ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ
КИЇВСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ ТАРАСА ШЕВЧЕНКА
КАФЕДРА ТЕОРІЇ ТА ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

"ПРОГРАМУВАННЯ: ТЕОРІЯ ТА ПРАКТИКА"

ЗБІРНИК МАТЕРІАЛІВ
ЗА РЕЗУЛЬТАТАМИ ІТ-ПРОЄКТУ
МІЖДИСЦИПЛІНАРНОЇ ІНТЕГРАЦІЇ

2022-2023 навчальний рік

КИЇВ, 2023

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

"PROGRAMMING: THEORY AND PRACTICE"

MATERIALS OF THE INTERDISCIPLINARY INTEGRATION IT PROJECT RESULTS

2022-2023 academic year

KYIV, 2023

УДК 004.9
П78

Рекомендовано до друку вченою радою факультету комп’ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка (Протокол № 12 від 14 червня 2023 року).

Рецензенти:
Марченко О.О., доктор фізико-математичних наук, професор кафедри математичної інформатики
Шкільняк О.С., кандидат фізико-математичних наук, доцент кафедри інтелектуальних програмних систем

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, асистент кафедри теорії та технології програмування

Програмування: теорія та практика. Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції / За редакцією Л.Л. Омельчук, О.М. Ткаченка, О.В. Шишацької. – Одеса: "Гельветика", 2023. – 213 с.

Збірник містить звіти про студентські роботи, виконані в рамках міждисциплінарного проєкту-експерименту інтеграції на прикладі теоретично орієнтованих курсів "Методи специфікації програм", "Коректність програм та логіки програмування" і практично орієнтованого курсу "Інструментальні середовища та технології програмування" (спеціальність 122 "Комп’ютерні науки").

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

ISBN

УДК 004.9
П78

Recommended for printing by the Academic Council of the Faculty of Computer Science and Cybernetics at Taras Shevchenko National University of Kyiv (Protocol No. 12, June 14, 2023).

Reviewers:
O. Marchenko, Doctor in Computer Science, Professor at the Department of Mathematical Informatics.
O. Shkilniak, Ph.D. in Computer Science, Associate Professor at the Department of Intelligent Software Systems.

General Editing:
L. Omelchuk, Ph.D. Computer Science, Associate Professor at the Department of Theory and Technology of Programming.
O. Tkachenko, Ph.D. in Computer Science, Associate Professor at the Department of Theory and Technology of Programming.
O. Shyshatska, Ph.D. in Physical and Mathematical Sciences, Assistant at the Department of Theory and Technology of Programming.

Programming: Theory and Practice. Materials of the interdisciplinary integration IT project results. Edited by L.Omelchuk, O.Tkachenko, O.Shyshatska. – Odesa: "Helvetyka", 2023. – 213 p.

The collection includes reports on student works carried out within the interdisciplinary integration project-experiment based on theoretically oriented courses "Program Specification Methods," "Program Correctness and Programming Logic," and practically oriented course "Development Tools and Programming Technologies" (Specialty 122 "Computer Science").

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

ЗМІСТ

CONTENTS

ІНТЕГРАЦІЯ НАВЧАЛЬНИХ КУРСІВ НА ОСНОВІ ПРОЄКТНОГО ПІДХОДУ ТА ГНУЧКИХ МЕТОДОЛОГІЙ УПРАВЛІННЯ У 2022-2023 У НАВЧАЛЬНОМУ РОЦІ

Людмила Омельчук, Олексій Ткаченко, Олена Шишацька

Дана робота висвітлює процес реалізації третього року виконання проєкту інтеграції навчальних курсів на основі гнучких методологій управління та проєктного підходу в межах освітньо-професійної програми "Інформатика" бакалаврського рівня вищої освіти, що реалізується на факультеті комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка за спеціальністю 122 "Комп'ютерні науки". В рамках спільного проєктного завдання для студентів другого та четвертого років навчання поєднано практичні частини наступних дисциплін: "Методи специфікації програм", "Коректність програм та логіки програмування" та "Інструментальні середовища та технології програмування".

Результати першого та другого років (2020-2021, 2021-2022 навчальні роки) реалізації проєкту оприлюднені в [1, 2]. Результати попередніх двох років проведення проєкту вплинули на внесення змін до освітньо-професійної програми "Інформатика" вцілому та на зміст задіяних освітніх компонент зокрема.

АКТУАЛЬНІСТЬ

Відповідо до дослідження [3], 77% студентів чи випускників університетів станом на 2023 рік, які працюють чи планують працювати в ІТ, рекомендували би абітурієнтам українські університети, 17% радили би їхати вчитися за кордон (що менше, ніж у 2019 році). Студенти вважають недостатнім рівень формування уміння працювати в команді: лише 37% опитаних задоволені рівнем формування цієї компетентності. При цьому тільки 30% респондентів вважають, що університет сприяв набуттю практичних навичок та співпраці з ІТ-компаніями. В умовах складної безпекової ситуації в Україні означені проблеми мають тенденцію до поглиблення.

Інтеграція навчальних курсів з використанням проєктного підходу та гнучких методологій управління покликала сприяти усуненню зазначених проблем. Розробка, впровадження та вдосконалення методики проведення інтегрованих курсів (далі – проєкт) здійснюється в межах освітньо-професійної програми "Інформатика" першого рівня вищої освіти, що реалізується на факультеті комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка за спеціальністю 122 "Комп'ютерні науки". В рамках спільного проєктного завдання для студентів другого та четвертого років навчання було поєднано практичні частини наступних дисциплін:

- "Методи специфікації програм" (вибіркова дисципліна, 4 рік навчання, викладач: к.ф.-м.н. Шишацька О.В., задіяно 49 студентів);
- "Коректність програм та логіки програмування" (вибіркова дисципліна, 4 рік навчання, викладач: к.т.н. Ткаченко О.М., задіяно 49 студентів);
- "Інструментальні середовища та технології програмування (обов'язкова дисципліна, 2 рік навчання, викладач: к.ф.-м.н. Омельчук Л.Л., задіяно 11 студентів).

Цілі проєкту:

(1) посилення практичної складової навчальних зазначених дисциплін;

(2) підвищення у студентів рівня розуміння:

- усіх етапів життєвого циклу програмного забезпечення;
- підходів до управління ІТ-проєктами;
- міждисциплінарних зв'язків;
- зв'язків між теоретичним і практичним матеріалом;

(3) підвищення рівня професійних та соціальних навичок;

(4) врахування результатів проєкту при перегляді та оновленні освітньої програми та її компонентів.

Для досягнення цілей проєкту поставлено такі задачі:

(1) інтеграція на прикладі теоретично орієнтованих курсів "Методи специфікації програм", "Коректність програм та логіки програмування" та практично орієнтованої курсу "Інструментальні середовища та технології програмування";

(2) проведення проєкту, в рамках якого для виконання завдань формуються команди за різними критеріями: володіння технологіями розробки ПЗ, методологіями управління ІТ-проєктом, складом (віковий, соціально-спрямованими вподобаннями, ін.);

(3) розробка критеріїв оцінювання успішності запропонованого підходу до інтеграції дисциплін;

(4) аналіз результатів успішності проєкту, формування пропозицій щодо його вдосконалення та впровадження як кейсу на постійній основі;

(5) розробка програмного продукту підтримки інтеграції навчальних дисциплін.

Цільовою аудиторією проєкту є:

- студенти та викладачі бакалаврської освітньої програми "Інформатика", які задіяні в рамках дисциплін "Методи специфікації програм", "Коректність програм та логіки програмування" та "Інструментальні середовища та технології програмування" (пряма аудиторія);
- ІТ-компанії, студенти і викладачі інших освітніх програм за ІТ-спеціальностями (опосередкована аудиторія);
- представники інших освітніх програм (як потенційні замовники).

Третій рік реалізації проєкту було організовано у другому семестрі 2022-2023 навчального року. Було визначено наступні показники досягнення цілей:

(1) підвищення рівня професійних та соціальних навичок у студентів, задіяних в проєкті-експерименті (інструмент перевірки – вихідне опитування);

(2) наявність програмних систем, розроблених студентськими командами;

(3) наявність звітів за результатами роботи кожної студентської команди;

(4) наявність звіту викладачів про результати проєкту;

(5) оприлюднення результатів проєкту.

ПРОВЕДЕННЯ ПРОЄКТУ

Для студентів четвертого року навчання, які вивчали дисципліни "Методи специфікації програм", "Коректність програм та логіки програмування" участь в проєкті була обов'язковою. Студентам другого курсу було запропоновано на добровільних засадах долучитися до проєкту. При цьому було враховано результати навчання (підсумковий бал більше 80) з обов'язкової дисципліни "Об'єктно-орієнтоване програмування", яку ці студенти вивчали в попередньому семестрі.

На початку семестру студентам четвертого року навчання було запропоновано пройти анкетування, яке включало такі питання:

- **чи маєте Ви досвід роботи в реальних колективних проєктах (не навчальних)** (*був досвід, не маю досвіду, зараз в проєкті);*
- **запропонуйте декілька варіантів предметних областей або конкретних систем Вашого майбутнього проєкту;**
- **вкажіть не більше, ніж 3 людини, з якими Ви б хотіли працювати в команді;**
- **на Вашу думку, в якій ролі в командній роботі Ви би** *(почувалися впевнено, НЕ хотіли себе бачити, хотіли "прокачати" себе в проєкті / керівник проєкту, модератор, фахівець з документації, аналітик, фронтенд, бекенд, тестувальник);*

• **які компетентності Ви би хотіли розвинути в рамках майбутнього проєкту** (поглибити теоретичні знання, розвинути практичні вміння, програмувати, оформляти документацію, працювати в команді, планувати і організовувати свій час, вміння презентувати результати діяльності, комунікаційні навички, інше);

• **досвід роботи з якою платформою і/або методологією управління ІТ-проєктами Ви маєте**;

• **в якій системі управління ІТ-проєктами (методології) Ви б хотіли працювати в проєкті**;

• **на Вашу думку, використання якої мови для Вас є** *(близькою, Ви нею володієте на впевненому рівні, не бажаною, Ви не впевнені, що володієте нею на хорошому рівні, бажаною в проєкті, бо Ви хочете "прокачати" себе в ній / C#, C++, Java, PHP, Python, інше)*;

• **ваші пропозиції щодо організації проєкту.**

На рис. 1-3 наведено деякі результати вхідного анкетування.

Рисунок 1. Результати анкетування щодо досвіду участі в колективних проєктах

Рисунок 2. Результати анкетування щодо бажаних компетентностей

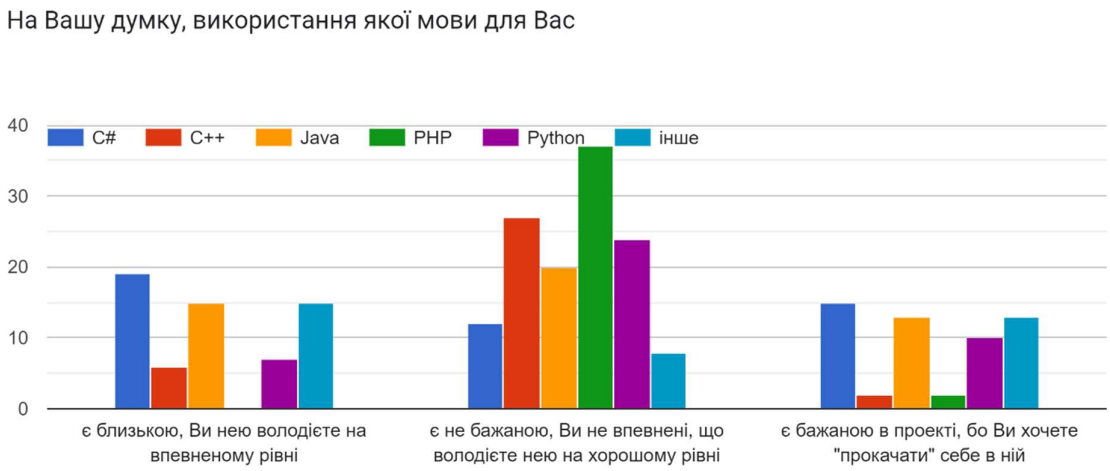


Рисунок 3. Результати анкетування щодо рівня володіння мовами програмування

За результатами анкетування для формування команд було побудовано соціометричний орієнтований граф, який враховував досвід роботи в командних проєктах, ступінь володіння технологіями, побажання щодо вибору колег по команді, активність на початковому етапі.

Для формування команд застосовувалися різні підходи:

- за роком навчання: команди, що складалися виключно зі студентів четвертого року навчання та команди, що складалися зі студентів четвертого та другого років навчання;
- за соціальними вподобаннями (з урахуванням вибору бажаних колег по проєкту): команди з тісними соціальними зв'язками (взаємний вибір);
- за досвідом участі в реальних колективних проєктах: усі учасники мали досвід, ніхто з учасників не мав досвіду, змішані команда;
- за технологіями: команди зі спільними побажаннями щодо технологій розробки та команди з різними вподобаннями.

В 2020-2021 н.р. зі здобувачів, що не заповнили анкети вхідного опитування було сформовано окрему команду і, як наслідок, це виявилася єдина команда, яка не змогла представити готовий проєкт до публічного захисту. В 2021-2022 н.р. студентів, що не виявили ентузіазму на початковому етапі і не заповнили анкети вхідного опитування, було розподілено в різні команди, в які не входили здобувачі другого року навчання. При цьому враховувалася місце зазначеної категорії учасників в побудованому соціометричному графі. Одним з позитивних результатів другого року реалізації проєкту було те, що така інтеграція слабомотивованих студентів до складу мотивованих команд сприяла тому, що усі студенти взяли активну участь у роботі своїх команд та успішно захистили реалізовані проєкти. У 2022-2023 н.р. при формуванні команд всі учасники були вмотивовані, але на початку виконання проєкту учасник однієї з команд (студент 2-го року навчання) виявив бажання повернутися з проєкту до стандартного формату навчання.

Було сформовано вісім команд, всі з яких успішно виконали проєкт. До складу п'яти команд входили студенти 2-го року навчання.

Особливістю третього року проведення проєкту був вільний вибір предметної області розробки. Для деяких команд узгодження теми виявилося непростою задачею зі сфери soft skills, з якою вони успішно справилися.

В рамках проєкту здійснювалися щотижневі зустрічі учасників студентських команд з викладачами та організовано фінальний публічний захист студентських розробок. На захисті були присутні представники кафедри теорії та технології програмуваня, інших кафедр факультету комп'ютерних наук та кібернетики, гаранти освітніх програм різних галузей, а також предстакники IT-компаній SoftServe, Infosoft та Revenue Grid. За результатами проезентації проєктів всі присутні таємним голосуванням визначили кращий проєкт 2022-2023 н.р., а також визначили переможців у таких номінаціях: краща ідея, краща реалізація, краща презентація проєкту, краща командна робота, за актуальність предметної області, найбільш інноваційний проєкт, найперспективніший проєкт, кращий дизайн.

Звіти про виконання колективних проєктів, розроблених в межах проєкту, подані в даному збірнику. З усіма командами викладачі проводили окремі зустрічі і не розголошували їх резульлати іншим учасниками проєкту.

У кожній задіяній дисципліні було передбачено оцінку за проєкт як частину семестрової оцінки, що прописано в робочих програмах. Для кожної дисципліни було встановлено свої вагові коефіцієнти за критеріями:

- формулювання вимог та формальна специфікація;
- програмна реалізація;
- тестування та верифікація;
- звіт (підготовка документу і презентація результатів);
- ведення проєкту;
- захист;
- взаємне оцінювання учасників команд.

По завершенню проєкту було проведено два вихідних опитування: анонімне та авторизоване. Було отримано ряд відгуків, зокрема (дослівно):

- *Після університету нам всім доведеться шукати роботу, цей проект - це дійсно гарний спосіб вивчити саме ті технології, які потрібні при працевлаштуванні. Протягом цього проекту, я здобув навичок в фронтенд розробці, і саме тих навичок, які знадобляться при пошуку роботи. Але я вважаю, що цього мало, і такий проект не зможе достатньо сильно "прокачати" усі навички, щоб після університету легко працевлаштуватися, все одно доведеться витрачати час на самонавчання. Я вважаю, що такі проекти повинні бути якщо не кожного семестру, то кожного року. Бо після участи в 4х таких проєктах, маючи такий досвід, працевлаштуватись буде набагато простіше. Я вважаю, що один такий проект на весь термін навчання - це дуже мало.*
- *Насамперед, було дуже класно поспілкуватися з одногрупниками майже вживу, для більшості - уперше з першого очного семестру... Завдяки цьому останній семестр видався пам'ятним. Хотілося б мати схожі проєкти й із інших предметів раніше, нехай і менші за обсягом. Проєкт сподобався й був корисним для розвитку як софт так і хард скілів. Протягом семестру було складно поєднувати роботу, проєкт та відвідування пар й виконання завдань із інших дисциплін. Утім, врешті саме здобуті софт скіли та командна робота дозволила впоратись із більшістю задач :)*
- *Система оцінювання за номінаціями так не працює і не може працювати. Це дивно, що команда може мати, наприклад, або найкращий дизайн, або найкращу ідею. Очевидно, що були команди, які заслуговували декілька номінацій, а були - що 0. Безвідносно до конкретної команди (просто знаю, що така проблема в деяких командах існувала), не завжди другий курс має в команді однакові права, що й четвертий, але, в силу своєї недосвідченості, не може з цим нічого зробити. Також не завжди четвертий курс зацікавлений в наданні достатнього фідбеку.*
- *Проект сподобався, шкода, що таких спільних проектів не було до цього на попередніх курсах, робота в команді дуже корисна і прокачує не тільки софт скіли, а й*

хард скіли, оскільки від тебе залежить результат твоїх колег по команді, що мотивує тебе виконувати свою роботу добре. Тому було б класно впроваджувати таке на усіх курсах.

- *Проєкт був цікавим, тут ми навчились працювати злагоджено, як команда. Дехто потренувався, а, можливо, навіть навчився чогось нового в хард скілах. Загалом цей проект дав одні плюси, від ближчого знайомства з людьми із команди до відточення професійних навичок.*
- *Формат групового проєкту дуже сподобався мені та допоміг отримати досвід командної розробки, якого мені бракувало. Робота в команді допомогла мені швидше опановувати нові для себе технології та цікаві рішення.*
- *По зауваженням - це не зовсім зрозумілі вимоги до презентації та незрозумілі критерії оцінювання, тобто кожна команда могла голосувати за себе (?) і не можна було одній команді виставити декілька балів.*
- *Хотілось б зробити мобільну версію додатку і показати їй на презентації, але на це було мало часу, тому зупинились лише на веб-застосунку. Може, це і на краще :)*
- *Цього разу, по відчуттям, захист був як Грандіозна подія. Дуже чудово проведений захист. Відсутність пропозицій щодо проекту зі сторони викладачів, що дозволяє стимулювати ідеї.*
- *Це був чудовий досвід роботи в команді, який дозволив спробувати різні ролі та відчути свій внесок при досягненні гарного результату у вигляду спільного проекту.*
- *Дуже вдячна за таку можливість взяти участь в командному проекті. Для мене, як для студентки 2-го курсу, цей перший досвід був надзвичайно цінним.*
- *Складно поєднувати людей з різним досвідом та різними технологіями в одній команді, а при великих навантаженнях це відчувається ще більш яскраво.*
- *Загалом, я був надзвичайно задоволений завершеним проектом і вважаю, що він матиме значний позитивний вплив на учасників нашої команди.*
- *Навчилась роботі в команді, деплою, роботі в gitlab, azure, trello покращила свої навички розробки та знання git.*
- *Дуже цікава ідея, тому б радив робити на 2-му курсі теж обов'язково, але за вибором, чи хочуть з 4-курсниками.*
- *Було цікаво попрацювати в команді, з побажань лище трохи більше часу, тижні 2 принаймні.*
- *Це був дуже корисний досвід роботи в команді. Дуже дякую за таку можливість!*
- *Все було чудово, але у деяких людей було небагато мотивації для роботи.*
- *Дуже класний проєкт і хороша практика, яка буде корисна в майбутньому.*
- *Мало часу та другокурсників та не ідеальний баланс стеків.*
- *Було трохи важко, але воно того варте :)*
- *Класна штука всім рекомендую.*
- *Наші задачі виконані успішно.*

По завершенню проєкту учасники надали ряд пропозицій (дослівно):

- *Якби протягом усіх 4-х років навчання в нас було меньше звичайних контрольних та лабораторних робіт, але були такі проєкти, було би набагато цікавіше вчитись і ми б вивчали популярні технології, які знадобляться у майбутньому при працевлаштуванні… Хоча це призведе до своїх недоліків, таких як: студенти, які нічого не роблять на командних проектах, але отримують стільки ж балів, скільки інші. Треба більш детально досліджувати діяльність кожної конкретної людини, бо будуть такі, хто за весь семестр зробив одну таску і все.*
- *Як на мене, щотижневі демо були недостатньо ефективними через неможливість приділити достатньої уваги всім групам. Можливо, замість проведення*

демо перед усіма викладачами, слід призначити викладача-куратора для декількох команд, який трохи більше занурився б у їх роботу (але не занадто). При цьому, не слід відмовлятися й від звичних спільних демо, можливо, в такому форматі зробивши їх раз на два тижні, а зустріч із куратором - щотижня.

- На мою думку, може бути цікавим та корисним, якщо задачу буде ставити окрема група студентів (інший курс, інший факультет, команди одна одній) або викладачі, імітуючи роботу із замовником. Це може допомогти з розвитком софт-скілів, бо у межах команди налагодити спілкування при наявності бажання в учасників не є складною задачею.

- Змінити опитування [визначення переможців – Авт.], яке було в кінці. Так, я розумію що не хочеться образити жодну команду, але з 8 команд можна було чітко визначити 3-4 команди по певним критеріям. А всі інші команди - суто рандом, бо не вразило взагалі, якщо чесно. Але і назвати ті проекти найперспективнішими і т.д. назвати не можна.

- Довший термін виконання проєктів, команди більш збалансовані за стеками та з більшою кількістю другокурсників, щоб 4-курсник був ментором мінімум одного 2-курсника, таким чином, "прокачував" свої навички в сторону Lead позиції.

- Нашій команді було складно розпочати роботу, і вже в кінці ми зрозуміли, що зробили деякі суттєві помилки на цьому етапі. Можливо, слід краще пояснити студентам, як правильно і ефективно розпочати командний проект.

- Краще проводити даний проєкт в першому семестрі, щоб було більше часу на реалізацію функціоналу, так як через велике навантаження в цьому семестрі не приділяєш стільки часу, скільки б хотілось.

- Відсутність достатньої кількості фронтенд-розробників може стати проблемою в такого виду проектах. Було б гарно мати можливість компонувати сбалансовані команди на основі вже набутих навичок.

- Було б класно працювати в такому форматі, у студентів дійсно є можливість удосконалювати та набувати певні знання та навички, які необхідні кожному впродовж життя та кар'єрного шляху.

- Можливо, непогано було б один раз показати усім, що хто робить. Щоб ті, хто побачив, що вони відстають від інших, доклали більше зусиль для реалізації проекту.

- Подовжити час (зачепити частину першого семестру), зробити загальні демо посеред терміну (щоб люди бачили рівень інших команд, це може трохи змотивувати).

- Покращити інструкції для презентації та, можливо, додати лекційний матеріал, спрямований саме на проект, щоб комунікація велась не лише раз у тиждень.

- Можливо, якісніше підбирати теми проектів, бо деякі були супер примітивні та нічим не відрізнялись від вже існуючих.

- Усе пройшло чудово, хіба лише 8 тижнів - це мало, щоб створити крутий проект, який справді зацікавить роботодавців.

- Хотілося б, щоб в подібних проектах брало участь більше другокурсників, бо 10 людей з усього потоку - це мало.

- Хотілося б мати список ролей із їх обовязками запропонованих зі сторони викладачів.

- Більш детальний контроль за якістю роботи в командах (особливо, коли є другий курс).

- Побільше спілкуватися з викладачами та командою.

- Краще балансувати команду (бекенд/фроненд)

- Перенести його в інший семестр :)

- Всі цікаві пропозиції вже були озвучені під час захисту. Вважаю, що організація була крута, а наступний раз буде ще крутіше!

Незважаючи на воєнний стан через зовнішню аграсію та окупацію частини території України, що спричинило масові переміщення громадян України, в тому числі студентів і викладачів, варто відзначити кращу мотивацію студентів, у порівнянні з попередніми роками.

ВИСНОВКИ

В результаті проведення проєкту:

(1) розроблено програмні продукти згідно з обраними предметними областями;

(2) підвищено рівень професійних та соціальних навичок у студентів, задіяних в проєкті;

(3) підвищено рівень розуміння всіх етапів життєвого циклу програмного забезпечення, підходів до управління ІТ-проєктами; міждисциплінарних зв'язків у межах освітньої програми; зв'язків між теоретичним та практичним матеріалом;

(4) посилено практичну складову навчальних дисциплін, задіяних в експерименті;

Подальшими кроками будуть:

(1) опис методики інтеграції навчальних курсів в галузі інформаційних технологій на основі проєктного підходу та гнучких методологій управління;

(2) вироблення рекомендацій щодо вдосконалення та впровадження методики інтеграції навчальних курсів в галузі інформаційних технологій на основі проєктного підходу та гнучких методологій управління;

(3) поширення досвіду проєкту на інші навчальні дисципліни;

(4) посилення практичної складової навчальних курсів;

(5) підвищення якості освітньої програми, зокрема, в контексті формування професійних та соціальних навичок у випускників ІТ-спеціальностей;

(6) залучення до проєкту представників освітніх програм з інших галузей як експертів прездметних областей;

(7) розширення співпраці з представниками ІТ-компаній в рамках проєкту.

ВИКОРИСТАНІ ДЖЕРЕЛА:

1. Омельчук Л.Л., Ткаченко О.М., Шишацька О.В. Впровадження методики інтеграції навчальних курсів на основі проєктного підходу та гнучких методологій управління / Програмування: теорія та практика. Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції. 2020-2021 навчальний рік. Рекомендовано до друку вченою радою факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка. - Одеса: Видавничий дім "Гельветика", 2021. - С.4-10.

2. Омельчук Л.Л., Ткаченко О.М., Шишацька О.В. Інтеграція навчальних курсів на основі проєктного підходу та гнучких методологій управління / Програмування: теорія та практика. Збірник матеріалів за результатами ІТ-проєкту міждисциплінарної інтеграції. 2021-2022 навчальний рік. Рекомендовано до друку вченою радою факультету комп'ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка. - Одеса: Видавничий дім "Гельветика", 2022. - С.4-11

3. Рейтинг ІТ-вишів 2023: УКУ — беззаперечний лідер, КПІ — в десятці, наприкінці кілька харківських вишів. [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/ukrainian-universities-2023/

СИСТЕМА ФОТОФІКСАЦІЇ ПОРУШЕНЬ ПДД "ФІКСУЙ"

Дмитро Поліщук, Любомир Маєвський, Максим Закорко, Діна Формакідов, Михайло Головацький, Поліна Голоцван, Яна Свіргуненко, Ольга Вішнєвська

ВСТУП

Постановка задачі. Задача полягає у розробці мобільного додатку для системи фотофіксації правил дорожнього руху з назвою "Фіксуй". Основна мета додатку - забезпечити безпеку на дорогах, запобігаючи порушення правил дорожнього руху.

Після фотофіксації порушення, додаток має автоматично відправити зображення на сервер системи фотофіксації, де проводиться аналіз та обробка даних. Користувачі мають доступ до своєї історії фотофіксацій, де можуть переглянути свої порушення правил дорожнього руху та дати коментар щодо кожного з них.

Актуальність роботи. Система фотофіксації ПДР є актуальною в сучасному світі, оскільки вона може допомогти забезпечити безпеку на дорогах та зменшити кількість дорожньо-транспортних пригод. Система фотофіксації може допомогти у швидкому виявленні порушень правил дорожнього руху та вживанні необхідних заходів щодо усунення таких порушень. Система "Фіксуй" може використовувати місцезнаходження, щоб визначити місце паркування, і можливість сфотографувати автомобіль, який порушує правила. Потім власник автомобіля отримує повідомлення з фотографією і інформацією про порушення та може бути оштрафований за нього.

Багато країн по всьому світу вже використовують системи фотофіксації порушень правил паркування, щоб зменшити кількість порушень та збільшити безпеку на дорозі.

Однак, необхідно забезпечувати права та інтереси громадян, які можуть бути зображені на фотографіях, отриманих з таких систем. Тому необхідно забезпечувати захист особистих даних та приватного життя громадян.

Обґрунтування необхідності системи "Фіксуй". Залучення громадян та активної частини громадянського суспільства до контролю у сфері безпеки дорожнього руху на прозорих правових засадах може бути ефективним кроком, який призведе до різкого зменшення кількості ДТП на дорогах України.

Якщо кожен свідомий громадянин, водій або пішохід зможе забезпечувати контроль за безпекою на дорозі навіть за відсутності поліцейського, іншого представника держави чи встановленого комплексу автофіксації, це унеможливить свавільне порушення ПДР, а у випадках допущення таких дій призведе до негайного покарання винного.

Саме до цього режиму фіксації порушень можливо залучити активних громадян, які зі свого боку забезпечили б "громадський контроль" за безпекою на тих ділянках доріг, що не контролюються ані посадовими особами безпосередньо, ані комплексами автоматичної фіксації.

ОГЛЯД НАЯВНИХ СИСТЕМ

DashcamUA. Мета цього проєкту – докорінне зменшення кількості аварій та смертності на дорогах, збільшення прозорості роботи Національної поліції України.

Новий законопроєкт затвердить право поліції використовувати відео, зняті через мобільний застосунок, для спрощеної процедури виписування штрафів. У подібний спосіб поліція наразі опрацьовує дані з камер автоматичної фіксації порушень швидкісного режиму.

