурованих баз даних із нормалізацією та оптимізацією запитів;

- використання Swagger UI: навчилися автоматично документувати API, що значно спростило тестування та інтеграцію.

Робота в команді:

- покращили навички ефективної комунікації, розподілу задач та взаємодії в умовах обмеженого часу;

- освоїли принципи роботи в команді за методологією Agile, зокрема проведення щотижневих зустрічей та ретроспектив.

Проектний менеджмент:

- зрозуміли важливість чіткого визначення пріоритетів для створення MVP;

- здобули досвід у веденні документації та підготовці технічних звітів.

Дизайн та UX:

- дізналися, як розробляти інтуїтивно зрозумілі інтерфейси, які відповідають потребам цільової аудиторії.

Робота над проектом дала змогу не лише покращити технічні знання, але й зрозуміти, як створювати реальні продукти, що відповідають вимогам користувачів.

ВИСНОВКИ

Основною метою роботи було створення мобільного застосунку для власників домашніх тварин, який об'єднує функції догляду та управління інформацією про тварин. Застосунок реалізує функції створення профілів тварин, ведення медичної картки.

Завдяки аналізу наявних рішень на ринку та опитування потенційних користувачів ми сформувавши структуру та функціонал застосунку, що максимально відповідає потребам цільової аудиторії.

Під час виконання проекту були досягнуті такі цілі:

- проаналізовано ринок застосунків для догляду за тваринами та визначено ключові тенденції;

- розроблено концепцію мобільного застосунку з урахуванням побажань користувачів;

- створено технічне завдання і архітектуру бази даних;

- вдосконалено навички роботи з інструментами розробки C#, React, HTML/CSS, Swagger UI;

- реалізовано функції створення профілю користувача та профілів тварин;

- розроблено медичну картку з можливістю додавання та редагування даних;
- проведено тестування застосунку та виправлено знайдені помилки.

Розробка застосунку дала можливість нашій команді не лише створити якісний продукт, але й здобути нові знання та навички, серед яких:

- планування роботи команди за методологією Scrum;
- поглиблення знань у роботі з базами даних, API, C# та React;
- розробка інтерфейсу користувача з урахуванням принципів UX/UI.

Для подальшого розвитку застосунку ми плануємо такі рішення:

- інтегрувати календар із нагадуваннями для важливих подій;
- додати карту з локаціями ветеринарів, магазинів та інших важливих місць;
- додавання спільноти для обміну досвідом між користувачами;
- розширення функціоналу медичної картки, включаючи історію вакцинацій та рекомендації щодо догляду;
- розробка CRM-системи для ветеринарних клінік.

Результатом роботи став функціональний мобільний застосунок, який відповідає сучасним вимогам і потребам власників домашніх тварин та має потенціал для подальшого розширення та вдосконалення.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. JetVet. URL: <https://jet.vet/uk/> (дата звернення: 20.04.2025).
2. SimilarWeb. URL: <https://www.similarweb.com/> (дата звернення: 20.04.2025).
3. my Pet. URL: <https://mypet.com.gr> (дата звернення: 20.04.2025).
4. DaySmart Vet. URL: <https://daysmart.com/vet/> (дата звернення: 20.04.2025).
5. C Sharp. Wikipedia. URL: https://uk.wikipedia.org/wiki/C_Sharp (дата звернення: 17.04.2025).
6. ASP.NET Core. URL: <https://dotnet.microsoft.com/en-us/apps/aspnet> (дата звернення: 22.04.2025).
7. React Native. URL: <https://reactnative.dev/> (дата звернення: 22.04.2025).
8. React Expo. URL: <https://expo.dev/> (дата звернення: 22.04.2025).
9. TypeScript. URL: <https://www.typescriptlang.org/> (дата звернення: 22.04.2025).
10. Електронний HTML і CSS довідник. URL: <https://html-css.co.ua/> (дата звернення: 22.04.2025).
11. Swagger UI. URL: <https://swagger.io/tools/swagger-ui/> (дата звернення: 22.04.2025).
12. Figma. URL: <https://www.figma.com/> (дата звернення: 22.04.2025).
13. Axios. URL: <https://axios-http.com/uk/docs/intro> (дата звернення: 22.04.2025).
14. React Query. URL: <https://tanstack.com/query/latest/docs/framework/react/overview> (дата звернення: 22.04.2025).
15. React Hook Form. URL: <https://react-hook-form.com/docs/> (дата звернення: 22.04.2025).
16. Tailwind CSS. URL: <https://tailwindcss.com/> (дата звернення: 22.04.2025).
17. NativeWind. URL: <https://www.nativewind.dev/> (дата звернення: 22.04.2025).
18. React-i18next documentation: Introduction. URL: <https://react.i18next.com/> (дата звернення: 22.04.2025).
19. AsyncStorage. URL: <https://reactnative.dev/docs/asyncstorage> (дата звернення: 22.04.2025).
20. ESLint. URL: <https://eslint.org/> (дата звернення: 22.04.2025).
21. Prettier. URL: <https://prettier.io/> (дата звернення: 22.04.2025).
22. Jest. URL: <https://jestjs.io/uk/> (дата звернення: 22.04.2025).
23. Testing Librar. URL: <https://testing-library.com/> (дата звернення: 22.04.2025).

24. OpenAPI Generator CLI. URL: <https://openapi-generator.tech/docs/usage/> (дата звернення: 22.04.2025).
25. Fitts's Law. URL: <https://lawsofux.com/fittss-law/> (дата звернення: 23.04.2025).
26. Hick's Law. URL: <https://lawsofux.com/hicks-law/> (дата звернення: 23.04.2025).
27. Miller's Law. URL: <https://lawsofux.com/millers-law/> (дата звернення: 23.04.2025).
28. Pareto principle. URL: <https://lawsofux.com/pareto-principle/> (дата звернення: 23.04.2025).
29. Jira. URL: <https://www.atlassian.com/software/jira> (дата звернення: 28.04.2025).
30. Notion. URL: <https://www.notion.com/> (дата звернення: 28.04.2025).
31. GitHub. URL: <https://github.com/> (дата звернення: 28.04.2025).
32. Confluence. URL: <https://www.atlassian.com/software/confluence> (дата звернення: 28.04.2025).
33. Про ідентифікацію та реєстрацію тварин : Закон України від 16.10.2016, № 42 : станом на 06.09.2022. URL: <https://zakon.rada.gov.ua/laws/show/1445-17#Text>.
34. Про внесення змін до деяких постанов Кабінету Міністрів України щодо надання послуг у галузі ветеринарної медицини і сфері ідентифікації та реєстрації тварин : Постанова від 12.07.2024, № 811. URL: <https://zakon.rada.gov.ua/laws/show/811-2024-%D0%BF#Text>.
35. Про прийняття за основу проєкту Закону України про ветеринарну медицину та благополуччя тварин : Постанова від 17.06.2020, № 718. URL: <https://zakon.rada.gov.ua/laws/show/718-20#Text>.
36. Про особливості надання публічних (електронних публічних) послуг : Закон України від 22.09.2021, № 47 : станом на 17.07.2024. URL: <https://zakon.rada.gov.ua/laws/show/1689-20#Text> (дата звернення: 20.04.2025).
37. Про захист персональних даних : Закон України від 23.02.2012, № 34. URL: <https://zakon.rada.gov.ua/laws/show/2297-17#Text>.

ЗАСТОСУНОК ДЛЯ ПІДГОТОВКИ ДО НМТ "OSVITA+"

**Олександр СТАРЖИНСЬКИЙ, Владислав ТАРАН,
Катерина ШЕМЕТ, Яна УТОЧКІНА, Іван БОЖЕВСЬКИЙ,
Максим СОМОВ, Євген ЧОПЕНКО, Владислав КУЧИНСЬКИЙ**

Дана робота описує процес розробки застосунку "Osvita+". Метою проєкту є надання інструменту для підготовки до іспиту у форматі НМТ з використанням цифрових технологій. Робота містить детальний опис повного циклу розробки продукту в команді. Описано етапи ініціації ідеї, вибору технологій, власне розробки серверної та клієнтської частин застосунку, тестування застосунку. Також робота описує етап планування задач, побудову комунікацій в команді, методи документації проєкту, правові норми регулювання відносин в предметній області продукту, інструменти аналізу та прийняття рішень.

ВЕБЗАСТОСУНОК, МОБІЛЬНИЙ ЗАСТОСУНОК, ОНЛАЙН-НАВЧАННЯ, НМТ, КОМАНДНА РОЗРОБКА, УПРАВЛІННЯ ПРОЄКТОМ.

APPLICATION FOR NMT PREPARATION "OSVITA+"

**Oleksandr STARZHYNISKYI, Vladyslav TARAN,
Kateryna SHEMET, Yana UTOCHKINA, Ivan BOZHEVSKYI,
Maksym SOMOV, Yevhen CHOPENKO, Vladyslav KUCHYNSKYI**

This work describes the process of developing the application "Osvita+". The goal of the project is to provide a tool for preparing for the exam in the NMT format using digital technologies. The work contains a detailed description of the full cycle of product development in a team. The stages of initiating an idea, choosing technologies, developing the server and client parts of the application, testing the application are described. The work also describes the task planning stage, building communications in the team, project documentation methods, legal norms regulating relations in the subject area of the product, analysis and decision-making tools.

WEB APPLICATION, MOBILE APPLICATION, E-LEARNING, NMT, TEAM DEVELOPMENT, PROJECT MANAGEMENT.

ВСТУП

Постановка задачі. Система освіти в Україні передбачає необхідність складання іспиту у форматі НМТ для вступу у вищий навчальний заклад. Зважаючи на це, результат складання даного іспиту має значний вплив на життя людей в Україні. Майбутній абітурієнт може мати високу успішність у навчанні, але неправильно обрана стратегія підготовки до НМТ унеможливує вступ до омріяного навчального закладу. Задачею проєкту є створення застосунку, що допомагає користувачам підготуватися до іспиту. Застосунок дозволяє учням навчатися онлайн, надаючи рекомендації щодо навчання. В свою чергу вчителям надається змога керувати процесом навчання учнів. Рішення має мобільну та вебверсію для навчання з використанням пристроїв різних типів.

Актуальність теми. Актуальність проекту впливає з процесу впровадження використання електронних пристроїв в навчальному процесі. Також війна в країні ускладнила процес навчання, зокрема підготовку до іспитів. В деяких регіонах навчальний процес можливий лише у форматі онлайн, а тому збільшилася потреба у засобах організації дистанційного навчання. Додатковим фактором є наявність категорії людей, що надають перевагу самонавчанню через інтровертність, але не в змозі організувати своє навчання самостійно. На ринку існують застосунки для дистанційного навчання, але вони не надають спеціалізовану підготовку до іспиту у форматі НМТ.

Методи дослідження. Методи дослідження включають наявних рішень на ринку та виділення необхідних функцій застосунку. Також необхідний аналіз структури тестів у форматі НМТ та процесу тестування в цілому для побудови ефективної стратегії підготовки до іспиту. Додатковим етапом є опитування експертів предметної області, зокрема вчителів. Також є доцільним проведення опитувань потенційних користувачів розроблюваного застосунку – майбутніх абітурієнтів.

Мета та завдання роботи. Метою роботи є створення інструменту, що допомагатиме організувати процес підготовки до іспиту у форматі НМТ. Завданнями роботи є розробка мобільної та вебверсії застосунку. Також метою проекту є покращення практичних навичок розробки з використанням обраних технологій, отримання досвіду роботи в команді над проектом за обраною методологією, отримання навичок ініціації, планування проекту, ознайомлення з правовими нормами, що регулюють відносини в сфері застосування продукту.

1. ПОШУК ТА ВАЛІДАЦІЯ ІДЕЇ

Опис пошуку ідеї продукту. На цьому етапі висунуто по 1-2 ідеї від кожного учасника команди. З цих ідей обрано перспективні та виконано SWOT-аналіз. Нижче наведено ідеї, що були розглянуті.

Продукт для управління фінансами. Перша ідея була спрямована на розробку застосунку для управління особистими та спільними фінансами. Основними функціями застосунку мали стати розподіл витрат між групою людей, ведення статистики власних витрат за категоріями типів покупок. Було створено Lean Canvas до даної ідеї, що зображений на рис. 1.



Рис. 1. Lean Canvas продукту для управління фінансами

Під час розгляду даної ідеї проведено SWOT-аналіз:

- сильні сторони ідеї: одна платформа для багатьох задач замість 5 додатків;
- слабкі сторони: продукт не може все автоматизувати, потрібне втручання користувача;
- ризики: сайт буде менш відвідуваним, висока конкуренція на ринку;
- можливості: простір для інтеграцій.

Після аналізу було вирішено відмовитися від ідеї, оскільки ризики та слабкі сторони переважали над можливостями та сильними сторонами.

Продукт для моніторингу цін на продукти. Друга ідея мала на меті розробити застосунок для перегляду та порівняння цін на продукти в різних магазинах. Основною функцією застосунку була б можливість моніторингу знижок на певні товари й обрання місця найвигіднішої покупки. На рисунку 2 зображено Lean Canvas що описує дану ідею.



Рис. 2. Lean Canvas продукт для моніторингу цін на продукти

Виконано огляд, що включає SWOT-аналіз:

- сильні сторони ідеї: можливість переглядати ціни на продукти в різних супермаркетах в одному місці, мала конкуренція, актуальність в економічній ситуації в країні;
- слабкі сторони: залежність від актуальності інформації зовнішніх ресурсів;
- ризики: важко слідкувати за актуальністю даних, потенційна відсутність API супермаркетів, актуальність лише в містах з наявністю вибору супермаркетів;
- можливості: простір для інтеграцій нового функціоналу.

Провівши огляд, було вирішено відмовитися від ідеї, оскільки залежність від сторонніх API та актуальність, що залежить від місця проживання користувачів, переважають над сильними сторонами.

Платформа для продажу і покупки автомобілів. Наступною ідеєю була розробка платформи для продажу й покупки автомобілів. Платформа мала надавати змогу покупцям перевіряти достовірність інформації про автомобілі, що продаються. В свою чергу продавці отримали б майданчик для продажу автомобілів. Ідею описано за допомогою Lean Canvas, що зображений на рисунку 3.

Для оцінки ідеї проведено SWOT-аналіз:

- сильні сторони ідеї: зручна та інтуїтивна платформа з мінімалістичним дизайном, перевірка історії автомобіля за допомогою блокчейн-технологій, індивідуальні рекомендації для користувачів на основі їхніх вподобань;
- слабкі сторони: неконкурентоспроможний функціонал в MVP версії;
- ризики: висока конкуренція;
- можливості: простір для інтеграцій нового функціоналу.

Зважаючи на результати аналізу ідею було відкинуто, оскільки на ринку даного типу продукту наявна висока конкуренція, що представлена конкурентом-монополістом.



Рис. 3. Lean Canvas платформи для продажу і покупки автомобілів

Застосунок для підготовки до НМТ. Остання ідея мала на меті розробку застосунку, що допомагав би учням готуватися до НМТ. Застосунок мав надавати учням можливість вивчати матеріали та проходити тестування, що спрямовані на підготовку до іспиту. Також застосунок мав надавати змогу вчителям керувати навчання учнів. За допомогою Lean Canvas, що зображений на рис. 4, описано дану ідею.



Рис. 4. Lean Canvas застосунку для підготовки до НМТ

Проведено SWOT-аналіз для дослідження ідеї:

- сильні сторони ідеї: постійний попит на даний тип продукту;
- слабкі сторони: необхідність кваліфікації співробітників для генерації контенту;
- ризики: конкуренція, складність підготовки контенту;
- можливості: простір для інтеграцій нового функціоналу.

Дана ідея сподобалася всім учасникам команди. Після аналізу було обрано саме цю ідею для реалізації, зважаючи на кваліфікацію членів команди та перспективність ідеї.

Огляд наявних на ринку ІТ-продуктів – конкурентів

Для створення корисного продукту був проведений огляд наявних на ринку готових рішень. Під час огляду було розглянуто функціонал, підходи до навчання, наявність контенту саме для підготовки до НМТ та варіанти монетизації продуктів.

Онлайн-школа математики "GIOS". Основний акцент на навчанні учнів 5-9 класів. В контексті НМТ має набір з 10 тестів у форматі НМТ, що охоплюють лише математику [1].

Основний функціонал:

– вивчення тем з математики для 5-9 класів, що містить відео до тем, короткий конспект після лекцій до відповідних занять;

– пошук репетитора;

– тренажер до НМТ – проходження тестів у форматі НМТ;

Увесь функціонал є платним. Ціна за доступ до тренажеру з НМТ – 500 гривень.

Переваги:

– має 10 створених тестів у форматі НМТ.

Недоліки:

– замало контенту для підготовки до НМТ;

– присутня лише математика.

Курси підготовки до НМТ "ZNOUA". Основний акцент на підготовці до НМТ. Надає доступ до платформи, де знаходяться лекції тривалістю 1-2 години, матеріал та тести до різних тем [2].

Основний функціонал:

– проходження тестів у форматі ЗНО та НМТ;

– проходження тестів за темами;

– перегляд відеолекцій за темами;

– перегляд конспектів та презентацій за лекціями;

1. Увесь функціонал є платним. Доступ до платформи коштує 3900 гривень.

Переваги:

– тести та курси спрямовані на підготовку до НМТ;

Недоліки:

– довгі лекції;

– увесь функціонал платний.

Онлайн-платформа "ОСВІТА.UA". Основний акцент на проходженні минулорічних тестів ЗНО та НМТ. Всі тести у вільному доступі, платформа автоматично перевіряє правильність наданих відповідей [3].

Основний функціонал:

– проходження попередніх тестів ЗНО та НМТ;

– проходження тестів з одної теми з питань, що були на ЗНО та НМТ.

Платний функціонал:

– перегляд пояснення до завдань;

– збереження свого прогресу.

Ціна платного функціоналу – 999 гривень.

Переваги:

- спеціалізується на ЗНО та НМТ: надає доступ до новин, рейтингів ВНЗ;
- можна без реєстрації використовувати безкоштовний функціонал.

Недоліки:

- велика кількість реклами.

Онлайн-платформа "МійКлас". Основний акцент на самостійному вивченні шкільного матеріалу та спрощенню онлайн-взаємодії вчителів та учнів. Має розподіл за предметами і темами. Також надає можливість створювати акаунти викладачів, які проходять перевірку, та після перевірки мають доступ до перегляду успішності учнів тієї ж школи. Вчителі можуть створювати тести, перевірочні роботи, свої предмети, використовуючи матеріали з сайту або додаючи свої завдання [4].

Основний функціонал учня:

- проходження тестів для тем 1-11 класів з багатьох предметів, зокрема для підготовки до ЗНО та НМТ;
- матеріали до тем в короткій формі;
- виконання тестів;
- можливість проходити минулорічні ЗНО та НМТ.

Основний функціонал вчителя:

- створення авторського предмету у редакторі курсів з можливістю використання завдань, що наявні в МійКлас або створюючи власні;
- створення перевірочних робіт та тестів.

Додатковий функціонал:

- перегляд вебінарів за додатковими темами;
- організація турнірів з призами;

Більшість функціоналу є платним.

Переваги:

- має велику кількість тестів за темами всього шкільного курсу;
- має тести ЗНО та НМТ за всі минулі роки;
- має від 5 до 11 тестів у форматі НМТ з основних предметів;

Недоліки:

- більшість функціоналу за платною підпискою.

Мапа продукту

Проблема, яку вирішує продукт. На ринку освітніх платформ наявна мала кількість рішень, що спеціалізуються на підготовці до НМТ. Наявні рішення зазвичай мають на меті надати можливість вивчати матеріал всієї шкільної програми, що є надлишковим під час підготовки до іспиту. Продукт, що розроблено, вирішує проблему підготовки до НМТ в режимі онлайн, зокрема для осіб, що надають перевагу самостійній підготовці через інтровертність.

Сегменти клієнтів. Під час складання мапи продукту виділено сегменти потенційних клієнтів:

- учні 9-11 класів, які хочуть успішно скласти НМТ;
- дорослі, які хочуть вступити до вищого навчального закладу;
- учні-інтроверти 10-11 класів, які звикли навчатись самостійно.

Унікальна ціннісна пропозиція. Застосунок орієнтований саме на підготовку до іспиту у форматі НМТ. Під час процесу підготовки використовуються різні методики навчання: використання технік інтервального повторення, елементи гейміфікації для заохочування до навчання. Також застосунок використовує ШІ як інструмент для допомоги у процесі навчання.

Рішення проблеми. Використовуючи застосунок, учні отримують інструмент для підготовки до НМТ в режимі онлайн у вигляді вивчення матеріалів, проходження тестів та аналізу власного прогресу. З іншого боку, вчителі отримують інструмент для керування процесом навчання учнів.

Канали просування. Для просування застосунку доречно використати канали просування:

- реклама в соціальній мережі Facebook [5];
- реклама в соціальній мережі Instagram [6];
- контекстна реклама Google Ads [7];
- реклама на платформі YouTube [8].

Джерела доходів. Можливі варіанти монетизації застосунку:

- підписка на сервіс – виділення частини платного функціоналу за підпискою;
- відображення реклами в безкоштовній версії.

Ключові метрики. Для аналізу функціонування продукту обрано метрики:

- MAU – число активних користувачів за місяць;
- LTV – загальна сума яку витрачає клієнт на продукт;
- SAC – вартість залучення одного клієнта.

Неринкова перевага. Маючи доступ до цільової аудиторії через викладачів, команда мала можливість тестувати новий функціонал та проводити опитування серед потенційних користувачів застосунку, що підвищило успіх фінальної версії продукту.

Сильні і слабкі сторони розроблюваного продукту. Виділено сильні та слабкі сторони продукту на основі попереднього аналізу.

Сильні сторони:

- наявний постійний попит на даний тип продукту;
- простір для інтеграцій нового функціоналу;

Слабкі сторони:

- необхідність кваліфікації співробітників для генерації контенту;
- наявна конкуренція на ринку;

Огляд використаних технологій

Розглянуто технології, які було обрано для створення застосунку для підготовки до НМТ. Для проєктування використано інструменти:

- App.diagrams.net – інструмент для створення діаграм та схем [9];
- Figma – інструмент для створення дизайну застосунку [10].
- Для розробки використано технології:

– .NET – платформа для створення програмного забезпечення, використано для створення серверної частини застосунку [11];

- ASP.NET Core – фреймворк для веброзробки на платформі .NET [12];
- React – фреймворк для створення клієнтської частини застосунку [13].

Для тестування використано технології: Nunit, Moq, Manual testing, RTL, Vitest, Appium.

Технології обрано зважаючи на їхню актуальність на ринку та досвід членів команди у використанні технологій.

2. ОПИС ПРОГРАМНОГО ПРОДУКТУ

Призначення створення застосунку

Застосунок створено для надання можливості підготовки до НМТ онлайн. Метою було створення застосунку, що дозволяє учням самостійно підготуватися до НМТ, а також вчителям керувати процесом навчання учнів.

Також застосунок дозволяє користувачам створювати акаунти типів учня або вчителя, вивчати обраний контент, проходити тестування, слідкувати за прогресом навчання.

Вимоги до застосунку

Застосунок розроблено з метою полегшення процесу підготовки до НМТ. Це включає можливість вивчати матеріали онлайн, проходити тестування. Застосунок містить елементи гейміфікації (щоденні завдання, рейтингова таблиця), що заохочують учня до навчання. Застосунок дозволяє створити акаунт, використовуючи електронну пошту. На рис. 5 зображено Use Case діаграму, що відображає можливості учня.

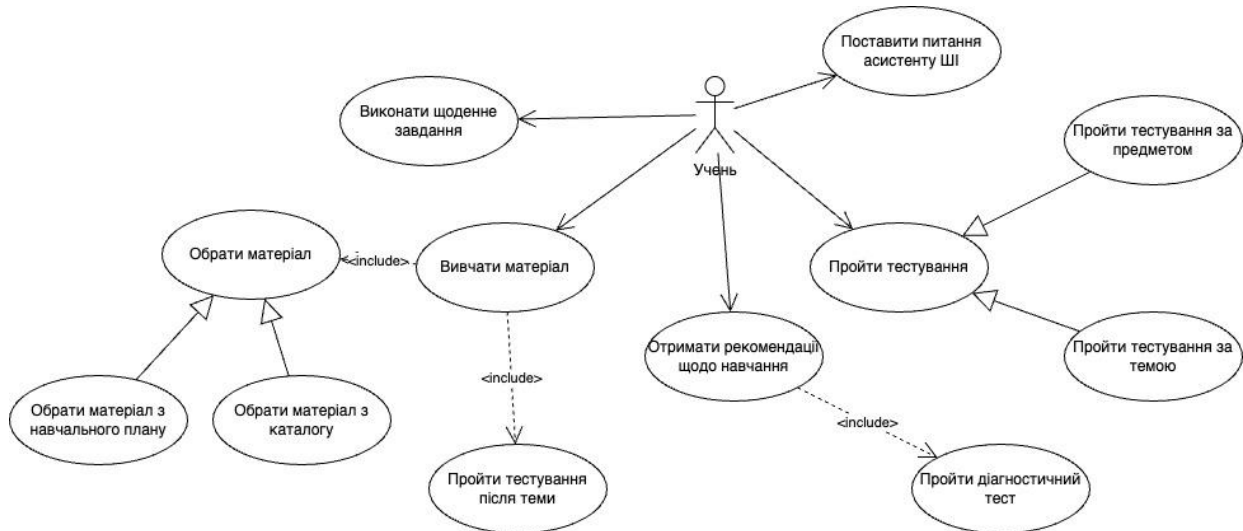


Рис. 5. Use Case діаграма учня

Застосунок дозволяє вчителям керувати процесом учнів, призначаючи завдання та переглядаючи успішність виконання. На рисунку 6 зображено Use Case діаграму, що відображає можливості вчителя.

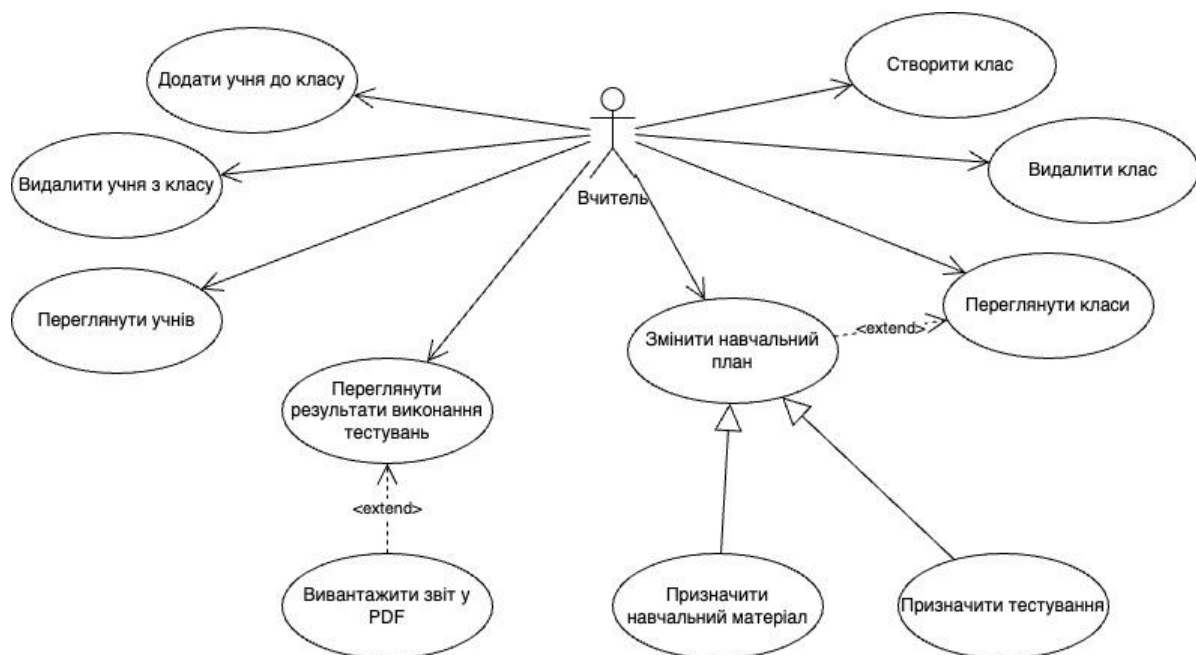


Рис. 6. Use Case діаграма вчителя

Застосунок має адміністративну панель, що дозволяє адміністраторам додавати навчальний контент та тести. На рисунку 7 зображено Use Case діаграму, що відображає можливості адміністратора.