Смартфон із активним мобільним застосунком постійно записує 30-секундні відео в циклічному режимі. Коли минає 30 секунд, попереднє відео стирається, і починає записуватися нове. Коли водій бачить на дорозі порушення, йому потрібно лише торкнутися екрана, щоб останні 30 сек відео збереглися в застосунку. Пізніше користувач зможе швидко надіслати відео до поліції, вказавши номер автівки порушника та тип порушення.

Ідея ініціативи dashcamUa народилася серед українських випускників західних вузів у 2015 році. У 2016 році була офіційно зареєстрована громадська організація ДешкемЮЄй. Основна мета – це значне зменшення кількості жертв на дорогах.

Тоді ж у 2016 була перша спроба запустити мобільний додаток із мінімальним функціоналом. На жаль, тоді мобільна передача даних була на рівні технології EDGE, а перспективи запуску мереж 3G були ще під питанням. Також камери мобільних телефонів були низькому рівні, а процесори телефонів не забезпечували виконання багатьох завдань. Користувачі мобільних телефонів не довіряли телефонам як зараз, коли захищені програми мобільних банків довели, що телефону можна довіряти не тільки персональну інформацію, але й власні фінанси.

Тільки до 2019 року стало зрозумілим, що роботу можна відновлювати. Рівень безпеки даних значно покращився. Сучасні мобільні телефони розвивалися і розвиваються великою мірою саме камерами та потужнішими процесорами. Це дозволило оновити концепт мобільного додатка та підготувати його прототип. Після запуску прототипу розробники побачили, що спільнота ще більше підтримує ідею ініціативи dashcamUa. Сумна статистика смертей на дорогах показувала необхідність якнайшвидших дій.

На початку 2020 року був підготовлений MVP (мінімальна робоча версія) та влітку здійснили запуск тестової версії на Android платформі. Очікування були набрати хоча б одну тисячу користувачів, щоб протестувати програму на різних моделях телефонів. Втім розробники були дуже здивовані, коли побачили, що буквально за кілька тижнів додаток встановили більше 15 тис користувачів.

Щоб гарантувати анонімність користувачів, всі отримані на сервер відео в поліцію пересилаються від імені ГО ДешкемЮЕй. Оскільки розробники впевнені у захищеності програми (неможливо підробити відео, дані або підставити іншу інформацію), вони беруть на себе відповідальність за подання заяв до регіональних відділень поліції відповідно до географічного місця фіксації порушення.

За їх поданнями до поліції є результати. Хоча більшість регіональних відльотів пишуть відписки, що вони не можуть встановити хто був за кермом авто, є все-таки добросовісні поліцейські, які проводять встановлені законом процедури і встановлюють всі деталі досконалого порушення. Є велика кількість результативних заяв за якими порушники були залучені за найбільш небезпечні порушення.

Наразі у Верховній раді на розгляді знаходиться законопроект #5798. Він уже отримав позитивне схвалення ГНЕУ (головне науково-експертне управління) та рекомендований до ухвалення парламентським комітетом з правоохоронної діяльності. Очікуваний час розгляду законопроекту у Парламенті - зима 2022

Окрім України у проекті зацікавлено ради інших країн: Об'єднані Арабські Емірати, Катар, Саудівська Аравія, Узбекистан, Грузія та інші. Отримали підтримку низки міжнародних організацій, зокрема Світового Банку, який веде окремі напрямки допомоги проектам у сфері безпеки дорожнього руху.

Переваги: бонуси, смарт-парковка, мобільний європротокол.

Смарт парковка. Розробники підготували унікальну систему, яка показуватиме вільні місця для паркінгу в центрі міста в реальному часі. Причому, як для платних, так і для безкоштовних місць.

Інтеграція з відеореєстраторами. Система розроблена таким чином, що для нормальної роботи досить просто смартфона з холдером. Але для більш високої зручності користування планується, що відеореєстратор зможе працювати як зовнішня камера смартфона.

Мобільний європротокол. Щоб машини, які потрапили в незначну аварію, не створювали пробок і не стояли на дорогах годинами, підготовлений простий і швидкий спосіб оформлення євро протоколу.

Недоліки: не популярність системи.

Анонімність. Всі подання в Поліцію робляться від імені ГО ДешкемЮЕй. Вони несуть всю відповідальність за якість кожної заяви. Дані усіх користувачі анонімні та не зберігаються на серверах

- Відповідно до законодавства України обов'язковою умовою фотофіксації ПДР є авторизація та верифікація особи [1]/

ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Frontend-частина:

Проект створено в інтегрованому середовищі розробки "AndroidStudio". Це IDE, що надає графічний інтерфейс для розробки додатків для Android. Вона включає в себе різні функції, такі як редактор коду, відладчик, емулятор та інші, які допомагають розробникам писати, тестувати і налагоджувати свої додатки.

JavaDevelopmentKit (JDK) - це комплект для розробки програмного забезпечення, який використовується для написання програм на Java. Він включає в себе середовище виконання Java (JRE) і компілятор Java, які необхідні для запуску і компіляції коду Java відповідно.

AndroidSoftwareDevelopmentKit (SDK) - це набір інструментів для розробки програмного забезпечення, який використовується для розробки додатків для платформ Android. Він включає різні бібліотеки, інструменти та API, які допомагають розробникам створювати додатки для Android.

AndroidVirtualDevice (AVD) - це емулятор, який використовується для імітації поведінки пристрою Android на комп'ютері. Він дозволяє розробникам тестувати свої додатки без використання фізичного пристрою.

GradleBuildSystem - це інструмент автоматизації збірки, який використовується для створення, тестування та розгортання додатків Android. Він керує залежностями і компілює код в APK-файл, який можна встановити на пристрій Android [2]

Android API - це набір класів та інтерфейсів, що надаються Android SDK. Він дозволяє розробникам отримати доступ до різних функцій Android-пристроїв, таких як камера, GPS, датчики тощо.

Backend-частина:

Entity Framework (EF) Core — це легка, розширювана кросплатформна версія технології доступу до даних Entity Framework із відкритим кодом. Дозволяє розробникам працювати з базою даних за допомогою об'єктів .NET.

ASP.NET Core Identity - API, яке підтримує функцію входу в інтерфейс користувача (UI). Керує користувачами, паролями, даними профілю, ролями, заявками, маркерами, підтвердженням електронної пошти тощо.

CsvHelper - бібліотека .NET для читання та запису файлів CSV.

FluentValidation - бібліотека перевірки для .NET, яка використовує плавний інтерфейс і лямбда-вирази для створення строго типізованих правил перевірки [3]

Mapster — це бібліотека, яка зіставляє один тип об'єкта з іншим типом об'єкта. Це картограф на основі конвенцій, який простий у налаштуванні та використанні. Mapster звільняє від написання шаблонного коду, схильного до помилок.

SendGrid - забезпечує протокол SMTP(Simple Mail Transfer Protocol), який дозволяє доставляти електронну пошту через зовнішні сервери замість власного клієнта чи сервера [4]

ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

Організація роботи в команді.

Планування, Методологія та система управління проєктами.

1. **Знайомство команди**

Знайомство команди відбулось на першому дзвінку в GoogleMeet, з якого і почалось обговорення і планування подальшої розробки. 09.02.2023 був проведений комплексний розподіл ролей в команді відповідно до знань студентів та враховано побажання кожного з учасників. Кожний студент має дві ролі: одну основну та одну допоміжну, яка необхідна для підтримки учасників іншої сфери у разі виникнення проблем або браку часу. Такий підхід допоможе реалізувати ефективну роботу в команді та досягнення поставлених цілей.

Учасник	**Роль учасника на початку проекту**
Поліна Голоцван	бекенд, тестувальник
Михайло Головацький	модератор, фахівець з документації
Максим Закорко	аналітик, фахівець з документації
Дмитро Поліщук	керівник проекту, фахівець з документації
Яна Свіргуненко	бекенд, тестувальник
Діна Формакідов	Тестувальник, модератор, бекенд, фронтенд,
Любомир Маєвський	модератор, бекенд, фронтенд
Ольга Вішнєвська	бекенд, аналітик

Першочерговим рішенням команди з планування роботи було визначення основного способу швидкої комунікації, а саме використання месенджера Telegram. Було створено 4 групи в телеграмі і поділено на підрозділи з відповідною кількістю учасників кожного. Наведені чати підрозділів команди:

- Основна група CLords
- Back-End команда
- Analitycs команда
- Testing команда

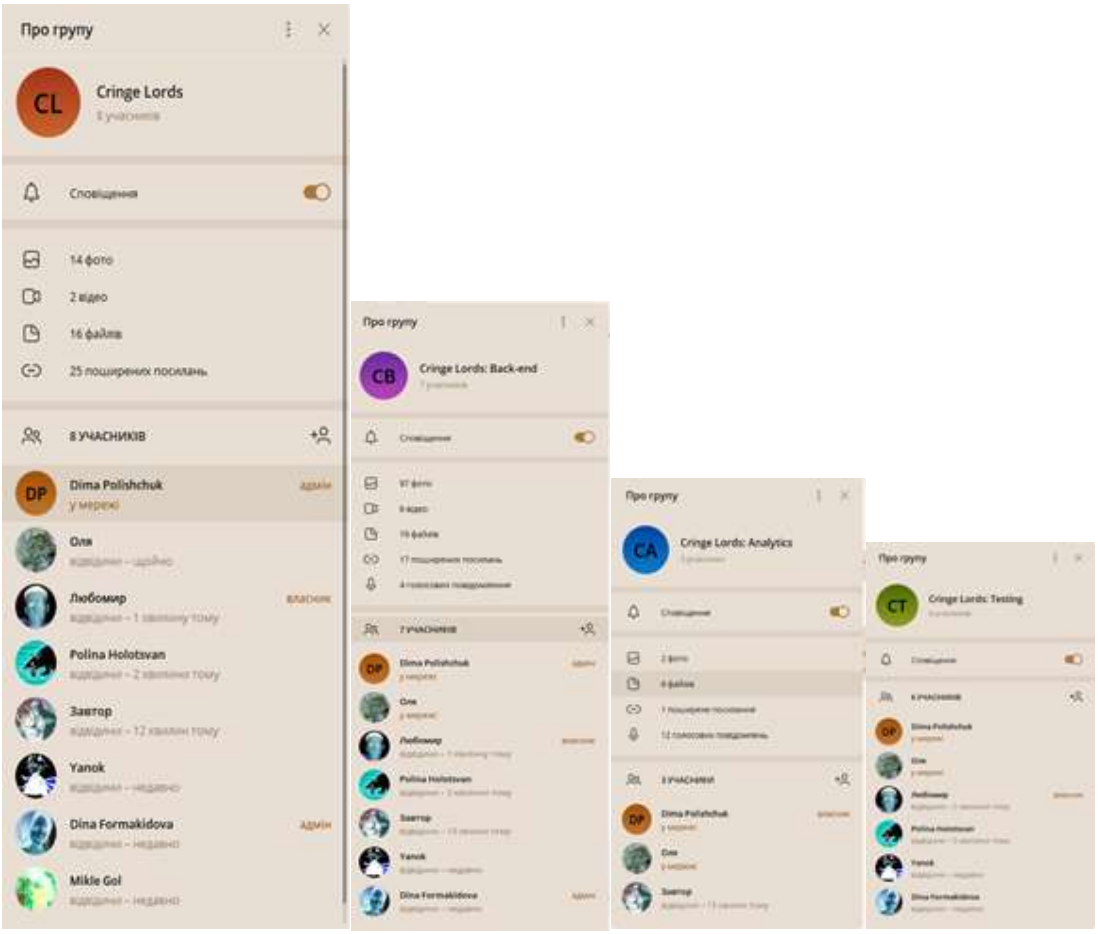


Рисунок 1. Огляд груп комунікації в месенджері Tekegram

Варто виділити основну групу телеграму, в котрій проводилось планування часу кожного дзвінка з урахуванням всіх бажань учасників. Основна мета розподілу – зручна паралельна робота кожного етапу розробки без перешкод для обговорень особливостей інших завдань кожної задачі.

2. Методологія управління розробкою

Також основними рішенням першої зустрічі було узгодження методології розробки, за основу якої ми взяли Scrum. Загальна думка полягала в формуванні ефективного керування проєктом, плануванні та контролю роботи з створенням зручних за обсягом спринтів і графіку основних мітів (meets). Було вирішено проводити 2 зустрічі в тиждень (середа, неділя). Зустрічі проводились в GoogleMeet, посилання було закріплено на всіх доступних учасникам ресурсах (групи Telegram, Classroom). Кожні задачі, якщо були питання, обговорювались в чатах Telegram або відводились на окремі зустрічі, котрі не завжди потрібно було відвідувати всім, але для залучення всіх учасників було вирішено проводити запис кожної зустрічі і завантажувати їх в групу. Основні блоки були розбиті на ("підготовчий" та "розробку")

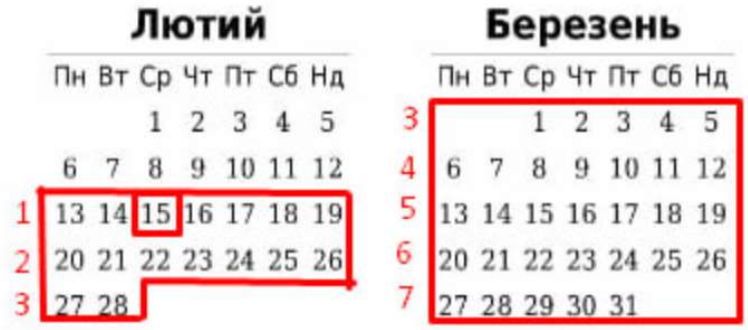


Рисунок 2. Огляд планування по спринтам

2 невеликі спринти полягли в основну "підготовчого блоку". Протяжність котрих була по 1 тижню (15.02.2023-22.02.2023, 22.02.2023-01.03.2023).

2 великі спринти полягли в основу "блоку розробки". Протяжність котрих була по 2 тижні з урахуванням досить короткого періоду роботи над проєктом, через що основні спринти мали своєрідні медіани з обговоренням і звітністю по роботі, котрі проходили в середині кожного спринта.

Звітність по спринтам:

15.02.2023 - 22.02.2023

Остаточне узгодження теми проєкту, обговорення очікувань від проєкту кожного з учасників.

Дмитро: створення дошки Trello, робота зі специфікацією, ведення Classroom, робота зі специфікацією

Любомир: робота з другокурсниками, розподіл задач, перевірка результатів роботи

Діна: перевірка результатів роботи другокурсників

Максим : пошук API, аналіз схожих систем на ринку, робота зі специфікацією, робота з законодавчою базою, бізнес - аналітика

Ольга: створення UseCase та DataBase діаграм; робота зі специфікацією, внесення правок

Поліна: створення UseCase та DataBase діаграм

Яна: створення UseCase та DataBase діаграм

Михайло: візуальна складова додатку.

22.02.2023 – 01.03.2023

Узгодження теми проєкту з викладачами, представлення специфікації та діаграм.

Дмитро: розподіл загальних задач між членами команди. Проведення мітів з командою

Любомир: робота з другокурсниками, розподіл задач, перевірка результатів роботи, внесення правок

Діна: перевірка результатів роботи другокурсників

Максим : створення логотипу, бізнес-аналітика, пошук існуючих систем

Ольга: отримання GitHubPro, встановлення GitKraken, розробка тестового проєкту ASP.NET WEB API

Поліна: отримання GitHubPro, встановлення GitKraken, розробка тестового проєкту ASP.NET WEB API

Яна: отримання GitHubPro, встановлення GitKraken, розробка тестового проєкту ASP.NET WEB API

Михайло: початок створення додатку на Java

01.03.2023 – 15.03.2023

Дмитро: розподіл загальних задач між членами команди, перехід від Trello до GitLab, створення UserExperience (UseCase), проведення додаткових зустрічей для формування форм ретроспективи за минулі спринти. Проведення мітів з командою

Любомир: робота з другокурсниками, розподіл задач, перевірка результатів роботи, внесення правок, допомога у вирішенні проблем

Діна: перевірка результатів роботи другокурсників, допомога у вирішенні проблем, початок тестування

Максим : створення логотипу, створення презентації зі звітністю, внесення правок до UserExperience (UseCase)

Ольга: створення бази даних, робота з базою даних, створення міграції в БД, внесення правок до UserExperience (UseCase)

Поліна: створення бази даних, push проєкту, робота з базою даних

Яна: створення бази даних, робота з базою даних, створення міграції в БД, зміна Express SQL Server на Development SQL Server

Михайло: початок створення додатку на Java

15.03.2023-29.03.2023

Дмитро: розподіл загальних задач між членами команди, початок роботи зі звітом. Проведення мітів з командою

Любомир: робота з другокурсниками, розподіл задач, перевірка результатів роботи, внесення правок, допомога у вирішенні проблем

Діна: перевірка результатів роботи другокурсників, допомога у вирішенні проблем, початок тестування, перехід від GitLab до AzureDevOps

Максим : створення презентації зі звітністю, початок роботи зі звітом

Ольга: Implement Asp.Net Identity (IdentityDbContext + User + Role). DataSeeding (ініціалізація в базі даних одного адміна та двох юзерів)

Поліна: Implement Controllers and Managers for Offence + OffenceLawArticle + LawArticle

Яна: Implement Controllers and Managers for Car + OffencePhoto (rename entity to Photos)

Михайло: розробка додатку на Java

29.03.2023-12.04.2023

Дмитро: розподіл загальних задач між членами команди, робота над звітом, огляд переходу між платформами. Розробка user-flow діаграм під верифікацію. Проведення мітів з командою

Любомир: робота з другокурсниками, розподіл задач, перевірка результатів роботи, внесення правок, допомога у вирішенні проблем, створення специфікації, верифікації за допомогою мови Dafny

Діна: перевірка результатів роботи другокурсників, допомога у вирішенні проблем, тестування, ведення AzureDevOps

Максим: робота зі звітом, верифікації за допомогою мови OCL, створення презентації зі звітністю

Ольга: розробка авторизації з використанням AccessToken та RefreshToken, розробка реєстрації з підтвердженням пошти. Зміна в можливості юзера змінити власну пошту (реалізація зміни пошти лише через підтвердження нової пошти)

Поліна: зміна стилю коду, завершення імплементації менеджерів, робота з валідацією, пошук даних про авто, тестування, робота над виправленнями, деплой бази даних та API в Azure

Яна: завершення імплементації менеджерів, внесення правок в базу даних, переробка Offence менеджеру, перенесення інформації з бази даних авто, тестування, робота над виправленнями, деплой бази даних та API в Azure

Михайло: завершення роботи над додатком на Java, опис використаних технологій

3. Система управління проєктами

Специфічною ознакою ведення проєкту стали часті переходи між системами управління, котрі наша команда змінювала двічі. Початок та планування були проведені в Trello, початок розробки та розробка в GitLab, розробка та кінець проєкту були проведені в AzureDevOps.

Trello

Обсяг та час проведеної роботи (2 спринта (15.02.2023 - 01.03.2023)) Дошка завдань команди:

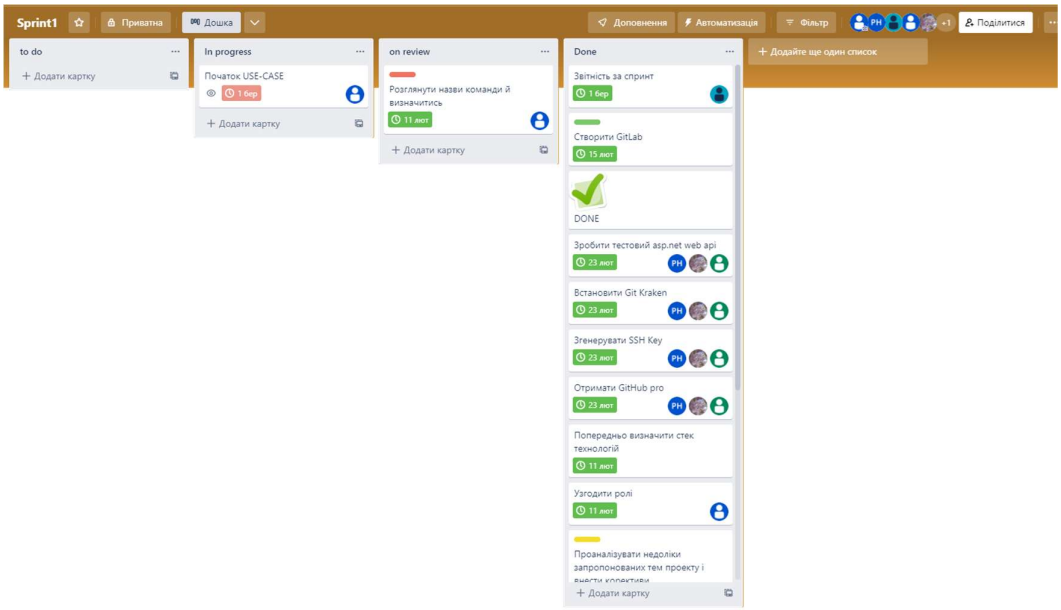


Рисунок 3 Дошка Trello

Trello є онлайн-інструментом, який дозволяє користувачам організувати роботу в команді та відслідковувати процес виконання завдань. Система базується на концепції дошок, на яких відображаються списки задач, які можуть бути легко перенесені з одного списку в інший, залежно від стану їх виконання. Trello є базовим інструментом планування з яскраво виділеними перевагами, але саме його ми заздалегідь обирали як систему для знайомства і початку проекту, що чітко видно на дошці планування, котру поповнив блок з підготовкою всіх супровідних процесів для переходу на GitLab.

GitLab. Обсяг та час проведеної роботи (1 спринт (01.03.2023- 15.03.2023)) **Дошка завдань команди**:

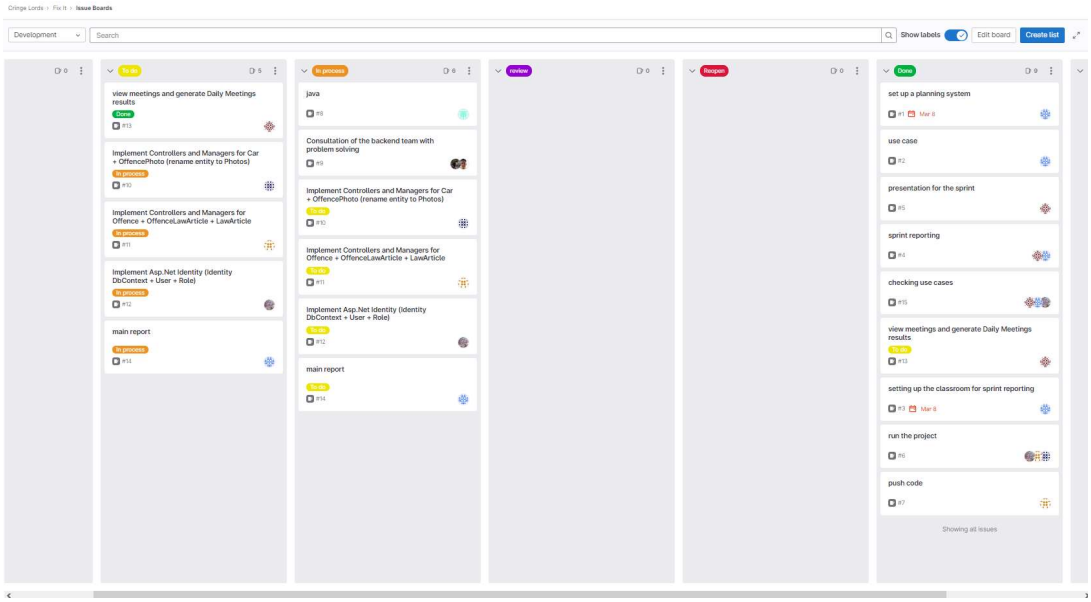


Рисунок 4. Дошка GitLab

GitLab - це веб-сервіс для керування кодом та проєктами розробки программного забезпечення. Основна функціональність GitLab включає систему контролю версій Git, систему управління проєктами та інтегрований засіб CI/CD.

GitLab, як система управління сподобалась більшості учасниками команди завдяки інтуїтивно зрозумілому розподілу дошок з тасками та злиття зі всім відомим GitHub. Також, через наявний досвід деяких учасників і зацікавленість інших вибір системи не був досить довгим. Система дає можливість індивідуального контролю роботи кожного учасника та зручне налаштування планів спринтів.

Основною проблемою системи виникла платна підписка, з котрою у команди виникли проблеми і було вирішено перейти до AzureDevOps.

AzureDevOps

Обсяг та час проведеної роботи (2.5 спринта (15.03.2023- 19.04.2023))

AzureDevOps - це хмарна платформа, надана Microsoft, яка пропонує набір інструментів та сервісів для команд розробників програмного забезпечення для планування, співпраці та розгортання програмних додатків.

Ось деякі з ключових компонентів AzureDevOps:

Azure Boards: це інструмент управління проектами, який дозволяє командам планувати, відстежувати та обговорювати роботу на всьому життєвому циклі розробки.

Azure Repos: це система контролю версій, яка дозволяє командам керувати вихідним кодом та відстежувати зміни з часом. Вона підтримує Git та Team Foundation Version Control (TFVC).

Саме ці основні пункти привернули увагу учасників проекту для подальшого переходу на платформу.

Задачі команди:

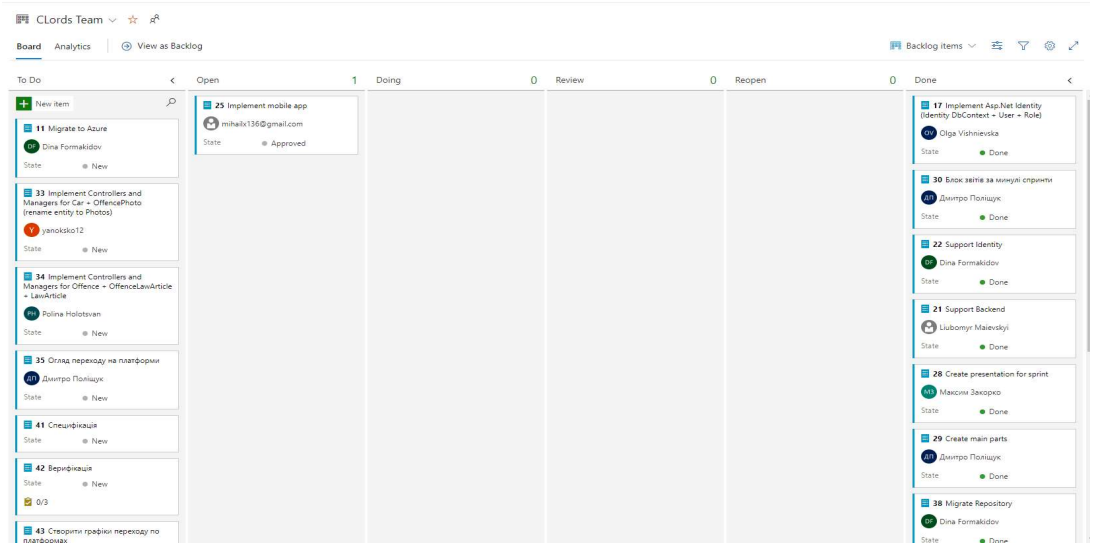


Рисунок 5. Дошка AzureDevOps

Проблеми переходу, відгуки та оцінки платформ учасниками: Проблем переходу між платформами не спостерігалось, до того ж, кожен перехід супроводжувався вивченям системи і новим досвідом роботи з ними для кожного учасника, що дало позитивні результати зі сторонни залучення кожного члена команди. Слід відзначити, що не всі системи отримали чудові оцінки і стали лідерами за оцінками учасників. Для визначення найзручнішої і уподобаної системи було проведено голосування з оцінками по основним критеріям, таким як:

- Функціональність – functionality – **F**
- Інтерфейс – Interface **-I**
- Масштабованість – Scability – **S**
- Сумісність з іншими системами – Copability–**C**

Оцінки є суто суб'єктивними і несуть за собою бачення зручності системи окремих учасників. Але це важливо, для визначення фаворита, для злагодженої роботи команди, що несе за собою зацікавленість зі сторони працівників, та якість виконання роботи :

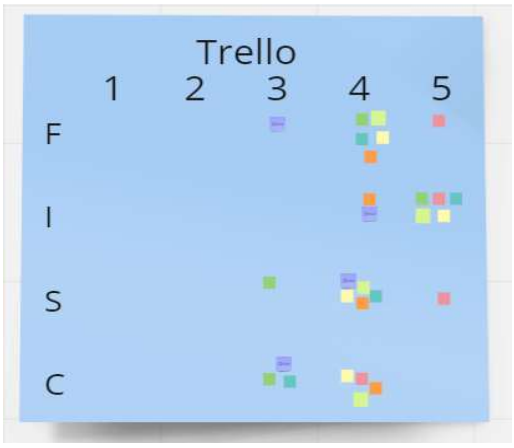


Рисунок 6. Дошка Відгуків Trello

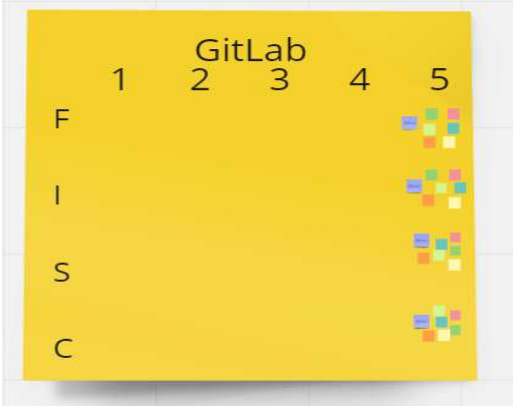


Рисунок 7. Дошка Відгуків GitLab

Рисунок 8. Дошка Відгуків AzureDevOps

Колір полотна учасника:

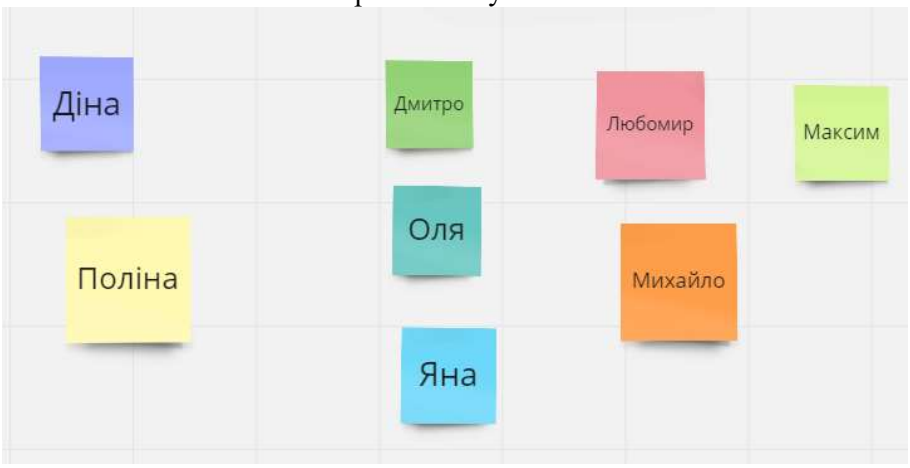


Рисунок 9. Дошка відповідності кольорів до голосів учасників

Підведемо підсумок з оцінок, отриманих опитуванням. Середній бал систем:

1. Trello: 27 голосів, середній бал - (4.2)
2. GitLab: 28 голосів, середній бал (5)

3. AzureDevOps: 28 голосів, середній бал - (4.17)

Отже, найкращою системою, на думку учасників проєкту став GitLab. Основні показники, котрі вивели GitLab лідери і були не такими гарними, як у інших платформ стали:

- Сумісність з іншими системами– **C (Trello)**
- Інтерфейс - **I (AzureDevOps)**

Ретроспектива. Для підтримки продуктивності і гарного настрою команди було вирішено проводити невеликий погляд в минуле і оцінювати проведену роботу разом з внесенням коректив або побажань. Приклад проведених оцінок по зустрічам в кінці спринтів:

Рисунок 10. Дошка відгуків за Sprint 1

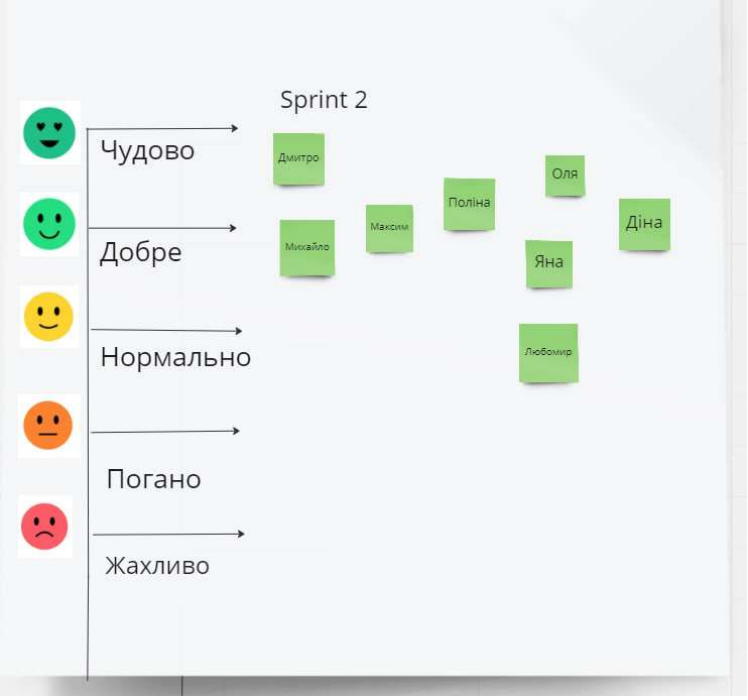


Рисунок11. Дошка відгуків за Sprint 2

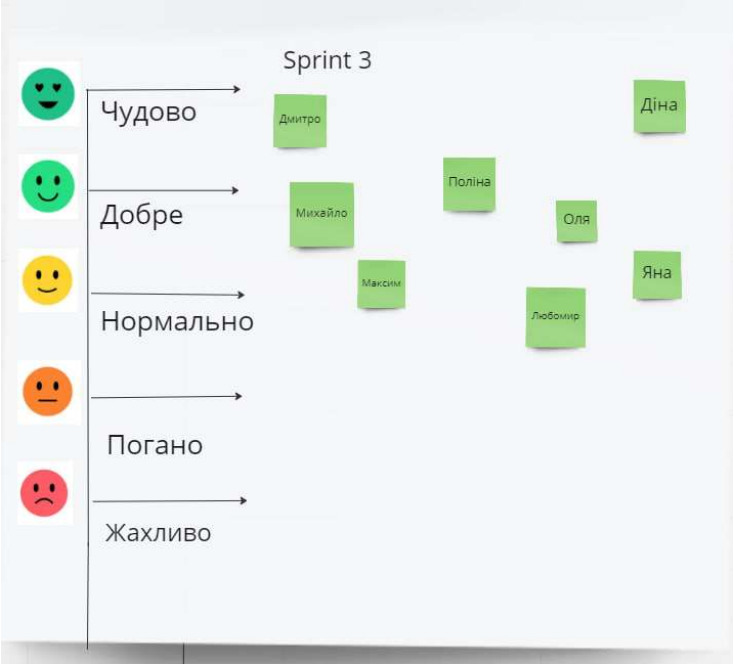


Рисунок 12. Дошка відгуків за Sprint 3

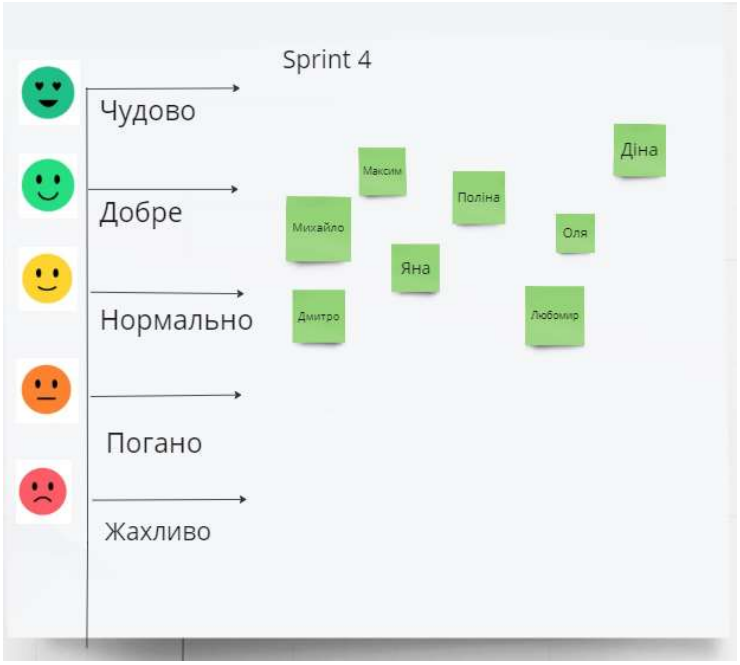


Рисунок 13. Дошка відгуків за Sprint 4

Побажання та проблеми:

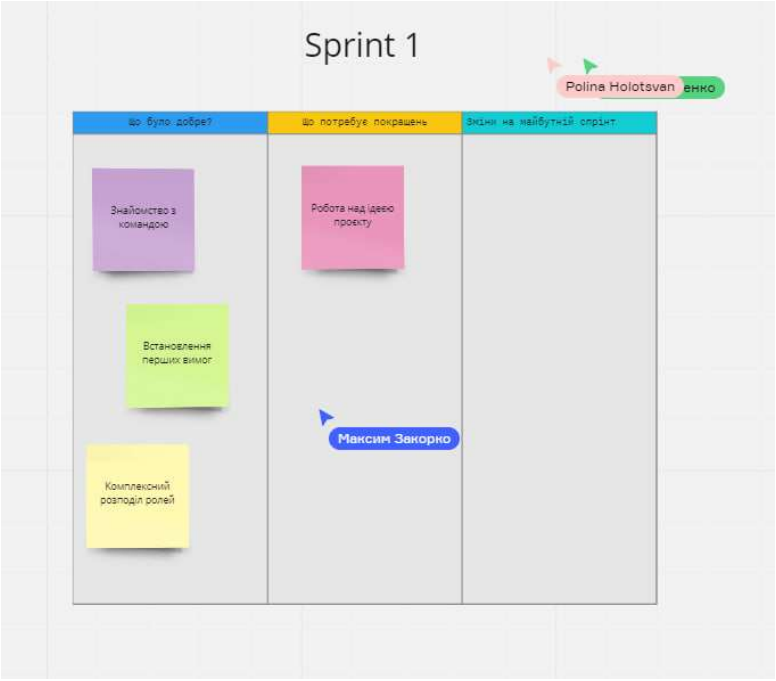


Рисунок 14. Дошка відгуків за Sprint 1

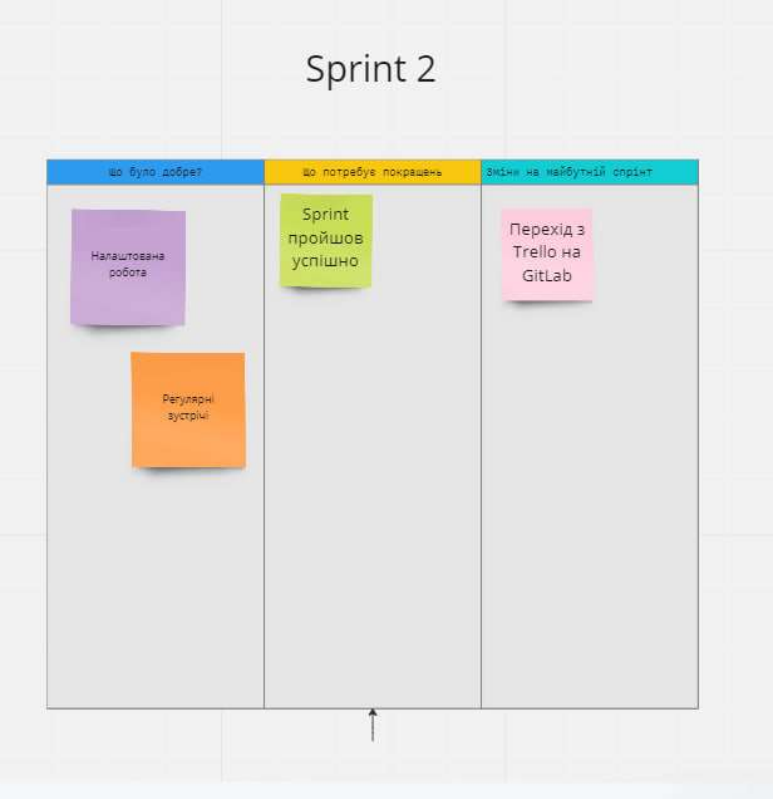


Рисунок 15. Дошка відгуків за Sprint 2

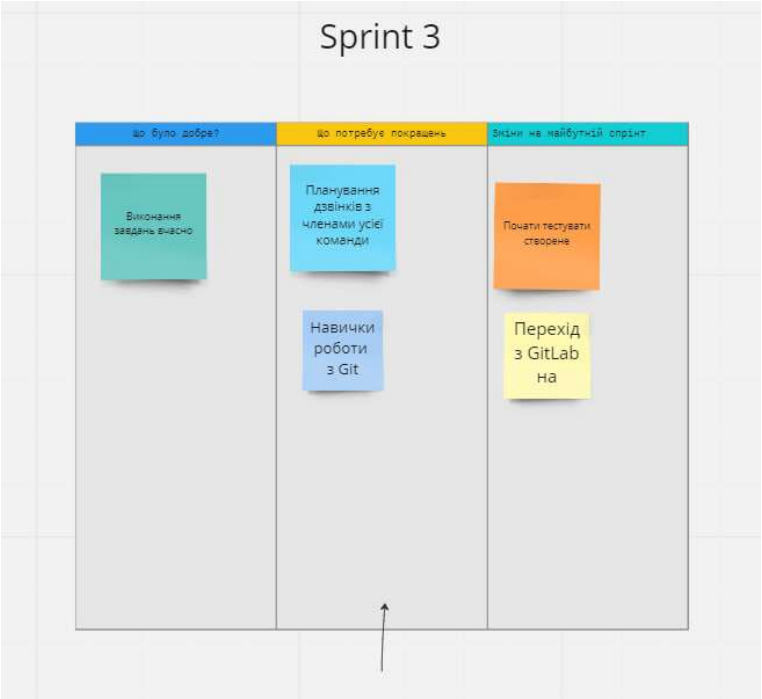


Рисунок 16. Дошка відгуків за Sprint 3

СПЕЦИФІКАЦІЯ

Загальний опис системи "Фіксуй"

Створення онлайн додатку для ведення історії правопорушень з всім необхідним обладнанням для висвітлення правопорушення в загальному доступі та винесення міри покарання (адмін-поліцейський). Перегляд основного списку правопорушень з розділенням на основні (ПДР) та всі інші (віддалені від основної мети застосунка).

Основні функції системи " Фіксуй ":

1) можливість додавати фото правопорушення (не менше 3)
2) можливість додавання точної дати та часу фіксації
3) можливість обрати країну реєстрації транспортного засобу
4) можливість додавання місцезнаходження правопорушника
5) можливість вибору правопорушення
6) можливість вписати номер авто правопорушника
7) можливість перегляду інформації із бази даних МВС про авто правопорушника
8) можливість вказати відмінність між інформацією із бази МВС та фактичною маркою/моделлю/кольору автомобіля
9) можливість додавання коментаря

Додаткові вимоги до системи:

1) можливість авторизуватися в системі
2) можливість підтвердити особистість
3) можливість переглянути попередні фотофіксації та результати перевірки
4) можливість отримати винагороду за повідомленням правопорушення у розмірі 10% від обсягу штрафу
5) можливість зайти в систему під роллю модератора для перегляду всіх наявних правопорушень

Для модератора:

1) можливість приймати рішення щодо ситуації
2) можливість залишати приватний/відкритий* коментар

Цілі створення системи:

Система фотофіксації правопорушень "Фіксуй" була створена для залучення учасників дорожнього руху, громадян та активних представників громадянського суспільства до безпосереднього контролю за безпекою на дорогах.

Цільовою авдиторією є користувачі "Фіксуй", які представлені в системі двома можливими ролями: модератора (він же керівник працівник правоохоронних органів) та користувачі. Тому в Системі передбачається виділення двох функціональних підсистем: адміністративна, призначена для модератора(ів) проекту та підсистема користувачів.

Призначення системи:

Система розробляється для посилення контролю за безпекою дорожнього руху за рахунок залучення учасників дорожнього руху, громадян та активних представників громадянського суспільства, зменшення випадків ДТП на дорогах і, як наслідок, кількості травмованих та загиблих в них людей, впровадження в роботу по контролю за безпекою дорожнього руху та правилами зупинки, стоянки, паркування сучасних технічних засобів та передових технологій.

Користувачі:

Продукт має підтримку різних груп користувачів, які мають різні права доступу для роботи з системою.

Групи користувачів:

- Звичайний користувач (фіксатор правопорушення) – користувач має змогу фіксувати правопорушення і викладати пости з необхідною інформацією про пригоду.

- Адмін (поліцейський)– користувач має змогу переглядати правопорушення з подальшим винесенням міри покарання у відповідності з силою порушення.

В системі передбачено функціонал створення нових користувачів.

Розділ правопорушення:

Головний екран включає:

- додавання правопорушення
- вибір перегляду списку існуючих
- інформація законодавчої бази стосовно діяльності проекту

Розділ створення правопорушення:

- вибір правопорушення
- завантаження фотографій порушення
- вибір місця реєстрації транспортного засобу
- коментар
- адреса правопорушення
- номер автомобіля
- дата та час фіксації
- чекбокс про відповідність авто з БД МВС

Розділ винесення покарання (адмін) (**додатково**)

- з головного екрану в вкладці перегляд правопорушень можна додати свій коментар-міру покарання по існуючому порушенню

Законодавча база

До Верховної Ради України було подано законопроєкт 5798, який передбачає фіксацію порушення правил дорожнього руху в автоматичному режимі або в режимі фотозйомки (відеозапису).

Номер, дата реєстрації: 5798 від 16.07.2021

Сесія реєстрації: 5 сесія IX скликання

Включено до порядку денного: 2911-IX від 07.02.2023

Редакція законопроекту: Основний

Рубрика законопроекту: Правова політика

Дати та стан проходження:	Очікує розгляду
20.12.2021	Надано висновок Комітету про розгляд
19.10.2021	Включено до порядку денного
08.10.2021	Надано висновок Комітету про включення до ПД
20.07.2021	Надано для ознайомлення
19.07.2021	Направлено на розгляд Комітету
16.07.2021	Передано на розгляд керівництву
16.07.2021	Одержано Верховною Радою

Рисунок 17. Дати прийняття законопроекту

Проект Закону про внесення змін до деяких законодавчих актів України щодо посилення ролі суспільства у заходах контролю за безпекою дорожнього руху та дотриманням правил зупинки, стоянки, паркування транспортних засобів.

Суть проекту Закону. Проектом Закону пропонується удосконалити чинний режим фотозйомки (відеозапису) та додати до переліку суб'єктів які можуть його здійснювати фізичних осіб, які є безпосередніми свідками порушення.

Режим фотозйомки (відеозапису) пропонується розповсюдити на всі порушення передбачені частинами першою, другою, третьою, п'ятою, шостою і сьомою статті 122 (крім перевищення встановлених обмежень швидкості руху транспортних засобів, порушення правил перевезення вантажів, буксирування транспортних засобів, порушення безпечної дистанції або інтервалу, порушення правил руху автомагістралями, використання зовнішніх освітлювальних приладів та їх переобладнання з порушенням вимог відповідних стандартів, користування під час руху транспортного засобу засобами зв'язку, не обладнаними технічними пристроями, що дозволяють вести перемови без допомоги рук, порушення правил навчальної їзди), а також на порушення передбачені частиною першою статті 123 та частинами першою, другою, шостою та восьмою статті 152-1 КУПАП.

Обов'язковою умовою для здійснення режиму фотозйомки (відеозапису) стане використання спеціальних програмних засобів, які будуть забезпечувати ідентифікацію користувача через "Дію", гарантуватимуть цілісність та автентичність створених матеріалів фотозйомки (відеозапису), забезпечать визначення часу, дати та географічних координат здійснення фіксації. За допомогою спеціальних програмних засобів буде реалізована технічна передача матеріалів фотозйомки (відеозапису) до органів, які розглядають адміністративні справи. Пропонується щоб порядок впровадження та експлуатації таких засобів встановлювався Міністерством внутрішніх справ України. Додатково пропонується покласти на Міністерство внутрішніх справ України повноваження з ведення переліку спеціальних програмних засобів, які можливо використовувати в режимі фотозйомки (відеозапису), на офіційному веб-сайті, а також встановлення вимог до таких засобів щодо захисту інформації в ЄІС МВС.

Громадяни, які зафіксували порушення в режимі фотозйомки (відеозапису) та з використанням спеціального програмного засобу звернулися з відповідною інформацією до уповноваженого органу, отримують статус сповіщувача та в разі притягнення винної особи до відповідальності та сплати нею накладеного штрафу можуть отримати винагороду в розмірі 10% від цього штрафу. Порядок таких виплат пропонується передати на затвердження Кабінету Міністрів України.

З метою недопущення зловживань з боку громадян, пропонується встановити кримінальну відповідальність за подання підроблених або штучно створених матеріалів фотозйомки (відеозапису).

У той же час, гарантується конфіденційність та нерозголошення відомостей про таких громадян.

Таким чином проектом Закону пропонується збалансований механізм залучення громадян до контролю за безпекою дорожнього руху, який передбачає що держава зможе приймати та розглядати заяви від таких осіб, і при цьому така інформація буде захищена. Недобросовісні користувачі в свою чергу можуть бути покарані.

Розгляд матеріалів від громадян пропонується покласти на органи Національної поліції. Враховуючи потенційно великі об'єми матеріалів фотозйомки (відеозапису) які можуть надходити, пропонується прибрати вимогу про обов'язкову наявність спеціальних звань для працівників Національної поліції, які будуть розглядають справи про адміністративні правопорушення зафіксовані в автоматичному режимі або в режимі фотозйомки (відеозапису).

Проектом Закону також пропонується доповнити статтю 251 КУПАП терміном "електронні докази", порядком їх подачі та зберігання.

Законопроектом пропонуються також інші зміни до Кодексу України про адміністративні правопорушення.

Приведення редакції статті 123 КУПАП у відповідність до інших положень Кодексу, адже чинна редакція є застарілою.

Викладення статті 121-1 КУПАП в новій редакції, оскільки діюча редакція, по-перше, звужує перелік підстав для заборони експлуатації транспортних засобів, наведених в статті 37 Закону "Про дорожній руху", а по-друге, необґрунтовано покладає відповідальність на особу, яка навряд чи є причасною до зазначених правопорушень – на водія. Він може керувати чужим транспортним засобом, орендованим тощо. Завдяки змінам сплачувати штраф за експлуатацію транспортного засобу з підробленими ідентифікаційними номерами складових частин, або непереєстрованого, буде особа, за якою зареєстровано транспортний засіб, а не особа, яка ним керує.

Розширення повноважень виконавчих комітетів сільських, селищних, міських рад на розгляд справи про адміністративні правопорушення, передбачені частиною шостою статті 122 КУПАП (зупинка на місцях для інвалідів), адже інспектори з паркування можуть більш ефективно розглядати такі справи, що у свою чергу знизить навантаження на роботу поліцейських.

Надається можливість складати протокол про адміністративне правопорушення та постанову про накладення адміністративного стягнення у формі електронного документа.

Прибирається обов’язок розміщення постанови у справі про адміністративне правопорушення під лобовим склом автомобіля, адже такий документ містить персональні данні особи та може призвести до розголошення таких даних, при цьому обов’язок розміщення повідомлення залишається.

Пропонується можливість за бажанням особи та відповідним її зверненням отримання не тільки інформації про правопорушення, але й електронної постанови.

У постановах по справах про адміністративні правопорушення, зафіксовані в режимі фотозйомки (відеозапису), обов’язково буде зазначатися назва спеціального мобільного засобу.

Постанови у справах про адміністративне правопорушення, зафіксовані в автоматичному режимі, можливо буде оскаржити виключно до суду.

У Кримінальному кодексі України пропонується внести зміни до статей 384 та 386, та криміналізувати підроблення доказів у справах про адміністративні правопорушення, зафіксовані в режимі фотозйомки (відеозапису), а також незаконний тиск на свідків у справах про адміністративне правопорушення взагалі.

У КАСУ пропонується передбачити, що держава оплачує витрати пов’язані із проведенням експертизи з подальшим стягненням таких витрат в порядку, встановленому цим Кодексом, за наявності всіх наступних умов:

1) якщо справа стосується оскарження рішення (постанови) у справі про адміністративне правопорушення у сфері забезпечення безпеки дорожнього руху або про порушення правил зупинки, стоянки, паркування транспортних засобів, зафіксоване в режимі фотозйомки (відеозапису) сповіщувачем;

2) якщо клопотання про проведення експертизи заявлено позивачем;

3) якщо експертиза стосується дослідження фотозйомки (відеозапису) на предмет його недостовірності або підробки.

Якщо за результатами проведеної експертизи не встановлено недостовірність або підробку фотозйомки (відеозапису), витрати на проведення такої експертизи стягуються з позивача. Якщо за результатами проведеної експертизи встановлено недостовірність або підробку фотозйомки (відеозапису), держава має право на зворотну вимогу (регрес)

до сповіщувача, який надав такі матеріали фотозйомки (відеозапису), в межах суми витрат на проведення такої експертизи.

Відповідно до постанови Кабінету Міністрів України від 26 жовтня 2011 р. № 1098 "Деякі питання надання підрозділами Міністерства внутрішніх справ, Національної поліції та Державної міграційної служби платних послуг" вартість експертизи, дослідження звуко- та відеозапису складає: простої складності - 1407 грн., середньої складності – 4688 грн., складні – 9870 грн. Особа, яка вважає що відеоматеріали є "сфабрикованими", зможе з цих підстав оскаржити постанову та заявити відповідну експертизу, не несучи фінансовий тягар для цього. Натомість держава в будь-якому випадку стягне кошти на її проведення – з недобросовісного сповіщувача який надав такі підроблені докази, або з самого скаржника якщо він оскаржував постанову безпідставно і відеоматеріали є цілісними. За даними Міністерства юстиції України наразі зареєстрованими є більше 10 методик проведення експертизи відеозвукозапису. Таким чином право особи, яка притягається до відповідальності на оскарження, буде збалансованим в даній категорії справ.

Також пропонується встановити строк на оскарження постанови – 10 днів з дня набрання законної сили, відповідно до положень КУПАП.

В Законі України "Про дорожній рух" пропонується надати Кабінету Міністрів України відповідні повноваження затверджувати порядок виплати винагороди сповіщувачам. Передбачається право громадян на отримання винагороди, вносяться інші технічні зміни.

В Законі України "Про безоплатну правову допомогу" передбачається право сповіщувачів, які повідомили інформацію про адміністративне правопорушення, зафіксоване в режимі фотозйомки (відеозапису), на отримання безоплатної правової допомоги [5]

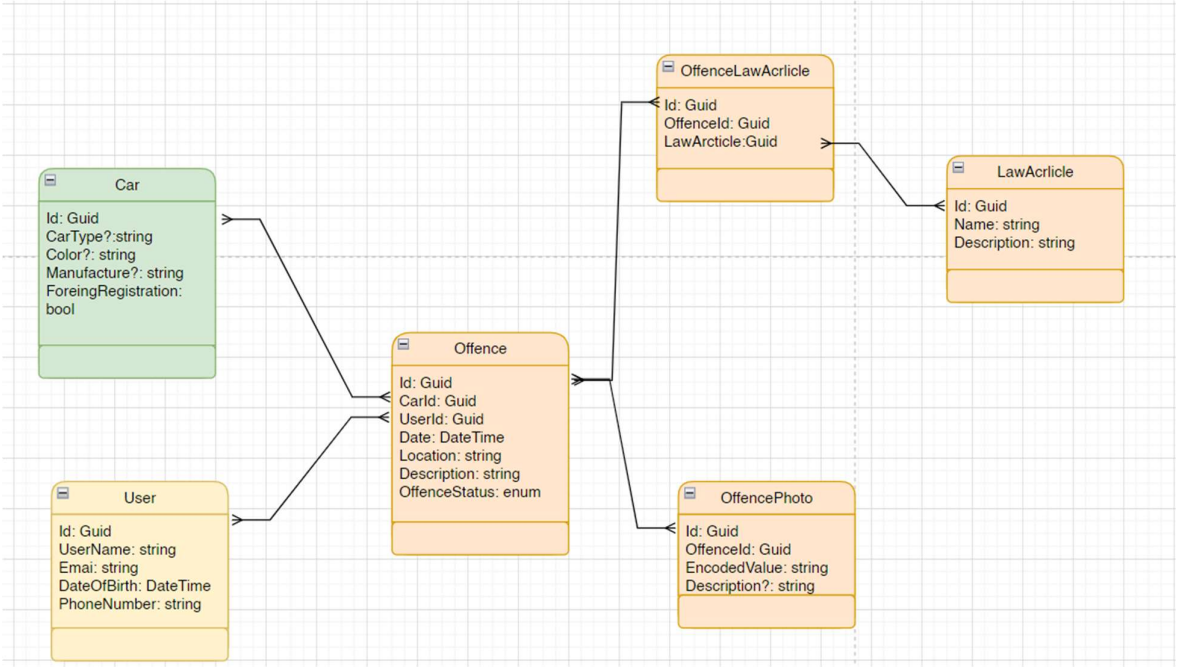


Рисунок 18. Специфікація OCL

Car

Обмеження:

- Id має бути унікальним та його значення має бути більше 0

- CarType має бути не порожнім
- Color має бути не порожнім
- Manufacrure має бути не порожнім

Context Car inv:
self. Id > 0 and Car.allInstances() -> isUnique (self. Id)
context Car inv:
self. CarType->notEmpty()
context Car inv:
self. Color ->notEmpty()
context Car inv:
self. Manufacrure ->notEmpty()
context Car:: ForeingRegistration: Boolean

User

Обмеження:

- Id має бути унікальним та його значення має бути більше 0
- UserName має містити більше двох але менше 50 символів
- Email має відповідати стандартам RFC та бути унікальним
- DateOfBirth повинна бути такою, щоб вік особи на сьогодні був більше 18 років та менше 130 років
- PhoneNumber має починатися з +380 та мати 9 цифр

Context User inv:
self. Id > 0 and Car.allInstances() -> isUnique (self. Id)

Context User inv:
self.UserName.size() > 2 and self.UserName.size() < 50
Context User inv:
self.DateOfBirth - currentDate >= 18
and self.DateOfBirth - currentDate <= 130
Context User
def PhoneNumver_before='+380'
inv
self.PhoneNumver.size() =9
Context User
def sq_email = sequence{1..self.email.size()} -> collect(i | email.at(i))
def email_before = sq_email -> subSequence(1, indexOf("@"))
def email_after = sq_email -> subSequence(indexOf("@")+1,self.email.size())
def allowed_after = Sequence('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '-', '.')
def allowed_before = allowed_after -> union(Sequence{'!', '#', '$', '%', '&', '"', '*', '+', '-', '/', '=', '?',
'^', '_', '`', '{', '|', '}', '~'})
inv
self.email.toLower() = self.email
and sq_email -> count("@") = 1
and email_before -> count(".") = 0
and email_before -> size() <= 64
and allowed_before -> includesAll(email_before)
and email_after -> count(".") > 0

and email_after -> first() <> “-”
and email_after ->last() <> “-”
and email_after -> size() <= 64
and allowed_after -> includesAll(email_after)
and Customer.allInstances() -> isUnique (self.email)

Верифікація (створення offence) було проведено на мові Dafny

User Experience

1. **Історія**: Як користувач сервісу FixIt я хочу створити пост зафіксованого мною правопорушення

Назва: Створити правопорушення (Createoffence)
Ціль: Створити пост правопорушення
Діюча особа: будь-які користувачі FixIt
Попередні умови:
- Користувач має бути зареєстрований в FixIt
- Користувач повинен мати доступ до інтернету

Успішний сценарій:
1. Користувач натискає кнопку "додати правопорушення"
2. Користувач вибирає тип правопорушення за наданими кнопками-типами
- правопорушення автомобіля з українською реєстрацію номерного знаку (Ukrainianregistration)
- автомобілі з іншою реєстрацією (Otherregistration)
3. Система показує меню з параметрами правопорушення для його створення
4. Користувач завантажує фотографії місця події та автомобіля
5. Користувач визначає час події
6. Користувач вписує номер реєстраційного номера автомобіля-правопорушника
7. Система відображає дані транспортного засобу по номеру автомобіля
8. Користувач має можливість вказати чи актуальні дані системи
9. Користувач вказує місцезнаходження правопорушення
10. Користувач має можливість залишити коментар до події
11. Користувач натискає кнопку "додати правопорушення"

Неуспішний сценарій:
1. Користувач має бути зареєстрований в FixIt
- Система виводить повідомлення "Ви повинні зареєструватися в системі"
2. Користувач не має доступу до інтернету
- Система виводить повідомлення "Немає доступу до Інтернету. Перевірте інтернет підключення"

2. **Історія**: Як користувач сервісу FixIt я хочу переглянути список створених мною правопорушень

Назва: Перегляд створених користувачем правопорушень
Ціль: Перегляд створених користувачем правопорушень
Діюча особа: будь-які користувачі FixIt
Попередні умови:
- Користувач має бути зареєстрований в FixIt
- Користувачмає створити перед цим хоча б одне правопорушення
- Користувач має натиснути кнопку "перегляд правопорушень"
- Користувач повинен мати доступ до інтернету

Успішний сценарій:

1. Користувач натискає кнопку “перегляд правопорушень завантажених мною” (‘‘myfixations’’)
2. Система показує меню з списком існуючих правопорушень
3. Користувач обирає правопорушення, котре цікавить його
4. Користувач натискає кнопку “перегляд”
5. Система відображає дані про правопорушення

Неуспішний сценарій:

1. Користувач має бути зареєстрований в FixIt

- Система виводить повідомлення "Ви повинні зареєструватися в системі"

2. Користувач не має доступу до інтернету

- Система виводить повідомлення "Немає доступу до Інтернету. Перевірте інтернет підключення"

3. Користувач не має жодного звернення

- Система виводить повідомлення "Ви повинні створити хоча б одне звернення"

3. **Історія**: Як адміністратор сервісу FixIt я хочу винести вирок зафіксованому і доданому в додаток правопорушенню

Назва: Винести вирок
Ціль:Винести міру покарання правопорушнику

Діюча особа: адміністратор-поліцейський FixIt

Попередні умови:

- Користувач має бути зареєстрований в FixIt
- Користувачумають бути надані права адміністратора
- Користувач повинен мати доступ до інтернету
- Користувач має зайти в розділ "список правопорушень"

Успішний сценарій:

1. Користувач вибирає правопорушення зі списку завантажених правопорушень в додатку
2. Користувач натискає кнопку "Перевірити правопорушення" (Checkingoffence)
3. Адміністратор детально описує в окремому коментарі до правопорушення для винесення вироку причину покарання
4. Адміністратор натискає кнопку “checked

Неуспішний сценарій:

1. Користувач має бути зареєстрований в FixIt

- Система виводить повідомлення "Ви повинні зареєструватися в системі"

2. Користувач не має доступу до інтернету

- Система виводить повідомлення "Немає доступу до Інтернету. Перевірте інтернет підключення"

4. **Історія**: Як користувач сервісу FixIt я хочу змінити дані про себе.