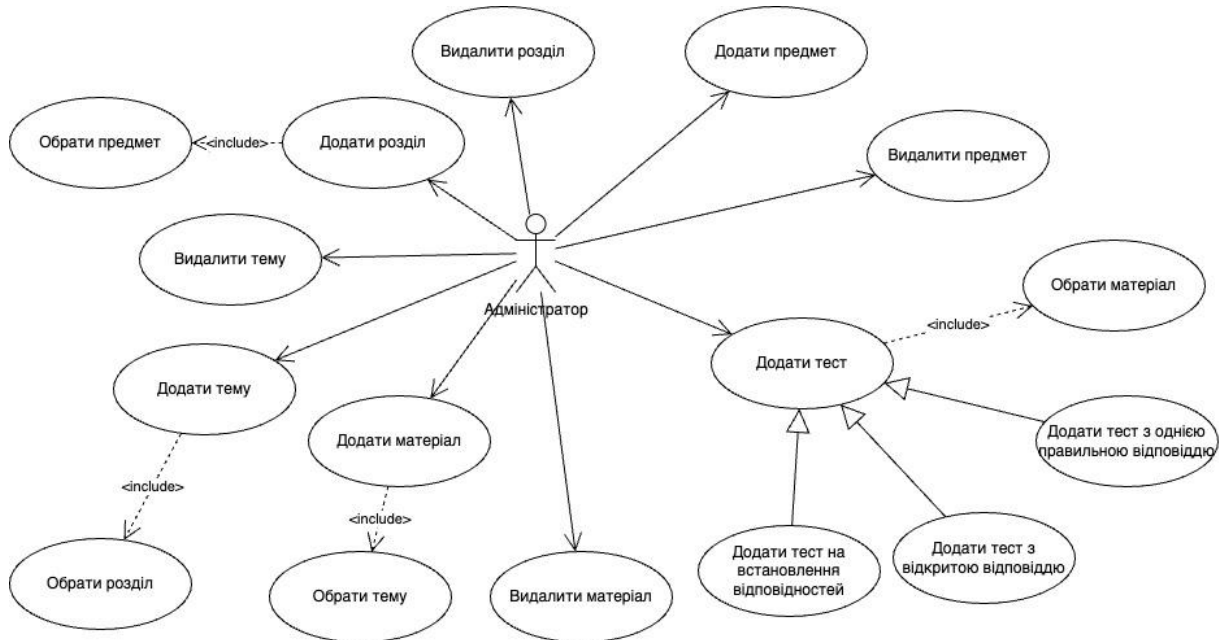


Рис. 7. Use Case діаграма адміністратора

Вимоги до функцій, які виконуються застосунком

Застосунок надає функціонал для вивчення матеріалів, проходження тестування учнями та керування процесом навчання учнів вчителями. Також застосунок надає можливість адміністраторам керувати навчальним контентом.

В застосунку виділено наступні функціональні підсистеми:

- підсистема учня: призначена для вивчення навчальних матеріалів та проходження тестування;
- підсистема вчителя: призначена для слідкування за прогресом навчання учнів та керування ним;
- підсистема адміністратора: призначена для керування контентом.

Підсистеми учня, вчителя та адміністратора мають функціонал, що наведений в таблицях 1-3.

Таблиця 1. Перелік функцій учнів

Функція	Задача
Робота з навчальними матеріалами	Перегляд навчальних матеріалів
	Додавання навчального матеріалу до навчального плану
Робота з тестами	Проходження тестування до предмету
	Проходження тестування до теми
	Проходження діагностичного тестування
	Проходження щоденного тестового завдання
Робота із прогресом навчання	Перегляд статистики проходження тестувань
	Вивантаження статистики проходження тестувань у форматі PDF

	Перегляд кількості днів проходження щоденного тестового завдання без пропуску
	Перегляд рейтингової таблиці успішності учнів проходження щоденних завдань
Робота з класами	Перегляд призначених тем для класу
	Перегляд та виконання призначених тестувань для класу
Робота із рекомендаціями	Отримання рекомендацій щодо вивчення тем після проходження діагностичного тесту
	Отримання щоденних рекомендацій щодо повторення тем
Робота з акаунтом	Реєстрація акаунту
	Введення інформації про себе
	Оновлення інформації
	Вступ до класу за запрошенням

Таблиця 2. Перелік функцій вчителів

Функція	Задача
Робота з учнями	Перегляд учнів
	Додавання учнів до класу
	Видалення учнів з класу
Робота з класами	Створення класів
	Видалення класів
	Перегляд класів
Робота з навчальними планами	Призначення навчальний матеріал учню
	Призначення тестування учню
Робота з акаунтом	Реєстрація акаунту
	Введення інформації про себе
	Оновлення інформації

Таблиця 3. Перелік функцій адміністраторів

Функція	Задача
Робота з навчальними матеріалами	Перегляд навчальних матеріалів
	Створення навчальних матеріалів
	Редагування навчальних матеріалів
	Видалення навчальних матеріалів
Робота з тестами	Перегляд тестів
	Створення тестів
	Редагування тестів
	Видалення тестів
Робота з користувачами	Перегляд користувачів
	Редагування користувачів
	Видалення користувачів
Робота з акаунтом	Введення інформації про себе
	Оновлення інформації

Системні вимоги

Виділено системні вимоги до застосунку, які були обов'язковими при розробці. Застосунок коректно відображається на найбільш популярних пристроях в інтернет-браузерах Google Chrome, Mozilla Firefox, Apple Safari, Opera версій, що створені 2020 року і пізніше.

Також на довільні некоректні дії користувача, які можуть бути оброблені системою, генеруються відповідні повідомлення про помилки українською мовою, в межах загального дизайну застосунку. Застосунок слідкує за доступом до частин, до яких має право звертатися лише певна група користувачів, блокуючи несанкціоновані переходи до цих секцій.

Систему розроблено з використанням технологій .NET та React. Джерелом даних для застосунку обрано СКБД MS SQL Server [14].

Опис інформаційної бази

Логічна структура бази даних

Для збереження інформації створено базу даних з використанням СКБД MS SQL Server. Схема діаграми класів, що відповідає частині створеної бази даних, яка зберігає дані про навчальний контент зображена на рис. 8.

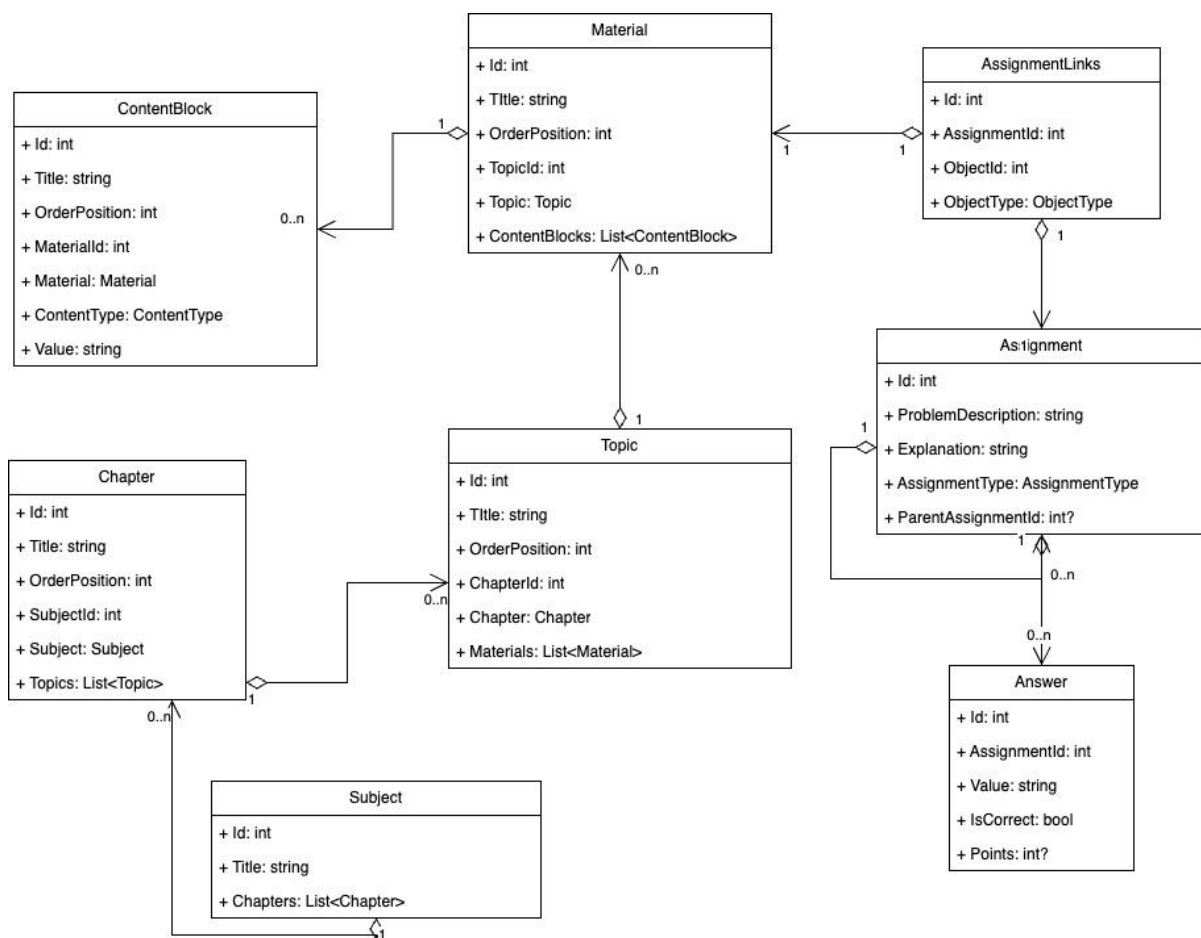


Рис. 8. Діаграма класів

Опис призначення таблиць бази даних, наведено у таблиці 4.

Детальний опис даних для таблиць, які зберігають дані про навчальний контент наведено в таблиці 5.

Також створена друга база даних для збереження інформації про користувачів, що використовується механізмом ASP.NET Identity.

Таблиця 4. Опис призначення таблиць бази даних

Номер	Таблиця	Опис
1	Subjects	Таблиця для збереження інформації про предмети
2	Chapters	Таблиця для збереження інформації про розділи
3	Topics	Таблиця для збереження інформації про теми
4	Materials	Таблиця для збереження інформації про навчальні матеріали
5	ContentBlocks	Таблиця для збереження інформації про блоки контенту навчальних матеріалів
6	Assignments	Таблиця для збереження інформації про тести
7	Answers	Таблиця для збереження інформації про відповіді до тестів
8	AssignmentLinks	Таблиця для збереження інформації про приналежності тесту
9	AssignmentSets	Таблиця для збереження інформації про набір тестів у тестуванні
10	Users	Таблиця для збереження інформації про користувачів
11	EducationPlans	Таблиця для збереження інформації про навчальні плани учнів
12	TopicPlanDetails	Таблиця для збереження інформації про теми в навчальному плані
13	AssignmentSetPlanDetails	Таблиця для збереження інформації про тестування в навчальному плані
14	EducationClasses	Таблиця для збереження інформації про навчальні класи
15	EducationClassPlans	Таблиця для збереження інформації про навчальні плани класів
16	Statistics	Таблиця для збереження інформації про статистику виконання завдань користувачем
17	TopicProgressDetails	Таблиця для збереження інформації про вивчення тем користувачем
18	AssignmentSetProgressDetails	Таблиця для збереження інформації про проходження тестувань користувачем
19	AssignmentProgressDetails	Таблиця для збереження інформації про проходження завдань у тестуванні
20	RecomendationMessages	Таблиця для збереження інформації про повідомлення з рекомендаціями
21	EducationClassInvitations	Таблиця для збереження інформації про запрошення учнів до класу
22	EducationClassUser	Таблиця для збереження інформації про зв'язки учнів та класів

Таблиця 5. Опис атрибутів таблиць бази даних

Атрибут	Тип	Опис
Таблиця Subjects		
Id	int	Ідентифікатор предмету
Title	nvarchar(MAX)	Заголовок предмету
Таблиця Chapters		
Id	int	Ідентифікатор розділу
Title	nvarchar(MAX)	Заголовок розділу
OrderPosition	int	Позиція у списку розділів
SubjectId	int	Ідентифікатор предмету

Таблиця Topics		
Id	int	Ідентифікатор теми
Title	nvarchar(MAX)	Заголовок теми
OrderPosition	int	Позиція у списку тем
ChapterId	int	Ідентифікатор розділу
Таблиця Materials		
Id	int	Ідентифікатор матеріалу
Title	nvarchar(MAX)	Заголовок матеріалу
OrderPosition	int	Позиція у списку матеріалів
ChapterId	int	Ідентифікатор теми
Таблиця ContentBlocks		
Id	int	Ідентифікатор блоку контенту
Title	nvarchar(MAX)	Заголовок блоку контенту
Value	nvarchar(MAX)	Значення блоку контенту
OrderPosition	int	Позиція у списку блоків контенту
ContentType	int	Тип контенту
MaterialId	int	Ідентифікатор матеріалу
Таблиця Assignments		
Id	int	Ідентифікатор тесту
ProblemDescription	nvarchar(MAX)	Питання тесту
Explanation	nvarchar(MAX)	Пояснення до тесту
AssignmentType	int	Тип тесту
ParentAssignmentId	int	Ідентифікатор батьківського тесту
Таблиця AssignmentLinks		
Id	int	Ідентифікатор приналежності тесту
AssignmentId	int	Ідентифікатор тесту
ObjectId	int	Ідентифікатор об'єкту
ObjectType	int	Тип об'єкту
Таблиця Answers		
Id	int	Ідентифікатор відповіді до тесту
Value	nvarchar(MAX)	Значення відповіді до тесту
IsCorrect	bool	Правильність відповіді
Points	int	Кількість балів за відповідь
AssignmentId	int	Ідентифікатор тесту

UI/UX-дизайн продукту

Під час створення дизайну продукту використано принципи та підходи, що підвищують ефективність взаємодії із користувачем.

Принцип узгодженості. Принцип узгодженості передбачає, що елементи інтерфейсу користувача повинні підтримувати однаковість у всіх інтерфейсах. Це покращує відчуття контролю користувача та зменшує когнітивне навантаження. Користувачі можуть швидко орієнтуватися в додатку, що забезпечує більш легку взаємодію.

На всіх сторінках присутні подібні елементи інтерфейсу користувача, такі як кнопки навігації, блоки для складання тестів та їх вміст, зберігають узгоджений стиль і поведінку.

На головній сторінці, що зображена на рис. 9, стилі кнопок ("Розпочати підготовку" та "Увійти") мають узгоджений дизайн на всіх сторінках, забезпечуючи однаковість.

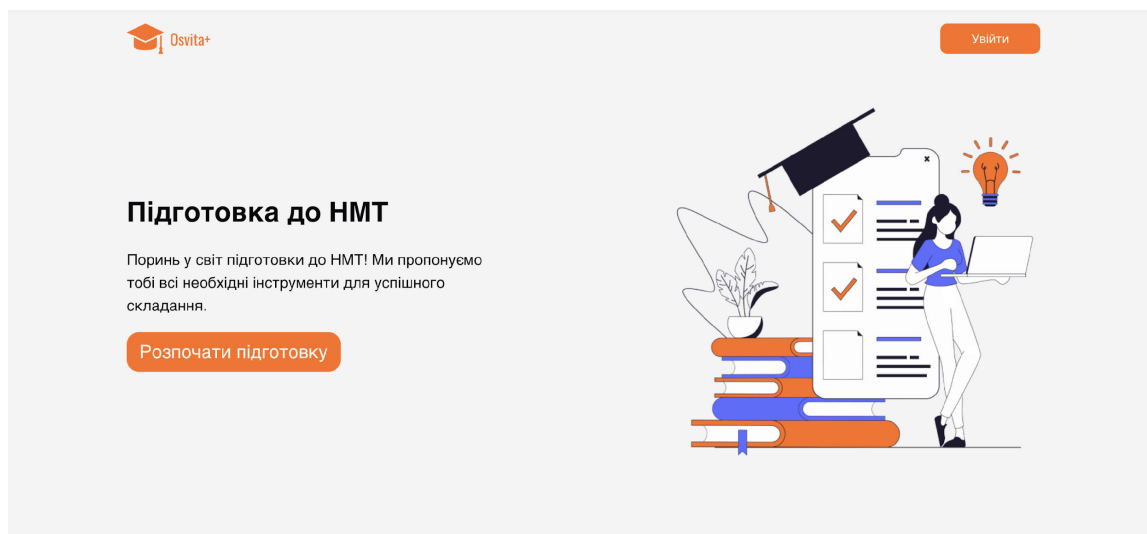


Рис. 9. Приклад принципу узгодженості стилю для кнопок

Ця узгодженість поширюється на всі форми, такі як вхід і реєстрація, що зображені на рис. 10, гарантуючи, що користувачі стикаються зі знайомими елементами інтерфейсу.

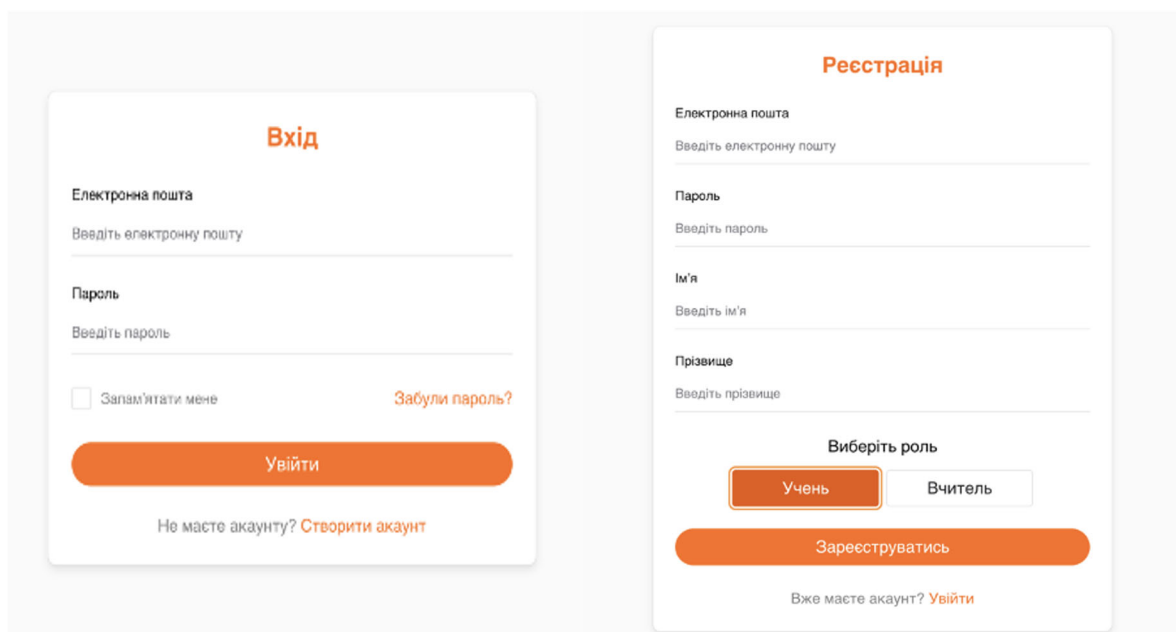


Рис. 10. Приклад принципу узгодженості на формах входу та реєстрації

Зворотній зв'язок і доступність. Закон зворотного зв'язку вказує на те, що користувачі повинні отримувати негайний візуальний, слуховий або тактильний зворотний зв'язок у відповідь на свої дії, щоб зрозуміти результат.

Коли користувач завершує тест, він отримує негайний зворотний зв'язок – відображення результатів тесту та правильних/неправильних відповідей. Приклад закону зворотного зв'язку зображено на рис. 11.

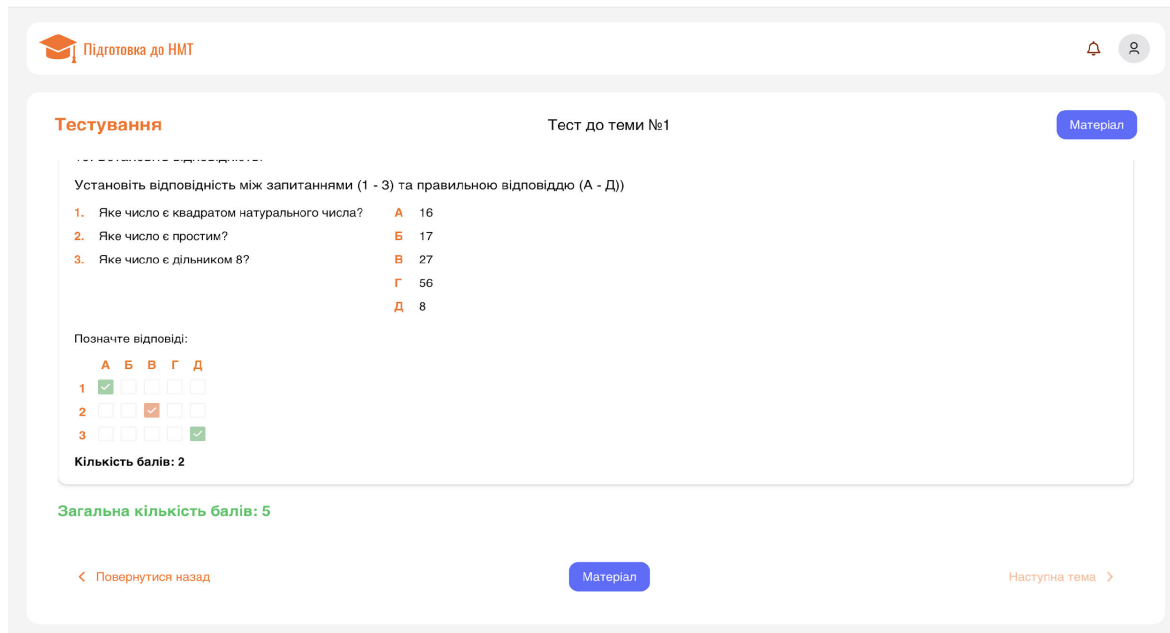


Рис. 11. Приклад закону зворотного зв'язку

Цей принцип запевняє користувачів, що їхні дії підтверджуються системою, і що вони можуть довіряти програмі, яка надасть корисну та своєчасну інформацію.

Закон Хіка. Закон Хіка стверджує, що час, необхідний для прийняття рішення, збільшується разом із кількістю та складністю вибору. Зменшення кількості опцій покращує швидкість прийняття користувачем рішень.

Тести згруповані за темами, а навігація є лінійною, щоб уникнути перевантаження користувачів надто великою кількістю варіантів одночасно. Приклад закону Хіка зображено на рис. 12.

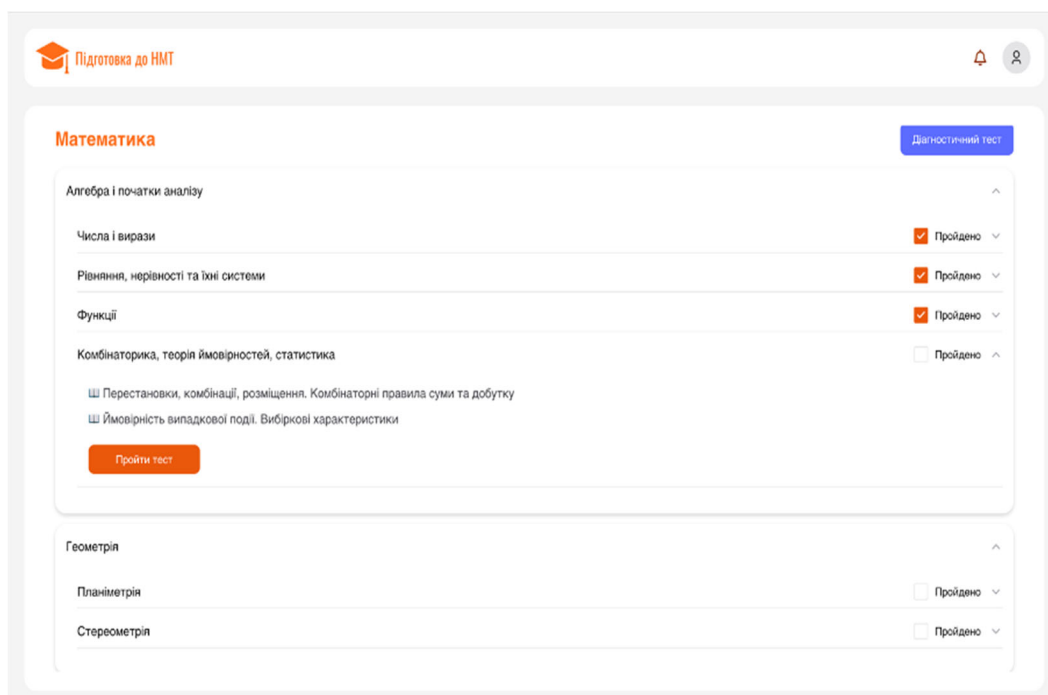


Рис. 12. Приклад закону Хіка

Обмеження видимих параметрів на кожному етапі допомагає користувачам робити швидкий і впевнений вибір, що покращує взаємодію з користувачем, особливо, коли під час підготовки до тесту бракує часу.

Принцип поступового розкриття інформації. Цей принцип передбачає, що користувачеві в будь-який момент часу слід показувати лише необхідну інформацію, а за потреби розкривати більш детальний зміст.

Дизайн дає змогу поступово розкривати зміст користувачам, починаючи з широких тем, як-от предметів, і звужуючи до конкретних тем з матеріалами та тестами.

Цей підхід допомагає уникнути перевантаження користувачів надто великою кількістю інформації одночасно, дозволяючи їм зосередитися на поточному завданні та поступово залучатися до більш детального змісту, якщо це необхідно.

Принцип контролю та свободи користувача. Цей принцип стверджує, що користувачі повинні завжди відчувати контроль над системою та мати можливість легко скасувати або повторити дії.

Дизайн дозволяє користувачам легко переміщатися між основним інтерфейсом складання тесту та іншими областями, як-от навчальний матеріал. Наявність таких функцій, як кнопки "Повернутися назад" і можливість пропускати або повертатися до попередніх розділів, забезпечує свободу та гнучкість. Приклад принципу контролю та свободи користувача зображено на рис. 13.

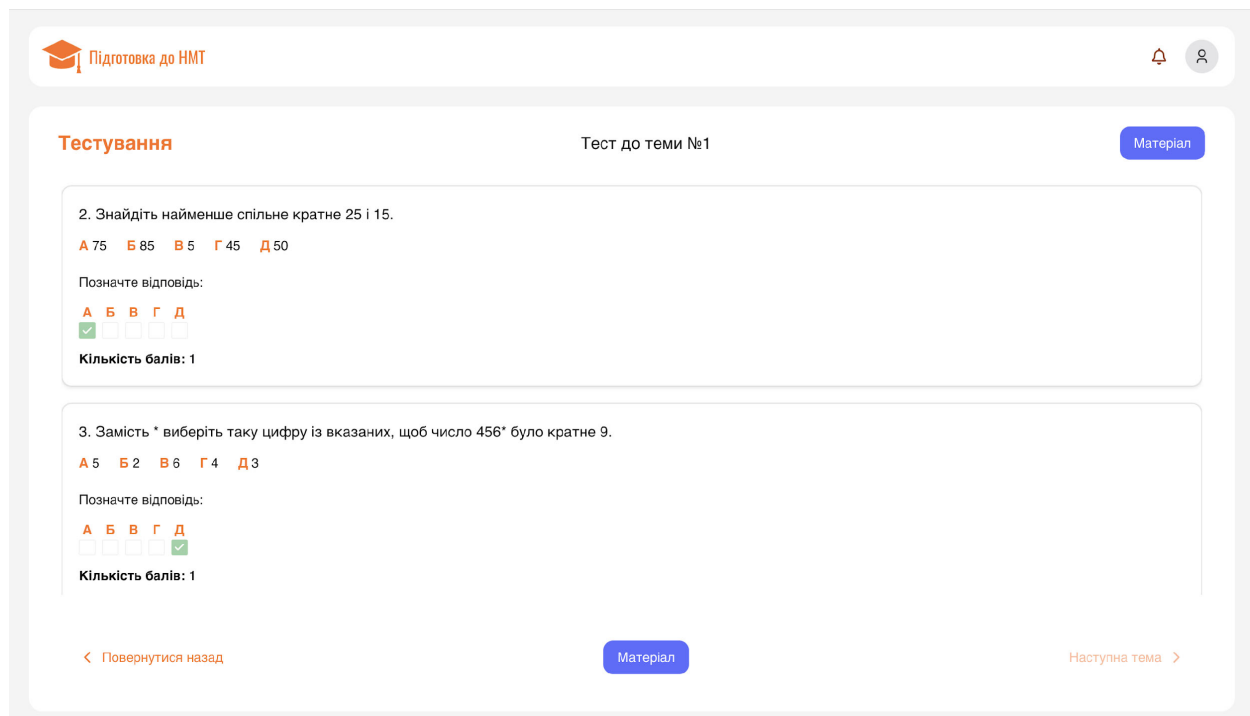


Рис. 13. Приклад принципу контролю та свободи користувача

Надаючи користувачам відчуття контролю та можливість виправляти помилки (наприклад, переглядаючи попередні розділи тесту чи змінюючи відповіді), користувачі відчувають себе комфортніше.