Назва: Змінити дані профілю
Ціль: Змінити дані про себе.

Діюча особа: будь-які користувачі FixIt

Попередні умови:

- Користувач має бути зареєстрований в FixIt
- Користувач повинен мати доступ до інтернету
- Користувач має зайти в розділ "Профіль"

Успішний сценарій:

1. Користувач натискає кнопку "Редагувати"

2. Користувач обирає, які саме дані про себе він хоче змінити (Email або номер телефону).

3. Користувач вводить нові дані.

4. Користувач натискає кнопку "Зберегти"

Неуспішний сценарій:

1. Користувач має бути зареєстрований в FixIt

- Система виводить повідомлення "Ви повинні зареєструватися в системі"

2. Користувач не має доступу до інтернету

- Система виводить повідомлення "Немає доступу до Інтернету. Перевірте інтернет підключення"

Альтернативні шляхи:

Назва: Недостатньо доказів для винесення вироку "Порушник немає понести покарання"

1. Адміністратор детально описуєпричину відхилення в окремому коментарі до правопорушення для винесення вироку

2. Адміністратор натискає кнопку “rejected

РОЗРОБКА ЗАСТОСУНКУ

Реєстрація:

```
[Route("api/[controller]")]
[ApiController]
Ссылок: 1
public class AuthController : Controller
{
    private readonly AuthManager _authManager;
    private readonly IdentityManager _identityManager;

    Ссылок: 0
    public AuthController(AuthManager authManager, IdentityManager identityManager)
    {
        _authManager = authManager;
        _identityManager = identityManager;
    }

    [HttpPost("Register")]
    [AllowAnonymous]
    [ProducesResponseType(StatusCodes.Status200OK)]
    [ProducesResponseType(StatusCodes.Status404NotFound)]
    [ProducesResponseType(StatusCodes.Status500InternalServerError)]
    Ссылок: 0
    public async Task<IActionResult> Register(UserCreateModel model)
    {
        if (await _authManager.Register(model) == true)
        {
            return StatusCode(201);
        }
        return BadRequest();
    }
```

Рисунок 19. Код реєстрації

Реалізацію виносимо в AuthManager:

```
Ссылок: 4
public class AuthManager
{
    private readonly UserManager<User> _userManager;
    private readonly AccessTokenSettings _authSettings;
    private readonly EmailSenderService _emailSender;
    private readonly IHttpContextAccessor _httpContextAccessor;


    Ссылок: 0
    public AuthManager(UserManager<User> userManager, IOptions<AccessTokenSettings> authSettings,
        EmailSenderService emailSender, IHttpContextAccessor httpContextAccessor)
    {
        _userManager = userManager;
        _authSettings = authSettings.Value;
        _emailSender = emailSender;
        _httpContextAccessor = httpContextAccessor;
    }

Ссылок: 1
public virtual async Task<bool> Register(UserCreateModel model)
{
    User user = model.Adapt<User>();
    IdentityResult res = await _userManager.CreateAsync(user, model.Password);
    if (!res.Succeeded)
    {
        throw new Exception("Invalid values");
    }

    var encodedToken = HttpUtility.UrlEncode(await _userManager.GenerateEmailConfirmationTokenAsync(user));
    var encodedEmail = HttpUtility.UrlEncode(user.Email);

    string host = _httpContextAccessor.HttpContext.Request.Host.Value;

    var confirmationlink = "https://"+host+"/api/Auth/ConfirmEmail?email=" + encodedEmail + "&token=" + encodedToken;

    await _emailSender.SendEmailAsync(user.Email, "Confirm Email", "Click on the link to confirm email: " +
        confirmationlink);
    await _userManager.AddToRoleAsync(user, "User");
    return true;
}
```

Рисунок 20. Код AuthManager

```
Ссылок: 6
public class EmailSenderService
{
    private readonly ISendGridClient _sendGridClient;
    private readonly SendGridSettings _sendGridSettings;

    Ссылок: 0
    public EmailSenderService(ISendGridClient sendGridClient,
    IOptions<SendGridSettings> sendGridSettings)
    {
        _sendGridClient = sendGridClient;
        _sendGridSettings = sendGridSettings.Value;
    }

    Ссылок: 2
    public async Task<bool> SendEmailAsync(string email, string subject, string htmlMessage)
    {
        var msg = new SendGridMessage()
        {
            From = new EmailAddress(_sendGridSettings.FromEmail, _sendGridSettings.EmailName),
            Subject = subject,
            PlainTextContent = htmlMessage
        };
        msg.AddTo(email);
        var response = await _sendGridClient.SendEmailAsync(msg);
        return response.IsSuccessStatusCode;
    }
}
```

Рисунок 21. Код надсилання листа на пошту

Після реєстрації користувачу на пошту надсилається лист з лінкою на RequestUrl, яка містить в собі певні значення параметрів(пошту та токен), і викликає ендпоінт ConfirmEmail. Надсилання листів відбувається за допомогою SendGrid.

```
[HttpGet("ConfirmEmail")]
[AllowAnonymous]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
public async Task<IActionResult> ConfirmEmail(string email, string token)
{
    if (await _authManager.ConfirmEmail(email, token) == true);
    return Ok();
}
```

```
Ссылок: 1
public virtual async Task<bool> ConfirmEmail(string email, string token)
{
    var user = await _userManager.FindByEmailAsync(email);
    if (user == null)
    {
        throw new Exception("The User with such email doesn't exist");
    }
    var result = await _userManager.ConfirmEmailAsync(user, token);
    if (!result.Succeeded)
    {
        throw new Exception("Unable to confirm email");
    }
    return true;
}
```

Рисунок 22. Код надсилання листа на пошту

Авторизація:

Увійти в систему можуть лише зареєстровані користувачі з підтвердженою поштою.

```
[HttpPost("Login")]
[AllowAnonymous]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
Ссылок: 0
public async Task<IActionResult> Login(LoginModel loginModel)
{
    TokenPairModel tokenPair = await _authManager.Login(loginModel);
    return Ok(tokenPair);

}
```

```
Ссылок: 1
public virtual async Task<TokenPairModel> Login(LoginModel loginModel)
{
    TokenPairModel tokenPair = new TokenPairModel();
    User user = await _userManager.FindByEmailAsync(loginModel.Email);
    if (user == null)
    {
        throw new Exception("The User with such email doesn't exist");
    }
    if (!await _userManager.CheckPasswordAsync(user, loginModel.Password))
    {
        throw new Exception("");
    }
    if (!await _userManager.IsEmailConfirmedAsync(user))
    {
        throw new Exception("Email is not confirmed");
    }

    string accessToken = GenerateJwtToken(user);

    var refreshToken = await _userManager.GenerateUserTokenAsync(user, "FixIt", "RefreshToken");
    await _userManager.SetAuthenticationTokenAsync(user, "FixIt", "RefreshToken", refreshToken);
    tokenPair.RefreshToken = refreshToken;
    tokenPair.AccessToken = accessToken;

    return tokenPair;
}
```

```
    Ссылок: 2
    public string GenerateJwtToken(User user)
    {
        var identity = GetClaims(user);
        if (identity == null)
        {
            return null;
        }

        var now = DateTime.UtcNow;
        var key = Encoding.ASCII.GetBytes(_authSettings.SigningKey);

        var jwt = new JwtSecurityToken(
                    issuer: _authSettings.Issuer,
                    audience: _authSettings.Audience,
                    notBefore: now,
                    claims: identity.Claims,
                    expires: now.Add(TimeSpan.FromMinutes(15)),
                    signingCredentials: new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256));
        var encodedJwt = new JwtSecurityTokenHandler().WriteToken(jwt);
        return encodedJwt;
    }
}
```

```
Ссылок: 1
private ClaimsIdentity GetClaims(User user)
{
    var claims = new List<Claim>
        {
            new Claim(ClaimsIdentity.DefaultNameClaimType, user.Id.ToString())
        };
    ClaimsIdentity claimsIdentity =
    new ClaimsIdentity(claims, "AccessToken", ClaimsIdentity.DefaultNameClaimType,
        ClaimsIdentity.DefaultRoleClaimType);
    return claimsIdentity;
}
```

Рисунок 23. Код авторизації

При успішній авторизації повертається пара згенерованих токенів: AccessToken та RefreshToken. AccessToken дає доступ до акаунтукористувача. Цей токен має expiriationtime 15 хвилин, після яких стає недійсним, і доступ за цим токеном стає неможливим. RefreshToken необхідний для оновлення пари токенів.

Оновлення токенів:

ЕндпоінтRefresh оновлює пару токенів. Як значення параметрів він приймає пару токенів – AccessToken та RefreshToken, і повертає нову пару.

```
[HttpPost("Refresh")]
[AllowAnonymous]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
Ссылок: 0
public async Task<IActionResult> Refresh(TokenPairModel tokenPair)
{
    TokenPairModel newTokenPair = await _authManager.Refresh(tokenPair);
    if (newTokenPair != null)
    {
        return Ok(newTokenPair);
    }
    return BadRequest();
}
```

```
Ссылок 1
public async Task<TokenPairModel> Refresh(TokenPairModel tokenPair)
{
    string accessToken = tokenPair.AccessToken;
    string refreshToken = tokenPair.RefreshToken;

    var tokenValidationParameters = new TokenValidationParameters
    {
        ValidateAudience = false,
        ValidateIssuer = false,
        ValidateIssuerSigningKey = true,
        IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(_authSettings.SigningKey)),
        ValidateLifetime = false
    };

    var tokenHandler = new JwtSecurityTokenHandler();
    var principal = tokenHandler.ValidateToken(accessToken, tokenValidationParameters, out SecurityToken securityToken);
    if (securityToken is not JwtSecurityToken jwtSecurityToken || !jwtSecurityToken.Header.Alg.Equals(SecurityAlgorithms.HmacSha256,
        StringComparison.InvariantCultureIgnoreCase))
        throw new SecurityTokenException("Invalid token");

    if (principal == null)
    {
        throw new Exception("Not existing access token.");
    }
    string id = principal.Identity.Name;

    var user = await _userManager.FindByIdAsync(id);
    var userToken = await _userManager.GetAuthenticationTokenAsync(user, "FixIt", "RefreshToken");
    var isValid = await _userManager.VerifyUserTokenAsync(user, "FixIt", "RefreshToken", refreshToken);

    if (user == null || userToken != refreshToken || !isValid)
    {
        return null;
    }

    var newAccessToken = GenerateJwtToken(user);

    await _userManager.RemoveAuthenticationTokenAsync(user, "FixIt", "RefreshToken");

    var newRefreshToken = await _userManager.GenerateUserTokenAsync(user, "FixIt", "RefreshToken");
    await _userManager.SetAuthenticationTokenAsync(user, "FixIt", "RefreshToken", newRefreshToken);

    tokenPair.AccessToken = newAccessToken;
    tokenPair.RefreshToken = newRefreshToken;

    return tokenPair;
}
```

Рисунок 24. Код оновлення токенів

ЕндпоінтWhoAmI повертає інформацію про користувача, який його викликає. Тобто він дає можливість користувачу переглянути інформацію про себе.

```
[HttpGet("WhoAmI")]
[ProducesResponseType(StatusCodes.Status200OK)]
[ProducesResponseType(StatusCodes.Status404NotFound)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
Ссылок 0
public async Task<IActionResult> WhoAmI()
{
    UserDetailedViewModel user = await _identityManager.WhoAmI();
    return Ok(user);
}
```

```
Ссылок 6
public class IdentityManager
{
    private readonly UserManager<User> _userManager;
    private readonly IHttpContextAccessor _httpContextAccessor;
    private readonly EmailSenderService _emailSender;

    Ссылок 0
    public IdentityManager(UserManager<User> userManager, IHttpContextAccessor httpContextAccessor, EmailSenderService emailSender
    {
        _userManager = userManager;
        _httpContextAccessor = httpContextAccessor;
        _emailSender = emailSender;
    }

    Ссылок 1
    public virtual async Task<UserDetailedViewModel> WhoAmI()
    {
        string id = _httpContextAccessor.HttpContext.User.Identity.Name;
        User user = await _userManager.FindByIdAsync(id);
        return user.Adapt<UserDetailedViewModel>();
    }
```

Рисунок 25. Код ендпоінта WhoAmI

Процес авторизації:

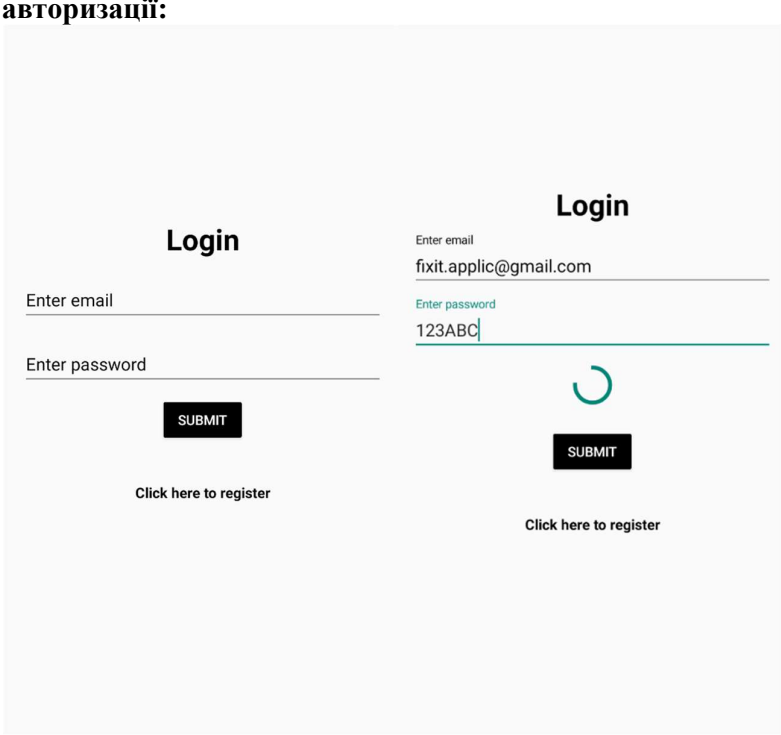


Рисунок 26. Сторінка авторизації

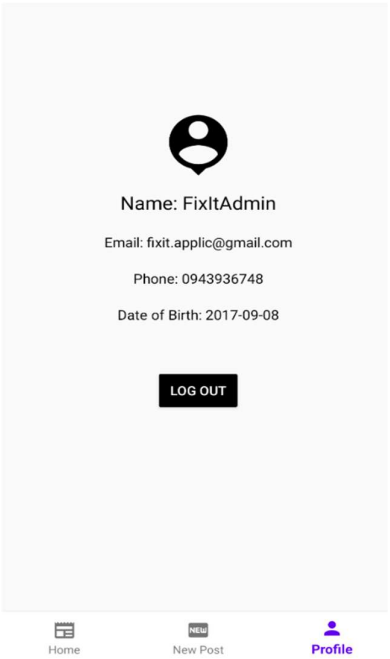


Рисунок 27. Сторінка авторизації

Головна функція login():

```
public void login() {
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                BufferedReader br = null;
                URL url = new URL( spec: "https://fixit228.azurewebsites.net/api/Auth/Login");
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Content-Type", "application/json;charset=UTF-8");
                conn.setRequestProperty("Accept","application/json");
                conn.setDoOutput(true);
                conn.setDoInput(true);
                JSONObject jsonParam = new JSONObject();
                jsonParam.put( name: "email", email);
                jsonParam.put( name: "password", password);
                DataOutputStream os = new DataOutputStream(conn.getOutputStream());
                os.writeBytes(jsonParam.toString());
                os.flush();
                os.close();
                br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
                JSONObject jsonObject = new JSONObject(br.readLine());
                authToken = jsonObject.getString( name: "accessToken");
                refreshToken = jsonObject.getString( name: "refreshToken");
                if(conn.getResponseCode() == 200){
                    checkUser();
                }
            } catch (Exception e) {e.printStackTrace();}
        }
    });
    thread.start();
}
```

Рисунок 28. Функція логіна

Увійти в систему можуть лише зареєстровані користувачі з підтвердженою поштою. Якщо вхід був успішним - користувачу видається AccessToken та RefreshToken, які мають свій строк життя - 15 хвилин.

Створення нового Offence:

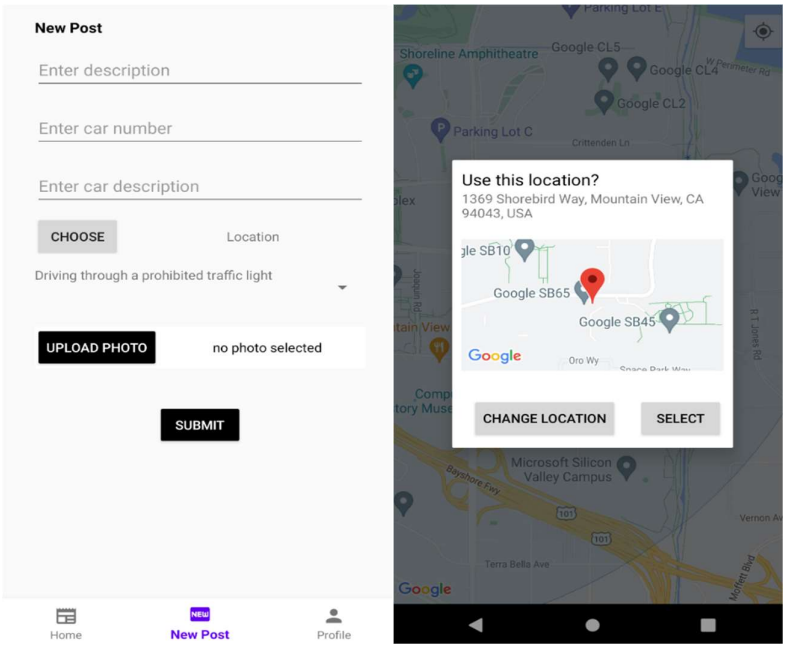


Рисунок 29. Створення offence і карта місця порушення

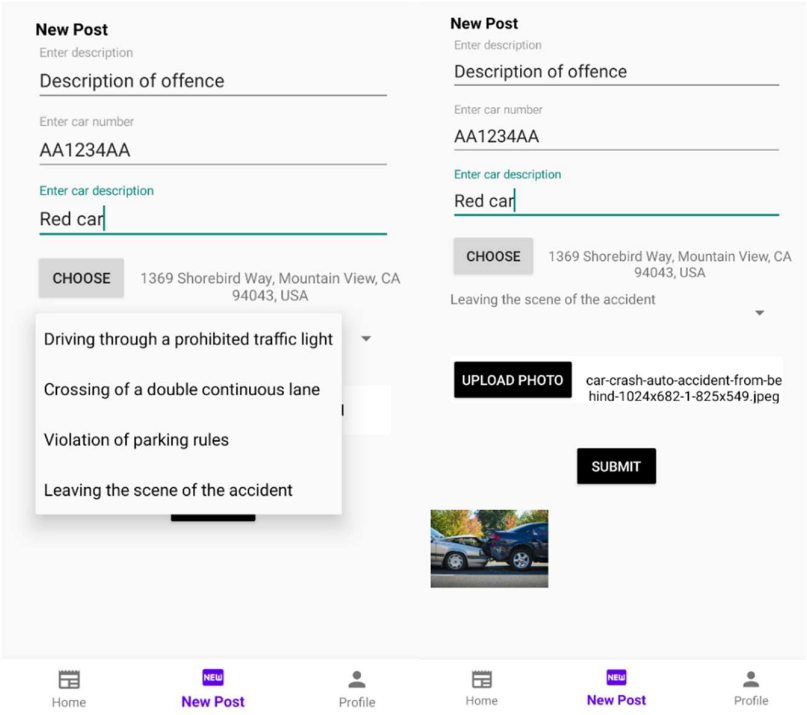


Рисунок 30. Демонстрація створення offence

Головна функція створення Offence, createPost():

```
private void createPost(String image) {
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                BufferedReader br = null;
                URL url = new URL( spec: "https://fixit228.azurewebsites.net/api/fix-it/Offence");
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Content-Type", "application/json;charset=UTF-8");
                conn.setRequestProperty("Accept","application/json");
                conn.setRequestProperty("Authorization","Bearer "+authToken);
                conn.setDoOutput(true);
                conn.setDoInput(true);
                JSONObject jsonParam = new JSONObject();
                jsonParam.put( name: "userId", userId);
                jsonParam.put( name: "latitude", location);
                jsonParam.put( name: "longitude", crime);
                jsonParam.put( name: "offenceDescription", description);
                jsonParam.put( name: "offenceStatus",  value: 0);
                jsonParam.put( name: "carNumber", carNumber);
                jsonParam.put( name: "isCarMatch",  value: 1);
                jsonParam.put( name: "carDescription", carDescription);
                DataOutputStream os = new DataOutputStream(conn.getOutputStream());
                os.writeBytes(jsonParam.toString());
                os.flush();
                os.close();
                br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
                if(conn.getResponseCode() != 200){
                    DataInputStream inputStream = new DataInputStream(conn.getErrorStream());
                }
                JSONObject jsonObject = new JSONObject(br.readLine());
                id = jsonObject.getString( name: "id");
                Looper.prepare();
                Handler handler = new Handler(Looper.getMainLooper()) {
                    @Override
                    public void handleMessage(Message msg) {
                        try {
                if(conn.getResponseCode() == 200){
                    uploadPhoto(image);
                }
                        } catch (IOException e) {e.printStackTrace();}
                    }
                };
                handler.sendEmptyMessage( what: 1);
            } catch (Exception e) {e.printStackTrace();}
        }
    });
    thread.start();
}
```

Рисунок 31. Головна функція створення offence

Підтягування з бази даних існуючих правопорушень:

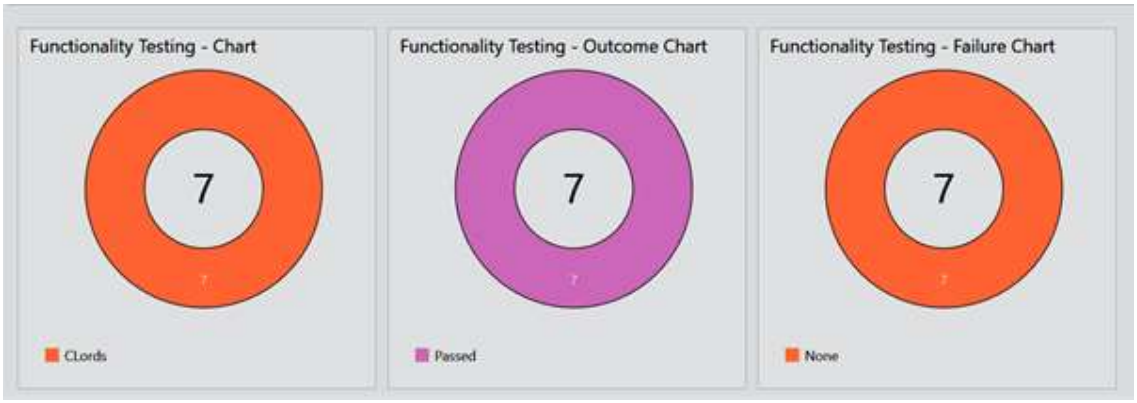
```
public void getCrimes(){
    Thread thread = new Thread(new Runnable() {
        @Override
        public void run() {
            try {
                BufferedReader br = null;
                URL url = new URL( spec: "https://fixit228.azurewebsites.net/api/fix-it/LawArticle");
                HttpURLConnection conn = (HttpURLConnection) url.openConnection();
                conn.setRequestMethod("GET");
                conn.setRequestProperty("Content-Type", "application/json;charset=UTF-8");
                conn.setRequestProperty("Authorization","Bearer "+authToken);
                br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
                JSONArray jsonArray = new JSONArray(br.readLine());
                try {
                    for (int i = 0; i < jsonArray.length(); i++) {
                        JSONObject jsonObj = jsonArray.getJSONObject(i);
                        crimes.add(jsonObj.getString( name: "name"));
                        crimesList.put(jsonObj.getString( name: "name"), jsonObj.getString( name: "id"));
                    }
                            Looper.prepare();
                            Handler handler = new Handler(Looper.getMainLooper()) {
                                @Override
                                public void handleMessage(Message msg) {
                                    ArrayAdapter crimesAdapter = new ArrayAdapter(getApplicationContext(),
                                            androidx.appcompat.R.layout.support_simple_spinner_dropdown_item, crimes);
                                    crimesAdapter.setDropDownViewResource(androidx.appcompat.R.layout.support_simple_spinner_dropdown_item);
                                    spinnerCrimes.setAdapter(crimesAdapter);
                                }
                            };
                            handler.sendEmptyMessage( what: 1);
                }catch (Exception ex){ex.printStackTrace();}
            }
            catch (Exception ex){ex.printStackTrace();}
        }
    });
    thread.start();
}
```

Рисунок 32. Підтягування з бази даних існуючих правопорушень

ТЕСТУВАННЯ

Було оформлено підписку на AzureDevopsTestPlans. Для функціонального тестування створено TestSuite, у якому створено Тест Кейси. У кожному Тест Кейсі є інструкція до виконання. За результатами роботи усі тести пройшли перевірку.

Нижче наведено діаграми виконання тестів:



Риснунок 33. Діаграми результатів тестування

Порядок виконання тестів:

За допомогою AzureDevops можливо відкрити поточний тест кейс та паралельно з перевіркою вносити Баг фікси або позначати, що тест пройдений:

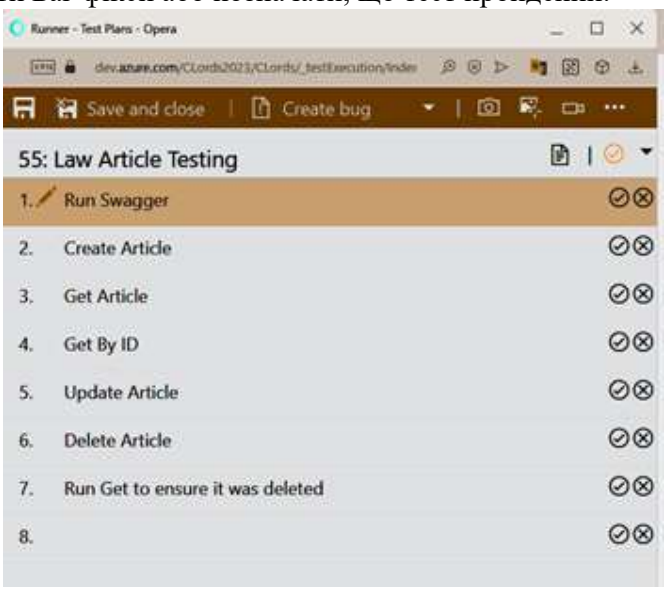


Рисунок34. Демонстрація відкриття тесту для внесення баг фіксів

У процесі функіонального тестування багів не було знайдено та тест кейси отримали статус Passed.