3. ІМПЛЕМЕНТАЦІЯ ПРОДУКТУ ТА ТЕСТУВАННЯ

Серверна частина

Для створення застосунку обрано підхід з використанням тришарової архітектури програмного забезпечення. Згідно з цим підходом застосунок поділяється на три рівні:

- рівень даних – реалізує інтерфейс збереження даних та доступу до них з використанням технологій EntityFramework Core [15];
- рівень логіки – реалізує бізнес-логіку системи;
- рівень презентації – реалізує REST API для доступу до системи [16].

Схема взаємодії шарів застосунку зображена на рис. 14.

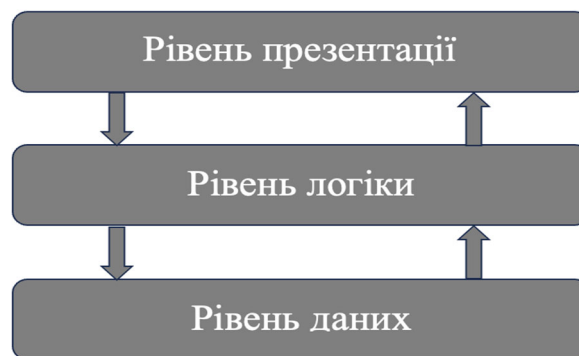


Рис. 14. Тришарова архітектура

Рівень даних реалізовано з використанням патернів Repository та UnitOfWork. На рис. 15 зображено схему патерну.

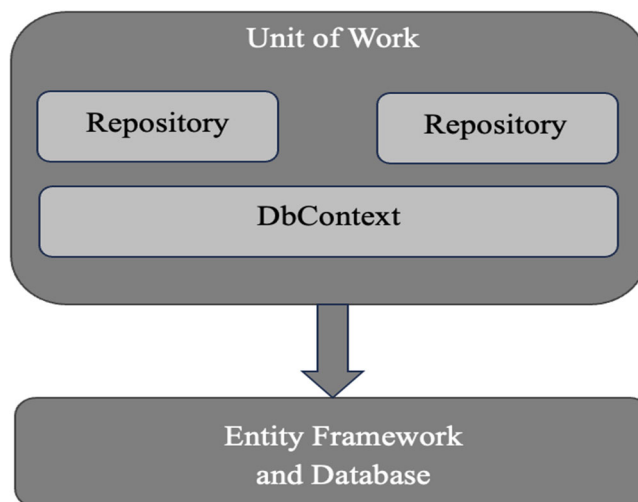


Рис. 15. Схема реалізації рівня даних

Рівень логіки реалізовано у вигляді сервесів, які реалізують відповідні інтерфейси, що дозволяє притримуватися принципу Dependency Inversion. Опис основних сервесів:

- SubjectService – відповідає за логіку, що пов'язана з предметами;
- ChapterService – відповідає за логіку, що пов'язана з розділами;
- TopicService – відповідає за логіку, що пов'язана з темами;

- MaterialService – відповідає за логіку, що пов'язана з матеріалами;
- AssignmentService – відповідає за логіку, що пов'язана з тестами;
- OpenAIService – відповідає за логіку, що пов'язана з ШІ;
- StatisticReportService – відповідає за логіку генерації статистики учня.

Рівень презентації реалізовано у вигляді REST API. Використання ASP.NET дозволяє створювати кінцеві точки у вигляді методів, що об'єднані в контролери. Опис основних контролерів:

- SubjectsController – відповідає за кінцеві точки що пов'язані з предметами;
- ChaptersController – відповідає за кінцеві точки що пов'язані з розділами;
- TopicsController – відповідає за кінцеві точки що пов'язані з темами;
- MaterialsController – відповідає за кінцеві точки що пов'язані з матеріалами;
- EducationClassesController – відповідає за кінцеві точки що пов'язані з навчальними класами;
- ReportsController – відповідає за кінцеві точки що пов'язані з генерацією звітів зі статистикою учня.

Приклад кінцевих точок для роботи з предметами зображено на рисунку 16.

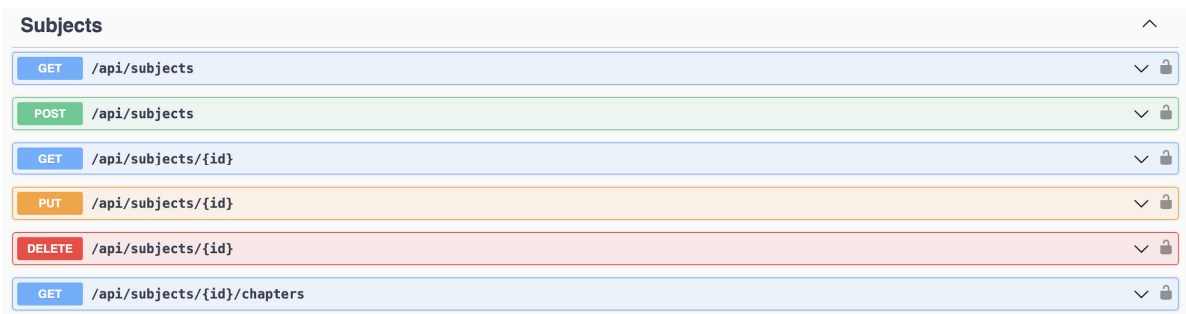


Рис. 16. Кінцеві точки для роботи з предметами

Клієнтська часина

Клієнтську частину реалізовано з використанням React та Vite. Створено загальні компоненти, які використовуються на різних частинах клієнтської частини. Клієнтська частина спілкується з серверною, надсилаючи запити з використанням бібліотеки Axios.

Використано Redux Toolkit для зберігання глобального стану користувача, токена, ролей.

Використано React Query для взаємодії з API: кешування, обробка помилок, повторні запити. Chakra UI v3 використано для розробки інтерфейсу, що підтримує адаптивність.

React Router v6 відповідає за налаштування маршрутів з перевіркою ролей.

Використано підхід FSD (Feature-Sliced Design) – поділ проєкту на модулі.

Мобільний додаток

Мобільний додаток реалізовано за допомогою .NET MAUI. Використано шаблон проєктування MVVM (Model-View-ViewModel) для відокремлення моделі та відображення. У додатку реалізовано адаптивний інтерфейс, який підлаштовується під розміри екранів різних пристроїв. У розробці враховано принципи UX/UI дизайну, що забезпечує інтуїтивно зрозуміле користування додатком.

Тестування

Проведено тестування серверної частини з використанням бібліотеки NUnit. На рис. 17 зображено результати запуску тестового проєкту.

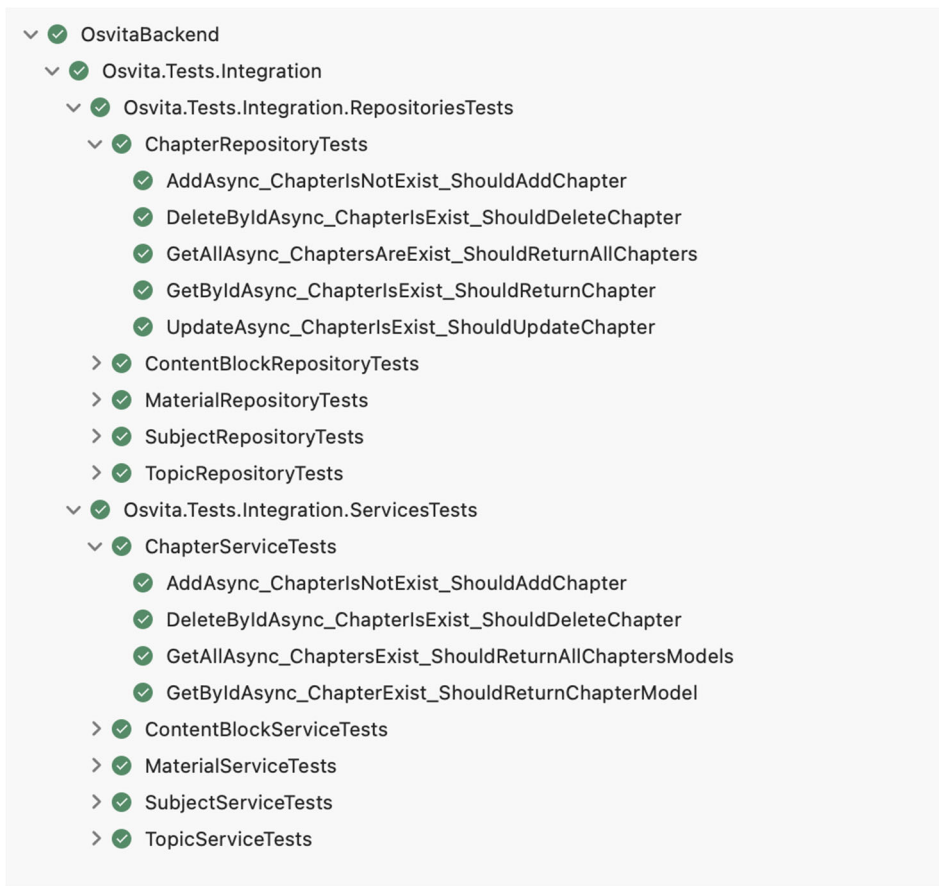


Рис. 17. Результати запуску тестового проєкту

Налаштовано автоматичне тестування збірки. На рисунку 18 зображено результати виконання дій під час злиття гілок.

✓ Merge pull request #19 from LironeA/OSV-124 Build All #72: Commit 46123ff pushed by Stargik	deve lop	📅 3 hours ago 🕒 58s	...
✓ Merge pull request #18 from LironeA/OSV-125-Write-unit-tests Build All #69: Commit 1e41b0c pushed by Stargik	deve lop	📅 5 days ago 🕒 53s	...
✓ Merge pull request #17 from LironeA/OSV-98-Tests-Backend Build All #66: Commit ec325cd pushed by Stargik	deve lop	📅 last week 🕒 53s	...
✓ Merge pull request #16 from LironeA/OSV-33-Front Build All #63: Commit 2f1ca54 pushed by JustVimeR	deve lop	📅 last week 🕒 59s	...
✓ Merge pull request #15 from LironeA/OSV-33-Front Build All #58: Commit b33aff9 pushed by JustVimeR	deve lop	📅 last week 🕒 56s	...
✓ Merge pull request #14 from LironeA/OSV-82-Backend-PL Build All #50: Commit 452ca8c pushed by Stargik	deve lop	📅 last week 🕒 53s	...

Рис. 18. Результати виконання дій під час злиття гілок

Проведено тестування головної сторінки сайту застосунку з використанням інструменту Lighthouse [17].

На рис. 19 зображено результати тестування для десктопної версії сайту.

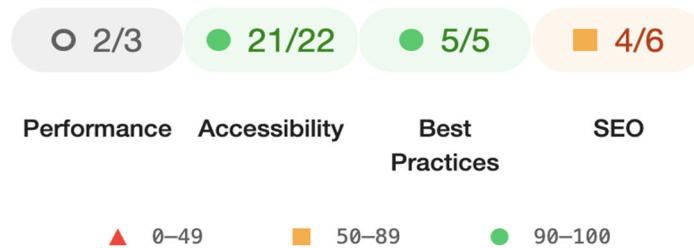


Рис. 19. Результати тестування десктопної версії сайту

На рис. 20 зображено результати тестування для мобільної версії сайту.

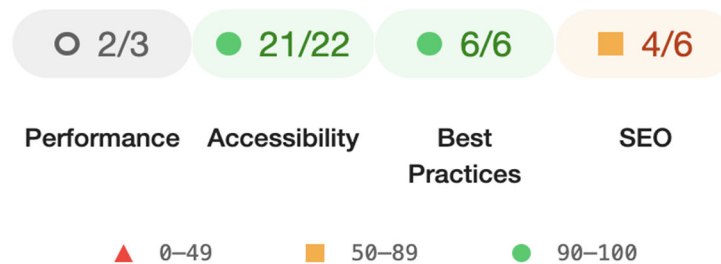


Рис. 20. Результати тестування мобільної версії сайту

Застосунок є оптимізованим до використання у найпопулярніших браузерях з використанням мобільних та десктопних пристроїв.

4. ДОКУМЕНТАЦІЯ ПРОЄКТУ

Важливою складовою успіху проєкту була його документація. Для ведення документації обрано модель "Перекидаємо через стіну". Згідно з цією моделлю документування проведено після реалізації застосунку. Для збереження документів використано платформу Confluence [18].

Основні документи проєкту та етапи, на яких вони були створені:

- SWOT-аналіз – на етапі формування ідеї та огляду конкурентів;
- Lean Canvas проєкту – на етапі формування ідеї;
- карта проєкту – на етапі ініціації проєкту;
- карта продукту – на етапі ініціації проєкту;
- аналіз конкурентів – на етапі планування функціоналу продукту;
- діаграма Ганта – на етапі планування;
- RACI-матриця ролей – на етапі планування;
- матриця Ейзенхауера – на етапі планування;
- UML-діаграми класів та активностей користувача – на етапі проєктування.

Інструкція користувача

Створено інструкцію користувача, в якій описано роботу із застосунком. Інструкція надає повний опис можливостей застосунку.

Головна сторінка сайту має вільний доступ. Сторінка описує мету функціонування застосунку і закликає до дії. Частина головної сторінки сайту зображена на рисунку 21.

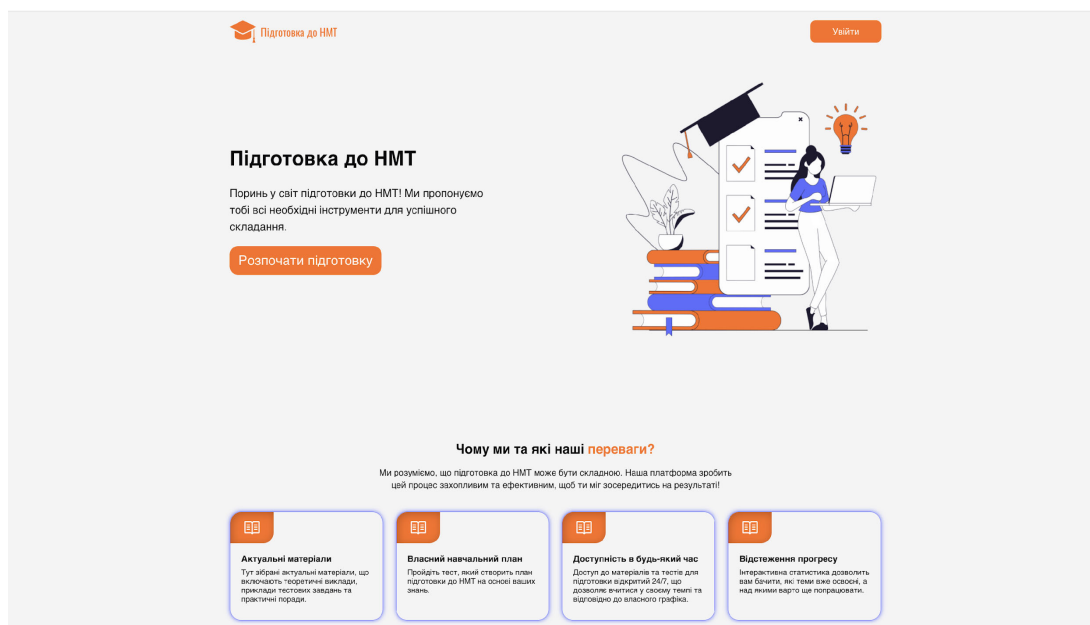


Рис. 21. Головна сторінка сайту

На головній сторінці сайту користувач може зареєструватися або авторизуватися, натиснувши на кнопку "Увійти". Для реєстрації необхідно обрати посилання "Створити акаунт". Також можливо одразу зареєструватися натиснувши кнопку "Розпочати підготовку".

На рис. 22 зображено форму авторизації.

Рис. 22. Форма авторизації

Під час реєстрації можна обрати роль: учень або вчитель. На рис. 23 зображено форму реєстрації.

Після авторизації користувач матиме різні можливості, залежно від його ролі.

Інструкція для учня. Учень потрапляє на сторінку з доступними предметами для навчання. Для доступу до навчальних матеріалів необхідно обрати предмет. На рис. 24 зображено сторінку предметів.

Реєстрація

Електронна пошта
Введіть електронну пошту

Пароль
Введіть пароль

Ім'я
Введіть ім'я

Прізвище
Введіть прізвище

Виберіть роль

Вже маєте акаунт? [Увійти](#)

Рис. 23. Форма реєстрації

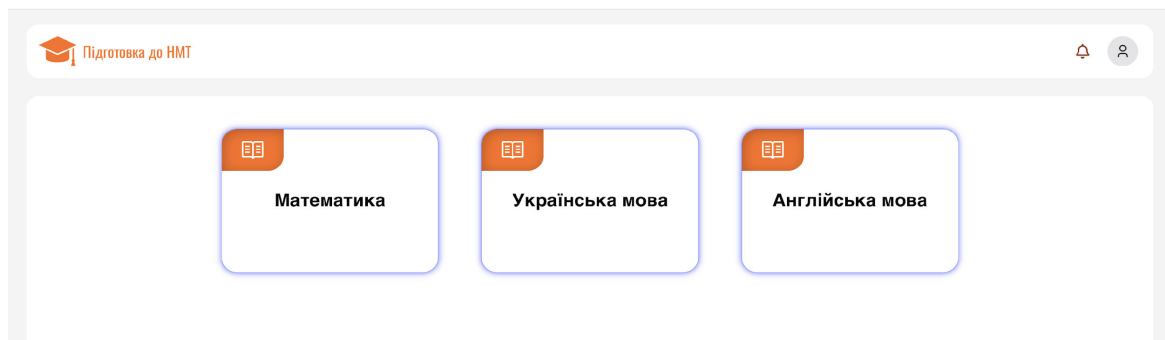


Рис. 24. Сторінка предметів

На сторінці предмету відображаються теми для вивчення. Темы згруповані у розділи. Кожна тема має теоретичний матеріал. Для перегляду теоретичного матеріалу необхідно натиснути на відповідне посилання. До кожної теми можна пройти тест. Учень може позначати теми як вивчені. Також для оцінювання знань учня з обраного предмету є можливість пройти діагностичний тест. Для проходження діагностичного тесту необхідно натиснути відповідну кнопку. На рис. 25 зображено сторінку предмету.

На сторінці теоретичного матеріалу учень переглядає контент. Є можливість переходу до наступного матеріалу, повернутися назад, пройти тест з теми матеріалу натиснувши відповідні кнопки. На рис. 26 зображено сторінку теоретичного матеріалу.

Застосунок пропонує тестові завдання трьох типів: з однією правильною відповіддю, з відкритою відповіддю, на встановлення відповідностей (згідно зі структурою тесту у форматі НМТ). Кожен тест має різний набір тестових завдань. На сторінці проходження тесту за темою учень має вписати або відмітити відповідь до тестових завдань. Учень може перейти до теоретичних матеріалів натиснувши відповідну кнопку. Після натискання кнопки "Завершити тест" відображається результат тестування. На рис. 27 зображено сторінку проходження тестування за темою.

Рис. 25. Сторінка навчального матеріалу

Рис. 26. Сторінка теоретичного матеріалу

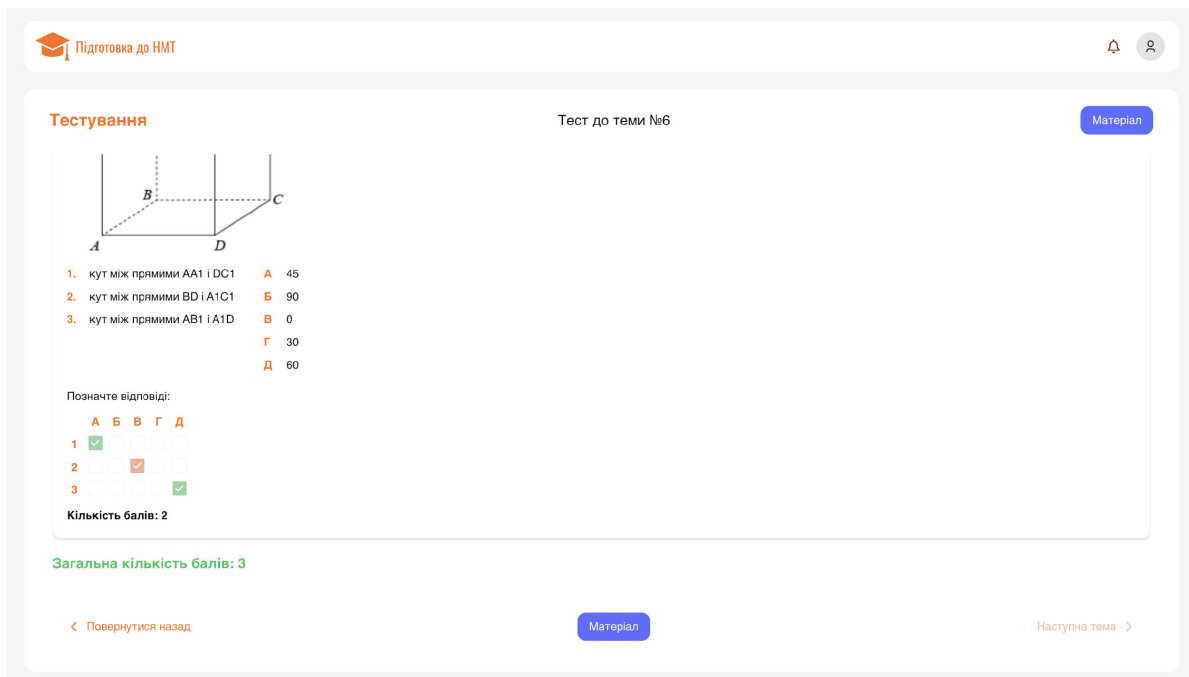


Рис. 27. Сторінка проходження тестування за темою

Діагностичний тест складається з тестових завдань всіх тем обраного предмету.

Після проходження діагностичного тесту учень бачить свій результат та отримує рекомендацію від AI-помічника щодо подальшого навчання. Учень може обрати тему для вивчення з рекомендації й перейти до неї натиснувши відповідну кнопку. На рисунку 28 зображено рекомендації після проходження діагностичного тесту.

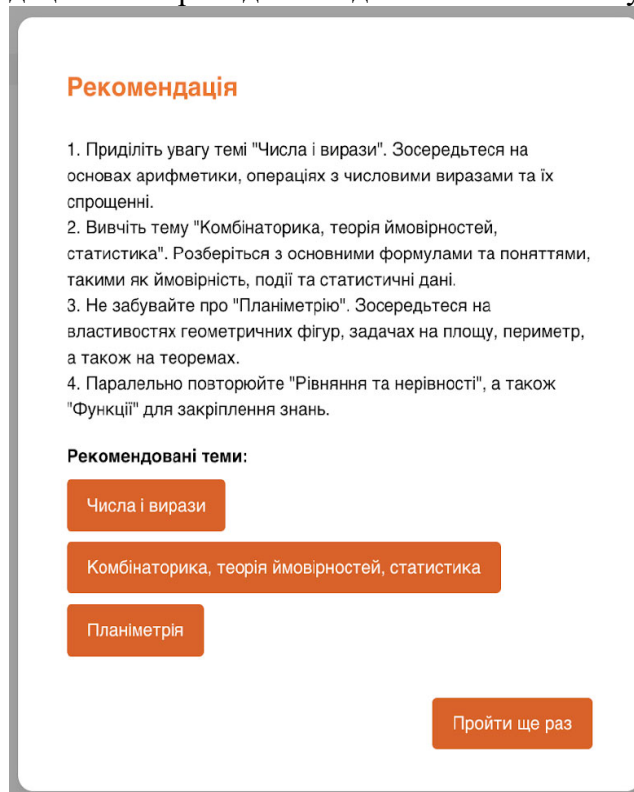


Рис. 28. Рекомендації AI-помічника

Для переходу в особистий кабінет користувач повинен натиснути на кнопку у верхньому правому кутку та обрати "Профіль" у випадаючому меню. На сторінці профілю учень може пройти щоденне завдання для повторення матеріалу. Щоденне завдання пропонується AI-помічником на основі успішності проходження учнем тестувань за темами. На сторінці профілю відображено кількість днів проходження щоденних завдань поспіль без пропуску, рейтингову таблицю учнів за кількістю набраних балів за проходження щоденних тестів. Відображено співвідношення правильно виконаних завдань до неправильних для предметів та окремо для тем. На рисунку 29 зображено сторінку профілю користувача.

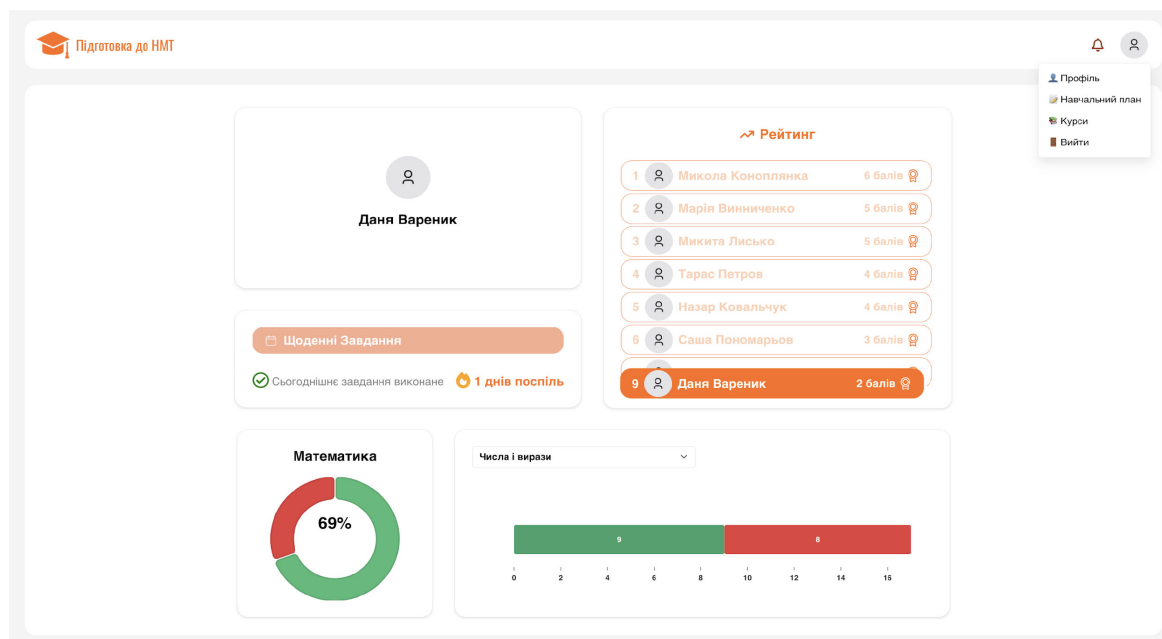


Рис. 29. Профіль користувача

Для перегляду навчального плану учень повинен натиснути на кнопку у верхньому правому кутку та обрати "Навчальний план" у випадаючому меню. На сторінці навчального плану відображено список класів та призначені завдання. На рисунку 30 зображено сторінку навчального плану.

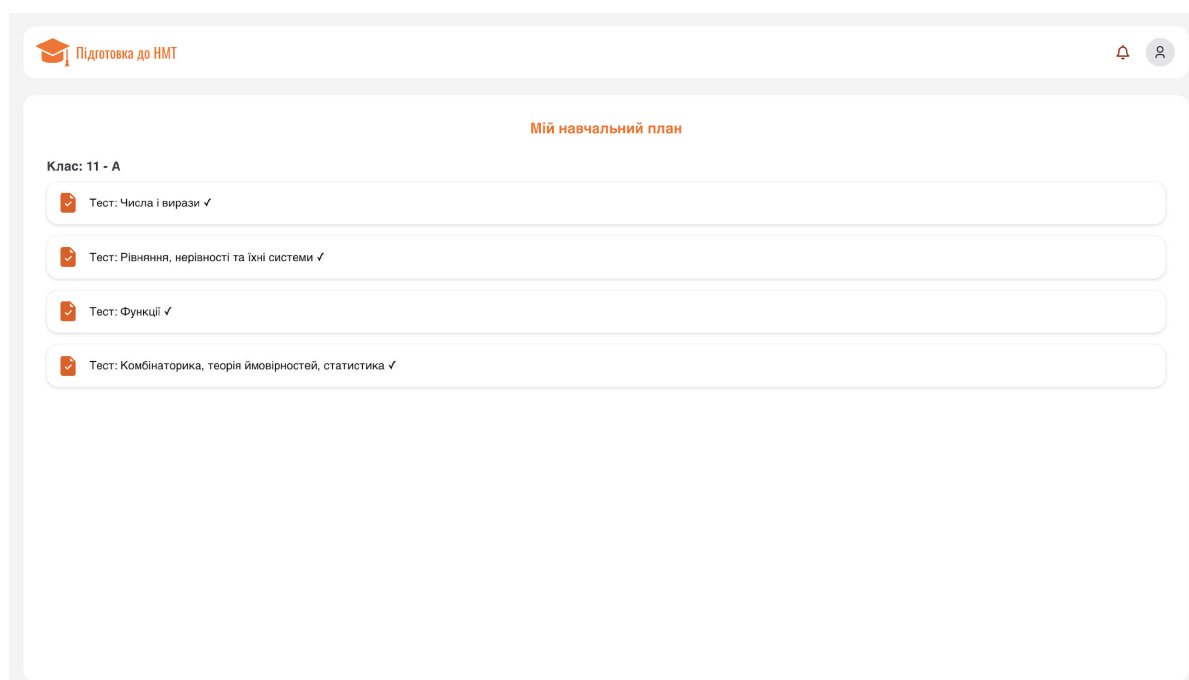


Рис. 30. Сторінка навчального плану

Натиснувши на кнопку у вигляді дзвіночка у верхньому правому кутку учню відображаються повідомлення від AI-помічника із закликом продовжити навчання та рекомендаціями. На рисунку 31 зображено повідомлення від AI-помічника.

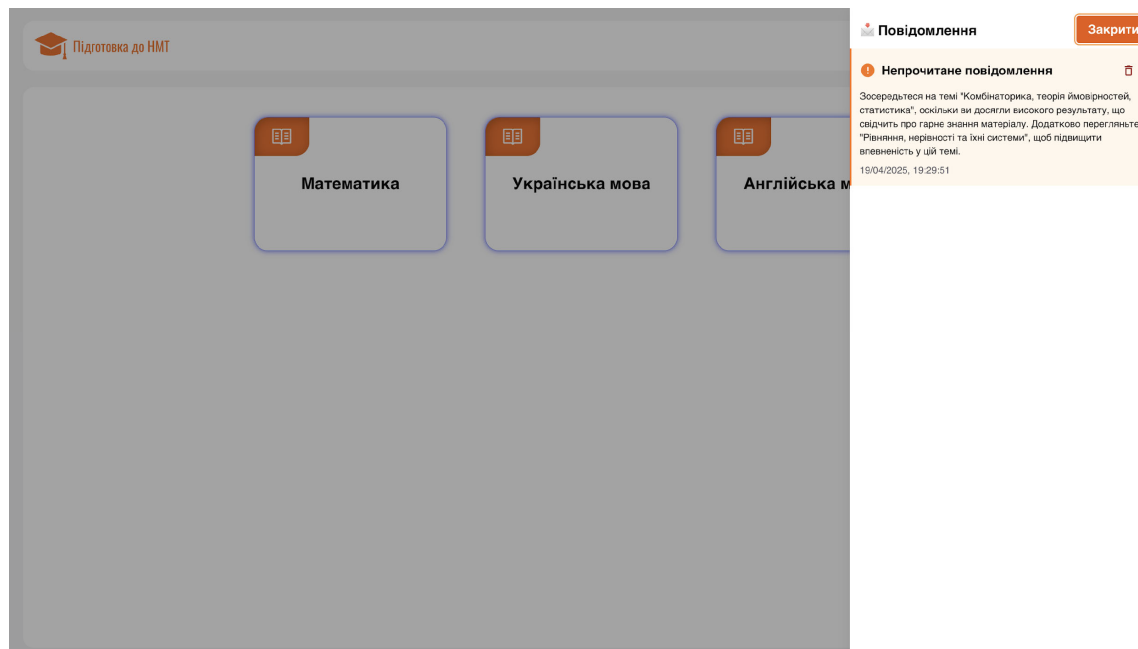


Рис. 31. Повідомлення від AI-помічника

Натиснувши на кнопку "AI-помічник" учень може поставити запитання щодо навчального матеріалу AI-помічнику. AI-помічник надасть відповідь, якщо питання пов'язане з вивченням матеріалу. На рис. 32 зображено консультацію AI-помічника.

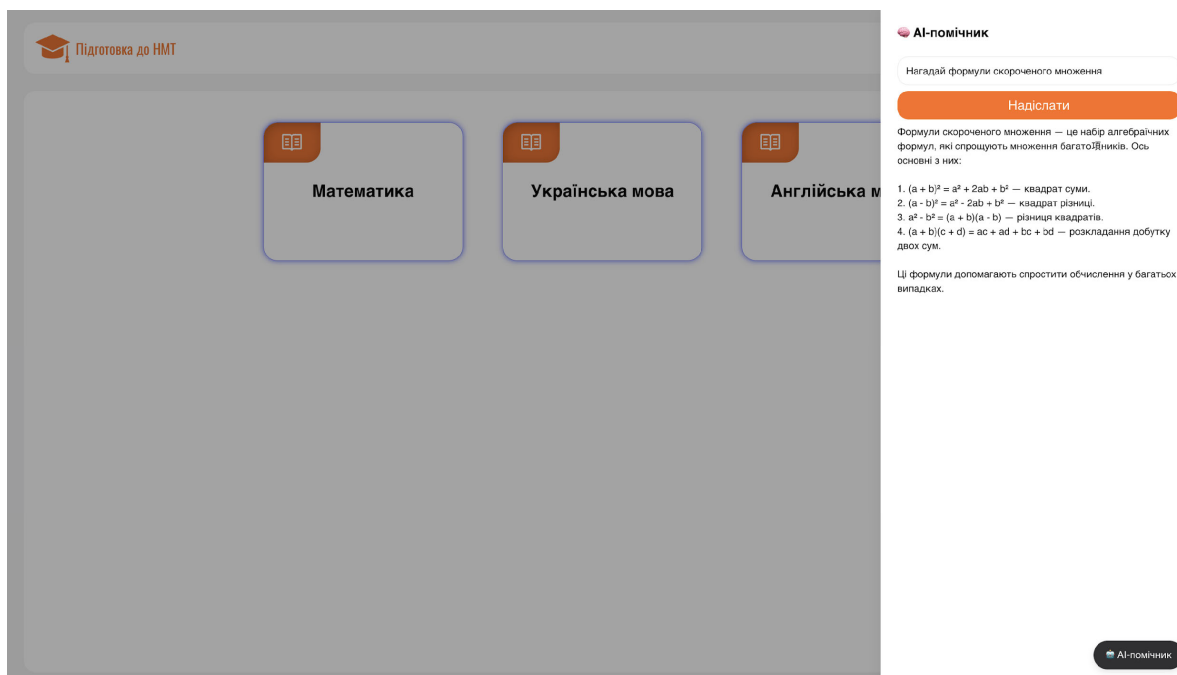


Рис. 32. Консультація AI-помічника

Для вступу до навчального класу учень повинен відкрити посилання, що буде надіслане вчителем на електронну пошту учня. На рисунку 33 зображено сторінку підтвердження вступу до класу.

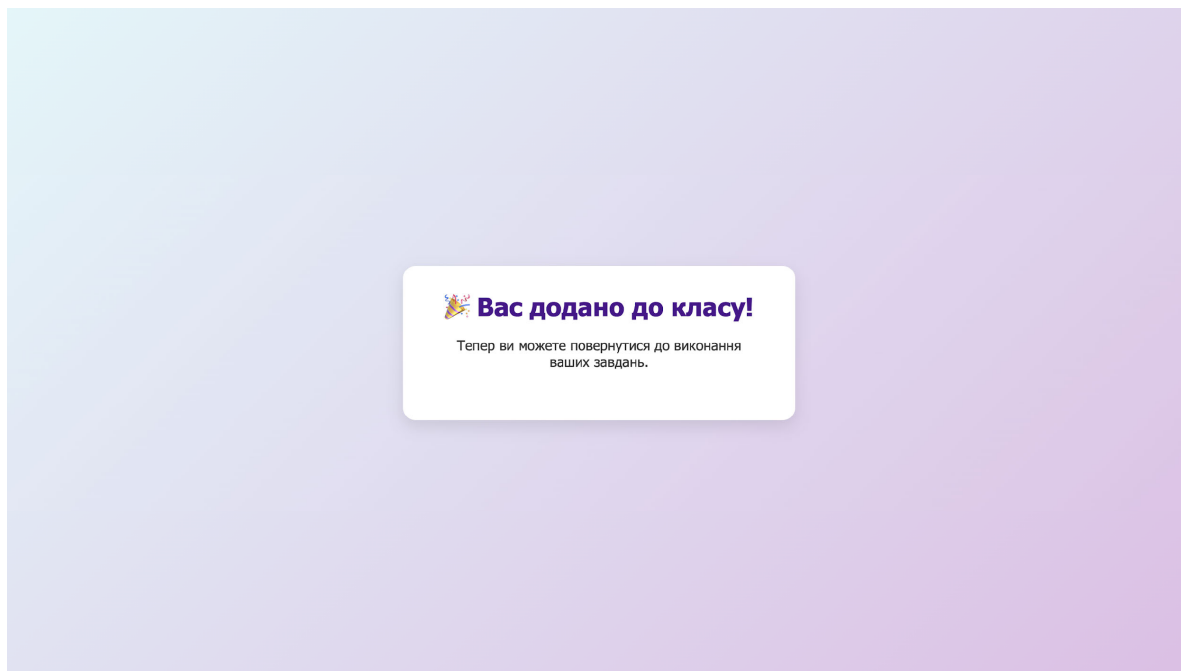


Рис. 33. Підтвердження вступу до класу

Інструкція для вчителя. Для перегляду класів вчитель повинен натиснути на кнопку у верхньому правому кутку та обрати "Вчитель" у випадваючому меню. Для створення нового класу необхідно натиснути кнопку "Створити клас" та ввести назву класу. На рис. 34 зображено процес створення класу.

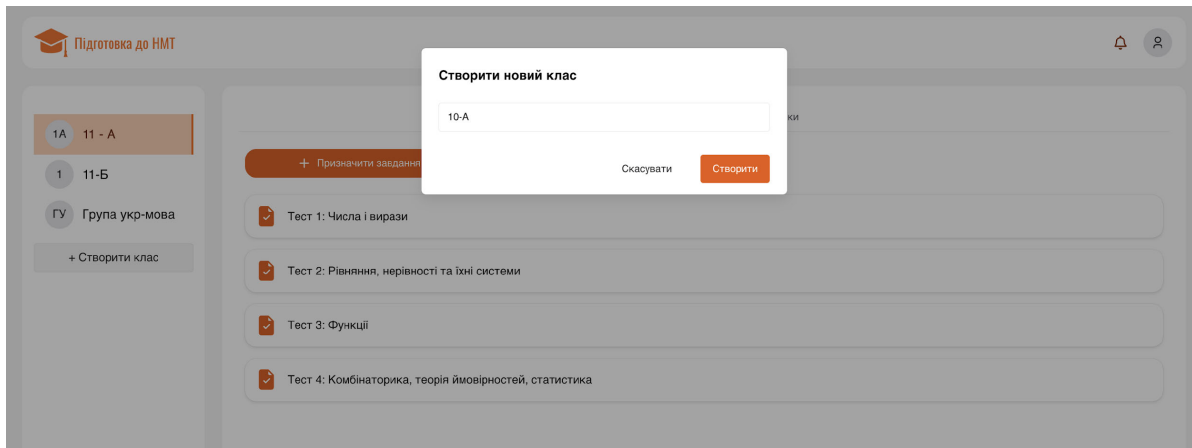


Рис. 34. Створення класу

Для перегляду інформації про клас необхідно обрати клас у бічному меню. На вкладці "Учні" зображено список учнів. Для надсилання запрошення до класу учню необхідно натиснути кнопку "Додати учня" та ввести електронну пошту учня. На рис. 35 зображено вкладку учнів класу.

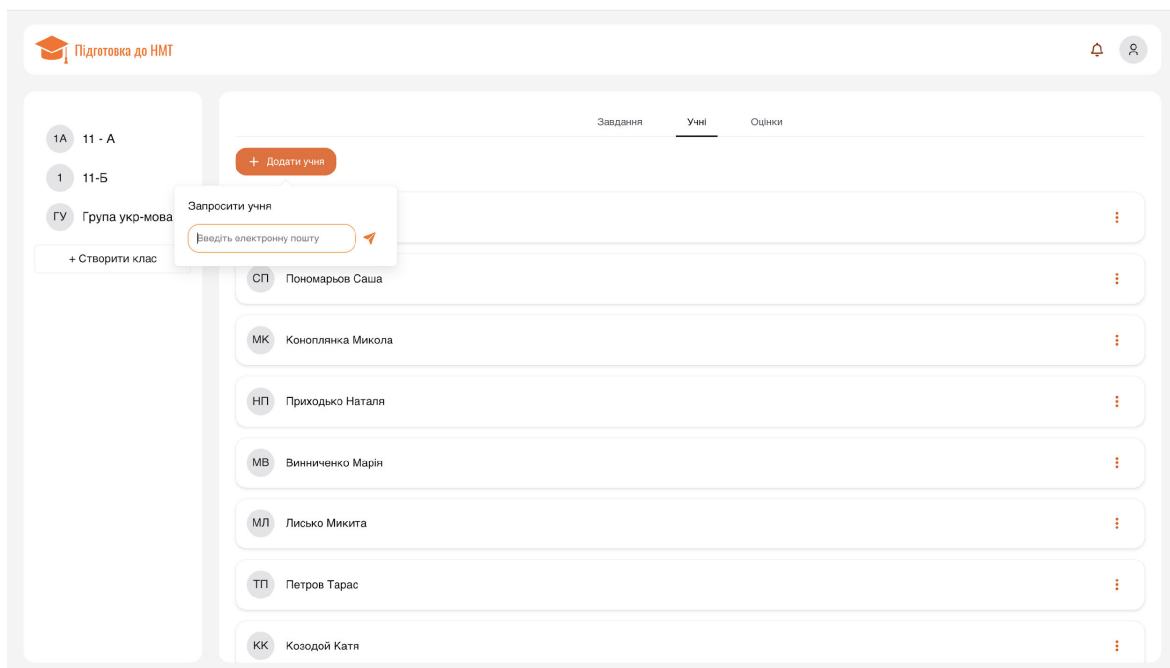


Рис. 35. Вкладка "Учні"

На вкладці "Завдання" відображено список завдань, що призначені для класу. Для призначення нового завдання необхідно натиснути кнопку "Призначити завдання". На рис. 36 зображено список завдань класу.

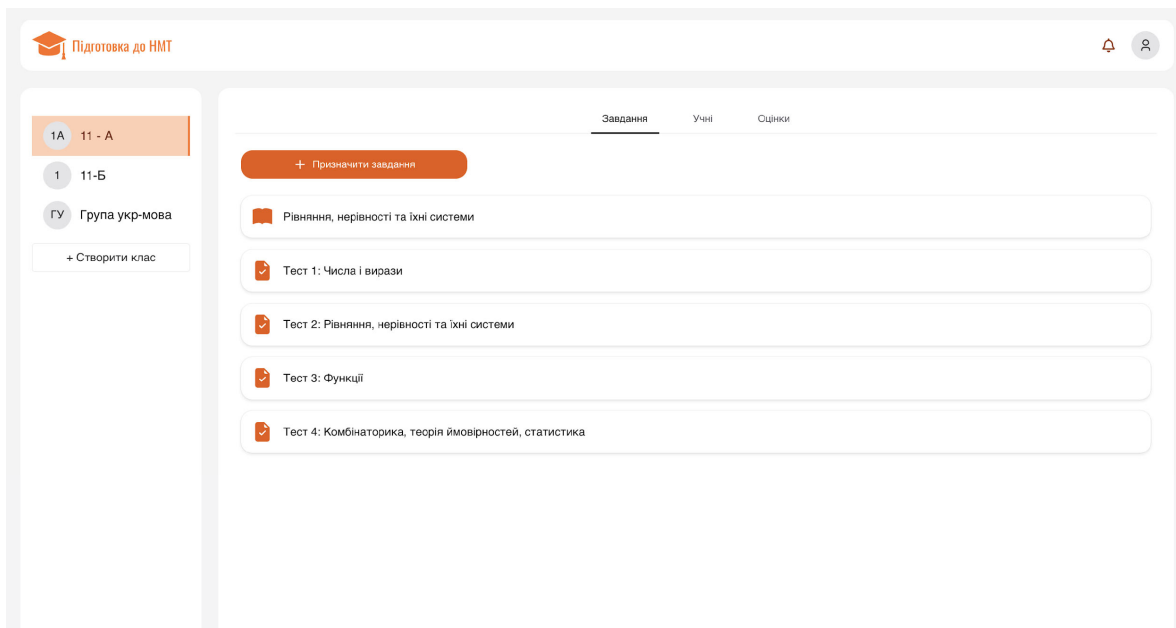


Рис. 36. Вкладка "Завдання"

Застосунок пропонує вчителю теоретичні матеріали та тестові завдання, які вчитель може призначати учням класу. Вчитель може призначити опрацювання теоретичного матеріалу теми або проходження тесту для класу. Для призначення тесту необхідно обрати предмет та тип тесту. Для призначення тесту за всіма темами предмету необхідно обрати "Загальний тест". Для призначення тесту за темою необхідно обрати "Проходження тесту" та обрати тему. Для призначення опрацювання теоретичних матеріалів теми необхідно обрати "Опрацювання матеріалу" та обрати тему. На рис. 37 зображено процес призначення завдання учням класу.

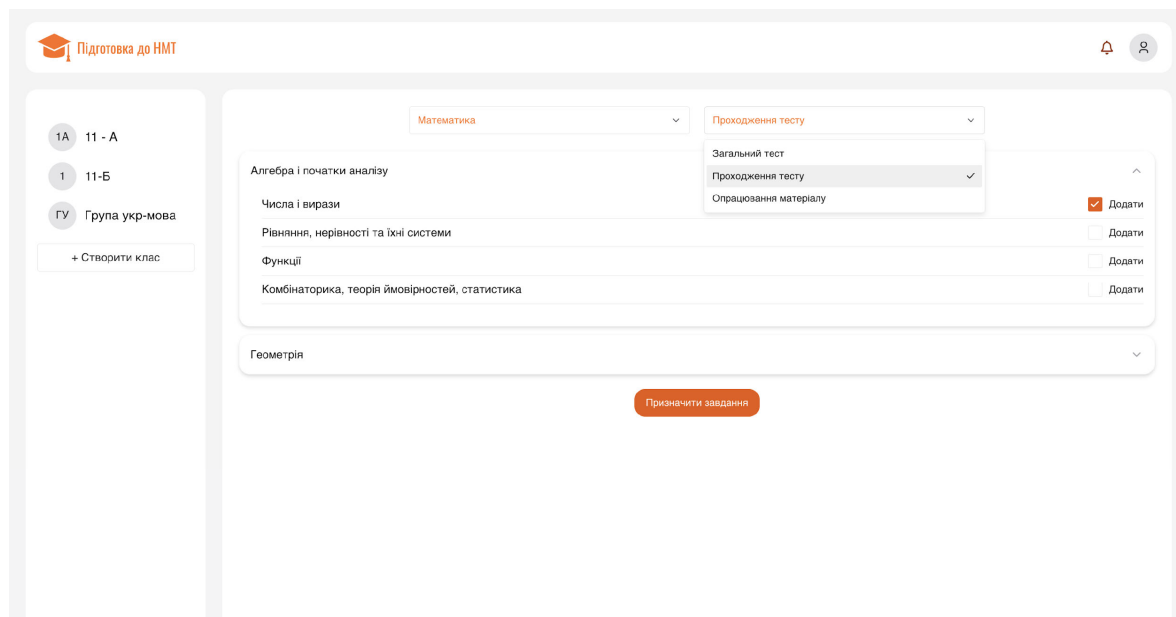


Рис. 37. Призначення завдання

На вкладці "Оцінки" відображено призначені тести класу та результати проходження тестів учнями. Вчитель може вивантажити результати виконання тесту учнями обравши відповідний тест та натиснувши кнопку "Вивантажити статистику PDF". Вчитель може переглянути детальніше виконання тесту учнями обравши відповідний тест та натиснувши кнопку "Переглянути детальніше". На рис. 38 зображено результати виконання тестів учнями.

ПІБ	Оцінки			
	Тест 1	Тест 2	Тест 3	Тест 4
Вареник Даня	9/17	8/14	13/14	
Пономарьов Саша	8/17	5/14	13/14	
Коноплянка Микола	8/17	11/14	9/14	13/14
Приходько Наталя	12/17	14/14	12/14	14/14
Винниченко Марія	14/17	14/14	12/14	12/14
Лисько Микита	15/17	14/14	11/14	12/14
Петров Тарас	15/17	14/14	11/14	14/14
Козодой Катя	15/17	14/14	11/14	14/14
Присяжнюк Аріна	15/17	14/14	12/14	14/14
Ковальчук Назар	15/17	14/14	12/14	14/14

Рис. 38. Призначення завдання

На сторінці детальної інформації проходження тесту учнями відображено кількість балів, що отримано учнем за тестове завдання. Для перегляду умови завдання необхідно натиснути на відповідну комірку. На рис. 39 зображено сторінку детальної інформації про проходження тесту учнями.

ПІБ	1	2	3	4	5	6	7	8	9	10	Сума
Вареник Даня	1/1	1/1	1/1	1/1	1/1	0/2	0/2	0/2	1/3	3/3	9/17
Пономарьов Саша	1/1	0/1	1/1	1/1	1/1	0/2	0/2	0/2	1/3	3/3	8/17
Коноплянка Микола	1/1	0/1	1/1	1/1	1/1	0/2	0/2	0/2	1/3	3/3	8/17
Приходько Наталя	1/1	0/1	1/1	1/1	1/1	2/2	2/2	0/2	1/3	3/3	12/17
Винниченко Марія	1/1	0/1	1/1	1/1	1/1	2/2	2/2	0/2	3/3	3/3	14/17
Лисько Микита	1/1	1/1	1/1	1/1	1/1	2/2	2/2	0/2	3/3	3/3	15/17
Петров Тарас	1/1	1/1	1/1	1/1	1/1	2/2	2/2	0/2	3/3	3/3	15/17
Козодой Катя	1/1	1/1	1/1	1/1	1/1	2/2	2/2	0/2	3/3	3/3	15/17
Присяжнюк Аріна	1/1	1/1	1/1	1/1	1/1	2/2	2/2	0/2	3/3	3/3	15/17
Ковальчук Назар	1/1	1/1	1/1	1/1	1/1	2/2	2/2	0/2	3/3	3/3	15/17

Рис. 39. Сторінка інформації про проходження тесту

Інструкція для адміністратора. Після авторизації адміністратор потрапляє в адміністративну частину застосунку. Адміністратор може переглянути предмети, розділи, теми, матеріали, тестові завдання обравши відповідний пункт бічного меню. На рис. 40 зображено адміністративну частину системи.

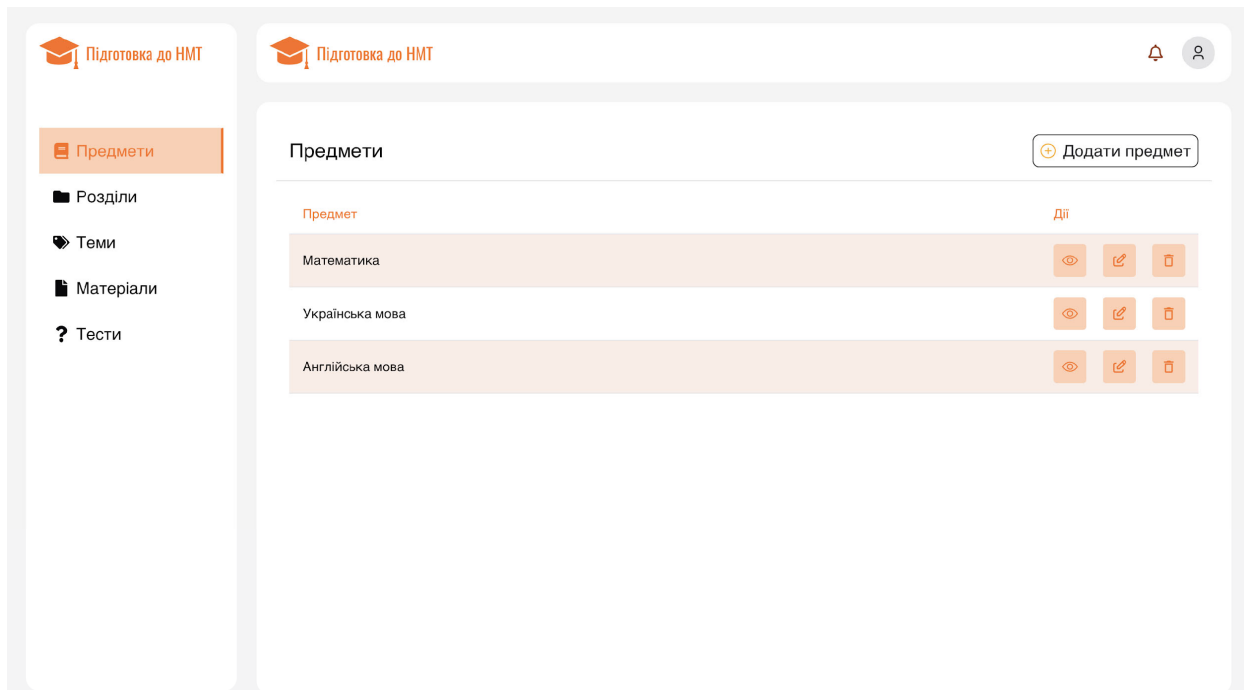


Рис. 40. Адміністративна частина застосунку

Для створення, редагування, видалення предметів, розділів, тем, матеріалів, тестових завдань трьох типів, необхідно натиснути відповідну кнопку та ввести необхідні дані. На рис. 41 зображено сторінку створення навчального матеріалу.

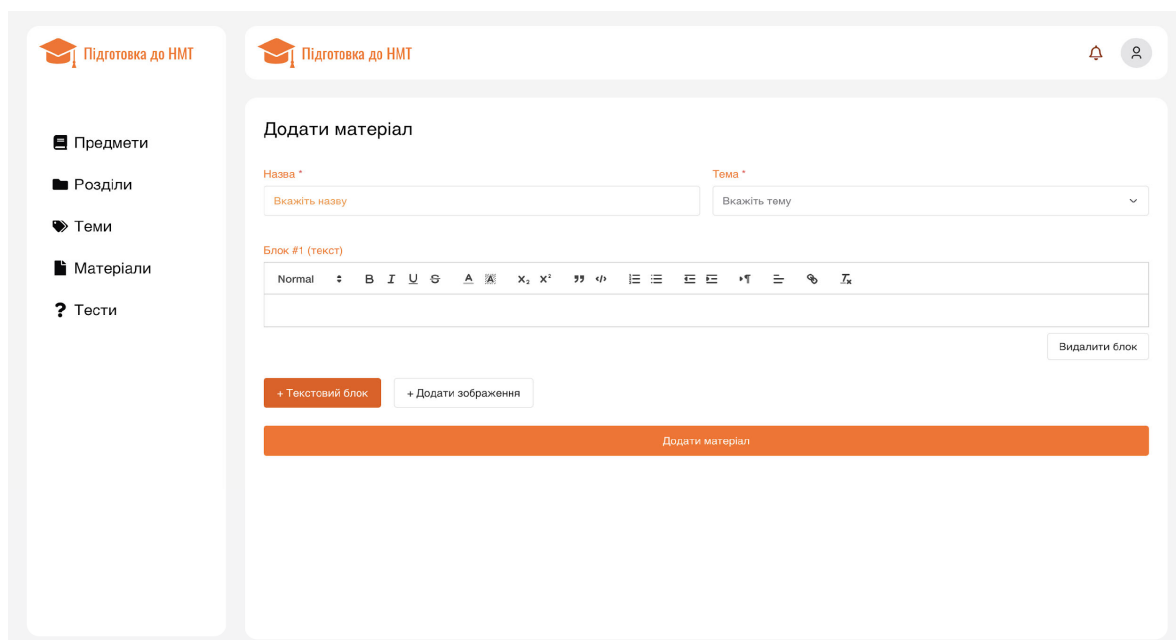


Рис. 41. Створення навчального матеріалу

Тестові завдання створюються для навчального матеріалу. На рис. 42 зображено сторінку створення тестового завдання.

Рис. 42. Створення тестового завдання

Передбачено імпорт з файлу формату Excel предметів, розділів, тем, матеріалів, тестових завдань трьох типів.

Інструкція користувача мобільного застосунку. Після відкриття застосунку першою сторінкою є сторінка авторизації користувача. Якщо користувач ще не має акаунту, він може перейти на сторінку реєстрації, натиснувши кнопку "Створити акаунт". Сторінки авторизації та реєстрації зображено на рис. 43.

Рис. 43. Сторінки авторизації та реєстрації

Після успішної авторизації відкривається сторінка вибору предмету. Активні предмети доступні для вибору, а неактивні відображаються сірим кольором. Сторінку вибору предмету зображено на рис. 44.

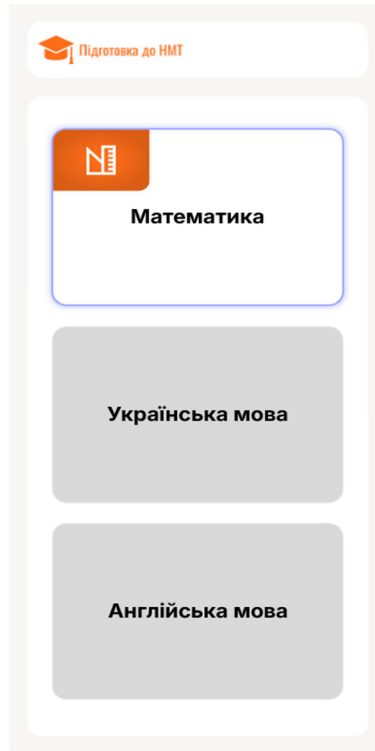


Рис. 44. Сторінка вибору предмету

При натисканні на активний предмет відкривається сторінка предмету, де відображаються розділи та теми. Наприклад, розділ "Алгебра і початки аналізу" містить теми: "Числа і вирази", "Рівняння, нерівності та їх системи" та інші.