ВИСНОВКИ

Під час ведення проекту по розробці мобільного застосунку для ведення списку правопорушень з винесенням вироку "Фіксуй" , можна зробити наступні висновки:

1. Розробка мобільних застосунків стає все більш важливою через зростання кількості користувачів мобільних пристроїв, які використовують їх для отримання інформації та виконання завдань. Розробка мобільного застосунку "Фіксуй" для ведення списку правопорушень може полегшити роботу правоохоронних органів та покращити ефективність системи правосуддя.
2. Мобільний застосунок "Фіксуй" забезпечує зручний та швидкий доступ до особистого списку зафіксованих правопорушень, що дозволяє користувачам ефективно вносити їх до бази даних та забезпечує точність інформації. Крім того, додаток є доступним в будь-який час та місце, що забезпечує швидкий доступ до необхідної інформації та полегшує ведення списку правопорушень.
3. У зв'язку зі зростанням кількості кібератак та порушень безпеки, забезпечення безпеки даних користувачів та відповідність законодавству щодо захисту персональних даних стає все важливішою задачею. Розробникам додатку необхідно приділяти належну увагу заходам забезпечення безпеки та захисту даних користувачів.
4. Для ефективної роботи мобільного застосунку "Фіксуй", необхідно співпрацювати з правоохоронними органами та забезпечувати взаємодію між різними системами. Така співпраця забезпечить ефективну обробку та аналіз інформації щодо правопорушень та покращить роботу системи правосуддя.

Отже, мета досягнута. Метою розробленого мобільного застосунку було створення інструменту, який сприятиме зменшенню кількості аварій та ситуацій некоректної поведінки водіїв. Результати показують, що застосунок може відігравати важливу роль у попередженні порушень правил дорожнього руху, завдяки збору і аналізу даних про

порушення, а також інформуванню користувачів про правильні способи поведінки на дорозі. Крім цього, важливою функцією застосунку, котра покращить, зробить прозорою і спростить систему правосуддя стала збільшена прозорість роботи поліції. За допомогою застосунку, користувачі могли фіксувати порушення і передавати цю інформацію поліції безпосередньо через застосунок. Це сприяло швидшій реакції поліції на порушення, забезпеченню ефективного контролю та вжиттю заходів щодо зменшення кількості порушень. Покращення безпеки дорожнього руху: Розроблений мобільний застосунок вніс значний внесок у покращення безпеки дорожнього руху. Шляхом реєстрації порушень правил дорожнього руху та передачі цих даних національній поліції, застосунок допоміг ідентифікувати небезпечні ділянки доріг, повторювані порушення та місця зі збільшеною ймовірністю аварій. Ця інформація може бути використана для прийняття відповідних заходів, таких як розміщення додаткових знаків, встановлення камер спостереження або поліцейських патрулів, що сприятиме запобіганню потенційних аварій та зменшенню травматизму на дорогах.

Кожен з учасників проєкту набув нових вмінь та практичного досвіду:

Поліна: Отримала досвід роботи в команді, роботи з GitKraken, Azure, GitLab, розробки API.

Оля: Отримала досвід роботи в командному проекті, розробки бекенду, в тому числі авторизації та реєстрації з підтвердженням по пошті.

Яна: Отримала досвід командної роботи, досвід розробки веб апі на c#, досвід роботи з Azure. Навчилася деплоїти бази даних та проекти.

Діна: Удосконалила навички ментерування. Робота з специфікацією та верифікацією.

Максим: Удосконалив свої навички роботи в команді, навчився працювати зі специфікацією та поглибив свої знання у бізнес - аналітиці. Також удосконалив свої вміння у процесі роботи з документацією, навчився коректно формувати звіт.

Дмитро: Навчився організації та керуванню командою, написання звітності та аналітики.

Любомир: Управління командною розробкою, навички навчання інших людей (експерементування, застосування різних підходів), постановка задач + та надання допомоги по ним.

Михайло: Набрався корисного досвіду спільної розробки програмного забезпечення, покращив навички програмування у сфері мобільних технологій.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Офіційний сайт системи, яка може виявляти автомобільні аварії DashcamUA [Електронний ресурс]: https://dashcam.in.ua/
2. Офіційний сайт емулятора GradleBuildSystem, який використовується для імітації поведінки пристрою Android на комп'ютері [Електронний ресурс]: https://gradle.org/
3. Документація бібліотеки перевірки для .NET FluentValidation [Електронний ресурс]: https://docs.fluentvalidation.net/en/latest/
4. Офіційний сайт ресурсу SendGrid, який дозволяє доставляти електронну пошту через зовнішні сервери [Електронний ресурс]: https://sendgrid.com/
5. Проект Закону про внесення змін до деяких законодавчих актів України щодо посилення ролі суспільства у заходах контролю за безпекою дорожнього руху та дотриманням правил зупинки, стоянки, паркування транспортних засобів [Електронний ресурс]: http://w1.c1.rada.gov.ua/pls/zweb2/webproc4_1?pf3511=725

САЙТ ПОЦІНОВУВАЧІВ ВИНА "LWINE"

Роман Богдан, Олександр Грабар, Ксенія Грищенко, Наталія Захарчук, Тетяна Третяк, Антон Троценко, Максим Ошийко

ВСТУП

Розвиток інтернет-технологій та зростання популярності електронної комерції вимагають від компаній розвитку новітніх веб-додатків, що забезпечують конкурентоспроможність на ринку. Однією з найбільш актуальних задач, яка стоїть перед розробниками веб-додатків, є створення програмного забезпечення, що дозволяє користувачам зручно вибирати та відстежувати вина за допомогою винної картки.

Веб-додаток винної карти має на меті допомогти користувачам знайти вина, які вони шукають, на основі різноманітних критеріїв, таких як країна виробник, сорт винограду, рік врожаю тощо. Проєкт є дуже актуальним для любителів вина, оскільки він дозволить їм зручно та швидко знаходити необхідну інформацію про вина.

Методи дослідження, що використовуються для розробки веб-додатку винної карти, включають аналіз вимог користувачів, проектування баз даних, розробку функціональності та інтерфейсу користувача, тестування та підтримку програмного забезпечення.

Основними цілями розробки веб-додатку винної карти є:

- забезпечення користувачів зручним та швидким способом знаходження інформації про вина, які вони шукають;
- покращення конкурентоспроможності веб-додатку на ринку;
- підвищення рівня задоволення користувачів від використання веб-додатку.

Крім прикладних цілей, розробка веб-додатку винної карти має на меті також закріпити знання та вивчити нові технології розробки програмного забезпечення. Для досягнення цього мети необхідно:

- розробити структуру бази даних, яка дозволяє зберігати інформацію про різноманітні вина, їх характеристики, фото та опис;
- розробити функціональність, що дозволяє користувачам знаходити вина на основі різноманітних критеріїв та відстежувати їх у своїй винній картці;
- створити зручний та привабливий інтерфейс користувача;
- провести тестування веб-додатку та виправити можливі помилки;
- забезпечити підтримку веб-додатку та вирішення технічних проблем, що можуть виникнути під час експлуатації.

Розробка веб-додатку винної карти є важливим проєктом, який забезпечує зручний та швидкий спосіб знаходження інформації про вина, а також дозволяє покращити конкурентоспроможність на ринку електронної комерції. Крім того, розробка веб-додатку винної карти є вагомою задачею для фахівців з розробки програмного забезпечення, оскільки дозволяє закріпити та поглибити знання з технологій розробки веб-додатків.

ОГЛЯД НАЯВНИХ НА РИНКУ СИСТЕМ

На основі наданих даних можна виділити кілька конкурентів для додатку "Веб-додаток винної карти".

Першим конкурентом є додаток “Vinnie”, який також пропонує можливість вибору вина в залежності від нагоди та вида події. Також є можливість пошуку та відстеження історії вибору вина.

Другим конкурентом є додаток “hellovino”, який пропонує вибір вина в залежності від смаку та цінового діапазону, а також надає інформацію про кращі вина в залежності від типу та з чим їх найкраще пити.

Третім конкурентом є додаток “vinomojo”, який пропонує квіз для визначення підходящого вина, а також надає відео-огляди та можливість відправки вина поштою.

Четвертим конкурентом є додаток “vivino”, який має багато категорій для вибору вина, такі як країна виробництва, регіон, тип винограду та рейтинг користувачів.

Однією з переваг додатку "Веб-додаток винної карти" є можливість пізнавального вивчення нових видів вина та поради щодо найкращих магазинів з вином за оптимальними цінами, а також тест, щоб дізнатися, яке вино підходить для користувача. Додаток не пропонує деяких функцій конкурентів, проте наявність інтерактивної частини може стати привабливою для користувачів.

ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Технології розробки : React, Django, DRF; технології проектування: app.diagrams.net, figma; технології тестування та верифікації: unittest, pytest.

React є однією з найбільш популярних бібліотек JavaScript для розробки користувацьких інтерфейсів. Вона пропонує декларативний підхід до створення складних UI компонентів, що спрощує процес розробки та забезпечує високу ефективність.

Django — це високорівневий фреймворк Python для швидкої розробки веб-додатків. Він пропонує широкий набір готових модулів та бібліотек, що дозволяє швидко та легко розробляти веб-додатки зі стандартними функціональними можливостями.

DRF (Django REST Framework) — це фреймворк для розробки RESTful API на базі Django. Він пропонує широкий набір інструментів для створення інтерфейсів API та забезпечення їх безпеки та перевірки валідності даних.

Для проектування інтерфейсу додатку ми використовуємо декілька інструментів. Першим з них є app.diagrams.net, який дозволяє створювати діаграми і схеми різного рівня складності. Ми використовуємо цей інструмент для проектування бази даних, визначення архітектури додатку та розробки прототипів інтерфейсу.

Другим інструментом є Figma - інструмент для дизайну інтерфейсів, який ми використовуємо для розробки дизайну додатку. За допомогою Figma ми створюємо макети інтерфейсу, дизайн іконок та інші елементи, які згодом інтегруються в додаток.

Для тестування додатку ми використовуємо два інструменти - unittest. Unittest є стандартним модулем для тестування у Python, який дозволяє створювати автоматизовані тести для перевірки функціональності окремих модулів додатку.

ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

Призначення системи

Призначенням системи "Веб-додаток винної карти" є автоматизація процесу пошуку вин під потреби користувачів. Цей процес створений за допомогою збору актуальної інформації з декількох перевірених ресурсів , її постійне оновлення та фільтрація вин за різними категоріями .

Крім цього, в системі присутні додаткові можливості , які допомагають підібрати вино відповідно до події та їжі , переглянути інформацію щодо конкретного напою , отримати перелік актуальних цін на це вино в різних магазинах , додати його до "обраних" , а також побачити запропоноване "вино дня".

Таким чином ми намагалися створити максимально зручний ресурс відповідно до побажань користувачів та додати важливі інструменти , яких немає у наших конкурентів.

Створений бот, доступ до якого є через сайт, призначений також для пізнавальних цілей, допомагаючи в ігровому форматі дізнатися більше про конкретні вина та збільшуючи зацікавленість клієнта в продукті.

Робота передбачає:

- аналіз методів, методик і моделей, що застосовуються для збору та обробки інформації щодо доступних вин в магазинах, її оновлення та фільтрація , поширення в зручному для користувача інтерфейсі , а також допомоги у вдалому підборі продукту під вимоги користувачів;
- аналіз наявних програмних засобів в галузях, пов’язаних з поширенням інформації про вина, їх підбір та продаж;
- проєктування та програмну реалізацію "Веб-додатку винної карти" та телеграм бота;
- апробацію "Веб-додатку винної карти".

Цілі створення системи

"Веб-додаток винної карти" створюється з метою:

- забезпечення збору, обробки та аналізу актуальної інформації про доступні вина в Україні;
- забезпечення можливості фільтрації вин відповідно до різних параметрів;
- забезпечення допомоги в підборі вина під подію та потребу користувача;
- забезпечення можливості проаналізувати ціни на відповідні вина в різних магазинах;
-забезпечення можливості в ігровому форматі дізнатися більше про конкретні вина;
-забезпечення зручного для користувача інтерфейсу.

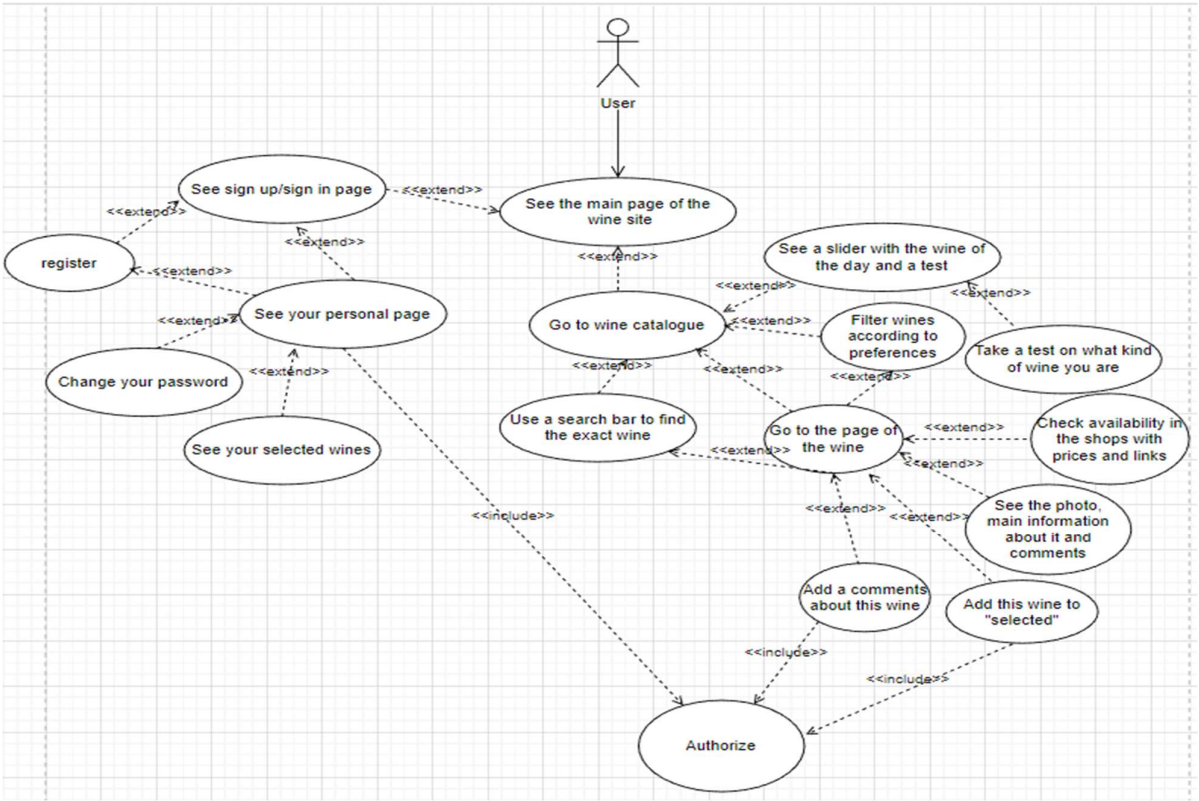


Рисунок 1. Use-Case diagram

Вимоги до системи

Вимоги до системи в цілому

Система "Веб-додаток винної карти" повинна бути простою та корисною в застосуванні. В Системі передбачається виділити наступні функціональні підсистеми:

- **адміністративна** — призначена для збереження, обробки та оновлення інформації , представленій на сайті та пов'язаною з даними користувачів , а також забезпечення функціонування веб-застосунку;
- **підсистема користувача —** призначена для забезпечення можливості авторизуватися в системі, переглядати та фільтрувати вина, а також зберігати їх та моніторити поточні ціни.

Вимоги до функцій, які виконуються системою

Підсистема адміністратора.

Таблиця 1. Перелік функцій, задач що підлягають автоматизації

Функція	**Задача**
Керувати роботою користувачів системи	редагування та видалення інформації про користувачів системи
	формування та візуалізація звітів про користувачів системи
Робота з інформацією, запропонованою на сайті	збір, обробка. сортування та оновлення інформації про вина
	збереження обраних користувачем вин
	надання можливості користувачам фільтрувати вина відповідно до параметрів
	пропонування "вина дня"
	надання доступу до тесту та телеграм боту

Підсистема користувача.

Таблиця 2. Перелік функцій, задач що підлягають автоматизації

Функція	**Задача**
Реєстрація	введення інформації про себе
	оновлення реєстрації
	зміна пароля для авторизації
	видалення інформації про себе
Робота з інформацією, запропонованою на сайті	фільтрація вин відповідно до бажаних параметрів
	збереження їх до обраних
	проходження тесту для підбору вин
	використання боту в цілях перевірки своїх знань щодо вин та ознайомлення з ними
	перегляд цін в різних магазинах на конкретні вина

Системні вимоги

Браузери: Сайт повинен коректно відображатися в інтернет-браузерах MS Internet Explorer 5.5 і вище, Mozilla 1.7 і вище, Opera 7.54 і вище.

На довільні некоректні дії користувача, пов’язані з введенням невірних даних, не заповненням обов’язкових полів введення в формах та інші, які можуть бути оброблені системою, генеруються відповідні повідомлення про помилки українською мовою, в межах загального дизайну сайта.

Операційна система: немає вимог щодо операційної системи, сучасні технології дають можливість уникати відмінностей різних ОС. Наприклад, на проекті працювали люди, які мали Windows та MacOS, при цьому жодного разу не виникло проблем.

Джерелом даних для Системи повинна бути інформаційна система. В проекті використовувалась велика кількість реальних даних, взятих за допомогою власних парсерів з інших сайтів. Тому для зберігання відносно великої кількості даних мало потужностей SQLite, які пропонуються в Django за замовчуванням. Було вирішено використовувати PostgreSQL СУБД. Для взаємодії з базою даних використовувався сервіс ElephantSQL, який надає достатньо багато можливостей навіть для безкоштовного плану.

ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

Логічна структура бази даних

Для нашої системи було вирішено використовувати PostgreSQL. Ця реляційна база даних є найбільш поширеною та чудово для проекту, крім того проект не потребував особливої бази даних.

Рисунок 2. Діаграма бази даних сайту "wine-map"

Оскільки, ми використовуємо django auth бібліотеку, то системою автоматично були створені таблички для авторизації. Також django створив деякі службові таблиці.

auth_group, auth_group_permissions, auth_permission, auth_user, auth_user_groups, auth_user_user_permissions - перелік таблиць, відповідальних ха авторизацію користувача.

django_admin_log, django_content_type, django_migrations, django_rest_passwordreset_resetpasswordtoken, django_session - перелік службових таблиць, створених django.

Таблиця 3. Перелік таблиць бази даних

Номер	**Таблиця**	**Опис**
1	wine_map_brand	Таблиця для збереження брендів вина
2	wine_map_country	Таблиця для збереження країн вина
3	wine_map_gamestate	Таблиця з даними ігрового бота
4	wine_map_quizanswer	Таблиця з відповідями користовачів до вікторини
5	wine_map_quizanswer_results	Таблиця з правильними відповідями до вікторини
6	wine_map_quizquestion	Таблиця питань вікторини
7	wine_map_wine	Таблиця де зберігаються всі вина
8	wine_map_in_favourites_of	Таблиця для зберігання улюблених вин
9	wine_map_additionalinfo	Таблиця з додатковою інформацією вина
10	wine_map_wineoftheday	Таблиця, що зберігає вино дня по даті
11	wine_map_comment	Таблиця для збереження комментарів

Опис таблиць.

Таблиця 5. Таблиця wine_map_brand

Атрибут	**Тип**	**Опис**
id	bigint	Ідентифікатор
name	varchar	Назва бренду

Таблиця 6. Таблиця wine_map_country

Атрибут	**Тип**	**Опис**
id	bigint	Ідентифікатор
name	varchar	Назва країни

Таблиця 7. Таблиця wine_map_gamestate

Атрибут	**Тип**	**Опис**
id	bigint	Ідентифікатор
user_id	bigint	Назва бренду
answers	varchar []	Масив відповідей
questions	jsonb []	Запитання, поставлені користувачу
total_questions	integer	Загальна кількість питань
points	integer	Кількість набраних очків
help_used	boolean	Використання підказок

Таблиця 8. Таблиця wine_map_quizanswer

Атрибут	**Тип**	**Опис**
id	bigint	Ідентифікатор

text	varchar	Текст запитання
for_question_id	bigint	Ідентифікатор запитання
next_question_id	bigint	Ідентифікатор наступного запитання

Таблиця 9. Таблиця wine_map_quizanswer_result

Атрибут	**Тип**	**Опис**
id	bigint	Ідентифікатор
quizanswer_id	bigint	Ідентифікатор відповіді
wine_id	bigint	Ідентифікатор вина

Таблиця 10. Таблиця wine_map_quizquestion

Атрибут	**Тип**	**Опис**
id	bigint	Ідентифікатор
text	varchar	Текст запитання
first	boolean	Перше запитання

Таблиця 11. Таблиця wine_map_wine

Атрибут	**Тип**	**Опис**
id	bigint	Ідентифікатор
name	varchar	Назва вина
wine_type	varchar	Тип вина
percent_of_alcohol	double	Відсоток алкоголю
brand_id	bigint	Ідентифікатор бренду
country_id	bigint	Ідентифікатор країни
pairs_with	varchar	Поєднується з
sweetness	varchar	Рівень солодкості
tastes	varchar	Смак вина
image_url	varchar	Посилання на фото
region	varchar	Регіон вина

Таблиця 12. Таблиця wine_map_in_favourites_of

Атрибут	**Тип**	**Опис**
id	bigint	Ідентифікатор
wine_id	bigint	Ідентифікатор вина
user_id	bigint	Ідентифікатор користувача

Таблиця 13. Таблиця wine_map_wineadditionalinfo

Атрибут	**Тип**	**Опис**
id	bigint	Ідентифікатор
name	varchar	Назва вина
price	double	Ціна вина
url	varchar	Посилання на вино
wine_id	bigint	Ідентифікатор вина

Таблиця 14. Таблиця wine_map_wineoftheday

Атрибут	**Тип**	**Опис**
id	bigint	Ідентифікатор
wine_id	bigint	Ідентифікатор вина
date	date	Дата, коли вино було вином дня

Таблиця 15. Таблиця wine_map_comment

Атрибут	**Тип**	**Опис**
id	bigint	Ідентифікатор
timestamp	timestamp	Час написання коментаря
content	text	Контент коментаря
autor_id	bigint	Ідентифікатор автора
wine_id	bigint	Ідентифікатор вина

РЕАЛІЗАЦІЯ СИСТЕМИ

Backend

У процесі розробки було вирішено використовувати мову Python для реалізації функціональної частини. Для нашої тему підходив Django framework. Model-template-view допоміг нам швидко та без лишніх проблем реалізувати нашу задачу, крім того купа бібліотек значно пришвидшила процес розробки. Для винної карти потрібно здебільшого повертати модель вина та показувати додаткові параметри, тому для економії часу обрали даний framework.

Щоб зчитати всі вина, було написано декілька парсерів, що зчитують дані з магазинів та заносять ці вина до бази даних. Парсер зчитує дані з веб застосунку “Winetime”, бере такі поля, як: посилання на фото, країна, бренд і так далі.

```
class WinetimeSpider(scrapy.Spider):
    name = "winetime"
    allowed_domains = ["winetime.com.ua"]
    start_urls = ["https://winetime.com.ua/ua/wine"]

    antos07
    def parse(self, response: Response):
        self.logger.info(f"Parsing catalog page {response.url}")

        wine_urls = response.xpath('//div[@class="products-main-slider-item"]'
                                   '/div[@class="products-main-slider-item-wrapper"]'
                                   '/div/a')
        yield from response.follow_all(wine_urls, callback=self.parse_wine_info)

        if wine_urls.get():
            yield self.follow_to_next_catalog_page(response)
```

Рисунок 3. Приклад парсеру

База даних та її параметри описані вище, у пункті чотири. Тут я додам, що для роботи з цією базою даних ми використовували наші власні міграції, для таблиці вин наприклад та вбудовані міграції з бібліотек Django - користувач та авторизація.

```
migrations.CreateModel(
    name="Wine",
    fields=[
        (
            "id",
            models.BigAutoField(
                auto_created=True,
                primary_key=True,
                serialize=False,
                verbose_name="ID",
            ),
        ),
        ("name", models.CharField(max_length=255)),
        (
            "wine_type",
            models.CharField(
                choices=[("red", "Red"), ("white", "White"), ("rose", "Rosé")],
                max_length=255,
            ),
        ),
        ("year", models.IntegerField()),
        ("percent_of_alcohol", models.FloatField()),
        (
            "brand",
            models.ForeignKey(
                on_delete=django.db.models.deletion.CASCADE, to="wine_map.brand"
            ),
        ),
```

Рисунок 4. Приклад міграцій

Для передачі даних на frontend ми використовуємо Rest API. Django Rest API бібліотека дозволила швидко та комфортно реалізувати всі наші ендпоінти. Усе розділено по категоріям і кожна частина проекту має свою власну директорію.

```
api = [
    path("wine/", include("wine_map.urls")),
    path("", include("djoser.urls")),
]

urlpatterns = [
    path("admin/", admin.site.urls),
    path("api/", include(api)),
    path("auth/", include("django.contrib.auth.urls")),
    path("auth/", include("authentication.urls")),
    path("api/schema/", SpectacularAPIView.as_view(), name="schema"),
    path(
        "api/doc/swagger/",
        SpectacularSwaggerView.as_view(url_name="schema"),
        name="swagger-ui",
    ),
]
```

Рисунок 5. Приклад шляхів для авторизації

Фрагмент коду на малюнку вище приймає базові шаблони шляху та переадресовує їх.

```python
comments = [
    path("", CommentListView.as_view(), name="comment-list"),
    path("create/", CommentCreateView.as_view(), name="comment-create"),
    path("delete/<int:id>/", CommentDeleteView.as_view(), name="comment-delete"),
    path("update/<int:id>/", CommentUpdateView.as_view(), name="comment-update"),
]

quiz = [
    path("start/", QuizStartView.as_view(), name="quiz-start"),
    path("questions/<int:pk>/", QuizQuestionView.as_view(), name="quiz-question")
]

favourites = [
    path("<int:wine_id>/", FavouriteWinesUpdateView.as_view(),
         name="update_favourites"),
    path("clear/", FavouriteWinesClearView.as_view(), name="clear_favourites"),
    path("", FavouriteWines.as_view(), name="retrieve_wines")
]

urlpatterns = [
    path("", WineListView.as_view(), name="wine-list"),
    path("<int:id>/", WineDetailView.as_view(), name="wine-detail"),
    path("wine-of-the-day/", WineOfTheDayView.as_view(), name="wine-of-the-day"),
    path("favourites/", include(favourites)),
    path("categories/", CategoriesView.as_view(), name="categories"),
    path("<int:wine_id>/comments/", include(comments)),
    path("<int:wine_id>/prices/", WineInShopsView.as_view(), name="wine-prices"),
    path("quiz/", include(quiz))
]
```

Рисунок 6. Приклад парсеру

Файл “wine_map/urls.py” відповідає за зберігання всіх можливих ендпоінтів для вина. Саме на цьому рівні ми викликаємо view данного ендпоінту.

```python
class WineOfTheDayView(generics.RetrieveAPIView):
    permission_classes = [AllowAny]

    romchik
    def get(self, request: Request, *args: tuple, **kwargs: dict) -> Response:
        today = date.today()
        try:
            wine = WineOfTheDay.objects.get(date=today).wine
        except WineOfTheDay.DoesNotExist:
            wine = Wine.objects.order_by("?").first()
            WineOfTheDay.objects.create(wine=wine, date=today)
        serializer = WineSerializer(wine)
        return Response(serializer.data)
```

Рисунок 7. Приклад view

Даний фрагмент коду містить логіку для отримання вина дня. В цілому, все наші ендпоінти слідують основному паттерну джанго. Після того як ми отримали реквест та перейшли до view частини - ми починаємо працювати з базої даних та серіалайзерами. На цьому прикладі видно, що спочатку ми ідентифікуємо вино дня за допомогою моделі вина та моделі вино дня, а після цього серіалізуємо результат.

```
class WineSerializer(serializers.ModelSerializer):
    brand = BrandSerializer()
    country = CountrySerializer()

    Sasha Hrabar
    class Meta:
        model = Wine
        fields = "__all__"
```

Рисунок 8. Приклад серіалайзеру

Потрібно зазначити, що наша програма включає в себе ще декілька додаткових частин. Одна з них - це вікторина. Коже користувач здатний пройти тест. Для реалізації цієї частини зі сторони ми маємо два класа типу “View”, які відповідають за початок вікторини та генерацію запитань.

```
class QuizStartView(generics.RetrieveAPIView):
    queryset = QuizQuestion.objects.prefetch_related("answers")
    serializer_class = QuizQuestionSerializer
    permission_classes = [AllowAny]

    antos07
    def get_object(self) -> QuizQuestion:
        try:
            question = self.get_queryset().get(first=True)
        except QuizQuestion.DoesNotExist:
            raise exceptions.NotFound
        return question


2 usages  antos07
class QuizQuestionView(generics.RetrieveAPIView):
    queryset = QuizQuestion.objects.prefetch_related("answers")
    serializer_class = QuizQuestionSerializer
    permission_classes = [AllowAny]
```

Рисунок 9. Приклад view

Останнім, що потрібно описати - є телеграм-бот. Він дуже схожий на вікторину, вбудовано в нашому веб застосунку, але має зовсім іншу реалізацію. Для реалізації цього боту - ми використовували бібліотеку telegram, яка дозволила описати всі можливі сторони нашого бота.

```python
def game(update: Update, context: CallbackContext, user_id: int = None) -> None:
    if user_id is None:
        user_id = update.effective_user.id
    conn = psycopg2.connect(
        host=os.getenv("DB_HOST"),
        database=os.getenv("DB_NAME"),
        user=os.getenv("DB_USER"),
        password=os.getenv("DB_PASSWORD"),
    )
    cur = conn.cursor()
    cur.execute(
        """
        SELECT name, country_id, sweetness, percent_of_alcohol, wine_type, image_url
        FROM wine_map_wine
        ORDER BY RANDOM()
        LIMIT 1
    """
    )
    wine_data = cur.fetchone()
```

Рисунок 10. Приклад функції для роботи з телеграм-ботом

Frontend

Основними технологіями, які були використані для розробки клієнтської частини, є JavaScript бібліотека React та Ant Design. React є оптимальним вибором для нашого проекту, оскільки вона має просту та логічну архітектуру, а також за допомогою Virtual DOM дозволяє створювати швидкі та ефективні веб-інтерфейси.