Щоб переглянути теми розділу, необхідно натиснути на експандер праворуч від назви розділу. Ліворуч від назви теми знаходиться індикатор її статусу:

- сірий кружечок – тема не пройдена;
- помаранчевий кружечок з прапорцем – тема пройдена.

Сторінка предмету зображена на рис. 45.

При натисканні на тему відкривається сторінка теми, яка містить список навчальних матеріалів, а також кнопку "Пройти тест з теми", що відкриває сторінку проходження тесту. Натискання на окремий матеріал (наприклад, "Дійсні числа") відкриває сторінку матеріалу. На рис. 46 зображені сторінки теми, проходження тесту, окремого матеріалу.

Тест містить 3 типи завдань:

- вибір однієї правильної відповіді;
- тест з відкритою відповіддю;
- тест на встановлення відповідності.

На рисунку 47 зображено відображення тестових завдань різних типів.

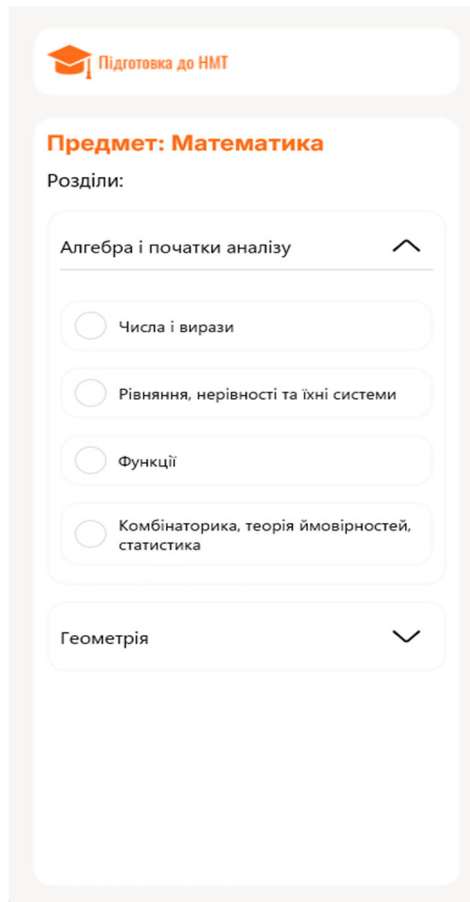


Рис. 45. Сторінка предмету

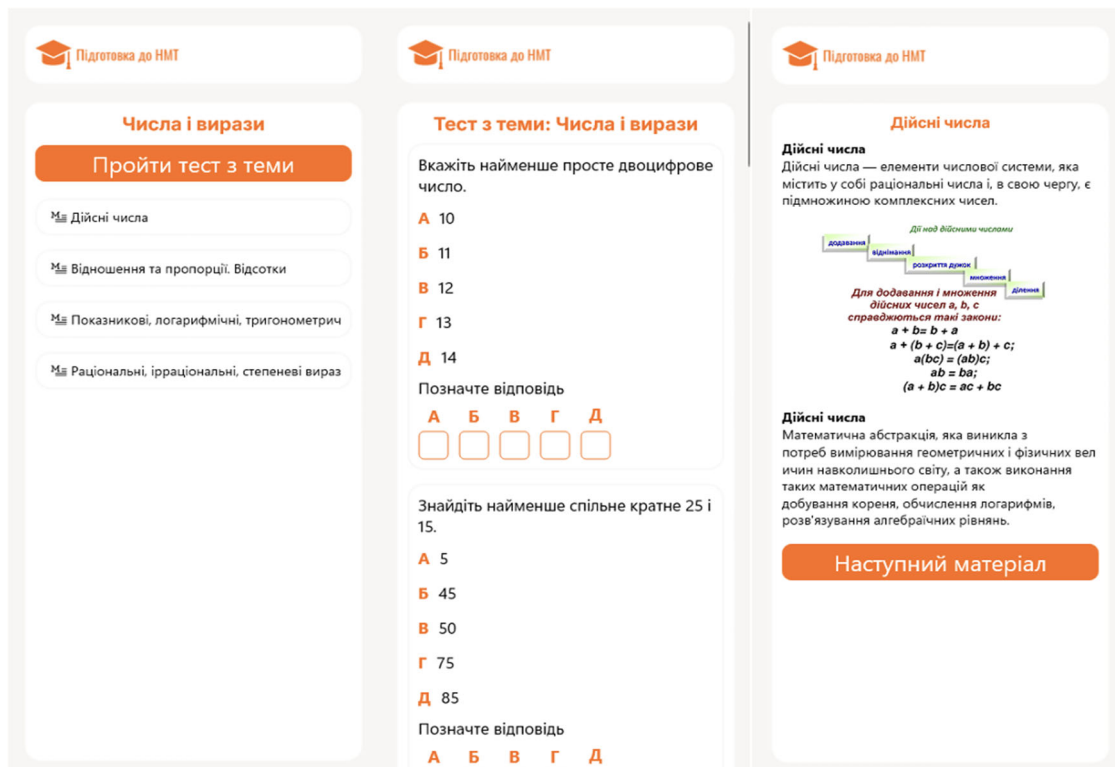


Рис. 46. Сторінки теми, тестів, матеріалів

Вкажіть найменше просте двоцифрове число.

A 10

B 11

V 12

G 13

D 14

Позначте відповідь

A **B** **V** **G** **D**

Обчисліть $2+2$
Впишіть відповідь:

Обчисліть $(6 \cdot 3) / 2 + 8$
Впишіть відповідь:

Додатне число А більше додатного числа В у 3,8 раза. На скільки відсотків число А більше за число В?
Впишіть відповідь:

Установіть відповідність між виразом та його значенням

1 4+4

2 4*4

3 4/4

A 8

B 16

V 1

G 10

D 2

Позначте відповідь

	A	B	V	G	D
1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Рис. 47. Тестові завдання

Правові норми, які регулюють відносини в сфері застосування програмного продукту

На діяльність програмного продукту впливають правові норми, що пов'язані з освітнім процесом в Україні, правом на володіння інформацією та врегулювання надання інформаційних послуг.

Наказ МОН "Про організацію та проведення у 2025 році національного мультипредметного тесту" [19] регулює укладання завдань згідно з програмами, що затверджені наказом "Про затвердження програм зовнішнього незалежного оцінювання для осіб, які бажають здобувати вищу освіту на основі повної загальної середньої освіти" [20]. Ці накази визначають тематику, структуру і типи тестів, що дозволяє побудувати ефективну підготовку до іспиту без надлишкової інформації.

Закон України "Про авторське право і суміжні права" [21] регулює авторські права на створений контент. Цей закон захищає авторські права на навчальний матеріал для підготовки до НМТ, який присутній у застосунку.

Закон України "Про інформацію" [22] визначає терміни "інформаційна продукція" та "інформаційна послуга". Глава 63 ЦК України "Послуги. Загальні положення" [23] визначає положення про надання послуг, зокрема інформаційно-консультаційних, та укладання відповідних договорів. Оскільки застосунок надає послуги, що пов'язані з поширенням інформації, ці правові норми визначають тип послуг, який надає застосунок, та порядок укладання договору у випадку використання даного застосунку для навчання.

5. УПРАВЛІННЯ ПРОЄКТОМ

Опис команди, обраних методології та інструментарію

Команда складалася з 8 осіб. У таблиці 6 наведені члени команди та їхні ролі в проєкті.

Для проєктного управління обрано методологію SCRUM [24]. Методологія є гнучкою, що дозволяє вносити зміни у планування під час розробки. Згідно з цією методологією на початку створюється загальний пріоритезований список задач. Далі робота ділиться на короткі етапи – спринти. Під час спринту учасники проєкту спілкуються на щоденних зустрічах і звітують про виконану роботу. Для керування проєктом обрано інструмент Jira [25].

Таблиця 6. Члени команди та їхні ролі в проєкті

Учасник	Ролі
Олександр Старжинський	Бекенд-розробник, тестувальник, фахівець з документації, заступник керівника проєкту
Владислав Таран	Фронтенд-розробник (Web), тестувальник
Катерина Шемет	Дизайнер, керівник проєкту, модератор
Яна Уточкіна	Бекенд-розробник, тестувальник
Іван Божевський	Фронтенд-розробник (Mobile), тестувальник
Максим Сомов	Бекенд-розробник, тестувальник, аналітик
Євген Чопенко	Бекенд-розробник, тестувальник
Владислав Кучинський	Бекенд-розробник, тестувальник, контент-менеджер

Демонстрація планування та виконання спринтів

На початку планування було створено беклог. Задачі розподілено на спринти тривалістю 2 тижні. Після кожного спринту проаналізовано ступінь виконання поставлених задач і перенесено невиконані задачі до наступного спринту. Комунікація між членами команди відбувалася у вигляді загальних зустрічей 2 рази на тиждень, згодом 1 раз на тиждень, оскільки робота над проєктом проводилася здебільшого у певні дні тижня, зважаючи на графік членів команди. Для обговорення термінових питань, що не потребували залучення усіх членів команди, проводилися зустрічі між частиною команди, яка відповідальна за вирішення цього питання. Також комунікація відбувалася у командному чаті. На рисунку 48 зображено таймлайн спринтів у Jira.

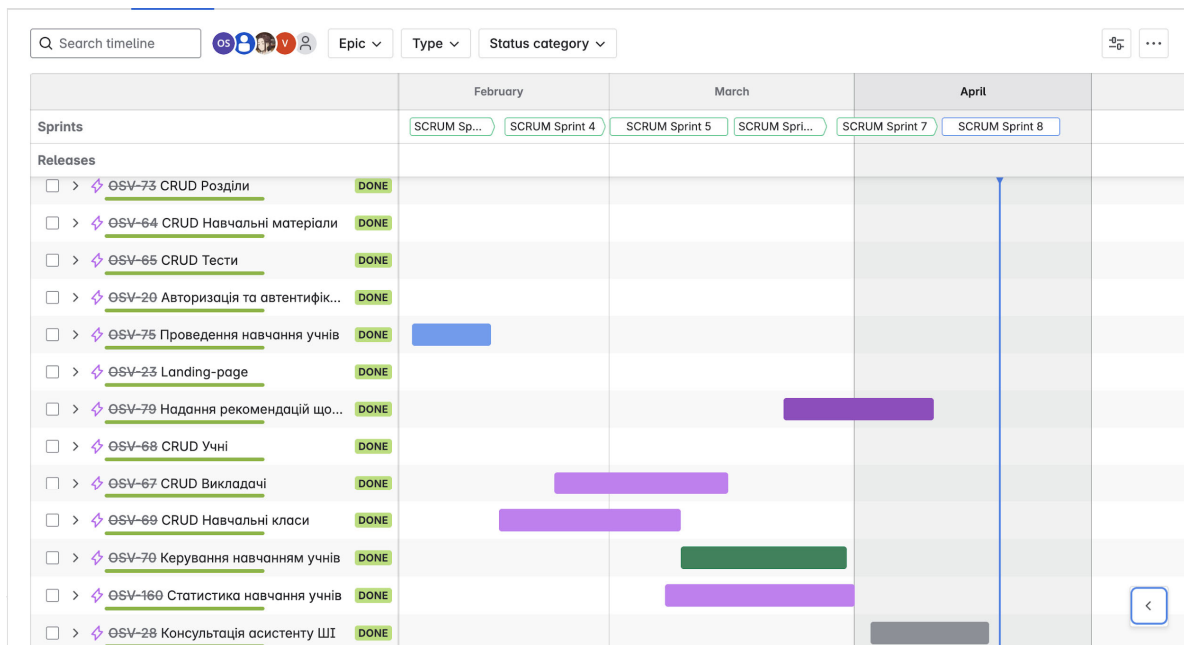


Рис. 48. Таймлайн спринтів у Jira

Після закінчення спринту відбувалося демо, для якого створювалася презентація з результатами виконаної роботи. Презентації та інші артефакти зберігалися на Confluence, де кожен учасник проєкту міг переглянути необхідні документи. Завдяки задокументованому функціоналу та інструкції новий член команди зміг налаштувати проєкт без складнощів. На рисунку 49 зображено документацію в Confluence.

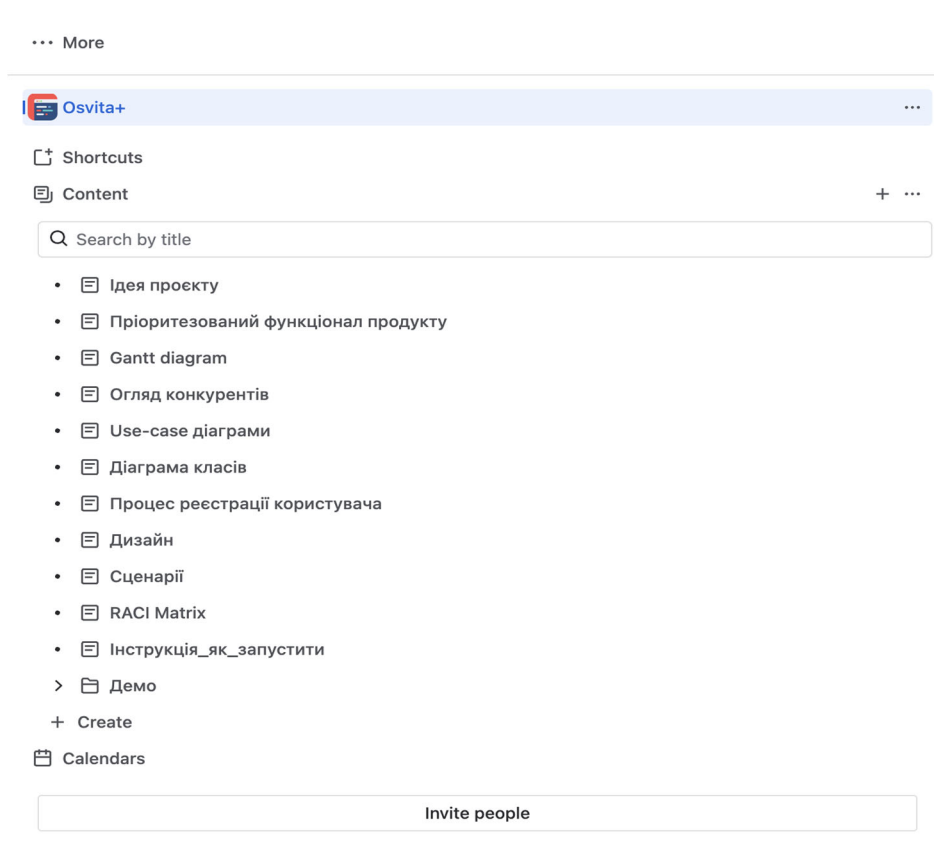


Рис. 49. Документація в Confluence

Рефлексія

На початку задачі з ініціації проєкту виконувалися під час командних зустрічей. Задачі виконувалися послідовно і потребували ствердження кожного учасника команди. Зустрічі були довгими та неефективними. Згодом вирішено на зустрічах лише розподіляти задачі між членами команди та звітувати про виконання. Це дало можливість розпаралелити процес виконання завдань та ефективніше використовувати ресурси.

На початку проєкту виникли складнощі з формулюванням чітких вимог до системи. Створення UML-діаграм дозволило задокументувати вимоги до системи і розпочати розробку.

На етапі моделювання були складнощі зі створенням моделі даних. Поради викладача допомогли створити модель даних.

Протягом розробки через нестачу часу винесені на спринти задачі іноді не були виконані у повному обсязі. Доводилося переносити задачі на наступний спринт. В результаті доводилося змінювати дедлайни етапів розробки.

Під час роботи над проєктом набуто досвід роботи в команді, планування проєкту, створено реалізацію продукту.

ВИСНОВКИ

Створено застосунок для підготовки до НМТ "Osvita+". Під час створення застосунку пройдено всі етапи розробки продукту. Під час пошуку ідеї використано інструменти аналізу та прийняття рішень. На етапі планування використано методи пріоритезації задач. Набуто досвід роботи в команді за методологією SCRUM, навички ініціації,

планування, документації проєкту. Покращено навички розробки з використанням технологій .NET, React. Також набуто навички використання нових технологій. Досліджено правові норми, що регулюють відносини в сфері застосування продукту. Члени проєкту набули досвід командної роботи над проєктом.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Онлайн-школа математики GIOS. URL: <https://gioschool.com/ua> (дата звернення: 20.04.2025).
2. Курси підготовки до НМТ/ЗНО ZNOUA. URL: <https://zno.ua/> (дата звернення: 20.04.2025).
3. ОСВІТА.UA. URL: <https://osvita.ua/> (дата звернення: 20.04.2025).
4. Українська електронна освітня система "МійКлас". URL: <https://www.miyklas.com.ua/> (дата звернення: 20.04.2025).
5. Facebook. URL: <https://www.facebook.com/> (дата звернення: 22.04.2025).
6. About Instagram. URL: <https://about.instagram.com/> (дата звернення: 22.04.2025).
7. Show up with Google Ads. URL: <https://business.google.com/us/> (дата звернення: 22.04.2025).
8. About YouTube. URL: <https://about.youtube/> (дата звернення: 22.04.2025).
9. App.diagrams.net. URL: <https://app.diagrams.net/> (дата звернення: 22.04.2025).
10. About Figma. URL: <https://www.figma.com/about/> (дата звернення: 20.04.2025).
11. .NET documentation. URL: <https://learn.microsoft.com/en-us/dotnet/> (дата звернення: 20.04.2025).
12. Overview of ASP.NET Core. URL: <https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core> (дата звернення: 20.04.2025).
13. Quick Start – React. URL: <https://react.dev/learn> (дата звернення: 20.04.2025).
14. SQL Server technical documentation. URL: <https://learn.microsoft.com/en-us/sql/sql-server/> (дата звернення: 20.04.2025).
15. Overview of EntityFramework Core. URL: <https://learn.microsoft.com/en-us/ef/core/> (дата звернення: 20.04.2025).
16. Вівчарик В. Повний огляд REST: нюанси, поради, приклади. Dou.ua. 19.09.2024. URL: <https://dou.ua/forums/topic/50364/> (дата звернення: 20.04.2025).
17. Introduction to Lighthouse. Chrome for developers. URL: <https://developer.chrome.com/docs/lighthouse/overview> (дата звернення: 20.04.2025).
18. Get started with Confluence. URL: <https://www.atlassian.com/software/confluence/resources/guides/get-started/overview> (дата звернення: 20.04.2025).
19. Про організацію та проведення у 2025 році національного мультипредметного тесту : Наказ Міністерства освіти і науки України від 30.12.2024, № 1806. URL: <https://mon.gov.ua/npa/pro-orhanizatsiiu-ta-provedennia-u-2025-rotsi-natsionalnoho-multypredmetnoho-testu> (дата звернення: 20.04.2025).
20. Про затвердження програм зовнішнього незалежного оцінювання для осіб, які бажають здобувати вищу освіту на основі повної загальної середньої освіти : Наказ Міністерства освіти і науки України від 03.02.2016, № 77: станом на 03.02.2016. URL: <https://zakon.rada.gov.ua/rada/show/v0077729-16#Text> (дата звернення: 20.04.2025).
21. Про авторське право і суміжні права : Закон України від 01.12.2022, № 57: станом на 15.11.2024. URL: <https://zakon.rada.gov.ua/laws/show/2811-20#Text> (дата звернення: 20.04.2025).

22. Про інформацію : Закон України від 02.10.1992, № 48 : станом на 15.11.2024. URL: <https://zakon.rada.gov.ua/laws/show/2657-12#Text> (дата звернення: 20.04.2025).
23. Цивільний Кодекс України : від 16.01.2003, № 40-44 : станом на 09.04.2025. URL: <https://zakon.rada.gov.ua/laws/show/435-15#Text> (дата звернення: 20.04.2025).
24. Scrum Guides. URL: <https://scrumguides.org/> (дата звернення: 20.04.2025).
25. Welcome to Jira. URL: <https://www.atlassian.com/software/jira/guides/getting-started/introduction> (дата звернення: 20.04.2025).

DOCDIRECT: AN ADVANCED SUPERVISION PLATFORM FOR PSYCHOLOGISTS, PSYCHOTHERAPISTS AND PSYCHIATRISTS

Volodymyr KRAVCHUK, Mykhailo STATNIK, Natalia MUSKA, Igor KOSTIUK, Nikita MASYCH, Mykyta IVANOV, Kateryna KYSIL, Laura MOSKALENKO

DocDirect is a web-based platform where psychotherapists, psychologists, and psychiatrists can both provide and receive supervision. In this project, we have implemented an MVP version of the product, which includes basic features: creating a personal account, selecting supervisors by criterias, booking supervision, and interaction between therapists. Market analysis was conducted, strategic development directions were identified, and knowledge in software development and project management methodologies was enriched. The resulting product makes it possible to improve the level of cooperation between therapists, simplify the acquisition of the necessary experience and support on the professional path, and improve people's mental health.

SUPERVISOR, SUPERVISION, MENTAL HEALTH, PSYCHOLOGIST, PSYCHOTHERAPIST, PSYCHIATRIST, WEB APPLICATION DEVELOPMENT, PRODUCT MANAGEMENT.

DOCDIRECT: ПРОФЕСІЙНА ПЛАТФОРМА СУПЕРВІЗІЙ ДЛЯ ПСИХОЛОГІВ, ПСИХОТЕРАПЕВТІВ ТА ПСИХІАТРІВ

Володимир КРАВЧУК, Михайло СТАТНИК, Наталія МУСКА, Ігор КОСТЮК, Нікіта МАСИЧ, Микита ІВАНОВ, Катерина КИСІЛЬ, Лаура МОСКАЛЕНКО

DocDirect – веб-платформа, де психотерапевти, психологи, психіатри можуть як надавати, так й отримувати супервізії. У цій роботі реалізовано MVP версію продукту, яка включає базові можливості: створення персонального кабінету, вибір супервізорів за критеріями, бронювання супервізії, взаємодія терапевтів між собою. Проведено аналіз ринку, визначено стратегічні напрямки розвитку, та збагачено знання в області розробки програмного забезпечення та методологій управління проєктами. Отриманий продукт дає можливість покращити рівень співпраці між терапевтами, спростити отримання потрібного досвіду та підтримки на професійному шляху й поліпшити ментальне здоров'я людей.

СУПЕРВІЗОР, СУПЕРВІЗІЯ, МЕНТАЛЬНЕ ЗДОРОВ'Я, ПСИХОЛОГ, ПСИХОТЕРАПЕВТ, ПСИХІАТР, РОЗРОБКА ВЕБ-ЗАСТОСУНКУ, МЕНЕДЖМЕНТ ПРОДУКТУ.

INTRODUCTION

Assessment of the current state of the research or development object. The modern field of mental health care is experiencing a growing demand for professional development, particularly in the form of supervision for psychologists and psychotherapists. Supervision plays a crucial role in enhancing practical skills, ensuring adherence to ethical standards, and maintaining professional competence. However, access to high-quality supervision remains a

challenge due to high costs, limited availability of certified supervisors, and the lack of centralized platforms for managing the supervision process.

Existing solutions are often fragmented, addressing narrow aspects of supervision while failing to provide a comprehensive approach. Many platforms lack transparent mechanisms for verifying supervisors' qualifications, affordable pricing models, and user-friendly features. These limitations highlight the need for an innovative solution that integrates advanced technologies to ensure accessibility, affordability, and efficiency in professional supervision.

The relevance of the work and rationale for its implementation. The development of a unified platform for professional supervision is highly relevant, given the increasing demand for accessible and transparent services in the mental health sector. The DocDirect [1] project seeks to address this need by offering a platform that combines verified expertise, user-friendly tools, and secure payment systems, providing a seamless and trustworthy supervision experience.

This work is essential as it responds to the global need for efficient solutions that reduce barriers to professional supervision. By integrating modern technology, the project aims to create a scalable and effective platform that serves as a valuable resource for professionals across various regions and levels of expertise.

Aim and objectives of the work. The aim of this work is to design, develop, and analyze the DocDirect platform, which facilitates access to certified supervisors and provides tools for managing sessions, transparent ratings, and secure payments.

To achieve this aim, the following objectives have been established:

- analyze the current state of existing solutions in the field of supervision;
- design platform architecture to meet modern standards of functionality and security;
- integrate mechanisms for supervisor verification, session management, and transparent financial operations;
- test the platform's functionality and assess its alignment with user needs;
- develop recommendations for scaling the platform to international markets.

Object and Methods of Research or Development. The object of this research is the process of organizing professional supervision in the field of mental health. The subject of the study is the development of an innovative platform to enhance access to supervision.

The methodology involves market analysis, technical design of the platform architecture, modeling the platform's operational processes, and testing integrated solutions. The development employs modern tools for data management, session scheduling, and secure financial transactions.

Potential Applications. The DocDirect platform can be applied in private practices, mental health clinics, educational institutions, and associations that support professional development. Additionally, it has the potential to integrate with existing health care and education platforms, expanding its impact.

1. IDEA RESEARCH AND VALIDATION

Description of the product idea research. The idea for the product came when our team participated in a medical hackathon. There were various areas of product development proposed, such as mental health, diabetes or cardiovascular disease. After a long search for ideas, 3 development options were identified:

- using an AI solution to reduce the cost of an MRI scan;
- anonymization of patients to increase their privacy and confidence in confidentiality;
- a supervision platform for medical professionals.

The latter idea came from our medical expert, who has been working as a psychotherapist for 3 years. So, we decided to focus on it based on number of reasons:

- this idea was easier, faster, and more profitable to analyze and develop;
- the problem came directly from a mental health worker who, in her experience and the experience of her colleagues, faced problems finding supervisors;
- in Ukraine (and potentially in Europe), there is a sufficiently large number of psychotherapists, psychologists, and psychiatrists;
- such a platform does not yet exist in Ukraine;
- our medical expert is a psychologist herself and has access to supervisors. Direct contact with this area will facilitate promotion and obtaining the necessary information from our target audience.

Review of competing IT products on the market. Competitor analysis is the main step before product creation. We found out and analyzed all possible competitors in Ukraine. The categories we have identified for analysis are abilities, relevance for us as competitor, strengths, weaknesses, localizations, size, communication channels, contact with supervisors, payment methods, payment provider, prices available. Such table is quite big to paste it here, so it will be available for investigation in our Notion [2] page [3].

Lean canvas of the product. The Lean canvas of our product includes the following points: problem description, customer segments, solution, unique value proposition, existing alternatives, unfair advantage, channels of communication, key metrics, revenue streams, cost structure, early adopters and high-level concept. Such table is quite big to paste it here, so it will be available for investigation in our Notion [2] page [4].

SWOT-analysis of the chosen idea. Strengths of the idea:

- sufficient ease of implementation (web-application);
- lack of direct competitors;
- the target audience is broad and diverse. It includes professional psychologists, psychotherapists, and psychiatrists – from beginners (1–2 years of experience) to experienced specialists (2+ years). They either need supervision to improve their skills, clarify professional problems, or are looking for someone to supervise others in order to develop their careers, gain new experience, and build their professional image. It also includes students of psychology, psychiatry, or psychotherapy from universities or courses. These individuals have only a basic understanding of supervision and can use our platform as a springboard for further professional growth;
- military psychologists/psychotherapists/psychiatrists. These are people who work in war or crisis situations and need to quickly find the right specialist in the relevant field.
- a large market share that we can take. Values were evaluated based on the research of our and foreign markets. See fig. 1.

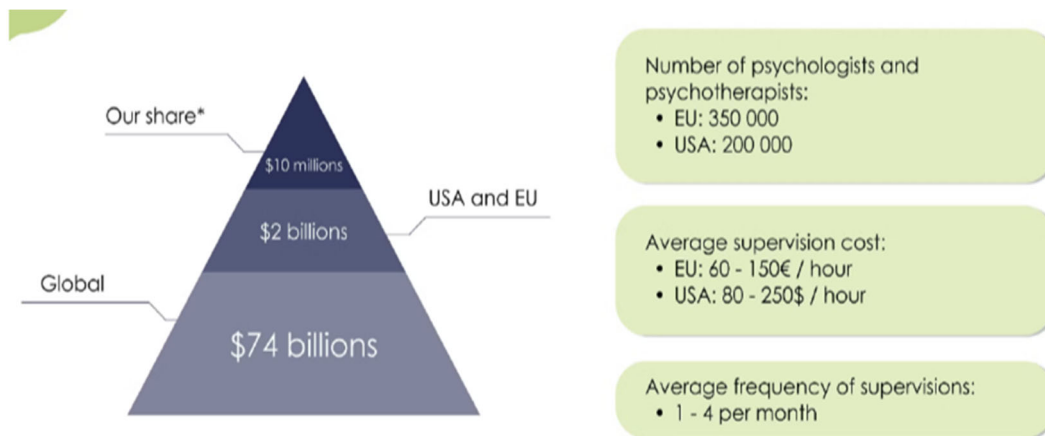


Fig. 1. DocDirect market share

Weaknesses of the idea:

- it can be difficult to attract supervisors to our platform. They may have their own customer base. We will need to give them some benefits to encourage them in the beginning;
- find investments to promote the idea due to martial law in Ukraine;

Risks:

- not retaining clients (supervisors) with our benefits or a small number of features;
- the Ministry of Education of Ukraine wants to digitalize some products for our target audience. Something will be implemented like ours. Solution is to integrate with them or become a base on which they will take further steps.