Ant Design - це бібліотека компонентів для розробки інтерфейсів користувача на основі React. Вона містить набір готових до використання компонентів, таких як кнопки, форми, таблиці, меню, модальні вікна та інші, які допомагають значно спростити процес розробки веб-інтерфейсу. Бібліотека має велику кількість налаштувань, що дозволяє змінювати стилі компонентів за потребами нашого проекту.

```tsx
export default function AntdConfigProvider({ children }: React.PropsWithChildren) {
  return (
    <ConfigProvider
      theme={{
        token: {
          colorPrimary: '#C92F49',
          borderRadius: 30,
          colorBgLayout: '#FFFFFF',
          colorLink: '#000000',
          colorLinkHover: '#C92F49',
          colorLinkActive: '#D65465'
        },
      }}
    >
      {children}
    </ConfigProvider>
  )
}
```

Рисунок 11. Приклад зміни стилів

Для реалізації маршрутизації було використано бібліотеку react-router, яка дозволяє описати сторінки та шляхи до них у виді React компонентів.

```
const router = createBrowserRouter(
  createRoutesFromElements(
    <Route path="/" element={<App />}>
      <Route index element={<Home />} />
      <Route path="login" element={<Login />} />
      <Route path="register" element={<Registration />} />
      <Route path="reset-password" element={<NewPassword />} />
      <Route path="user-profile" element={<UserProfile />} />
      <Route path="wines/:id" element={<WinePage />} />
      <Route path="favourites" element={<FavouritesPage />} />
      <Route path="wines" element={<Catalog />} />
      <Route path="quiz" element={<Quiz />} />
    </Route>
  )
);
```

Рисунок 12. Приклад роутерів

Для спілкування з сервером було створено сервіс з усіма необхідними функціями.

Оскільки наша система може взаємодіяти як з авторизованим, так і з не авторизованим користувачем, то функції поділяються на ті, що додають інформацію в запит до сервера про авторизованого користувача або не додають відповідно.

```
const getRequestWithoutAuthorization = async (URL: string, config?: AxiosRequestConfig) => {
  return axiosClient.get(URL, config).then((response) => response);
};

const getRequest = async (URL: string) => {
  return axiosClient
    .get(URL, {
      headers: {
        Authorization: `Bearer ${localStorage.getItem('access')}`,
      },
    })
    .then((response) => response);
};

const postRequestWithoutAthorization = async (URL: string, payload: Object) => {
  return axiosClient.post(URL, payload).then((response) => response);
};

const postRequest = async (URL: string, payload: Object) => {
  return axiosClient
    .post(URL, payload, {
      headers: {
        Authorization: `Bearer ${localStorage.getItem('access')}`,
      },
    })
    .then((response) => response);
};
```

Рисунок 13. Приклад функції з обмеженням доступу для неавторизованих користувачів

Наприклад, сервіс вище використовується на сторінці каталогу для отримання інформації про вина відповідно до заданих значень фільтрів.

```
const fetchWines = async (offset: number) => {
  setIsLoading(true);
  try {
    const params = {
      offset,
      limit,
      ...filter,
    };

    const { data } = await getRequestWithoutAuthorization(`/api/wine/`, {
      params,
    });

    return { wines: data.results, count: data.count };
  } catch (error) {
    if (axios.isAxiosError(error)) {
      const err = error as AxiosError<{ detail: string }>;
      api.error({
        message: (err.response?.data.detail) || 'Помилка',
        placement: 'top',
      });
    }
  } finally {
    setIsLoading(false);
  }
};
```

Рисунок 14. Приклад функції з фільтрами

Можна бачити, що також присутня обробка помилок, якщо відповіді з сервера не буде або вона буде негативною, то буде показано відповідне повідомлення з описом помилки. У випадку позитивного результату вина будуть збережені в стан компоненти і відображені на сторінці.

```
return (
  <>
    {contextHolder}
    <Row gutter={[35, 40]} className="wine-list">
      {wines.map((wine) => (
        <Col key={wine.id} span={8}>
          <WineCard
            wine={wine}
            isFavourite={favourites.map(x => x.id).includes(wine.id)}
            reloadFavourites={getFavourites}
          />
        </Col>
      ))}
      {isLoading ? (
        <Col span={24}>
          <Row justify="center">
            <Spin indicator={<LoadingOutlined style={{ fontSize: 40 }} spin />} />
          </Row>
        </Col>
      ) : null}
      {(wines.length < count) ? (
        <Col span={24}>
          <Row align="middle" justify="center">
            <Button
              type="primary"
              size="large"
              style={{ fontSize: '18px' }}
              onClick={loadMore}
            >
              Показати більше
            </Button>
          </Row>
        </Col>
      ) : null}
      {wines.length === 0 && !isLoading ? (
        <div className="no-wines">Вина не знайдені</div>
      ) : null}
    </Row>
  </>
```

Рисунок 15. Приклад компоненти каталогу

Для зручності пошуку вина на сторінці каталогу також було використано збеження значень фільтрів в browser local storage. За допомогою цього механізму, якщо користувач перейде на іншу сторінку або перезавантажить її, то повернувшись назад на сторінку каталогу його фільтри будуть збережені.

```
export default function useLocalStorage<T>(initialValue: T, key: string): [T, Dispatch<SetStateAction<T>>] {
  const getValue = () => {
    const storage = localStorage.getItem(key);

    if (storage) {
      return JSON.parse(storage);
    }

    return initialValue;
  };

  const [value, setValue] = useState<T>(getValue);

  useEffect(() => {
    localStorage.setItem(key, JSON.stringify(value));
  }, [value, key]);

  return [value, setValue];
}
```

Рисунок 16. Приклад використання local storage

Для реалізації форм було використано компоненту “Form” з Ant Design, яка дозволяє зручно та гнучко керувати як станом форми так і її відображенням на веб-інтерфейсі.

```
return (
  <Row align="middle" justify="center" className="content">
    <Col span={10}>
      <div className="title">Зміна паролю</div>      kseniia, last month • new
      <Form
        layout={'vertical'}
        form={loginForm}
        initialValues={{ layout: 'vertical' }}
        onFinish={changePassword}
        onInput={hideAlert}
      >
        <Form.Item name="password" rules={TextRules}>
          <Input.Password placeholder="Пароль" />
        </Form.Item>
        {!isAlertVisible && (
          <Form.Item>
            <Button type="primary" size="large" block htmlType="submit">
              Змінити
            </Button>
          </Form.Item>
        )}
      </Form>
      {isAlertVisible && (
        <div className="response-block">
          <Alert
            message={errorMessage || 'Пароль успішно змінено'}
            type={errorMessage ? 'error' : 'success'}
            showIcon
          />
          {!errorMessage && (
            <Link to="/login">
              <Button type="primary" size="large" block className="login-btn">
                Увійти
              </Button>
            </Link>
          )}
        </div>
      )}
    </Col>
  </Row>
```

Рисунок 17. Приклад використання форми

ТЕСТУВАННЯ

В цьому розділі ми розглянемо результати тестування, виконаних на проекті. Наведено детальний огляд процесу тестування, від підготовки та планування до аналізу результатів. Також ми розглянемо, як отримані результати впливають на проект як в цілому, так і на рівень його придатності.

Етап підготовки почався з порівняння технологій для тестування, які можуть забезпечити повноцінне тестування бекенд-частини. Оскільки наше api написано на мові програмування Python, то розглядались наступні варіанти:

1) Unittest – це модульне тестування для бекенду на Python. Це стандартна вбудована бібліотека Python, тому доступ до неї є завжди, без встановлення додаткових пакетів. За допомогою Unittest можна запускати тести з різними об'єктами і параметрами, а також визначати пройдені та непройдені тести.

Недоліки: не має зовнішніх бібліотек для автоматизації, відсутність автоматизації та неможливість запускати всі тести відразу, лише за допомогою команд.

2) Pytest – це популярна бібліотека тестування для бекенду на Python. Ця бібліотека дозволяє писати та запускати тести з простим API. Pytest має багато плагінів та зовнішніх бібліотек для автоматизації, що дозволяє автоматизувати тести. Також вона має вбудовані інструменти для обробки виключень та завантажень.

Недоліки: потребує більше налаштувань та завантажень.

Зваживши всі недоліки та переваги, було обрано unittest як основний інструмент для тестування.

Основною метою тестування було прописати поведінку відпрацювання api та зафіксувати яким чином мають відбутися або не відбутися зміни в базі даних. Також, тести прописані для відслідковування рівнів прав та доступів для авторизованих та неавторизованих користувачів.

Додатково було додано запуск тестів при кожному створенні Pull Requestів. Таким чином одразу фіксується чи дані зміни не порушують логіку виконання коду.

Отже, перейдемо до детального розгляду написання тестів. Було протестовано кожен модуль. Для дотримання DRY(don’t repeat yourself) принципу створення екземплярів класу було винесено в окремий файл, який знаходиться на рівні з конфігураціями та налаштуваннями бекендівської частини проекту. Даний файл містить функції, які створюють екземпляр класу з якимись дефолтними значення. Звичайно, додано можливість змінювати параметри при створення об’єкту на ті, що передаються у функцію при виклику. Всі ці функції знаходяться у файлі sample_data_functions.py

Ось декілька прикладів функцій з цього файлу з поясненням:

1) Функція створення користувача.

За замовчуванням при виклику цієї функції створюється користувач з параметрами, які вказані всередині словника defaults. Якщо при виклику функції передати іменовані параметри, наприклад sample_user(first_name=“Ivan”), то ця функція замінить ім’я користувача на передане. Це приклад функції, яка створює вбудовану в Django модуль користувача.

```python
def sample_user(**params) -> get_user_model():
    defaults = {
        "first_name": "first_name",
        "last_name": "last_name",
        "username": "username",
        "email": "test@test.com",
        "password": "testpassword123",
        "is_active": True,
    }
    defaults.update(params)

    return get_user_model().objects.create_user(**defaults)
```

Рисунок 18. Функція створення користувача для тестів

2) Функція створення вина.

Особливістю цієї функції, на відміну від попередньої, є те, що тут створюється екземпляр нашої кастомної моделі, яка всередині має зв'язки типу 1-to-n.

```python
def sample_wine(brand: Brand, country: Country, **params) -> Wine:
    defaults = {
        "name": "Test wine",
        "wine_type": "red",
        "sweetness": "sweet",
        "image_url": "https://test-image.com",
    }
    defaults.update(params)

    return Wine.objects.create(country=country, brand=brand, **defaults)
```

Рисунок 19. Функція створення екземпляру моделі зі зв'язками

На доданому зображені можна побачити, що для створення даного екземпляру мають додатково передаватись екземпляри зв'язних таблиць.

Перейдемо до розгляду реалізації самих тестів. Розглянемо два варіанти.

1) Тести ендпоінтів, до яких мають доступ усі користувачі.

Прикладом таких тестів є сторінка каталогу, оскільки переглядати її можуть всі. Було розглянуто випадки, що ендпоінт має коректно відпрацьовувати коли в базі даних немає вин, та коли в каталозі є вина. Також, йде перевірка на створення пагінації.

```python
class WineListApiViewTest(TestCase):
    def setUp(self) -> None:
        self.user = sample_user()
        self.client = APIClient()
        self.client.force_authenticate(self.user)

    def test_get_list_of_wines_when_there_is_no_wines(self) -> None:
        wines = Wine.objects.all()
        serializer = WineSerializer(wines, many=True)
        response = self.client.get(WINE_LIST_URL)

        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertEqual(response.data["count"], 0)
        self.assertEqual(response.data["results"], serializer.data)

    def test_get_list_of_wines_with_wines(self) -> None:
        self.country = sample_country()
        self.brand = sample_brand()
        sample_wine(country=self.country, brand=self.brand)
        sample_wine(name="New test wine", country=self.country, brand=self.brand)

        wines = Wine.objects.all()
        serializer = WineSerializer(wines, many=True)
        response = self.client.get(WINE_LIST_URL)

        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertEqual(response.data["count"], 2)
        self.assertEqual(response.data["results"], serializer.data)
```

Рисунок 20. Тестування функціоналу, який доступний для неавторизованих користувачів

2) Тести ендпоінтів, до яких мають доступ лише авторизовані користувачі.

Прикладом таких ендпоінтів є ендпоінт створення коментарю. Коментар може створювати в нашій системі виключно авторизований користувач. Тому тут потрібно було створити два класи з тестами. Один клас, де неавторизований користувач намагається створити коментар до вина:

```python
class UnauthenticatedCommentCreationViewSetTests(TestCase):
    def setUp(self) -> None:
        self.client = APIClient()

    def test_auth_required(self) -> None:
        self.country = sample_country()
        self.brand = sample_brand()
        self.wine = sample_wine(country=self.country, brand=self.brand)
        response = self.client.post(
            wine_comments_create_page_url(self.wine.id), data={"content": "New comment"}
        )

        self.assertEqual(response.status_code, status.HTTP_401_UNAUTHORIZED)
```

Рисунок 21. Тестування функціоналу, який доступний лише для авторизованих користувачів

А також другий приклад, де авторизований користувач створює коментар, і він додається до бази даних.

```python
class CommentCreateApiViewTest(TestCase):
    def setUp(self) -> None:
        self.user = sample_user()
        self.country = sample_country()
        self.brand = sample_brand()
        self.wine = sample_wine(country=self.country, brand=self.brand)

        self.client = APIClient()
        self.client.force_authenticate(self.user)

    def test_success_comment_creation(self):
        response = self.client.post(
            wine_comments_create_page_url(self.wine.id), data={"content": "New comment"}
        )
        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        self.assertEqual(Comment.objects.count(), 1)

    def test_success_comments_creation(self):
        response = self.client.post(
            wine_comments_create_page_url(self.wine.id), data={"content": "New comment"}
        )
        response_second = self.client.post(
            wine_comments_create_page_url(self.wine.id),
            data={"content": "One more comment"},
        )

        self.assertEqual(response.status_code, status.HTTP_201_CREATED)
        self.assertEqual(response_second.status_code, status.HTTP_201_CREATED)
        self.assertEqual(Comment.objects.count(), 2)
```

Рисунок 22. Тестування функціоналу, який доступний для неавторизованих користувачів

Тепер покажемо загальний результати виконання тестів:

```
(venv) ➜  backend git:(main) ✗ python manage.py test
Found 45 test(s).
Creating test database for alias 'default'...
System check identified some issues:

WARNINGS:
wine_map.WineOfTheDay.date: (fields.W161) Fixed default value provided.
        HINT: It seems you set a fixed date / time / datetime value as default for this field. This may not be what you want. If you want to have the current date as default, use `d
jango.utils.timezone.now`

System check identified 1 issue (0 silenced).
.............................................
----------------------------------------------------------------------
Ran 45 tests in 3.059s

OK
Destroying test database for alias 'default'...
(venv) ➜  backend git:(main) ✗
```

Рисунок 23. Результат виконання тестів

Отже, тестування допомагає забезпечити більш високу якість коду проекту. За допомогою тестування можна виявити помилки, знаходити проблеми та зробити програму більш стабільною. Також тестування допомагає відслідковувати якість програмного забезпечення. Відповідно до цього, цей процес допомагає підвищити рівень довіри і задоволення користувачів.

ВИКОРИСТАННЯ СИСТЕМИ

1. При переході на сайт за посиланням. можна переглянути “Вино дня” або пройти тест для підбору вина. Щоб його знайти треба на слайдері натиснути на стрілку вправо.

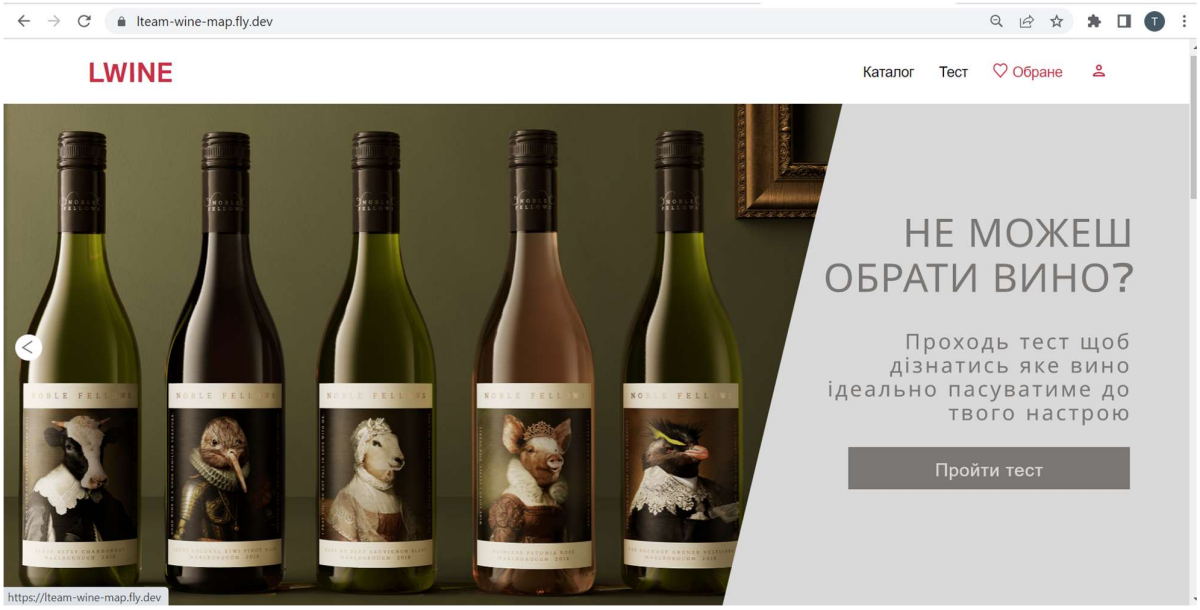


Рисунок 24. Слайдер з головної сторінки

2. При натисканні на “Каталог” в правому верхньому куті, відкривається сторінка каталогу , де можна переглядати та фільтрувати наявні вина

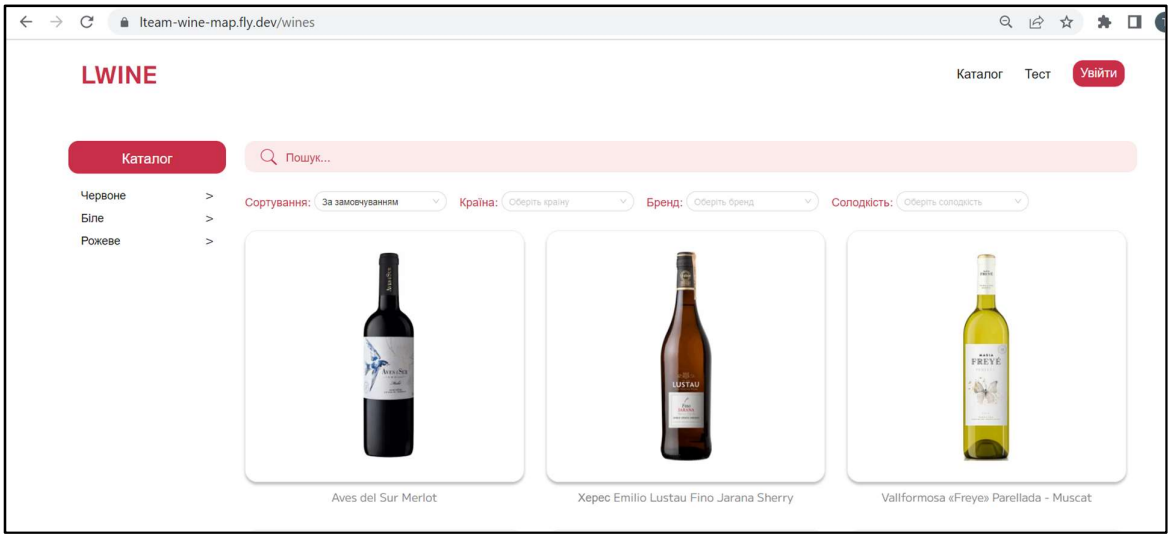


Рисунок 25. Каталог

3. В рядку сортування , натиснувши на стрілку вниз, можна обрати відповідні параметри та додати фільтрацію за ними.

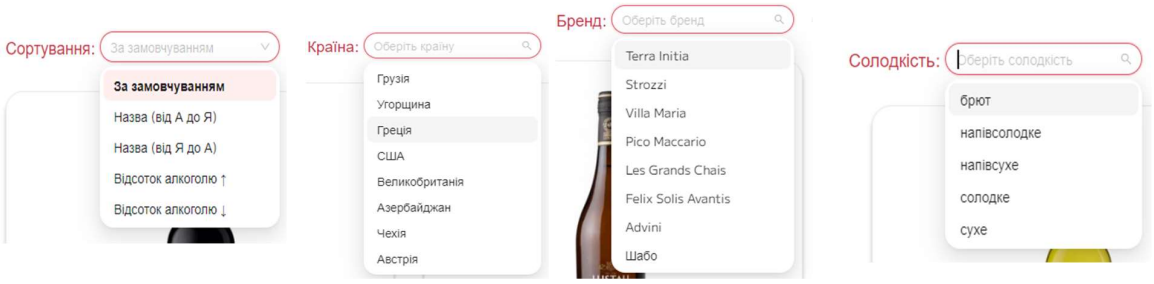


Рисунок 26. Застосування фільтрів

Як результат отримаємо відфільтровані за побажаннями вина:

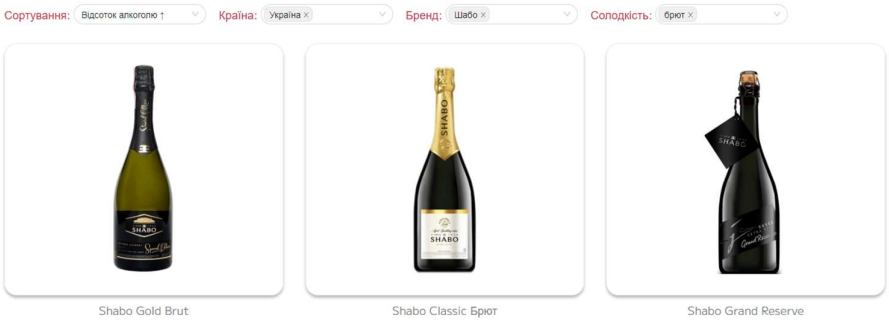


Рисунок 27. Результат застосування фільтрів

4. Також під каталогом можна відсортувати запропоновані вина за їх кольором
5.

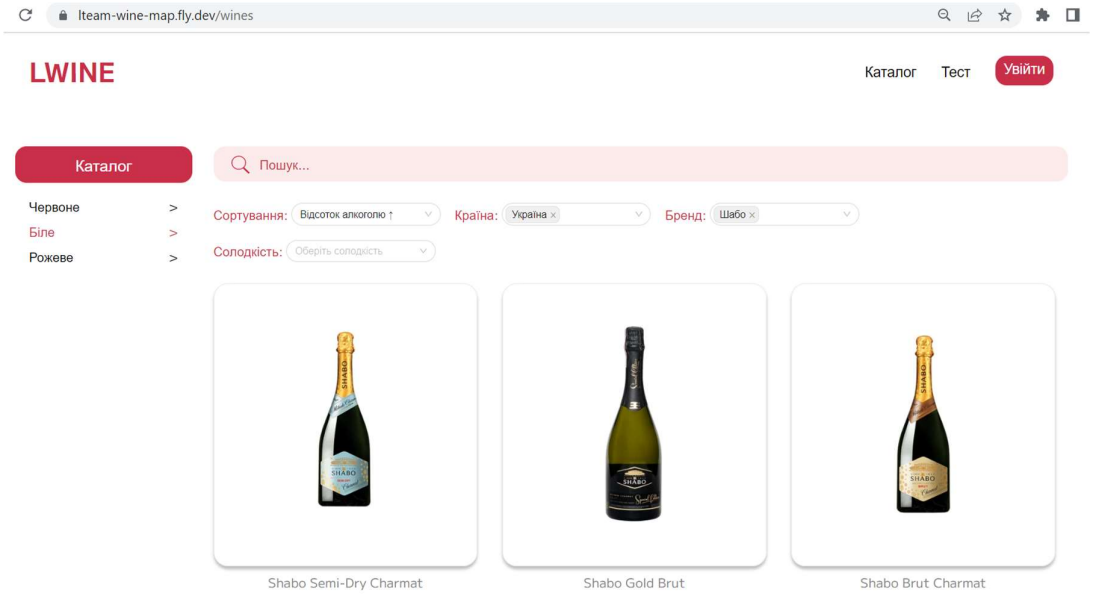


Рисунок 28. Сортування каталогу вин

6. Натиснувши на обране вино, можна з легкістю перейти на його сторінку ⇨

Рисунок 29. Детальна сторінка вина

На сторінці описані важливі параметри вина, а також ціни та наявність в магазинах.

7. Є можливість залишати коментарі, але для цього треба увійти.

Натиснувши на підсвічене слово “Увійдіть”, можна перейти на сторінку з входом у свій аккаунт або реєстрацію.

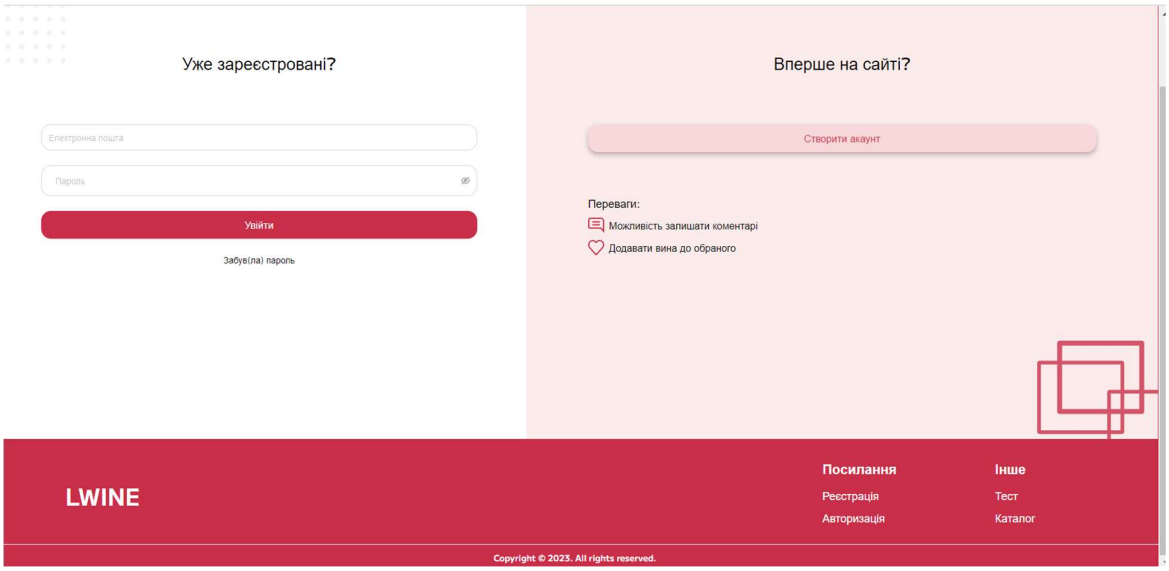


Рисунок 30. Авторизація

Якщо ви вперше на сайті, заповнивши декілька простих полів, можна швидко створити акаунт

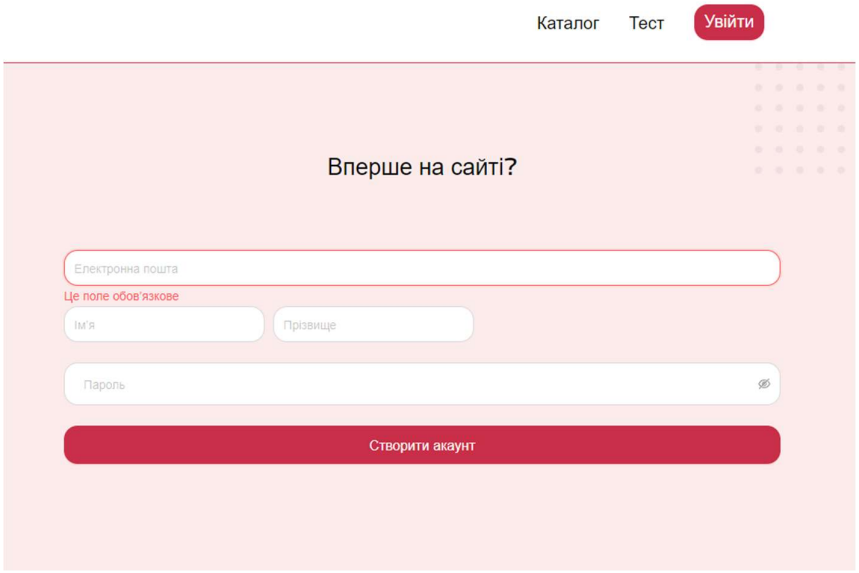


Рисунок 31. Реєстрація

8. Якщо ж ви вже зареєстровані, заповнивши поля електронної пошти та паролю, можна увійти в свій кабінет. У випадку, якщо ви забули пароль, його можна відновити за допомогою електронної пошти, натиснувши “Забули пароль”.

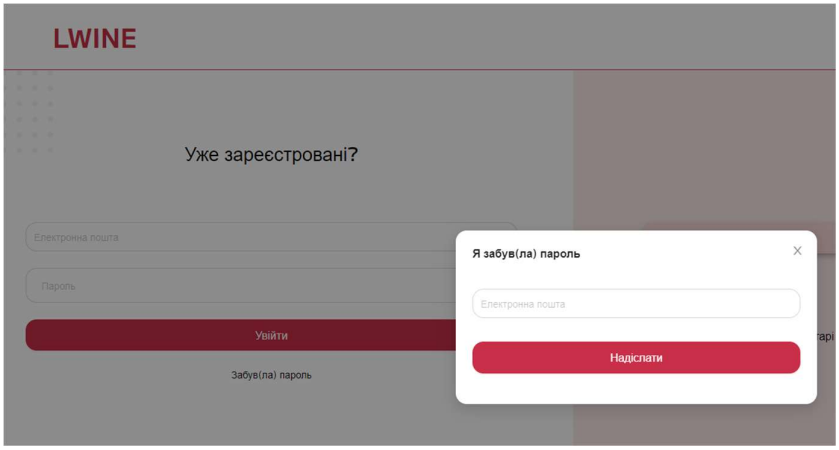


Рисунок 32. Скидання паролю

9. Також відкривається можливість додати вина до обраних. Для цього потрібно натиснути на серце в правому верхньому куті на фотографії вина.

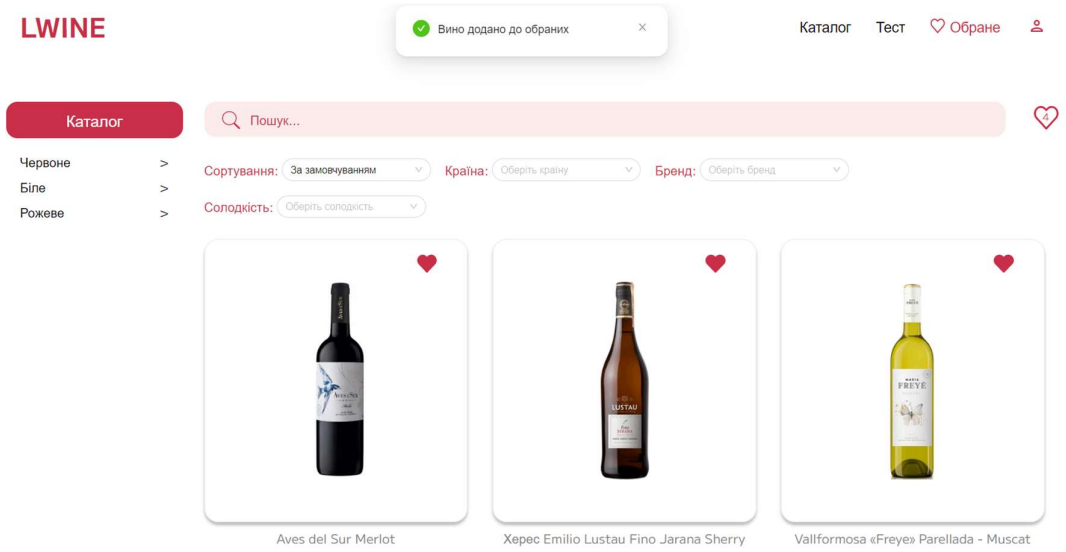


Рисунок 33. Додавання вина до “Обраних вин”

10. Переглянути обрані вина можна на сторінці “Обране”. посилання на яку знаходиться в правому верхньому куті сайту. Також цю сторінку можна очистити , натиснувши кнопку “очистити все” в правому нижньому куті.

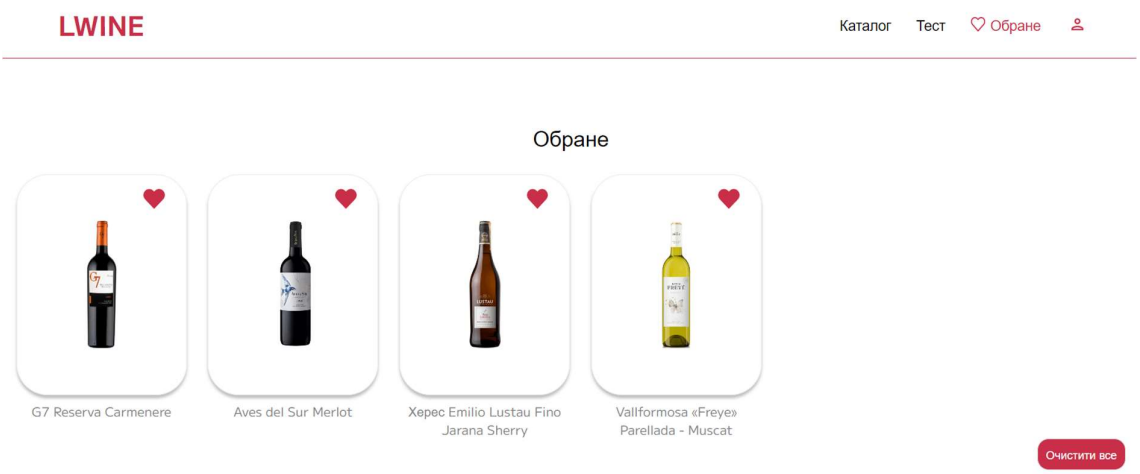


Рисунок 34. Сторінка обраних вин

11. Тепер вже відкрита можливість додавати коментарі під вином

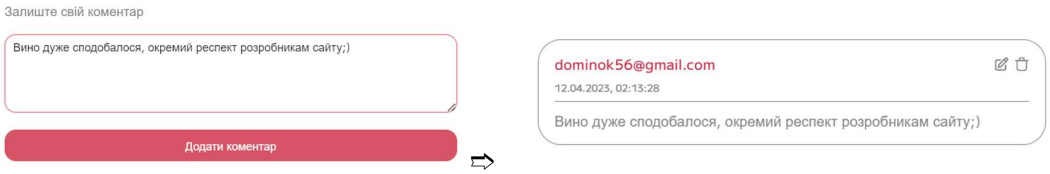


Рисунок 35. Створення коментарю до вина

12. Щоб зайти в особистий кабінет, або навпаки вийти з власного акаунту, достатньо натиснути на іконку в правому верхньому куті.

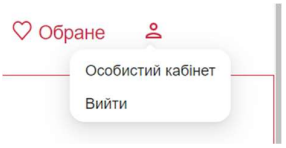


Рисунок 36. Особистий кабінет

13. В особистому кабінеті є доступ до зміни особистої інформації та паролю

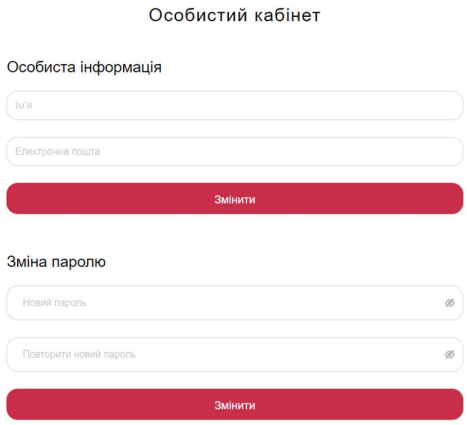


Рисунок 37. Зміна особистої інформації

Інструкція використання боту.

1. При переході в бот висвічується кнопка “start”. Натиснувши її можна активізувати бота.

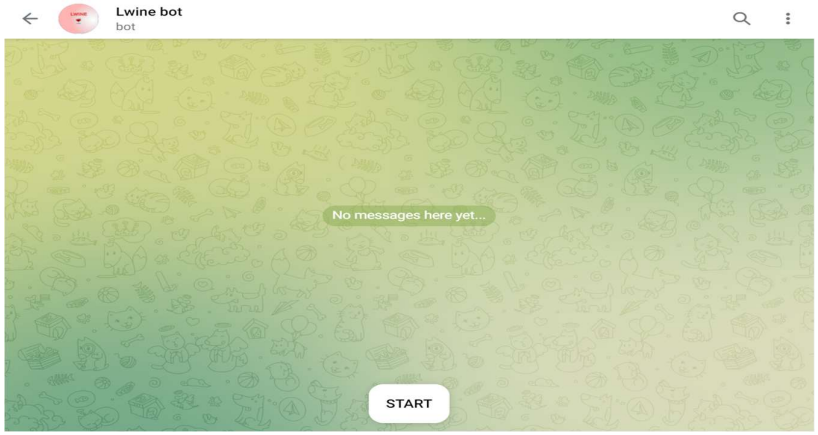


Рисунок 38. Запуск телеграмм-боту

2. Натиснувши “/help” можна ознайомитися з правилами боту

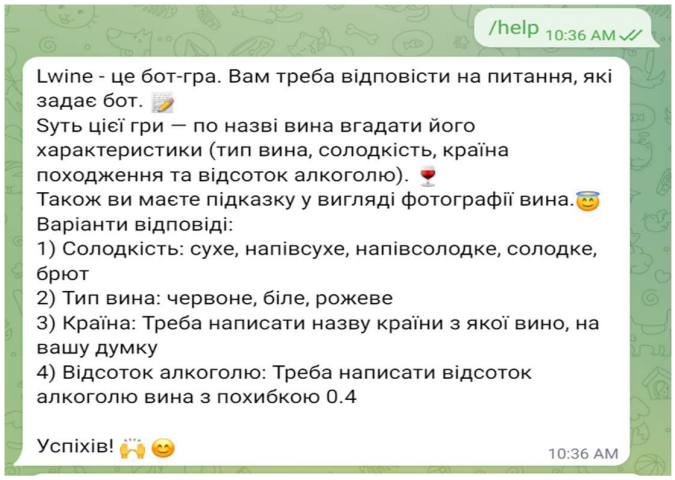


Рисунок 39. Команда “/help”

3. Після привітання бота, можна напряму в його повідомленні натиснути “/game” , щоб розпочати гру.

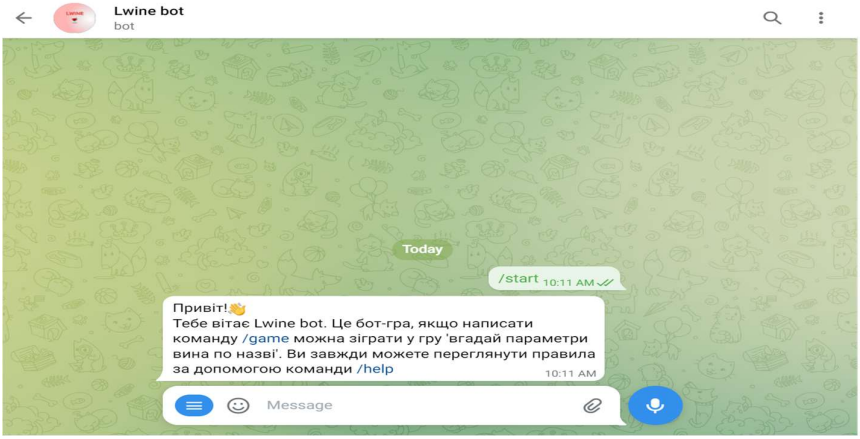


Рисунок 40. Початок гри

4. В результаті цього бот надає назву вина та питання по параметру.

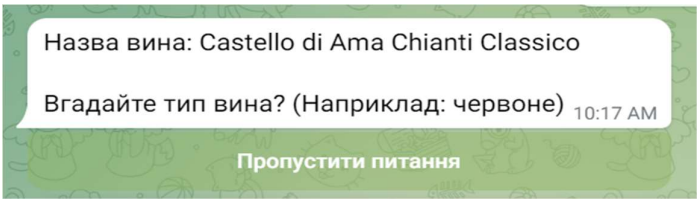


Рисунок 41. Приклад питання в грі

Відповідь варто надрукувати українською в полі повідомлення та надіслати боту у відповідь.

5. Також є можливість пропустити це питання або взяти підказку. В разу натискання “пропустити питання” бот згенерує наступне питання.

6. Якщо ж ви хочете взяти підказку, варто натиснути на відповідне поле та бот надасть вам фотографію вина. Інших типів підказок протягом гри немає.

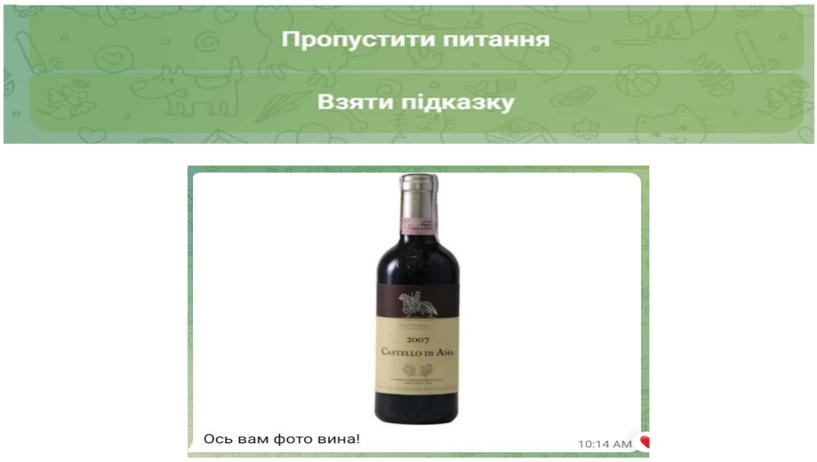


Рисунок 42. Підказка в грі

7. Після відповіді на питання, бот надішле вам результати та запропонує придбати його в одному з магазинів. Правильні відповіді також будуть вказані.

8.

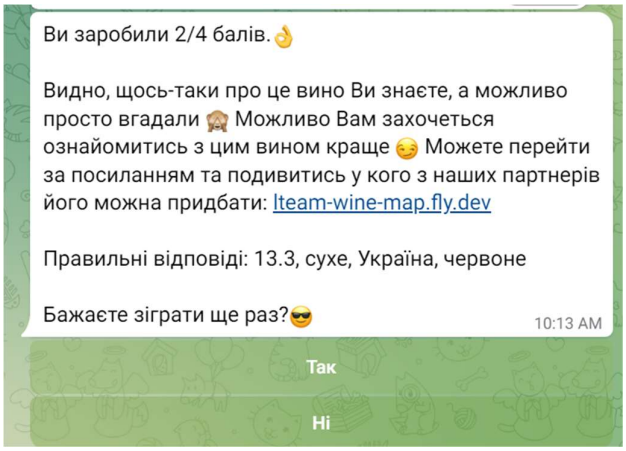


Рисунок 43. Результат гри

Також є можливість продовжити грати або завершити гру, для цього треба натиснути “так” чи “ні” відповідно.

ФОРМАЛЬНА СПЕЦИФІКАЦІЯ ТА ВЕРИФІКАЦІЯ

Специфікацію було вирішено розробляти на мові Dafny. Даний спосіб перевірки програм включає передумови та післяумови, мова підтримує загальні класи та динамічне видалення, а також є об’єктивною, послідовною та імперативною, вона чудово підходить для нашої системи. У нашій системи присутня реєстрація користувача - мова Dafny може на сто відсотків верифікувати цю частину коду.

Для специфікації ми використовуємо клас “User” з полями email, username та password. Ми верифікуємо валідацію даних користувача. Сама програма включає перевірку на унікальність користувача, довжина паролю та логіну, та валідацію поштової адреси.

```
class User {
  var Email: array<char>;
  var Password: array<char>;
  var UserName: array<char>;

  constructor(email: array<char>, password: array<char>, userName: array<char>) {
    this.Email := email;
    this.Password := password;
    this.UserName := userName;
  }
}

class WineProgram {
  var users: set<User>;

  constructor(users: set<User>)
  modifies users
  {
    this.users := users;
  }

  static method IsUnique(user: User, users: set<User>)
  returns (r: bool)
  {
    r := user !in users;
  }

  method ContainsAt(a: array<char>)
  returns (r: bool)
  {
    var index := 0;
    while index < a.Length
      invariant 0 <= index <= a.Length
      invariant forall k :: 0 <= k < index ==> a[k] != '@'
    {
      if a[index] == '@' { return true; }
      index := index + 1;
    }
```

Рисунок 44. Клас верифікації

Сам процес верифікації включає в себе створення нового користувача та перевіряє коректність кожного з полів класу. Після перевірки ми додаємо верифікованого користувача до списку.

```
method RegisterUser(user: User)
returns (res: bool)
modifies this
{
  var unique := IsUnique(user, this.users);
  var correctEmail := ContainsAt(user.Email);
  var correctPassword := user.Password.Length >= 8;
  var correctUsername := user.UserName.Length >= 3;

  if (unique || correctPassword || correctUsername) {
    return false;
  }

  this.users := this.users + {user};
  res := user in this.users;
  return true;
}
```

Рисунок 45. Процес верифікації

Отже, нам вдалося успішно верифікувати створення нового користувача. Виконати валідацію на кожне поле базового класу та переконатися, що наш код працює.

ОПИС УПРАВЛІННЯ РОЗРОБКОЮ

Початковим було рішення використовувати методологію kanban та trello для ведення спринтів. Тому було обрано дану методологію:

- Візуалізація процесів: канбан полягає у відображенні процесів розробки або виробництва на дошці з картками, що відображають стадії робіт. Це дозволяє команді зрозуміти, яка частина роботи вже зроблена, яка ще потребує уваги та де виникають затримки.
- Керування підтримкою якості: канбан допомагає команді зосередитись на підтримці якості продукту. Це досягається шляхом встановлення обмежень на кількість незавершених робіт і вимагає від команди завершення однієї роботи, перш ніж починати іншу.
- Постійна оптимізація: канбан дозволяє команді постійно вдосконалювати свої процеси, щоб зменшити час циклу та підвищити якість продукту. Це досягається шляхом внесення змін у картки на дошці та встановлення нових обмежень.

Але вже на першому тижні ми відчули значні недоліки даної методології. Наприклад, висока залежність від команди. Kanban залежить від високо ефективної команди, для досягнення цілей в межах визначеного часу, що спочатку було певною перевагою, але для першого проекту стало занадто складною умовою.

Також, дана методологія потребує адаптації. Kanban вимагає великої адаптації до змін протягом виконання проекту. Отже, після першого спринту було прийнято колективне рішення змінити методологію на більш гнучку – Scrum. Чому було прийнято таке рішення? Тому що, на відміну від Kanban, який є однією з реалізацій agile методології, методологія scrum більше фокусується на адаптації до потреб користувачів та бізнес-потреб. Також scrum підходить до проекту як до системи, а не як до складного процесу, що відрізняє його від kanban.

Ось такі особливості scrum методології ми визначили до того, як почали працювати за нею:

- Методологія Scrum використовує відбірковий підхід до процесу управління проектом з подальшим аналізом результатів протягом циклу розробки.

- Робочі процеси Scrum зосереджуються на вирішенні поставлених перед командою задач та завершенні проекту у визначений строк.
- Головний принцип методології Scrum - керування проектом поступово та командно, що дозволяє більш ефективно та зручно проводити роботу.
- Scrum використовує концепцію прагнення до вирішення проблем та завдань по максимально короткому часу.

Такі переваги ми виявили під час роботи в цій методології:

1. Забезпечує постійний процес поліпшення.
2. Забезпечує реактивне реагування на зміни в проекті.
3. Дозволяє відслідковувати перебіг робіт і відстежувати прогрес на кожному етапі.
4. Стимулює відвідування постійних зустрічей і встановлення пріоритетів.
5. Уникає створення масштабних планів і дозволяє вирішувати проблеми до завершення проекту.
6. Надає можливість надавати пріоритети завданням та адаптуватися до змін у реальному часі.
7. Забезпечує по-справжньому інтегровану команду і підтримує постійну взаємодію.
8. Забезпечує високий рівень відповідальності і прозорості.

Scrum методологія включає підхід, який спрямований на постійне вдосконалення результатів процесу розробки програмного забезпечення, за допомогою постійного використання великої кількості миттєвих ітерацій. Заснована на принципах адаптації, командної роботи, єднання бізнес-потреб та технологій та регулярних демонстраціях, які постійно реалізують велику ймовірність успіху проекту.

Дійсно, зі зміною методології, команда стала продуктивніше та більш результативно працювати.

Що найбільше з даної технології мало вплив на успішне виконання запланованих задач:

- регулярність командних зустрічей. Ми знайшли баланс та дійшли до висновку, що зустрічі тричі на тиждень дають можливість швидко вирішувати всі питання, стежити за прогресом задач (уникати їх блокування та невиконання) та планувати задачі на наступний тиждень.
- ведення всіх задач в trello. Це візуалізація прогресу спринту, що дає можливість уникнути не завершення задач до кінця спринту. За декілька днів до кінця спринту ми аналізували, які задачі не виконані — обговорювали та вирішували як можна виправити дану ситуацію.

Як вже зазначалось, ведення задач відбувалось за допомогою дошки Trello. Ми обрали Trello, оскільки це потужний і зручний безкоштовний інструмент для керування проектами та роботи з командою. Його гнучкість дозволяє відтворювати будь-який процес та цілі. Особливість, через яку ми обрали даний інструмент це те, що Trello має доступ до ряду додаткових функцій, таких як перегляд активності, можливість прикріплювати документи.

Ось такий вигляд має дошка: до кожної задачі назначено відповідальну людину та виконавців, прописано до якої частини команди це відноситься (аналітика, дизайн, фронтенд, бекенд або вся команда). По завершенню до опису задачі ми прикріплювали документи, посилання, пояснення або Pull Request, щоб не загубити де саме та коли вона була виконана.

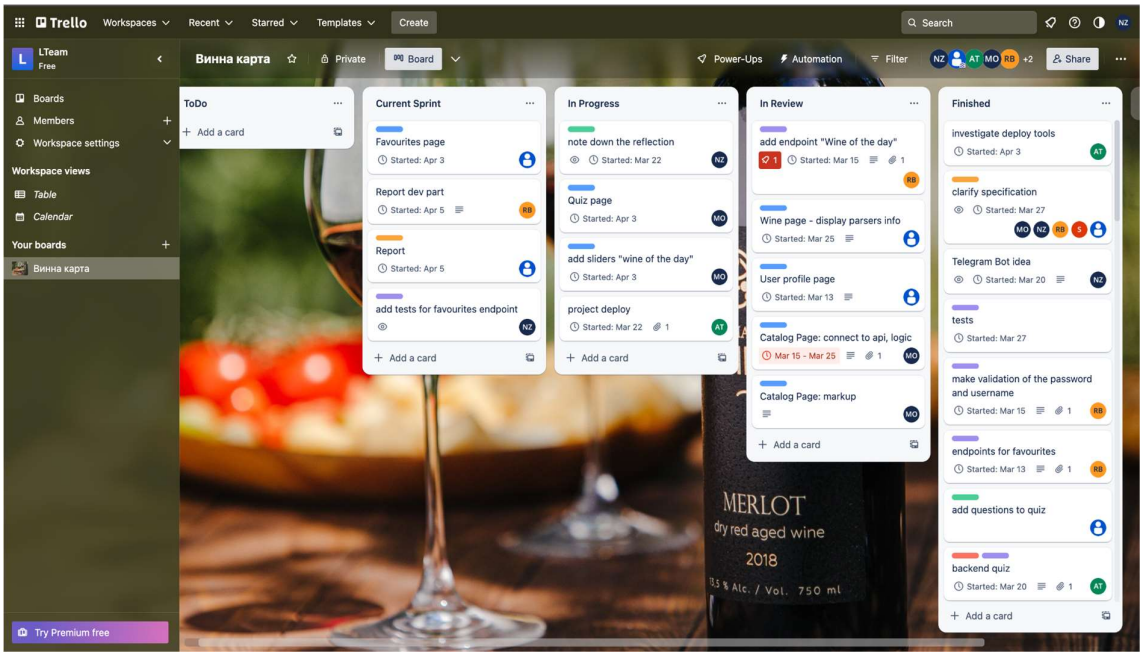


Рисунок 45. Дошка Trello

Спринти планувались на тиждень. Майже завжди вдавалось розподілити навантаження таким чином, щоб по виконанню спринту не було багато невиконаних задач. Але й не було такого, що хтось з команди не мав задач. Розподілялись задачі порівно між усіма членами команди, в силу навиків та можливостей людини.

Під час роботи в команді були проблеми, які виникали протягом сумісної роботи. Деякі проблеми виникали одноразово, але деякі повторювались, але ми швидко їх визначали та намагались виправити. Наведемо приклади таких ситуацій:

- **Недовершене планування.**

Це було найбільшою проблемою, яка найчастіше виникала. Варто зазначити, що наслідки недовершеного планування щоразу проявлялись у новій формі. Наприклад, завелика кількість задач на один спринт, завелика або недостатня кількість задач на одну особу, виконання глобальної задачі без чітких меж.

Способи вирішення даної проблеми:

1) Більше ознайомлення з технічними навичками та досвідом кожного з членів команди. Розрахування ймовірності успішного виконання задачі з цим фактором.

2) Детальне обговорення задачі на моменті її створення та призначенні її для користувача. Якщо виникають питання щодо виконання задачі, то її обговорення виносилось на кінець дзвінку.

3) Розбиття великої задачі на підзадачі. Це значно покращило розуміння прогресу виконання спринту.

- **Блокування задач.**

Іноді незавершеність спринта була лише за рахунок тих задач, які залежали від виконання інших задач. Здебільшого це стосувалось різних задач, які були назначені для різних людей.

Спосіб вирішення даної проблеми: розбиття подібних задач на різні спринти (хоча якщо людина була в змозі зробити обидві задачі за один спринт, то це вважалось більш бажаним варіантом) або назначення залежних задач для однієї людини, тоді їй/йому не доводиться чекати, він/вона робить свої задачі послідовно.

- **Недостатня кількість комунікації.**

Задля уникнення даної задачі було зроблено чіткий графік з достатньою кількістю дзвінків. Також, домовились вести постійну комунікацію в чатах, щоб не чекати дзвінка.

На даному проекті ми навчилися співпраці та роботі в команді, плануванню, комунікації та управлінню часом. Ми мали змогу робити командну роботу в одному напрямку, щоб разом досягти заданих цілей, на що була сильна мотивація та зацікавленість. Також ми навчилися передавати інформацію в проекті, доносити свої думки та контролювати завершення особистих задач, що в подальшому стало вдалим завершенням проекту. Можна впевнено сказати, що проект завершився вдало, оскільки команда досягла поставлених на початку цілей. Звичайно, залишається питання монетизації та подальшого розвитку проекту.

З технічної точки зору, кожен учасник мав нові для себе задачі, що дало можливість розвиватись як спеціаліст та професіонал в певній галузі. Ще ми отримали знання у використанні гібких методологій розробки.

ВИСНОВКИ

У межах даної лабораторної роботи наша команда вирішила займатись створенням системи "Винна карта". В результаті спільних зусиль ми успішно розробили веб-сайт, здійснили аналітику, створили дизайн, унікальний тест та телеграм-бот. Робота над проектом була надзвичайно продуктивною та цікавою.

Працювати в команді було дуже захоплююче. Відчуття спільної відповідальності та рівність учасників сприяли успішному виконанню завдань та вирішенню проблем. Кожен член команди вніс свій вклад, допомагаючи іншим у разі необхідності. Ми багато чому навчились, розвиваючи навички спілкування, організації та управління проектами.

Працюючи над проектом, ми навчились працювати з такими технологіями, як Django, Django REST framework, React, Ant Design, Figma та Trello. Використовуючи ці технології, ми змогли створити сучасний, ефективний та привабливий веб-сайт.

Наша команда з гордістю може відзначити успіхи проекту. Ми отримали позитивні відгуки від університетських викладачів та студентів. Створений нами телеграм-бот та унікальний тест допомагають користувачам знайти найбільш підходящі вина та дізнатись цікаву інформацію про них.

Працюючи над проектом, ми значно розширили свої навички та знання в різних сферах програмування та розробки. Ось деякі з них:

1. Розробка веб-сайту: ми покращили свої навички роботи з Django та Django REST framework, що дозволило нам створити масштабовану та високопродуктивну веб-платформу.

2. Frontend-розробка: ми вивчили React та Ant Design, що дозволило нам створити ефективний та адаптивний інтерфейс користувача.

3. Дизайн: ми опанували роботу з Figma, яке допомогло нам створити естетичний та привабливий дизайн веб-сайту.

4. Управління проектами: ми навчились ефективно використовувати Trello для організації роботи, стеження за прогресом та спілкування в команді.

5. Розробка телеграм-бота: ми навчились створювати та інтегрувати телеграм-бота, який допомагає користувачам у виборі вина та надає інформацію про них.