Opportunities:

- to develop the mental health area and bring it to a whole new level;
- enter the European and American markets;
- investments in Mental Health have begun to grow, which means that there are more chances to get funding for the development of the idea.

Overview of technologies used. We've chosen Notion [2] and Superthread [5] for documentation and task scheduling. The former was chosen because of its popularity and ease of use. But after we ran out of free documentation volumes, we switched to Superthread, which offers the same features.

For the development part, we chose the programming languages, frameworks, and systems in which we have experience. For the frontend part it's Vue.js framework [6] and TypeScript code language [7]. For the backend part it's Go language [8]. For the infrastructure part it's DigitalOcean cloud platform [8] and NGINX service [10].

2. SYSTEM AND PRODUCT DESCRIPTION

Purpose of the system. DocDirect shall fill the gap resolving problems aroused by lack of transparent pricing formation for supervision services, quick and reliable ways of finding one, suitable for your personal needs.

Goals of the system. The achievement of the aforementioned purpose may be accomplished by building a unified platform for both supervisors and supervisees, with a focus on affordable services and a verified supervision experience. See fig. 2.

General requirements from the system. Platform is in a format of web application, with focus on desktop version and correspondingly UI adopted for usage on desktop as a first priority. The

system shall be reliable, with access for users on a full day and night basis. First iteration planning encompasses Ukrainian market and corresponding domestic language translation, as well as conformity with local regulations from the financial and judicial perspective.

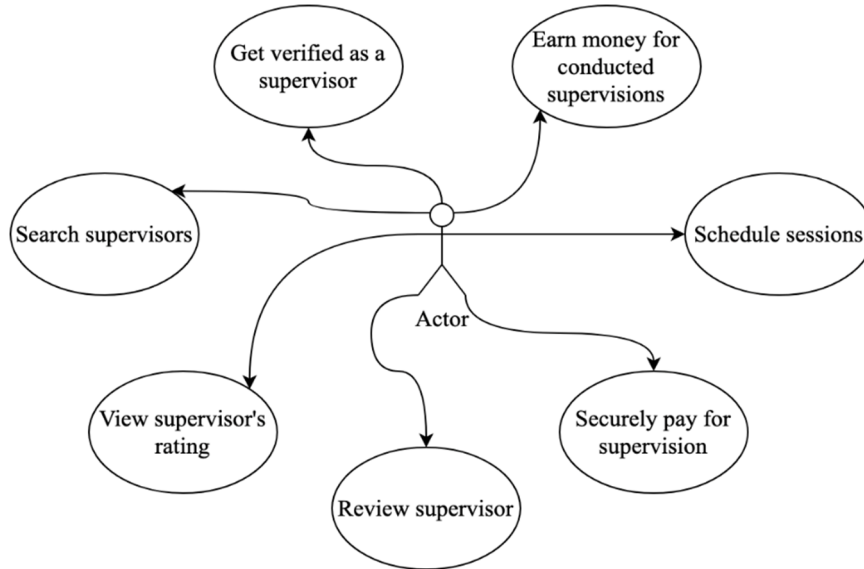


Fig. 2. Use case diagram for user

Requirements for system structure and functionality. In particular, the system shall be separated into client and admin panel perspectives. The former, allows for a user account with basic settings and traditional web2 formed identity linkage. Requirements are such as to meet the goals described in paragraph **Goals of the system**. The latter, in MVP scope, must allow review of supervisors' verifications and grant corresponding statuses.

Requirements for separate functions. Requirements for separate functions are described at tables 1-2 below.

Table 1. User system functions

Function	Purpose
Account management	Register account with email or Google OAuth2. Logout from the system. Edit name, email, password, detail info.
Supervisions	View list of supervisors with filters. View rating of the supervisor. Leave review about supervisor. Schedule supervision. Cancel supervision.
Chats	Send/receive text message to/from the user. Delete chat.
Payments	Pay for the supervision. Revert payment, cancelling supervision. Receive payment from a supervisee.

Table 2. Admin functions

Function	Purpose
Profiles	Review verification requests. Approve/Reject verification request. Send failed verification reason.
Supervisors	Review verification requests. Approve/Reject verification request. Send failed verification reason.

System requirements. The system must check that the data the user enters is valid. For example, upon registration it is required by the system to check email address to conform with the standard as defined in RFC 5322. See an example at fig. 3.

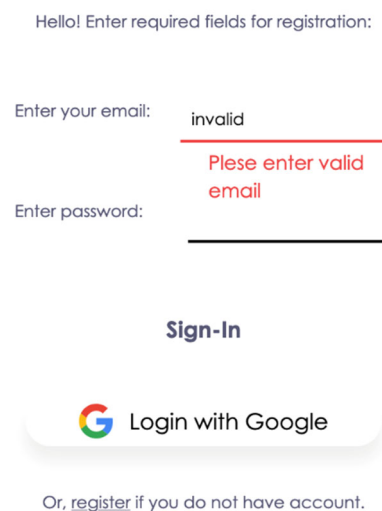


Fig. 3. Validation of user data during registration

Other than that, web version must work correctly on at least the following browsers:

- Google Chrome: Version 131.0.6778.69 for Linux, and 131.0.6778.69.70 for Windows and macOS;
- Mozilla Firefox: Version 132.0.2;
- Microsoft Edge: Version 130.0.6723.126 for Windows and macOS;
- Apple Safari: Version 15.2.

We implement system web client using Vue.js framework [6]. Backend is made using Go language [8]. Additionally, we require DNS domain and Kubernetes [11] cluster to host all system via helm charts. In order to send emails, we use 3-rd party orchestrators, such as Mailjet [12]. In order to conduct payment, we use gateway with support of escrow transfers such as RozetkaPay [13]. In order to save contacts who want to get our updates, we use external marketing service Brevo [14]. Also, we rely on Nginx [10] as a reverse proxy for backend microservices connection with client.

Logical structure of the database. We use a relational database management server named PostgreSQL [15]. Additionally, we rely on S3 buckets [16] for storing images and documents. If a backend service needs a database, only it has direct access to it. See relations between services and tables at table 3.

Table 3. Services and tables they have access to

Service name	Tables
auth-svc	user_ admin_ email_verification_token password_verification_token refresh_token oauth2_state
payment-svc	recipient payment
admin-svc	verification_request verification_log
profile-svc	certificate speciality method profile
supervision-svc	time_slot recurrent_slot supervisor supervision review
chat-svc	chat chat_participant message

Tables definitions

Below at fig. 4-9, we present a relation of the tables with respect to the services and tables definition as well.

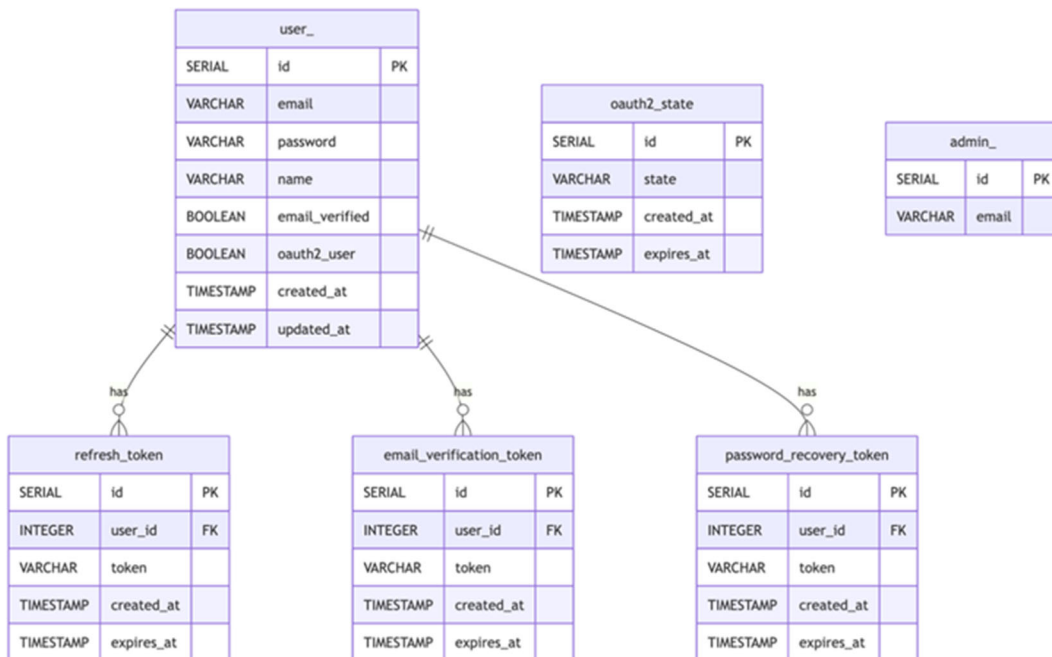


Fig. 4. Schema of the auth-svc database

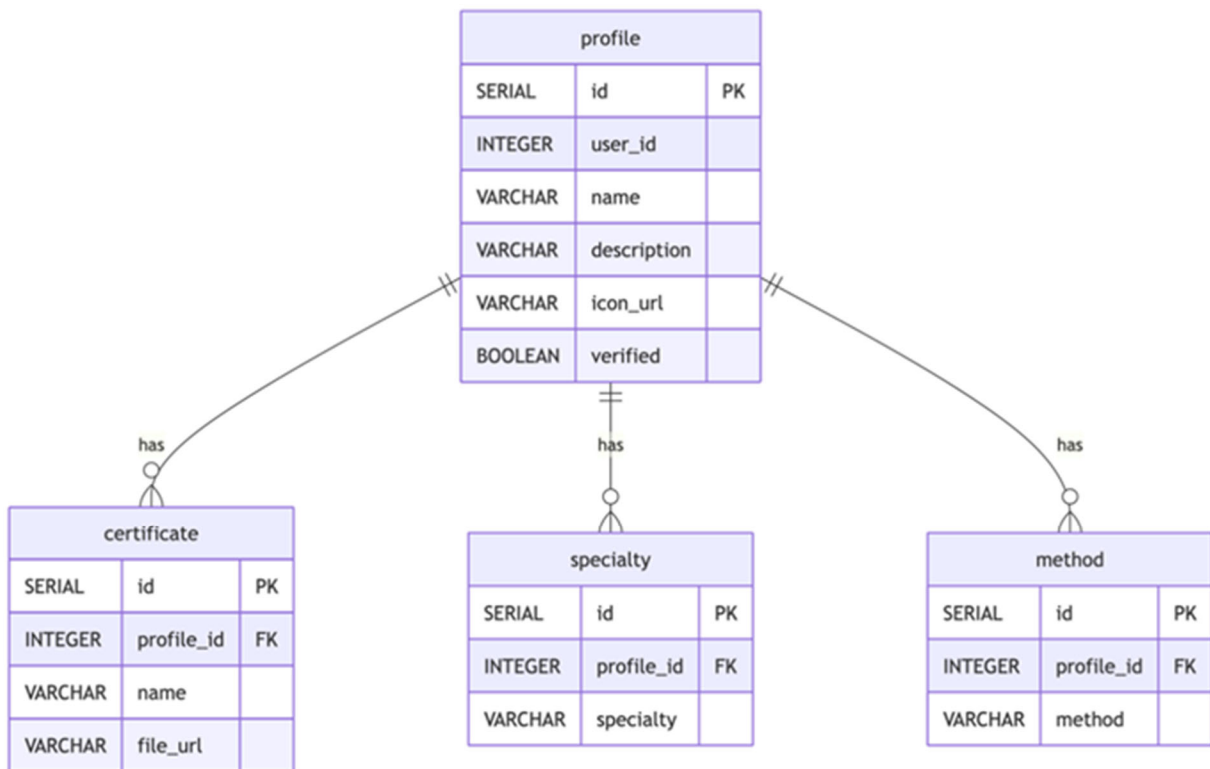


Fig. 5. Schema of the profile-svc database

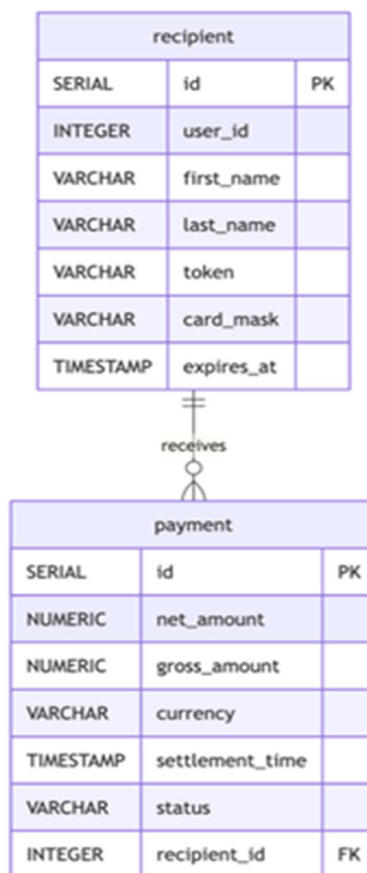


Fig. 6. Schema of the payment-svc database

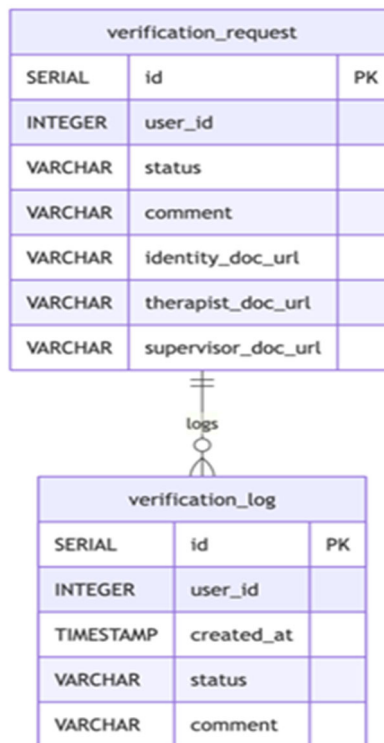


Fig. 7. Schema of the admin-svc database

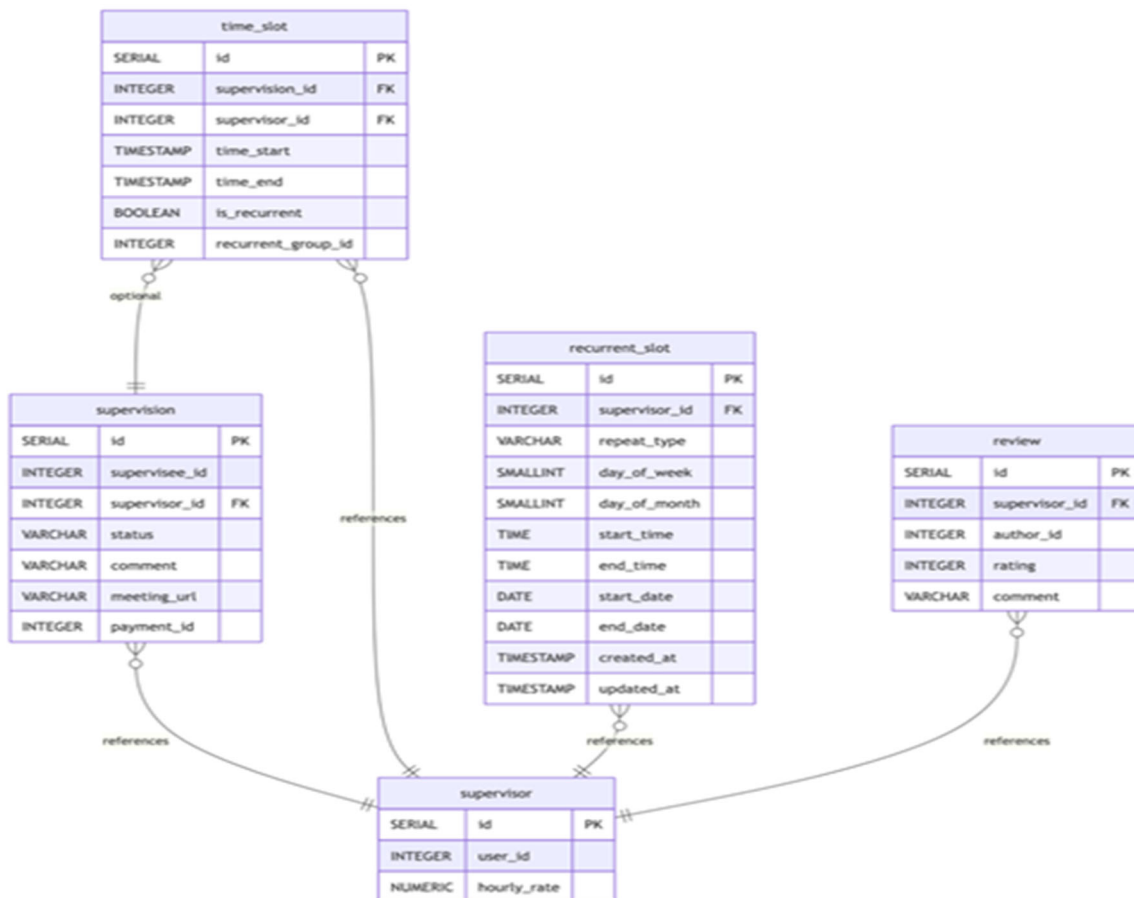


Fig. 8. Schema of the supervision-svc database

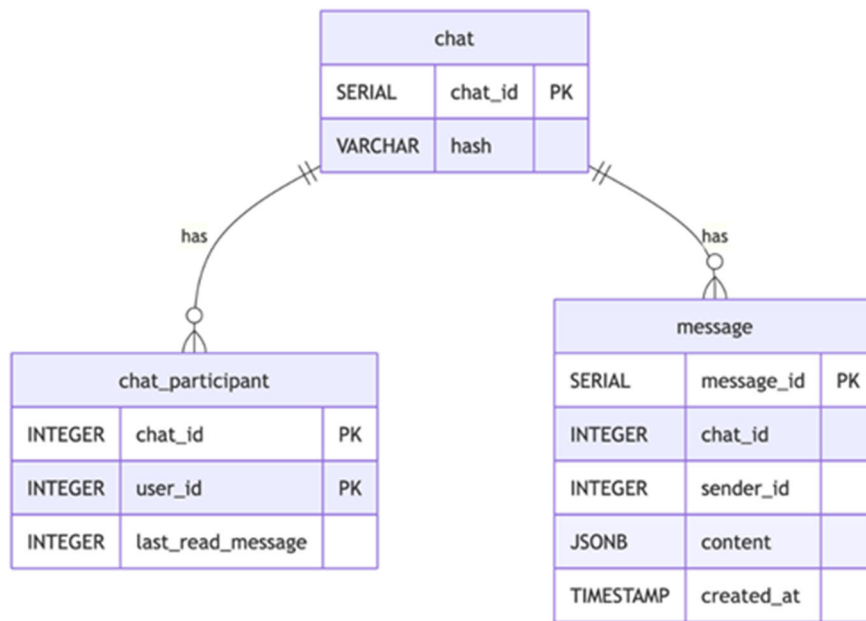


Fig. 9. Schema of the chat-svc database

UI/UX design of the product

Used UI/UX principles. At DocDirect, we prioritized user experience by adhering to key UI/UX principles in our website and application design. The following principles were used:

- consistency. We maintained a cohesive look and feel throughout DocDirect by using consistent fonts, colors, and button styles across all pages and features. This uniformity helps users intuitively navigate the platform, creating a seamless experience whether they are exploring the website or using the app;
- simplicity. Our design emphasizes simplicity and direct communication. The interface avoids unnecessary complexity, presenting information and actions in a straightforward manner. Clear labels, intuitive icons, and logical layouts ensure that users can easily understand how to interact with the platform;
- visual Hierarchy. To guide users effortlessly, we implemented a strong visual hierarchy. Larger, more prominent elements highlight key actions and information, while secondary content is styled subtly to avoid competing for attention. This approach helps users naturally navigate the interface and focus on what's most important.

Example of usage. First let's look at the Consistency. We have several major communication channels:

- social media;
- landing page;
- emails;
- application.

To make the brand more memorable and create a "web of associations" (e.g. a person will relate company's social media to the landing page, and to the product) we developed a color palette. Color palette was chosen to be of calm colors with red one being "contrast color" for emphasizing important elements (drag one's attention). Below at fig. 10, you can see above mentioned palette:



Fig. 10. Main palette of DocDirect

Then all this colors are being incorporated into different channels. Example at fig. 11-12. Apart from colors we also used unified style of Icons, Pictures, fonts across all channels.

Another UI/UX principle we applied so far is Simplicity. It is hard to overestimate the number of questions people can ask when first trying to use a new product. We understand that most of our users do not spend a lot of time on the internet, therefore we first pinpointed the goal of the user when he/she first tries out DocDirect and made all the UI around it.

We decided not to overpopulate the screen with icons and animations. Main page of the application contains the most important information:

- supervisors;
- filters (to find supervisor if you have a very specific needs);
- menu to navigate the application.

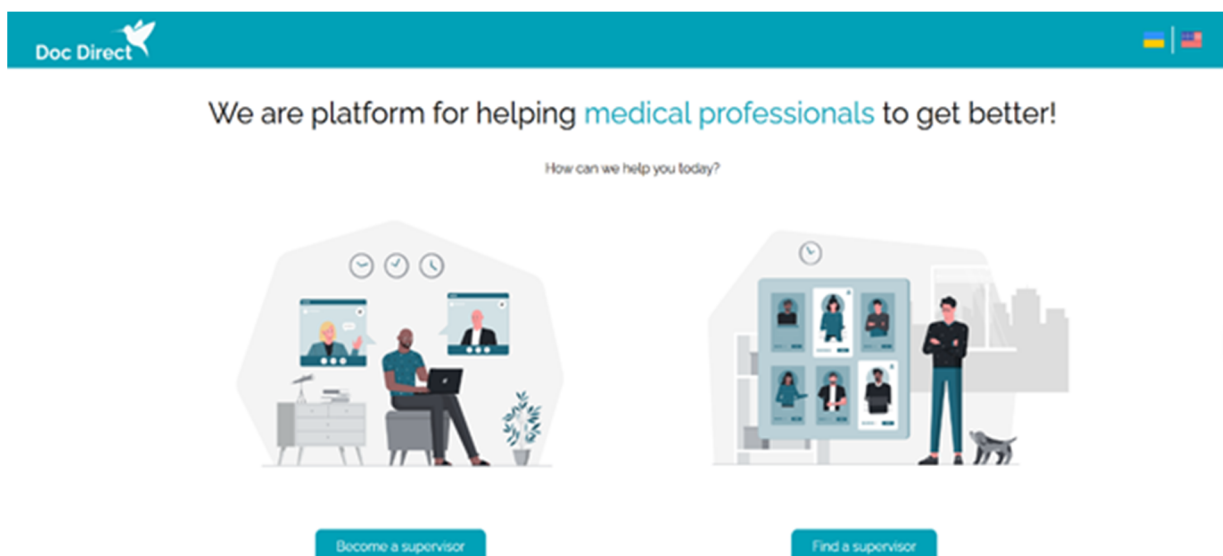


Fig. 11. Landing page of DocDirect

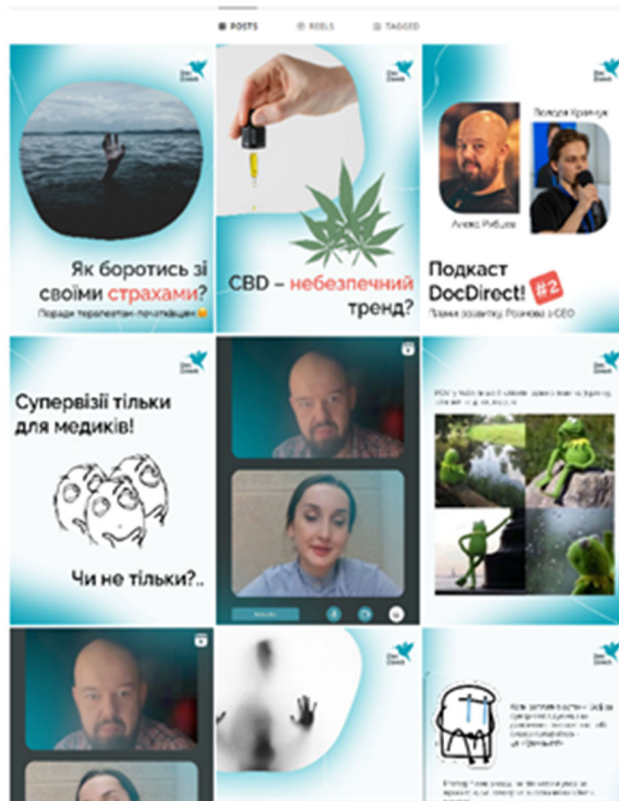


Fig. 12. Posts from Instagram page of DocDirect

As for Visual Hierarchy, we made important elements bigger and used more vibrant colors, while reserving calm colors for more of an additional information. At fig. 13 an example of this can be found when user first books supervision.

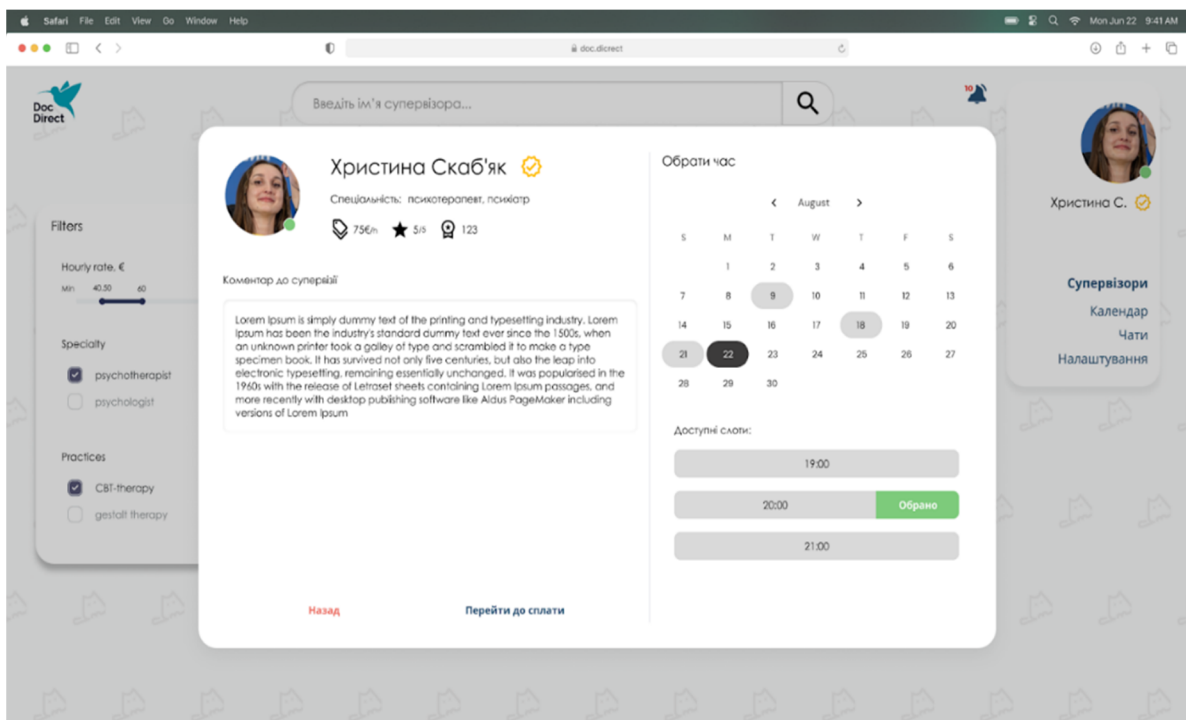


Fig. 13. Visual hierarchy principle from booking supervisor popup

Functions. Main functions of used UI/UX principles:

- consistency. We maintained a cohesive look and feel throughout DocDirect by using consistent fonts, colors, and button styles across all pages and features. This uniformity helps users intuitively navigate the platform, creating a seamless experience whether they are exploring the website or using the app.

- simplicity. Our design emphasizes simplicity and direct communication. The interface avoids unnecessary complexity, presenting information and actions in a straightforward manner. Clear labels, intuitive icons, and logical layouts ensure that users can easily understand how to interact with the platform.

- visual Hierarchy. To guide users effortlessly, we implemented a strong visual hierarchy. Larger, more prominent elements highlight key actions and information, while secondary content is styled subtly to avoid competing for attention. This approach helps users naturally navigate the interface and focus on what's most important.

Influence on the end user. Main influence on the end user:

- consistency. Consistency creates familiarity and trust for users. By maintaining uniform fonts, colors, and button styles across all pages and features, users can easily predict how the platform will behave. This reduces cognitive load, enabling users to focus on their tasks instead of figuring out how to use the interface. It builds confidence and reliability, ensuring a frictionless experience across both the website and the app;

- simplicity. Simplicity eliminates unnecessary complexity, allowing users to complete actions quickly and without confusion. A straightforward interface with clear labels and intuitive layouts means users can achieve their goals efficiently, regardless of their technical skill level. This simplicity fosters satisfaction, reduces frustration, and encourages continued engagement with the platform;

- visual Hierarchy. A strong visual hierarchy ensures users can effortlessly find what they need. By emphasizing key actions with larger and more prominent elements while toning down secondary content, the interface naturally guides users' attention. This approach minimizes decision fatigue, making the platform feel intuitive and efficient. Users appreciate how the design "works with them," helping them focus on their primary tasks without distraction.

3. PRODUCT IMPLEMENTATION AND TESTING

Overview. Our product is a sophisticated web platform with modern architecture, designed to ensure scalability, maintainability, and high performance. It utilizes a combination of state-of-the-art technologies, modular design principles, and automation to deliver a seamless user experience and efficient development workflow.

Frontened part. The frontend is crafted using Vue.js framework [6] with TypeScript language [7], prioritizing maintainability, scalability, and developer productivity. Key aspects include:

- dynamic UI leveraging Vue.js capabilities for responsive and interactive interfaces;
- efficient state management through Pinia library [17], ensuring consistent data flow across components;
- client-side routing with Vue Router for fast and intuitive navigation;
- integration of other standard tools and best practices for Vue.js projects, making the codebase both powerful and easy to maintain.

To ensure consistency between frontend and backend, the team employs an OpenAPI-driven development approach, generating API client code automatically and packaging it as a npm module for seamless consumption in the frontend.

Backend part. The backend is implemented using a microservices architecture, with each service written in Go language [8] for performance and reliability. The services are designed to operate independently, ensuring modularity and flexibility. These include:

- authentication service for user login, registration, and session management;
- supervision service to handle specific domain-related monitoring tasks;
- profile service for managing user account and profile data;
- chat service providing real-time or asynchronous messaging capabilities;
- contact service for organizing and maintaining user relationships;
- email service enabling robust email notification and communication systems;
- payment service driving financial transactions;
- admin service for elevated access controls.

Infrastructure and middleware part. The platform is hosted on DigitalOcean [9], leveraging its cloud solutions for reliable and scalable deployment. To optimize performance and handle traffic efficiently, NGINX [10] is used as a reverse proxy and load balancer, ensuring fast and secure communication between the client and server. The overall system architecture presents at fig. 14.

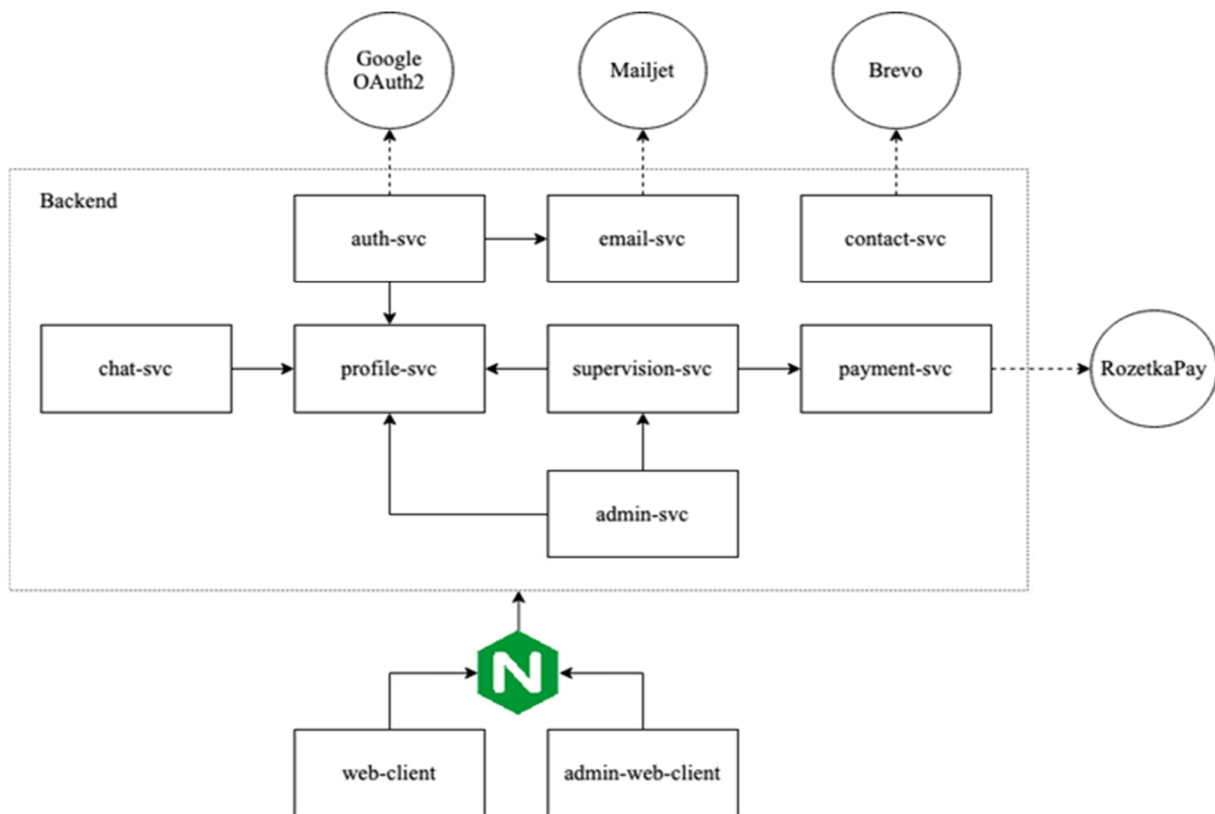


Fig. 14. System architecture

Automation and workflow. A comprehensive CI/CD pipeline underpins the development workflow, ensuring consistent quality and fast iteration cycles:

- code quality checks: automated linting and Prettier formatting during pull requests;
- version control discipline: enforced semantic commit specifications for meaningful and structured version history;
- continuous deployment: automated rollout to DigitalOcean [9] upon successful merges, minimizing manual intervention.

System and product testing.

Unit tests. Unit tests for our system are present at table 4. Part of testing journal is present at table 5.

Table 4. Unit tests

Test Name	Test Purpose	Other Information
axiosClient - should include the Authorization header in the request	Verifies that the Authorization header is set correctly before making a request.	Focuses on the correct behavior of the Axios request interceptor.
axiosClient - should handle non-Axios errors gracefully	Focuses on the correct behavior of the Axios request interceptor. axiosClient - should handle non-Axios errors gracefully. Verifies that non-Axios errors (e.g., JS errors) are handled properly.	Ensures graceful handling of non-Axios errors in the axiosClient.
PopupManager - should open a popup and add it to the stack	Tests if the PopupManager correctly opens and adds a popup to its stack.	Focuses on the behavior of the open method of PopupManager.
PopupManager - should not open a popup if open returns false	Verifies that the PopupManager does not add a popup if it cannot be opened.	Ensures that the open method behaves as expected when the popup cannot open.
PopupManager - should close a popup and remove it from the stack	Verifies that the PopupManager can successfully close a popup and remove it.	Focuses on the behavior of the close method of PopupManager.
PopupManager - should not close a popup that is not in the stack	Ensures that PopupManager does not close popups that are not in the stack.	Verifies that the close method does not erroneously remove popups not in the stack.
PopupManager - should close all popups	Verifies that the PopupManager can close all currently open popups.	Ensures the behavior of closeAll is correct when multiple popups are open.
create_new_client	Verifies that the function creates a new client instance correctly with the required base setup.	Ensures the client is initialized with proper configurations.
create_a_new_client_correctly_without_trailing_slash	Checks that the client is created even if the base URL does not have a trailing slash.	Helps standardize the base URL structure.
create_new_client_with_default_headers_to_add_in_request	create_new_client_with_default_headers_to_add_in_request	create_new_client_with_default_headers_to_add_in_request

Table 5. Testing journal of unit tests

Test Name	Execution Date	Test Status	Notes
axiosClient - should include the Authorization header in the request	20.11.2024	Passed	Authorization header added successfully.
axiosClient - should handle non-Axios errors gracefully	20.11.2024	Passed	Non-Axios errors handled gracefully.
PopupManager - should open a popup and add it to the stack	20.11.2024	Passed	Opened popup is added to stack correctly.
PopupManager - should not open a popup if open returns false	20.11.2024	Passed	Popup is not added when open returns false.
PopupManager - should close a popup and remove it from the stack	20.11.2024	Passed	Popup closed and removed from stack.
PopupManager - should not close a popup that is not in the stack	20.11.2024	Passed	Correct behavior when popup is not in the stack.
PopupManager - should close all popups	20.11.2024	Passed	All popups were closed as expected.
create new client	20.11.2024	Passed	
create a new client correctly without trailing slash	20.11.2024	Passed	
create_new_client_with_default_headers_to_add_in_request	20.11.2024	Passed	

Integration tests. Integration tests for our system are present at table 6. Part of testing journal is present at table 7.

Table 6. Integration tests

Test Name	Test Purpose	Other Information
PopupManager Integration - should be able to open a popup from within the app	Ensures that the popup manager can open a popup and add it to the stack.	Verifies that the PopupManager correctly interacts with a mounted component.
PopupManager Integration - should open and close popups through the popupManager within the app	Tests opening and closing popups via PopupManager and checks the popup stack.	Verifies that popups can be both opened and closed correctly.
PopupManager Integration - should close all popups when calling closeAll	Ensures that closeAll closes all currently open popups.	Verifies that multiple popups are handled correctly when closing all.
axiosClient - should refresh token and retry the request when receiving 401	Tests that a 401 error triggers a token refresh and the request is retried.	Verifies that the token refresh process works when encountering 401 errors.
axiosClient - should not refresh token if the error is not 401	Ensures that a non-401 error does not trigger a token refresh.	Verifies that the axiosClient correctly handles non-401 errors.

Table 7. Testing journal of integration tests

Test Name	Execution Date	Test Status	Notes
PopupManager Integration - should be able to open a popup from within the app	20.11.2024	Passed	Works as expected without any issues.
PopupManager Integration - should open and close popups through the popupManager within the app	20.11.2024	Passed	All interactions are handled correctly.
PopupManager Integration - should close all popups when calling closeAll	20.11.2024	Passed	Passed
axiosClient - should refresh token and retry the request when receiving 401	20.11.2024	Passed	Token refresh works as expected.
axiosClient - should not refresh token if the error is not 401	20.11.2024	Passed	Non-401 errors are handled properly.

Output of unit tests. Output of some unit tests you can find at fig. 15-16.

```

✓ tests/integration/PopupManager/PopupManager.integration.spec.js (3)
✓ tests/unit/PopupManager/PopupManager.spec.js (5)
✓ tests/unit/axiosClient/axiosClient.spec.js (4)

Test Files  3 passed (3)
Tests      12 passed (12)
Start at   15:48:33
Duration   3.05s (transform 801ms, setup 0ms, collect 2.19s, tests 367ms, environment 2.79s, prepare 544ms)
    
```

Fig. 15. Output of some unit tests from frontend

```

✓ Test Results 210 ms ✓ Tests passed: 4 of 4 tests - 210 ms
><4 go setup calls>
=== RUN   TestNewClient
=== RUN   TestNewClient/create_new_client
--- PASS: TestNewClient/create_new_client (0.00s)
=== RUN   TestNewClient/create_a_new_client_correctly_without_trailing_slash
--- PASS: TestNewClient/create_a_new_client_correctly_without_trailing_slash (0.00s)
=== RUN   TestNewClient/create_new_client_with_default_headers_to_add_in_request
--- PASS: TestNewClient/create_new_client_with_default_headers_to_add_in_request (0.00s)
--- PASS: TestNewClient (0.00s)
PASS
Process finished with the exit code 0
    
```

Fig. 16. Output of some unit tests from backend

Traffic on landing page. We are using google analytics [18] to monitor user flow to our landing page. It is connected via google tags on the frontend and enables us to see traffic on the page.

Here are some example statistics. Screenshots with traffic data on landing page are present at fig. 17.

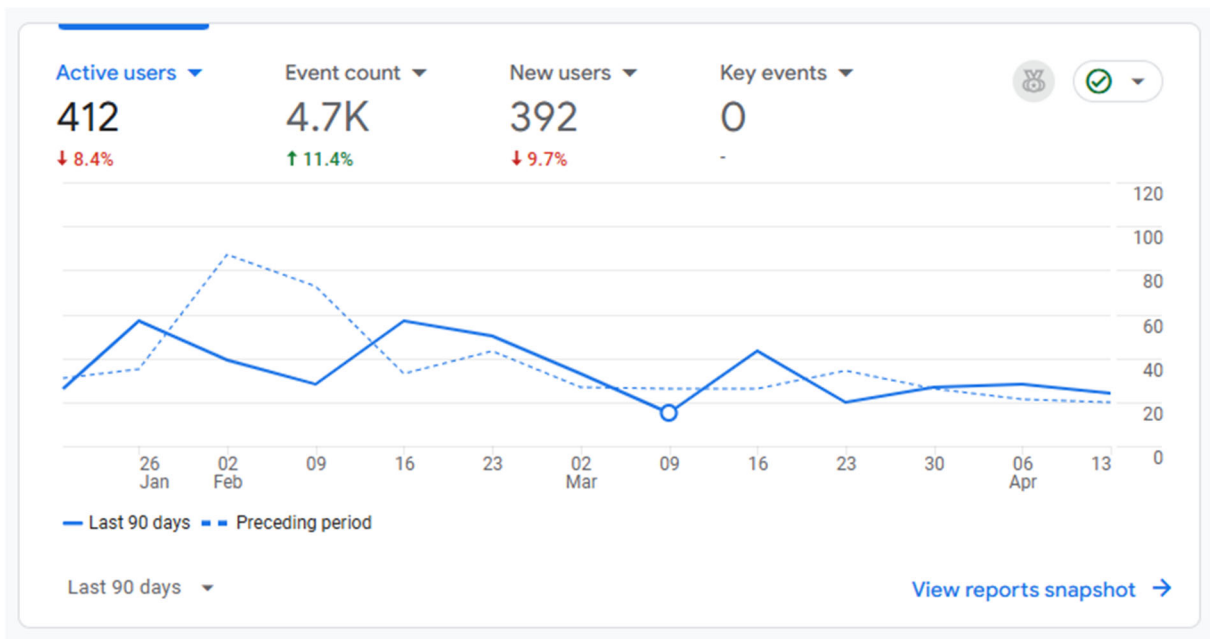


Fig. 17. Statistic of active users on landing page

Clean code tools. Maintaining a clean, consistent, and error-free codebase is critical in any development workflow, especially in collaborative projects. To achieve this, we integrate a suite of clean code tools, such as linting, Prettier, and naming conventions, into our CI pipeline. They test our code each run and if something goes wrong we can easily fix our code, or adjust our conventions.

In particular linting plays a pivotal role in testing code for security vulnerabilities and reliability issues during development. By analyzing source code against a set of predefined rules, linters help developers catch potential risks early, ensuring robust and secure applications. Example of testing results is present at fig. 18.

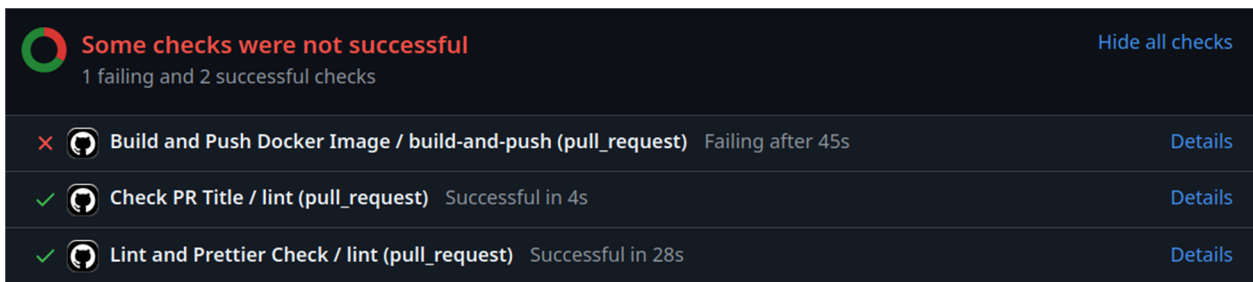


Fig. 18. CI pipeline with tests

QA testing. To ensure that all use cases are evaluated, we have created a QA testing journal. It includes tests for all out main feature-blocks: Authorization, Supervisors, Calendar, Chats, Page settings, Profile, Admin panel. As the outcomes of QA testing we create needed ticket for fixing. See an example from our journal of testing at fig. 19.

Тест	Шлях	Очікуваний результат	Фактичний результат	Статус
[Admin Panel] Авторизація як адміністратор.		Без авторизації як адміністратор немає доступу до платформи. Переїдає на сторінку логіна. У разі авторизації доступний весь функціонал.	*	PASSED
[Profile settings] Верифікація як терапевт із коректними документами.		Можна завантажити 2 відповідних документи. У користування відображається як документи він завантажив ????. Залишити коментар. При надходженні даних, заявка з'являється на адмінській панелі, статус у профілі змінюється на: Рев'ю. Після апрува заявки - статус змінюється на approved."	Прод. [{"title": "Bad Request", "detail": "Your request was invalid in some way", "status": "400"}]	PARTIALLY PASSED
[Profile settings] Сторінка верифікації містить актуальні статуси верифікації.		Відображено актуальні статуси верифікації як терапевта і супервізора.	Прод. Verify для терапевта є, для супервізора ні, помилка: https://api.doc.direct/verification-requests/supervisor/404 Not Found	PARTIALLY PASSED
[Profile settings] Верифікація як терапевт із некоректними документами.		При спробі надіслати без потрібних 2 документи - нічого не відбувається. При редакції заявки - статус змінюється на rejected."	Запроси не відправляються без документів. Дані не вдається протестувати через помилку. [{"title": "Bad Request", "detail": "Your request was invalid in some way", "status": "400"}]	PARTIALLY PASSED
[Profile settings] Загрузка нового фото.		При натисканні на кнопку завантажити нове фото, дається можливість завантажити фото з комп'ютера. При оновленні, воно оновлюється у формі. Але на сайті оновлюється тільки тоді, коли натискається кнопка збереження заявки.	Заргувати фото можна. При нажатті на save changes нічого не відбувається, не відправляється запит навіть на бек.	PARTIALLY PASSED
[Profile settings] Зміна інформації профіля.		Є можливість змінити поточну інформацію профілю. Але вона змінюється тоді, коли натискається кнопка зберегти зміни.	Заргувати фото можна. При нажатті на save changes нічого не відбувається, не відправляється запит навіть на бек.	PARTIALLY PASSED
[Profile settings] Верифікація як супервізор з коректними документами.		Під час спроби поміняти пароль, видно підказки, що паролі не збігаються. При натисканні на кнопку підтвердити пароль змінюється: "Можна завантажити 3 відповідних документи. У користування відображається як документи він завантажив ????. Залишити коментар. При надісланих даних, заявка з'являється на адмінській панелі, статус у профілі змінюється на: Рев'ю. Після апрува заявки - статус змінюється на approved. Користувач сповіщений поштою про зміну статусу ????. При спробі надіслати ще один запит, коли вже 1 в ревію - ????"	Відсутня кнопка Verify. https://api.doc.direct/verification-requests/supervisor/404 Not Found	FAILED
[Profile settings] Верифікація як супервізор з некоректними документами.		Можна завантажити 3 відповідних документи. У користування відображається як документи він завантажив ????. Залишити коментар. При надісланих даних, заявка з'являється на адмінській панелі, статус у профілі змінюється на: Рев'ю. При спробі відправити без потрібних 3 документи - сповіщається про помилку ????. Після редакції заявки - статус змінюється на rejected. Користувач сповіщений поштою про зміну статусу ????. При спробі надіслати ще один запит, коли вже 1 в ревію - ????"	Відсутня кнопка Verify. https://api.doc.direct/verification-requests/supervisor/404 Not Found	FAILED

Fig. 19. Example from QA journal of testing

4. SYSTEM AND PRODUCT DOCUMENTATION

Approach and tools we used. At the initial stages of the project, we opted for Notion [2] as the primary tool for documentation management. This choice was driven by its intuitive interface, robust structuring capabilities, and seamless collaboration features. At fig. 20, there is an example of our Notion workspace (for instance, our scrum board was taken).

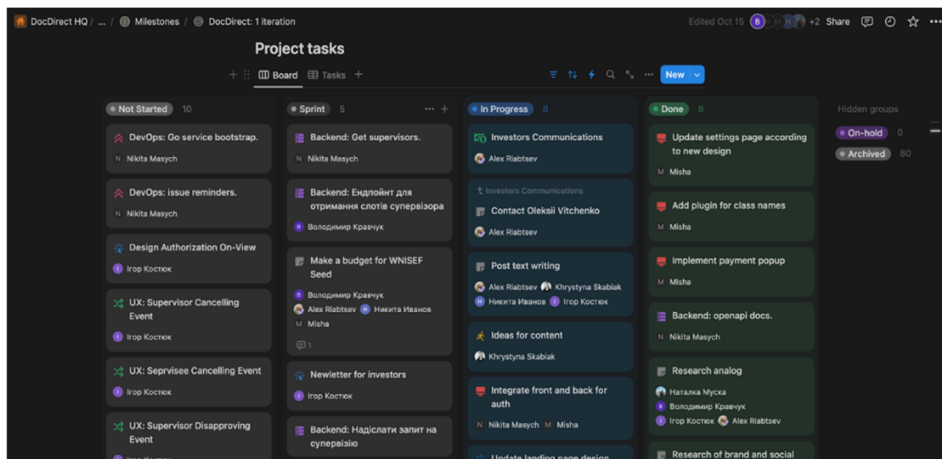


Fig. 20. Scrum board from Notion workspace

We used Notion as an ultimate knowledge database, so in order to keep it as the core of our information system, we stored there links to other services as well.

However, as the volume of documentation grew and the team expanded, a need for a more specialized solution arose as well as being more cost efficient. Consequently, we migrated to Superthread [5], a platform better suited to our requirements for integration, structured documentation, and access control. Superthread enabled more efficient document management and simplified the process of updating and synchronizing information across departments. See how looks our board at Superthread at fig. 21.

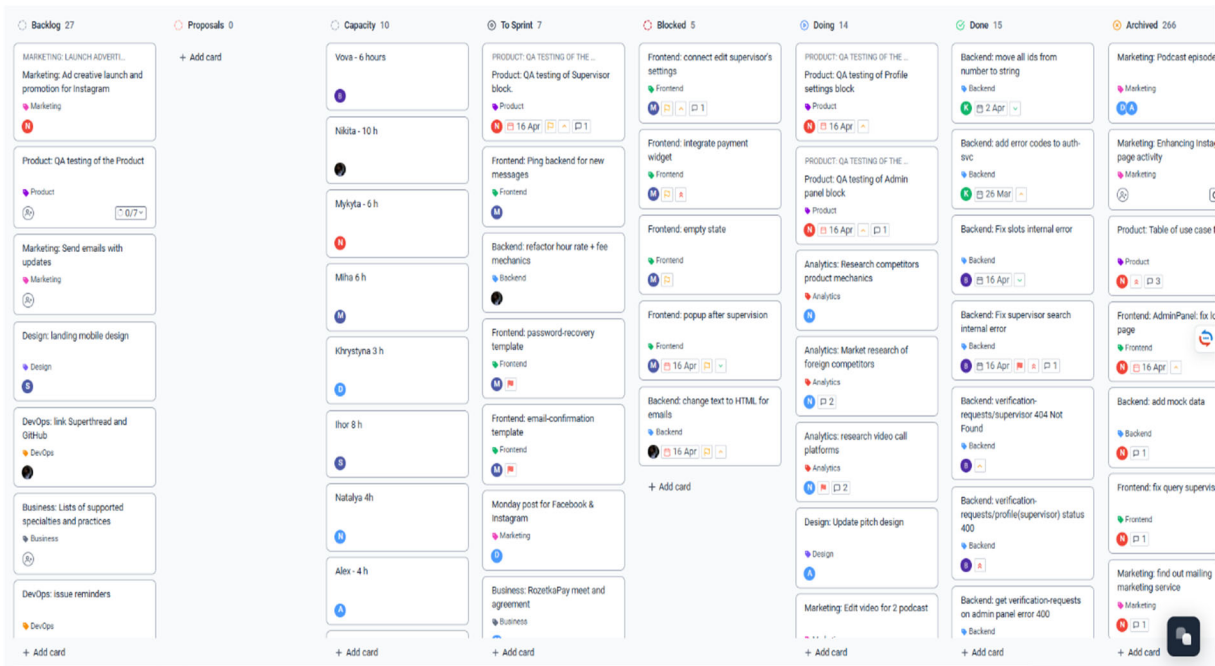


Fig. 21. New board looks at Superthread

Superthread has similar features to Notion, however it has some strong advantages that forced us to initiate migration:

- cost efficiency. Superthread has a free plan for teams that has no critical disadvantages compared to Notion;
- simplicity and minimalism. Superthread way more simple and user friendly that makes it easier to onboard newcomers;
- logotype. It is the same bird as DorDirect logotype, so we decided it is a crucial sign of the faith to choose Superthread.

Types of documents we used. In order to keep documentation clean and understandable, we have separated our knowledge base into different types.

Internal documents. These are the backbone of our internal operations. They include meeting notes, team updates, strategy outlines, and process guidelines—essential resources for seamless collaboration and knowledge sharing within the organization. Internal documents are separated into such different types:

- product Documentation. A comprehensive collection of resources dedicated to understanding and improving our product. This includes feature descriptions, user guides, release notes, and product roadmaps, ensuring everyone is aligned on delivering value to our users;
- technical Documentation. For our developers and technical teams, this section contains code repositories, API references, system architecture diagrams, and troubleshooting guides. It's the go-to resource for building and maintaining our technological infrastructure;
- MarTech Documentation. This category focuses on marketing technology—tools, analytics setups, campaign strategies, and automation processes. It bridges the gap between marketing goals and the technical implementations required to achieve them;
- accounting Tables. A structured repository for all financial documentation, including budgets, expense tracking, revenue forecasts, and tax filings. This ensures transparency and accuracy in our financial management. See fragment of this table at fig. 22.

Recurring payments			
Name	Price	Price per month	Once in a
Total		\$65.69	Month
Domain Renewal	\$28.26	\$2.36	Year
Cluster Cost	\$43.33	\$43.33	Month
GitHub Team	\$20	\$20	Month

Fig. 22. Fragment from the accounting table

Documents for external people. Resources tailored for clients, partners, and collaborators. These might include service agreements, FAQs, onboarding guides, and collaboration tools designed to strengthen external relationships. Such documents are separated into such different types:

- pitch decks. Carefully crafted presentations aimed at investors, stakeholders, or potential partners. They highlight our vision, achievements, and future goals in a visually engaging and impactful format. Example of our pitch deck you can see at figure 23;
- public Superthread Pages. Public-facing documents that consolidate our most valuable insights, updates, or announcements. They are designed to engage and inform external audiences while reflecting our expertise and values.

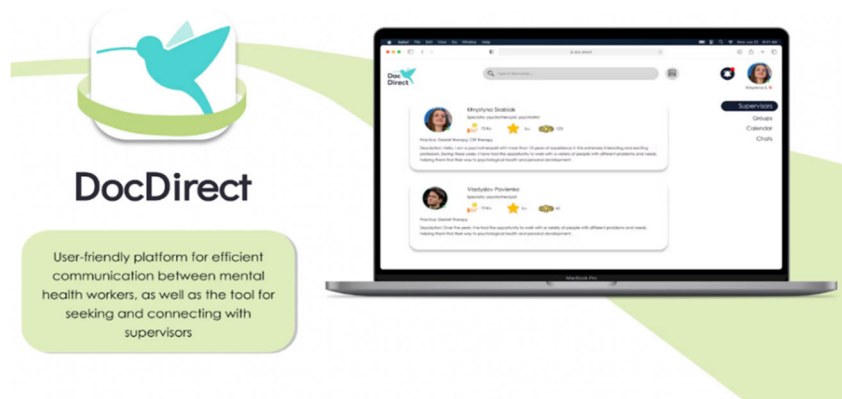


Fig. 23. Slide from DocDirect pitch deck

By segmenting our knowledge base in this way, we ensure that each document is easy to locate, tailored to its audience, and serves its purpose effectively.

Instruction of using our product. Using our platform is simple and intuitive. Before the registration process, the user gets on the landing page of DocDirecy [1], where user can choose to become either supervisor or supervisee. See an example at fig. 24.

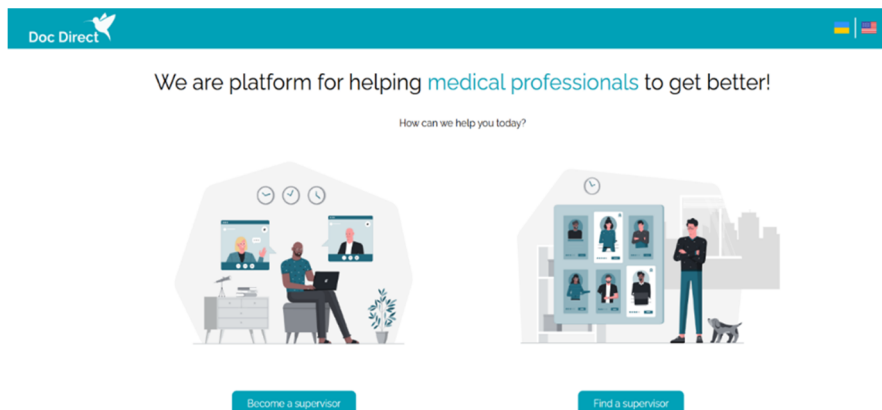
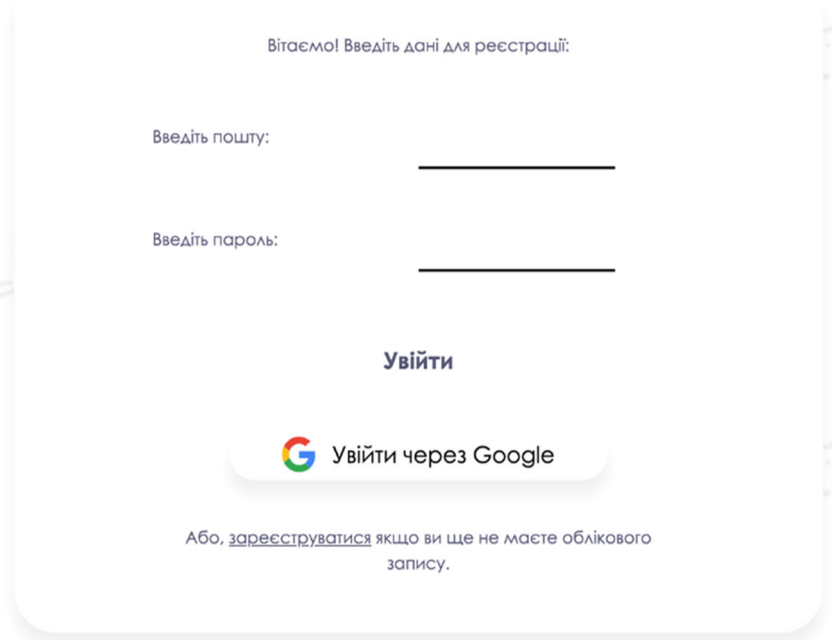


Fig. 24. Landing page of DocDirect

Follow these steps to get started and connect with a supervisor tailored to your needs:

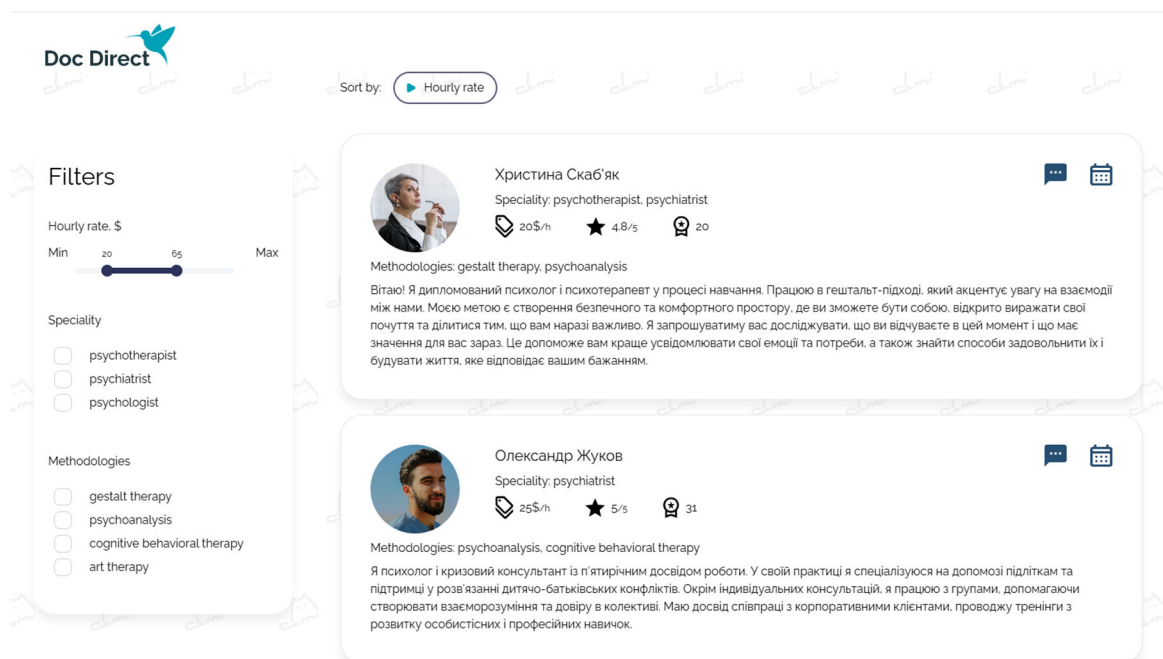
- register with Ease. Sign up effortlessly using your Google account or email address. This ensures a quick and secure onboarding experience. See view of the registration form at fig. 25;



The registration form is titled "Вітаємо! Введіть дані для реєстрації:" (Welcome! Enter data for registration:). It contains two input fields: "Введіть пошту:" (Enter email) and "Введіть пароль:" (Enter password). Below the fields is a blue "Увійти" (Log in) button. Underneath is a button with the Google logo and the text "Увійти через Google" (Log in with Google). At the bottom, there is a link: "Або, [зареєструватися](#) якщо ви ще не маєте облікового запису." (Or, [register](#) if you don't have an account yet.)

Fig. 25. Registration form

- Explore the List of Supervisors. Browse through a diverse and qualified pool of supervisors, each with detailed profiles to help you make an informed choice. Use our advanced filtering options to find the supervisor that best fits your preferences. Filter by expertise, language, availability, or other criteria. See example at fig. 26;



The screenshot shows the "Doc Direct" platform interface. At the top left is the logo. Below it, there's a "Sort by:" dropdown menu set to "Hourly rate". On the left side, there are "Filters" for "Hourly rate, \$" (with a slider from 20 to 65), "Speciality" (with radio buttons for psychotherapist, psychiatrist, and psychologist), and "Methodologies" (with radio buttons for gestalt therapy, psychoanalysis, cognitive behavioral therapy, and art therapy). The main content area displays two supervisor profiles. The first profile is for Христина Скаб'як, a psychotherapist and psychiatrist, with a 20\$/h rate, a 4.8/5 rating, and 20 reviews. Her bio mentions gestalt therapy and psychoanalysis. The second profile is for Олександр Жуков, a psychiatrist, with a 25\$/h rate, a 5/5 rating, and 31 reviews. His bio mentions psychoanalysis and cognitive behavioral therapy.

Fig. 26. List of supervision view

- choose the needed supervisor. Click on your desired profile to get more information about supervisor. See an example at fig. 27;
- schedule an appointment. Choose a time that works for you and book your appointment directly through the platform. See an example at fig. 28;
- attend your meeting. Join your scheduled meeting with your chosen supervisor to start your personalized journey toward growth and insight. See an example at fig. 29.

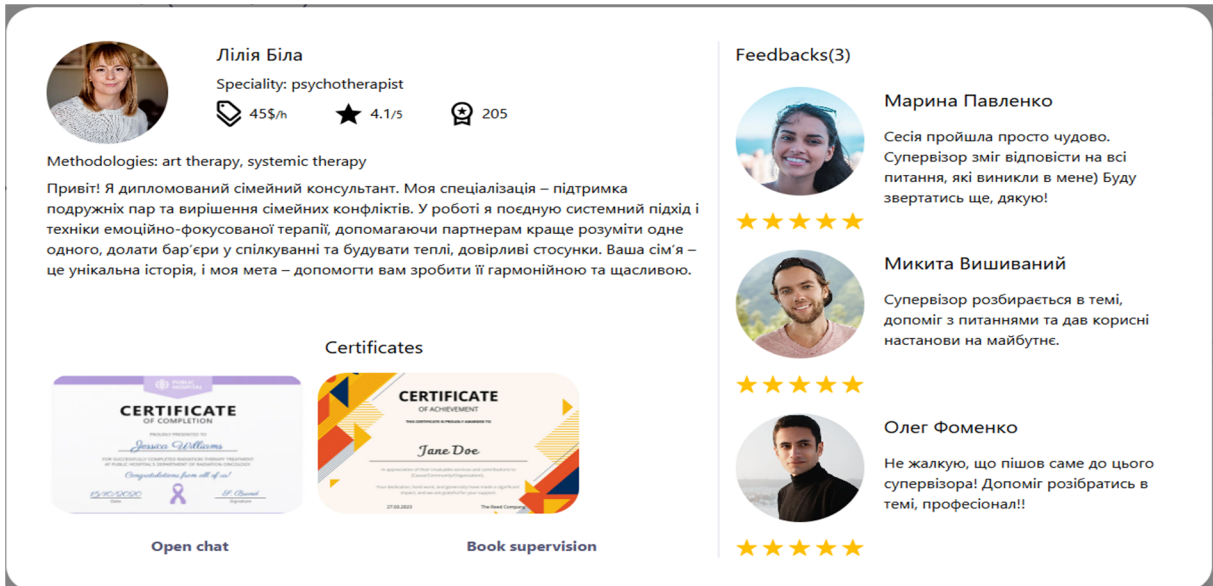


Fig. 27. Supervisor info view

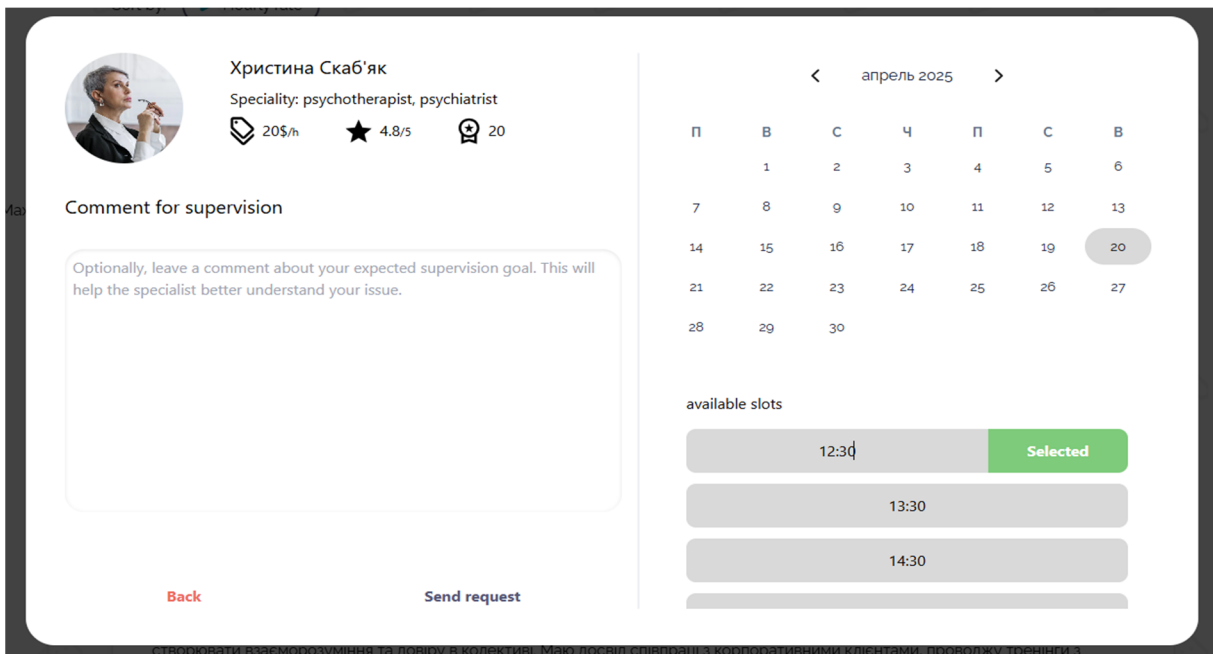


Fig. 28. Schedule appointment view

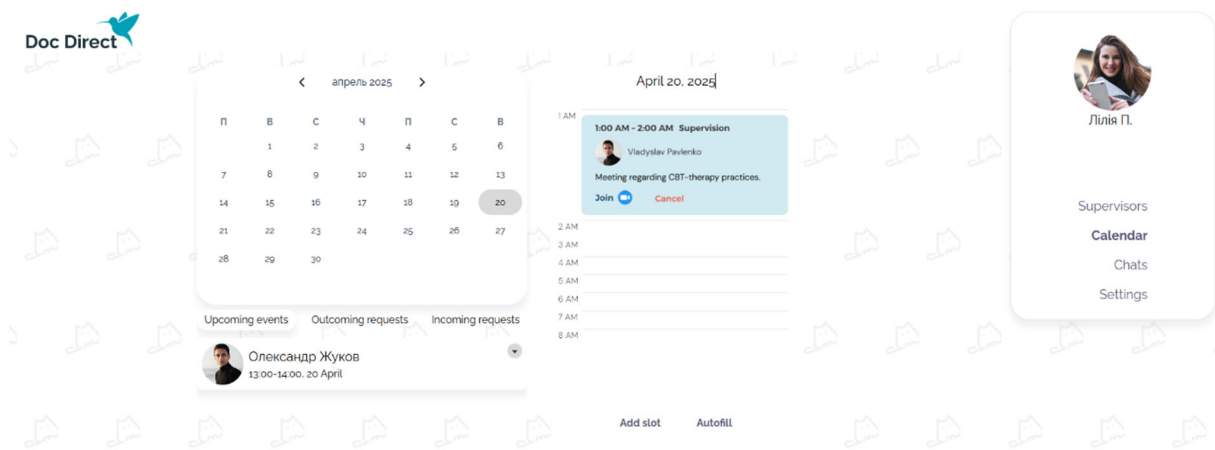


Fig. 29. Calendar view

By these easy steps users can book supervision.

Legal framework. We have reviewed a range of normative acts in the areas of telemedicine, healthcare, electronic document management, e-commerce, communications, and information/privacy protection. The two primary laws most directly relevant to DocDirect are:

- The Law of Ukraine "On Personal Data Protection" [19]. It establishes requirements for the collection, processing and storage of personal data, ensuring patient confidentiality and compliance with data-privacy standards.
- The Law of Ukraine "On the Mental Health Care System" [20]. It establishes a unified mental-health care system by defining the roles of central and local authorities, setting professional standards for specialists and crucially mandating the creation of a state-maintained Register of Mental-Health Service Providers (and self-regulatory organizations) interoperable with other government databases; this registry framework will allow DocDirect to integrate directly for real-time credential validation, automated onboarding, and transparent verification of all therapists and supervisors on our platform.

5. DESCRIPTION OF THE DEVELOPMENT PROCESS

Team members. Team members and their roles with responsibilities are described in table 8.

Table 8. Team members with roles and responsibilities

Member Name	Roles	Detailed description
Volodymyr Kravchuk	CEO / Backend developer	Communication with Investors. Pitching, overseeing long-term goals. Helping develop a server part of web-application.
Mykyta Ivanov	Marketing specialist, frontend developer	Building communication with the community through social networks/ads. Developing visual part of web-application. Helping develop a visual part of web-application.
Mykhailo Statnik	Frontend developer, Finance specialist	Developing a visual part of web-application. Helping manage company finances.
Kostiuk Ihor	Designer	Designing all visual parts of the product (web-app, landing, etc.). Overseeing short-term goals.

Natalia Muska	Product Analyst	Researching the market. Helping build a more in-depth profile of our potential clients.
Nikita Masych	CTO, Backend Developer	Developing a server part of web-application. DevOps guru.
Alex Riabtsev	Sales Specialist	External member. Communication with investors. Advisor on long-term strategy.
Khrystyna Skabiak	Medical Expert	External member. Communication with medical specialists, guidance through nuances of the medical system in Ukraine.
Kysil Kateryna	Backend trainee	Help with developing of the server part of web-application.
Laura Moskalenko	Designer	Designing visual parts of the marketing campaign.

Methodology and instruments we used. At DocDirect, we have adopted the Agile methodology with a Kanban board approach to streamline our development process and maintain flexibility in our workflows. This approach enables us to deliver consistent improvements while responding to feedback and adapting to changing requirements.

We operate in 1-week sprints, allowing our team to focus on manageable, incremental goals within a short timeframe. This ensures that progress remains steady and that we can adapt quickly to any challenges or changes in priorities. At the end of each sprint, we conduct a sprint retrospective to evaluate the previous week's work. During this session, we discuss:

- what went well;
- what could be improved;
- actions we can take to optimize future workflows.

These retrospectives are instrumental in fostering a culture of continuous improvement and collaboration.

Our Kanban board provides a clear visual representation of tasks and their statuses. The board is divided into columns that reflect the stages of our workflow, such as:

- backlog: tasks awaiting prioritization or planning;
- capacity: to plan how much hours we are planning to use in the sprint;
- blocked: to know what tickets currently blocked;
- in progress: current active work;
- completed: successfully finished work;

Example of our board before starting of the sprint is at figure 30.

This system enables team members to track progress immediately, manage workloads effectively, and ensure that no task falls through the cracks.

While our workflow is structured, we understand the importance of adaptability. To that end, we encourage internal meetings between teams whenever needed to address blockers, brainstorm solutions, or align specific objectives. These ad-hoc sessions foster communication and prevent delays, ensuring that cross-functional collaboration remains efficient and productive.

Instruments we used:

- Notion for task tracking, documentation creating.;
- EasyRetro for retrospective sessions;
- Google Meet for meetings;
- Telegram for internal communication.

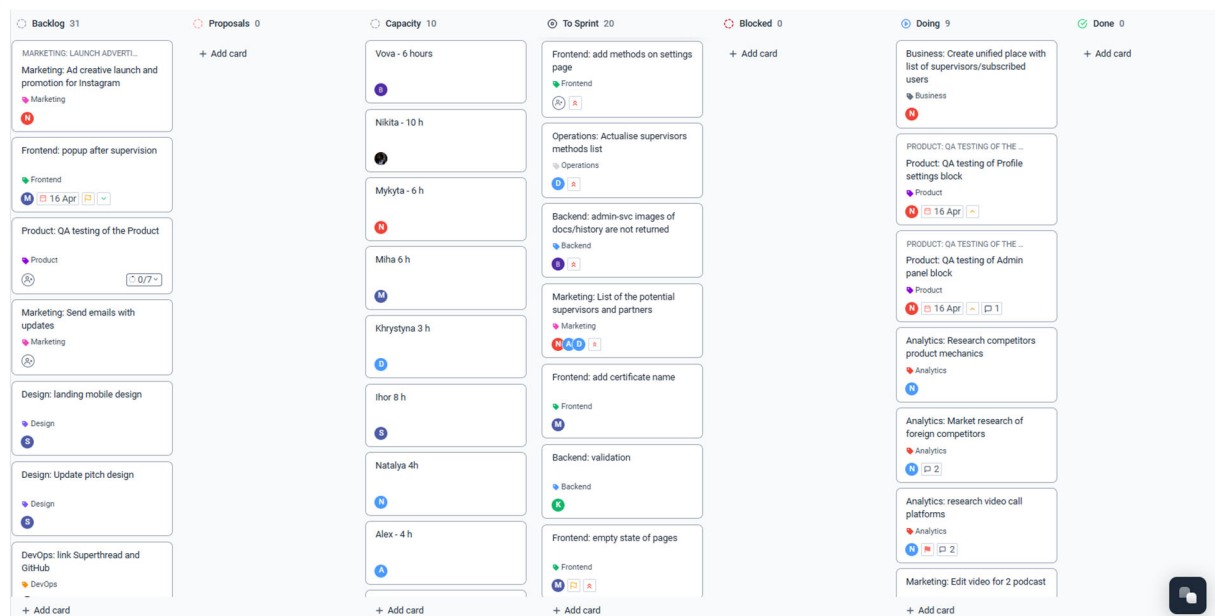


Fig. 30. Board view

Reflection. We've participated in multiple pre-acceleration and acceleration programs, gaining access to invaluable mentorship and resources. These programs not only provided insights into refining our business model and strategy but also allowed us to expand our professional network significantly. This expanded network has connected us to industry leaders, potential partners, and fellow entrepreneurs who continue to inspire and support our journey.

On the development side, we've made significant progress:

- frontend development: web applications for users and for admins are created. They are on the stage of testing. After that it will be ready for production;
- backend: full integration with frontend, almost all processes were tested, created all needed for MVP services. Backend part is on the stage of QA testing. After that it will be ready for production.

Our development also included active communication with external products. For example, to create payment workflow, we have communicated with a lot of possible business that could give us the necessary possibility.

Through our efforts, we've caught the attention of several investors interested in DocDirect. This is a strong indicator of the value and potential of our product. These interactions have also given us insights into what investors prioritize, helping us tailor our approach to meet their expectations while staying true to our vision.

We've begun fostering meaningful connections with both supervisors and supervisees, the core users of our platform. These conversations help us better understand their needs, refine our value proposition, and build anticipation for the product. This early engagement is critical for ensuring that DocDirect aligns closely with the real-world requirements of our audience.

All the above successes are due to hard work of all team members. And we are moving forward to achieve new heights! Unfortunately, there were some setbacks:

- the demanding nature of working on a startup while balancing other professional commitments has been a considerable challenge. For many of our team members, DocDirect is like a second full-time job—but without a salary. This intense workload has often led to exhaustion, forcing us to move deadlines forward. As a result, we have the following:
 - slower progress on certain milestones;
 - occasional dips in morale and productivity due to burnout.

But we've realized the importance of setting realistic goals, prioritizing tasks, and allowing room for flexibility. Moving forward, we're exploring ways to better distribute workloads and leverage external resources where possible;

- at the start, we dedicated very little attention to our social media and communication channels. While we were focused on product development and internal operations, we missed opportunities to engage with our audience, build a following, and share our journey with potential users and supporters. As a result, we have the following:

- missed chances to generate early interest and build brand awareness;
- slower community growth around the product.

But now, we recognize that social communication is not just a secondary task but an essential part of building a startup. A consistent and engaging presence on social platforms helps foster community, attract users, and show investors that there's demand for the product. We've begun to address this by integrating social communication into our strategy.

Learned lessons. Building DocDirect has been an incredible journey filled with valuable lessons that have shaped how we operate and grow. Here are some of the most impactful insights we've gained:

- **mastering time management, task prioritization, and risk assessment.** In the fast-paced world of startups, every moment and every decision carries weight. We've learned to structure our days around high-impact activities—breaking work into clear, achievable milestones and using tools like Kanban boards to stay focused on what truly moves the needle. This disciplined approach not only keeps our team aligned and productive but also empowers us to anticipate challenges, pivot swiftly when needed, and ensure that our resources are always steered toward the most valuable opportunities;

- **the importance of a strong first impression.** In the competitive world of startups, first impressions matter deeply. Whether it's presenting to investors, engaging with users, or networking, how your startup is perceived in the early moments can significantly impact opportunities. We've learned to focus on polished pitches, a professional online presence, and clarity in conveying our mission;

- **demonstrating traction for investors.** Investors are not just interested in great ideas—they want proof that your startup has momentum. We've learned that showing traction, whether through user growth, engagement metrics, or revenue, is essential for attracting investor interest. For us, this meant prioritizing strategies that deliver measurable results and aligning product development with key performance indicators;

- **balancing acceleration programs with product development.** Programs like pre-accelerators and accelerators have been invaluable for gaining insights, mentorship, and networking opportunities. However, we realized the importance of maintaining balance. While these programs provide immense value, dedicating enough time to actual product development is equally crucial. A strong product is ultimately what drives long-term success, so we've learned to carefully allocate time and resources to ensure progress on all fronts;

- **the critical role of market research.** Market research isn't just a phase – it's an ongoing process. Understanding our target audience, their pain points, and the competitive landscape has been fundamental to our growth. This insight has allowed us to align our product offerings with user needs and position DocDirect strategically within the market. Taking the time to validate assumptions and gather user feedback has proven to be one of the most reliable investments in our journey.

CONCLUSIONS

During the group project, the following tasks were completed:

- analysis of the current state of supervision practices and existing solutions in the mental health sector;
- comprehensive evaluation of the platform idea, including its strengths and weaknesses;
- design and development of the DocDirect platform to meet modern standards of functionality and security;
- analysis of opportunities for integrating features such as supervisor verification, session management, and transparent financial operations;
- creation of the company's brand and initial steps in social media promotion;
- participation in various startup projects, which provided valuable experience and a strategic direction for future development.

A significant amount of work was accomplished during the developing of project, contributing to both the development of the platform and the professional growth of the team. The project provided valuable experience in teamwork, including effective communication, task delegation, and collaboration. It also enhanced skills in documentation management, ensuring clarity and structure in presenting ideas and progress.

The team gained practical experience in designing and implementing solutions to a wide range of problems, from technical challenges to market positioning. Numerous aspects of the platform's concept were analyzed, including its strengths, weaknesses, and opportunities for improvement, which helped refine the overall approach.

Through branding efforts, participation in startup initiatives, and problem-solving tasks, the project allowed the team to develop a strategic vision and actionable steps for the platform's future. The insights and skills gained during the project lay a strong foundation for further professional development of the DocDirect.

LIST OF REFERENCE SOURCES

1. DocDirect. URL: <https://doc.direct/> (access date: 24.04.2025).
2. Notion. URL: <https://www.notion.com/> (access date: 24.04.2025).
3. DocDirect Notion site. URL: <https://doc-direct.notion.site/12e57d8cb4218185a42af3871d12bd2c?v=12e57d8cb421812e9f0000c9cb5ff2b> (access date: 24.04.2025).
4. Lean Canvas for DocDirect. URL: <https://doc-direct.notion.site/DocDirect-Lean-Canvas-11957d8cb4218138bfa5d9ef9d7b073f?pvs=4> (access date: 24.04.2025).
5. Superthread. URL: <https://superthread.com/> (access date: 24.04.2025).
6. Vue.js. URL: <https://vuejs.org/> (access date: 24.04.2025).
7. TypeScript. URL: <https://www.typescriptlang.org/> (access date: 24.04.2025).
8. Go. URL: <https://go.dev/> (access date: 24.04.2025).
9. DigitalOcean. URL: <https://www.digitalocean.com/> (access date: 24.04.2025).
10. NGINX. URL: <https://nginx.org/> (access date: 24.04.2025).
11. Kubernetes. platform URL: <https://kubernetes.io/> (access date: 24.04.2025).
12. Mailjet. URL: <https://www.mailjet.com/> (access date: 24.04.2025).
13. RozetkaPay. URL: <https://rozetkapay.com/> (access date: 24.04.2025).
14. Brevo. URL: <https://www.brevo.com/> (access date: 24.04.2025).
15. PostgreSQL. URL: <https://www.postgresql.org/> (access date: 24.04.2025).

16. DigitalOcean Spaces. URL: <https://www.digitalocean.com/products/spaces> (access date: 24.04.2025).
17. Pinia. URL: <https://pinia.vuejs.org/> (access date: 24.04.2025).
18. Google Analytics. URL: <https://developers.google.com/analytics> (access date: 24.04.2025).
19. Про захист інформації в інформаційно-комунікаційних системах : Закон України dated 05.07.1994, No. 80/94-ВР : as of 20.04.2025.
20. Про систему охорони психічного здоров'я в Україні : Закон України dated 15.01.2025, No. 4223-ІХ.

Наукове видання

ПРОГРАМУВАННЯ ТЕОРІЯ ТА ПРАКТИКА

ЗБІРНИК МАТЕРІАЛІВ
ЗА РЕЗУЛЬТАТАМИ ІТ-ПРОЄКТУ
МІЖДИСЦИПЛІНАРНОЇ ІНТЕГРАЦІЇ

2024-2025 навчальний рік

Друкується за авторською редакцією

Оригінал-макет виготовлено ВПЦ "Київський університет"



Формат 60x84^{1/8}. Обл.-вид. арк. 16,64. Ум. друк. арк. 26,16. Наклад 100. Зам. № 225-11407.
Гарнітура Times New Roman. Папір офсетний. Друк офсетний. Вид. № К8.
Підписано до друку 24.06.25

Видавець і виготовлювач
ВПЦ "Київський університет"

Б-р Тараса Шевченка, 14, м. Київ, 01601, Україна

☎ (38044) 239 32 22; (38044) 239 31 58; (38044) 239 31 28

e-mail: vpc@knu.ua; vpc_div.chief@univ.net.ua; redaktor@univ.net.ua

[http: vpc.knu.ua](http://vpc.knu.ua)

Свідоцтво суб'єкта видавничої справи ДК № 1103 від 31.10.02