Проект "Карта вин" став важливим досвідом для нашої команди. Ми отримали гарний досвід у роботі з новими технологіями та розширили свої навички у програмуванні та дизайні. Цей проект довів, що спільними зусиллями та відповідним підходом до роботи можна досягти чудових результатів. Ми пишаємося своїми досягненнями та сподіваємося на подальше використання здобутих знань та навичок у майбутніх проектах.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Ant Design. Офіційний веб-сайт. [Електронний ресурс]. Режим доступу: https://ant.design/
2. Django. Офіційний веб-сайт. [Електронний ресурс]. Режим доступу: https://www.djangoproject.com/.
3. Django Channels. Документація. [Електронний ресурс]. Режим доступу: https://channels.readthedocs.io/en/stable/.
4. Django Girls Tutorial. [Електронний ресурс]. Режим доступу: https://tutorial.djangogirls.org/.
5. Django Rest Framework. Офіційний веб-сайт. [Електронний ресурс]. Режим доступу: https://www.django-rest-framework.org/.
6. Figma. Офіційний веб-сайт. [Електронний ресурс]. Режим доступу: https://www.figma.com/.
7. GitHub. [Електронний ресурс]. Режим доступу: https://github.com/.
8. Official Git Documentation. [Електронний ресурс]. Режим доступу: https://git-scm.com/doc.
9. Official Python Documentation. [Електронний ресурс]. Режим доступу: https://docs.python.org/.
10. Official React Documentation. [Електронний ресурс]. Режим доступу: https://reactjs.org/docs/getting-started.html.
11. PostgreSQL. Офіційний веб-сайт. [Електронний ресурс]. Режим доступу: https://www.postgresql.org/.
12. React. Офіційний веб-сайт. [Електронний ресурс]. Режим доступу: https://reactjs.org/.
13. Redux. Офіційний веб-сайт. [Електронний ресурс]. Режим доступу: https://redux.js.org/.
14. Vivino. Веб-сайт. [Електронний ресурс]. Режим доступу: https://www.vivino.com/CA/en/
15. Hello vino. Мобільний додаток. Режим доступу: http://www.hellovino.com/
16. Vinomojo. Веб-сайт. [Електронний ресурс]. Режим доступу: https://vinomojo.co.uk/
17. Vinnie. Мобільний додаток. Режим доступу: https://vinnie.app/

СИСТЕМА УПРАВЛІННЯ ПОРТФОЛІО КОМПАНІЙ "OPTIMUM PORTFOLIO"

Анна Алєксєєнко, Олександр Лихопуд, Богдан Огородній, Володимир Чучканов, Дмитро Шабанов, Максим Юдкін, Нікіта Орляк , Ярослав Приходько

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

Для спрощення читання і розуміння даного документа будуть використовуватися наведені нижче абревіатури або скорочення, які можуть стосуватися як технічних моментів, так і позначення персон або термінів.

Таблиця 1. Абревіатури та скорочення

№ з/п	Термін	Абревіатури, скорочення	Коментар
1	Зареєстрований / незареєстрований користувач	користувач	-
2	База даних	БД	-
3	Технічне завдання	ТЗ	-
4	Адміністратор системи	Адміністратор	Особа або група осіб, що повністю ознайомлені з функціональною та програмно-апаратною структурою програмного продукту і контролюють коректність її використання
5	Хмарні обчислення	Хмара	Сукупність сервісів, що надають доступ до використання обладнання на вимогу та адмініструють управління наданими ресурсами
6	Серверна частина проєкту	Бекенд	-
7	Користувацький інтерфейс	UI	-
8	UI частина проєкту	Фронтенд	-
9	Система керування базами даних	СКБД	-
10	Google Cloud Platform	GCP	-
11	Integrated development environment	IDE	-
12	Програмний продукт	Система	-

ВСТУП

Мета й завдання роботи. Система управління портфоліо компаній “Optimum Portfolio” (далі – програмний продукт) розробляється як комплексне рішення для організації проєктів компаній в портфоліо, якими можна ділитися за допомогою посилань. Для досягнення цієї мети поставлено такі завдання.

- Проаналізувати існуючі застосунки для огляду портфоліо компаній та визначити функціонал, якого не вистачає більшій кількості користувачів.
- Обрати засоби та інструменти розробки застосунку.
- Спроєктувати архітектуру застосунку, схему бази даних, план тестування та верифікації.
- Розробити користувацький інтерфейс та дизайн продукту.
- Розробити сервер для обробки запитів клієнта.
- Дослідити методи запуску клієнта та сервера у хмарному середовищі.

Актуальність проєкту. Під час пошуку компанії, що зможе задовольнити конкретні бізнес потреби, користувачі обов’язково оглядають портфоліо компаній, щоб впевнитися в їх експертизі. Основними критеріями вибору підрядника є кількість та якість виконаних в певній області проєктів, відповідальність та надійність компанії, чітке слідування умовам контракту. Актуальність проєкту полягає в наданні систематизованої платформи для ефективного створення та надання доступу до портфоліо компанії, а також можливості пошуку компаній чи розроблених ними проєктів за обраними областями експертизи чи технологіями. Верифікація компаній адміністраторіми та наявність відгуків на платформі допомагає користувачам бути впевненими у надійності підрядника, а ефективниий пошук та швидка комунікація економить суттєву долю часу користувачів.

Методи дослідження. Основними методомами дослідження є розробка програмного продукту, аналіз та порівняння застосунків, що містять подібний функціонал, вивчення документації використаних технологій.

Цілі і завдання. Дослідити принцип роботи систем управління портфоліо компаній, імплементувати власне рішення, що вирішує більшість недоліків існуючих систем, поглибити знання з технологій розробки, тестування та верифікації клієнт-серверних застосунків, а також здобути досвід роботи над проєктом в команді.

АНАЛІЗ ІСНУЮЧИХ НА РИНКУ РІШЕНЬ

Командою було проаналізовано наступні конкурентні рішення:

- **Crunchbase** - платформа, яка надає інформацію про компанії, стартапи, інвесторів та інших гравців на ринку. Вона містить дані про фінансування, новини, продукти та технології компаній, а також про контактні дані представників компаній. Crunchbase має велику базу даних, яка оновлюється щоденно.
- **CB Insights** - платформа, яка збирає дані про стартапи, інвестиції, ринкові тренди та інші аспекти технологічної екосистеми. Вона надає інсайти про те, які компанії отримують інвестиції, які технології та бізнес-моделі перебувають у тренді, а також допомагає знаходити потенційних клієнтів та партнерів.
- **PitchBook** - платформа, яка надає дані та інсайти про приватні компанії та ринок інвестицій. Вона містить інформацію про фінансування, угоди, умови угод та інші аспекти приватних компаній. PitchBook також надає інструменти для пошуку та аналізу компаній.
- **SimilarWeb** - платформа, яка надає дані про трафік та використання веб-сайтів та мобільних додатків. Вона допомагає розуміти, які сайти та додатки використовують конкуренти, які канали маркетингу найбільш ефективні та як можна покращити свій веб-сайт та мобільне додаток.

Висновки стосовно кожного продукту можна підсумувати у вигляді таблиці 2.

Таблиця 2**.** Аналіз конкурентів на ринку

Продукт	**Переваги**	**Недоліки**
Crunchbase	- Велика база даних про компанії, що оновлюється щоденно; - Наявність API та інтеграція з іншими системами; За результатами аналізу переваг було розроблено API сервера з можливістю подальшого надання публічного доступу.	- Висока вартість платної версії платформи; - Не всі дані доступні в безкоштовній версії; - Інтерфейс може бути не дуже зручним для користувачів; За результатами аналізу недоліків продукту було розроблено більш зручний UX системи.
CB Insights	- Дані про стартапи, інвестиції та ринкові тренди, що допомагають зрозуміти технологічну екосистему; - Інформація про потенційних клієнтів та партнерів; - Легкий доступ до аналітичних звітів та інсайтів;	- Висока вартість платної версії платформи; - Інтерфейс може бути складним для новачків; - Не всі дані доступні в безкоштовній версії;
PitchBook	- Дані про приватні компанії та ринок інвестицій; - Легкий доступ до інформації про угоди та умови фінансування; - Інструменти для пошуку та аналізу компаній;	- Система фільтрації може бути складною для користувачів; - Обмежені дані (в різних версіях). - Незвичайний, складний інтерфейс; За результатами аналізу недоліків було спроєктовано систему фільтрації з фокусом на зручність користувача.
SimilarWeb	- Дані про трафік та використання веб-сайтів та мобільних додатків; - Допомагає зрозуміти поведінку конкурентів та їхню стратегію маркетингу; - Підтримка багатьох мов та географічних регіонів;	- Обмежені можливості аналізу соціальних мереж та інших каналів маркетингу; - Висока вартість платної версії платформи; - Недостатньо детальна інформація про деякі веб-сайти;

ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Python. Це інтерпретована мова програмування загального призначення, яка зазвичай використовується для широкого спектру завдань програмування, таких як розробка веб-додатків, наукових обчислень, штучного інтелекту та багатьох інших. Python має простий синтаксис, що робить його легким для вивчення, і багато стандартних бібліотек, що спрощують розробку програм [1]. Також, Python має велику спільноту розробників, яка активно працює над покращенням мови та розробкою нових бібліотек та фреймворків. Мова Python була обрана командою через легкість освоєння та розвинені фреймворки для створення веб-застосунків, доступні для цієї мови.

Flask є мінімалістичним фреймворком для розробки веб-додатків на мові Python. Flask надає базовий набір функцій для створення веб-додатків, таких як маршрутизація URL-адрес, робота з формами, керування сесіями та інші [2]. Фреймворк дозволяє розробникам вільно розпоряджатися архітектурою свого додатку, що дає більшу свободу в розробці. За замовчуванням Flask не накладає жорстких обмежень на вибір бібліотек

та інструментів, що використовуються в проекті, що дозволяє розробникам додатків вибрати найкращий набір інструментів для своїх потреб. Flask має велику спільноту розробників та багато документації, що спрощує розробку та підтримку додатків. Flask було обрано через мінімалістичність фреймворку, легке налаштування локального оточення для розробки та швидкість освоєння.

JavaScript - це скриптова мова програмування, що використовується для розробки веб-додатків та взаємодії з веб-сторінками. JavaScript може використовуватися для динамічної зміни вмісту веб-сторінки, валідації форм, асинхронних запитів до сервера та багато іншого. JavaScript може використовуватися як на стороні клієнта, так і на стороні сервера. JavaScript є однією з найпоширеніших мов програмування, яка має велику спільноту розробників та багато різноманітних бібліотек та фреймворків, що значно спрощують розробку веб-додатків [3]. Більшість сучасних веб-сторінок використовують JavaScript для динамічної зміни вмісту та взаємодії з користувачем. Мову JavaScript було обрано через популярність її використання на проєктах, а також наявність великої кількості фреймворків для імплементації користувацького інтерфейсу.

React - це бібліотека JavaScript, що використовується для розробки користувацьких інтерфейсів. React забезпечує спрощений підхід до створення веб-додатків, що дозволяє розробникам зосередитися на створенні користувацьких інтерфейсів без великої кількості додаткового коду. React працює з компонентами, які є незалежними і перевикористовуваними блоками інтерфейсу [4]. Компоненти дозволяють розробникам створювати складні інтерфейси з простих блоків. React також використовує віртуальний DOM для оптимізації роботи з веб-сторінкою, що дозволяє виконувати мінімальну кількість змін на сторінці при оновленні інтерфейсу. React має велику спільноту розробників та багато бібліотек та фреймворків, які спрощують розробку веб-додатків на базі React. React також часто використовується в поєднанні з іншими технологіями, такими як Redux, для керування станом додатку, або React Native, для розробки мобільних додатків.

Google Cloud Platform - це хмарна платформа, що надає різноманітні сервіси для зберігання даних, обчислень, машинного навчання та іншого. Google Cloud Platform надає розробникам можливість використовувати потужні інструменти та сервіси, не потребуючи великих витрат на інфраструктуру та обслуговування [5]. До основних сервісів Google Cloud Platform належать зберігання даних в обласній мережі, аналіз даних, обробка даних у реальному часі, машинне навчання, віртуальні машини, сервіси контейнерів та інші. Google Cloud Platform дозволяє розробникам створювати масштабовані додатки, які можуть бути використані у великому обсязі користувачів та високо завантажених середовищах. Google Cloud Platform також надає широкі можливості для інтеграції з іншими сервісами Google, такими як Google Analytics, Google Maps, Google Drive та інші. Розробники можуть використовувати Google Cloud Platform як для публічних, так і для приватних хмарних сервісів, що дозволяє налаштувати середовище розробки за потребами проекту.

Pytest та Unittest - це бібліотеки для тестування коду на Python. Обидва інструменти дозволяють писати тести, які автоматично виконуються та перевіряють правильність роботи функцій та методів [6]. Pytest є більш простим та легким у використанні, в той час як unittest надає більше можливостей для розширення функціональності тестів.

Postman - це інструмент для тестування API, який дозволяє тестувати та перевіряти роботу веб-сервісів, шляхом створення запитів до API та отримання відповідей в різних форматах, таких як JSON, XML та інші. [7] Postman дозволяє автоматизувати тести та забезпечує можливості моніторингу та аналізу відповідей API. Postman було обрано

через можливість створення колекцій запитів, якими можна ділитися в межах команди, що значно пришвидшує спільну розробку продукту.

app.diagrams.net - це безкоштовний веб-інструмент для створення різноманітних діаграм, таких як блок-схеми, діаграми потоків, діаграми баз даних та інші. [8] Цей інструмент дозволяє швидко створювати та редагувати діаграми у веб-браузері, зберігати їх у хмарі та експортувати у різні формати файлів.

Figma - це веб-інструмент для дизайну інтерфейсів користувачів та спільної роботи над проектами. Цей інструмент дозволяє створювати макети веб-сайтів та мобільних додатків, редагувати графічні елементи та робити анімацію [9]. Figma надає можливості спільної роботи над проектами в реальному часі, що дозволяє команді легко спілкуватись та координувати свою роботу. Інструмент також має можливості для експорту проектів у різні формати, такі як PNG, SVG, PDF, а також можливість інтеграції з іншими інструментами розробки.

Trello - це онлайн-інструмент для керування проектами та організації робочих процесів [10]. Trello базується на концепції дошок, які містять карточки з завданнями або ідеями, які можуть бути розподілені між користувачами та організовані за категоріями або статусами. Користувачі можуть створювати дошки для будь-якої роботи, включаючи проектні управління, технічну підтримку, ведення списку завдань, планування подій, тощо. Кожна карточка має свій заголовок, опис, прикріплені файли, мітки, терміни та коментарі, що дозволяє розробникам та іншим співробітникам бачити повний опис завдання та всіх його атрибутів. Trello дозволяє створювати та керувати командами, додавати співробітників до дошок та карточок, забезпечуючи зручний інтерфейс для спілкування та співпраці. Також Trello інтегрується з іншими популярними сервісами, такими як Google Drive, Dropbox, Slack, GitHub та інші, що дозволяє розширювати функціональність інструменту та забезпечувати його інтеграцію з іншими робочими процесами.

ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

Призначення системи: управління портфоліо компаній є автоматизація процесу організації проєктів компаній в портфоліо, що дає можливість ділитися проєктами за допомогою посилань, можливість перегляду інформації про компанії, що уже існують на ринку, їхніх експертиз і технологій. Робота передбачає:

- аналіз методів, методик і моделей, що застосовуються для розв’язання задач створення вебсайтів
- аналіз наявних програмних продуктів в галузі
- проєктування та програмну реалізацію системи
- апробацію системи “Optimum Portfolio”

Цілі створення системи

Система управління портфоліо компаній створюється з метою:

- забезпечення збору, обробки та аналізу інформації про технології для проєктування реалізації веб систем
- організації проєктів компаній в портфоліо, якими можна ділитися за допомогою посилань
- можливості проаналізувати існуючі продукти на ринку, враховуючи їхні переваги та недоліки, при розробці системи

Також система призначена для більш зручної комунікації між користувачами та компаніями, що представлені на сайті. Більш детальний опис функціоналу системи зображено на рисунку 1.

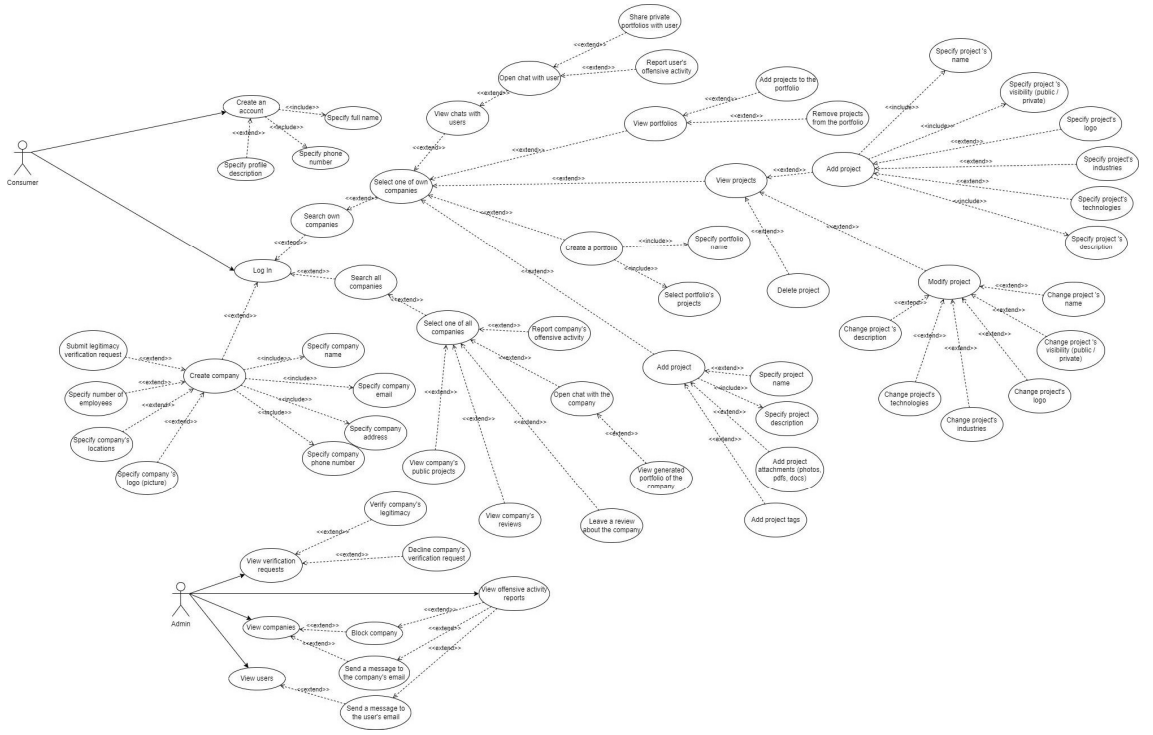


Рисунок 1. Use-case діаграма системи

Вимоги до системи в цілому

Вимоги до структури та функціонування системи. Система “Optimum Portfolio” повинна забезпечувати реалізацію основних завдань:

- створення нових компаній користувачами
- створення нових проєктів, групування проєктів у портфоліо
- пошук компаній, які зареєстровані на сайті
- спілкування користувачів з представниками компаній

Мова системи. Основна мова системи – англійська.

Вимоги до стилістичного оформлення Системи. Стилістичне оформлення Системи має відповідати дизайн-макету, погодженого з керівником Замовника через керівника проєкту та розробленого з використанням застосунку Figma.

Вимоги до графічного дизайну Системи. Система має бути зручною у використанні для користувача, UI має бути адаптивним і коректно відображатися на екранах різного розміру. Повинен використовуватися мінімалістичний дизайн, компоненти (такі як кнопки, спливаючі вікна та текстові поля) повинні мати один стиль та не сильно відрізнятися за розмірами.

В Системі передбачається виділити наступні функціональні підсистеми:

- **адміністративна**, призначена для підтвердження даних компаній, можливості обмежувати доступ для компаній та користувачів, змінювати їх дані
- **користувача**, призначена для роботи зі створенням та підтримкою власних компаній та їхніх портфоліо

Вимоги до функцій, які виконуються системою

Підсистема адміністратора.

Таблиця 3. Перелік функцій, задача що підлягають автоматизації

Функція	**Задача**
Керувати роботою користувачів системи	Редагування та видалення інформації про користувачів системи
	Формування та візуалізація звітів про користувачів системи
	Можливість обмежувати доступ до сайту користувачам
Робота з запитами	Проведення верифікації даних про компанію
	Розгляд скарг на користувачів / компанії

Підсистема користувача

Таблиця 4. Перелік функцій, задач що підлягають автоматизації

Функція	**Задача**
Реєстрація	Введення інформації про себе
	Оновлення реєстрації
	Видалення інформації про себе
Створення та редагування даних про власну компанію зареєстрованими користувачами	Створення та редагування сторінки власної компанії
	Створення та редагування проєктів власної компанії
	Створення та редагування портфоліо власної компанії
	Можливість вести спілкування з користувачами від імені компанії
Можливість спілкуватися з іншими користувачами за допомогою чату	Переглянути свої існуючі чати
	Почати новий чат з користувачами
	Можливість написати представнику компанії

Системні вимоги. Браузери: Сайт повинен коректно відображатися в інтернет-браузерах MS Internet Explorer 5.5 і вище, Mozilla 1.7 і вище, Opera 7.54 і вище, Google Chrome 99.0 і вище, Safari 14.1 і вище. На довільні некоректні дії користувача, пов’язані з введенням невірних даних, не заповненням обов’язкових полів введення в формах та інші, які можуть бути оброблені системою, генеруються відповідні повідомлення про помилки англійською мовою, в межах загального дизайну сайта.

Операційна система: Windows 10 і вище.

Джерелом даних для Системи повинна бути інформаційна система PostgreSQL.

ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

Логічна структура бази даних

Під час розробки використовувалась база даних PostgreSQL. На рисунку 2 наведено детальну схему бази даних. У таблиці 5 описано основні таблиці та їх призначення в системі.

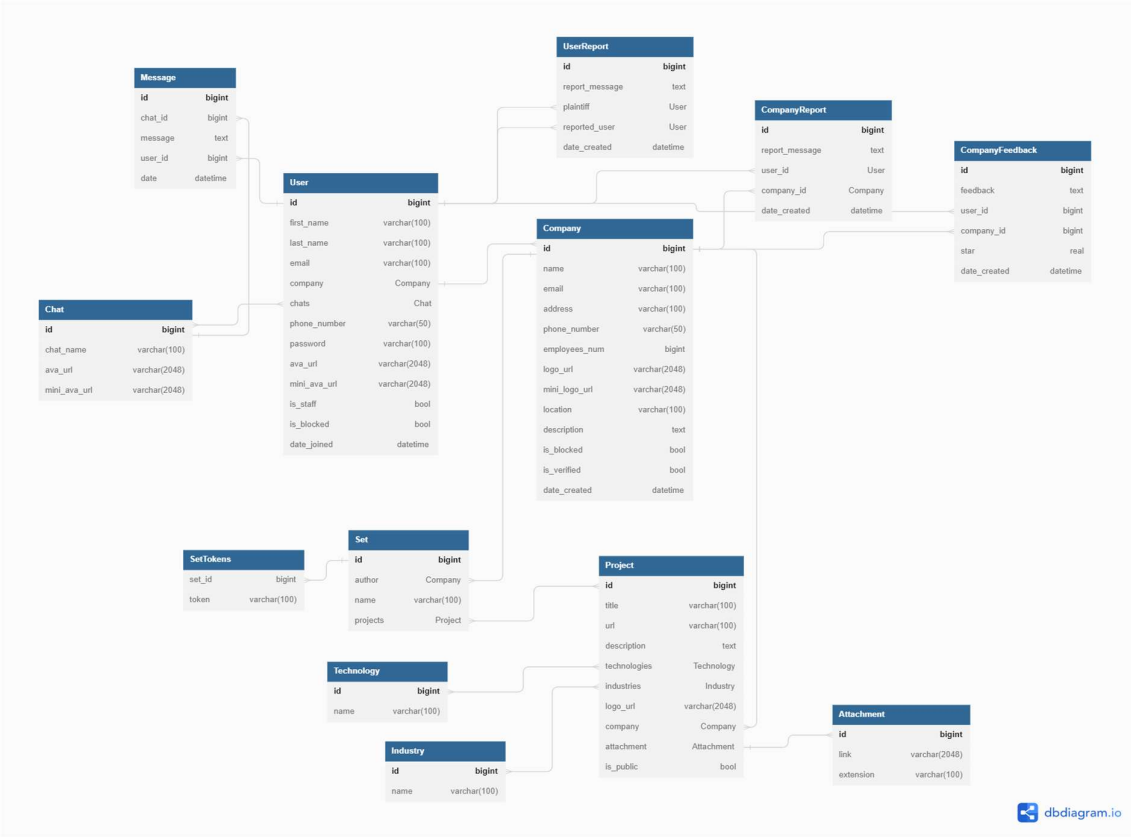


Рисунок 2. Схема бази даних

Таблиця 5. Повний перелік таблиць.

Номер	Таблиця	Опис
1	User	Таблиця для збереження інформації про користувачів
2	Company	Таблиця для збереження інформації про компанії
3	Industry	Таблиця для збереження інформації про індустрії
4	Technology	Таблиця для збереження інформації про технології
5	Project	Таблиця для збереження інформації про проєкти
6	Set	Таблиця для збереження інформації про набори проєктів, які буде показувати компанія
7	SetTokens	Таблиця для збереження інформації про токен доступа URL
8	Chat	Таблиця для збереження інформації про чат
9	Message	Таблиця для збереження інформації про повідомлення
10	CompanyFeedback	Таблиця для збереження інформації про фідбек компаній
11	CompanyReport	Таблиця для збереження інформації про скарги на компанії
12	UserReport	Таблиця для збереження інформації про скарги на користувачів
13	Attachment	Таблиця для збереження інформації про вкладення

Опис таблиць

У наступних таблицях приведено опис даних у кожній з таблиць.

Таблиця 6. Таблиця User

Атрибут	Тип	Опис
id	bigint	Ідентифікатор користувача
first_name	varchar(100)	Ім’я користувача
last_name	varchar(100)	Прізвище користувача
email	varchar(100)	Адреса електронної пошти користувача
phone_number	varchar(50)	Номер телефону
password	varchar(100)	Захешований пароль користувача
ava_url	varchar(2048)	Посилання на оригінальне фото користувача
mini_ava_url	varchar(2048)	Посилання на оброблене фото користувача маленьких розмірів
is_staff	bool	Прапорець, що визначає чи є користувач адміністратором
is_blocked	bool	Прапорець, що визначає, чи заблокований даний користувач
date_joined	datetime	Дата реєстрації користувача на платформі

Таблиця 7. Таблиця Company

Атрибут	Тип	Опис
id	bigint	Ідентифікатор компанії
name	varchar(100)	Назва компанії
email	varchar(100)	Адреса електронної пошти компанії
address	varchar(100)	Адреса головного офісу компанії
phome_mumber	varchar(50)	Телефон компанії
employees_num	bigint	Кількість працівників
logo_url	varchar(2048)	Посилання на оригінальний логотип компанії
mini_logo_url	varchar(2048)	Посилання на оброблений логотип компанії маленьких розмірів
location	varchar(100)	Загальна локація компанії
description	text	Опис компанії
is_blocked	bool	Прапорець, що визначає чи заблокована компанія
is_verified	bool	Прапорець, що визначає чи компанія верифікована адміністраторами
date_created	datetime	Дата створення компанії

.

Таблиця 8. Таблиця Industry

Атрибут	Тип	Опис
id	bigint	Ідентифікатор індустрії
name	varchar(100)	Назва індустрії

Таблиця 9. Таблиця Technology

Атрибут	Тип	Опис
id	bigint	Ідентифікатор технології
name	varchar(100)	Назва технології

Таблиця 10. Таблиця Project

Атрибут	Тип	Опис
id	bigint	Ідентифікатор проєкту
title	varchar(100)	Назва проєкту
url	varchar(100)	Посилання на сторінку з описом проєкту
description	text	Опис проєкту
logo_url	varchar(2048)	Посилання на оригінальний логотип проєкту
company_id	bigint	Ідентифікатор компанії, до якої належить проєкт

Таблиця 11. Таблиця Set

Атрибут	Тип	Опис
id	bigint	Ідентифікатор портфоліо
company_id	bigint	Ідентифікатор компанії, що є власником портфоліо
name	varchar(100)	Назва портфоліо

Таблиця 12. Таблиця SetTokens

Атрибут	Тип	Опис
id	bigint	Ідентифікатор токену
toket	varchar(100)	Токен, за яким можна отримати доступ до портфоліо

Таблиця 13. Таблиця Chat

Атрибут	Тип	Опис
id	bigint	Ідентифікатор чату
chat_name	varchar(100)	Назва чату

Таблиця 14. Таблиця Message

Атрибут	Тип	Опис
id	bigint	Ідентифікатор повідомлення
chat_id	bigint	Ідентифікатор чату, до якого належить повідомлення
message	text	Текст повідомлення
user_id	bigint	Ідентифікатор користувача, що є автором повідомлення
date	datetime	Дата відправки повідомлення

Таблиця 15. Таблиця CompanyFeedback

Атрибут	Тип	Опис
id	bigint	Ідентифікатор відгуку
feedback	text	Текст відгуку
user_id	bigint	Ідентифікатор користувача, що надіслав відгук
company_id	bigint	Ідентифікатор компанії, про яку надіслано відгук
date_created	datetime	Дата створення відгуку

Таблиця 16. Таблиця CompanyReport

Атрибут	Тип	Опис
id	bigint	Ідентифікатор скарги
report_message	text	Текст з описом скарги
user_id	User	Ідентифікатор користувача, що є автором скарги
company_id	Company	Ідентифікатор компанії, на яку надійшла скарга
date_created	datetime	Дата створення скарги

Таблиця 17. Таблиця UserReport

Атрибут	Тип	Опис
id	bigint	Ідентифікатор скарги
report_message	text	Текст з описом скарги
plaintiff	User	Ідентифікатор користувача, що є автором скарги
reported_user	User	Ідентифікатор користувача, на якого було створено скаргу
date_created	datetime	Дата створення скарги

Таблиця 18. Таблиця Attachment

Атрибут	Тип	Опис
id	bigint	Ідентифікатор
link	varchar(2048)	Посилання на файл
extension	varchar(100)	Розширення файлу

РЕАЛІЗАЦІЯ СИСТЕМИ

Інтерфейс користувача веб-сайту спочатку був побудований за допомогою Figma, після чого був реалізований у коді фронтенду. Фронтенд частина додатку використовує UI framework під назвою React. Для управління станом веб-додатку і відправки запитів до серверу було використано паттерн Redux та бібліотека Redux-Saga.

Бекенд частина системи використовує фреймворк Flask і бібліотеку для взаємодії з БД SQLAlchemy. На рисунку 3 прикладі продемонстрований ендпоінт для створення компанії. Інші ендпоінти програми побудовані аналогічно. Для побудови серверу використовувалися конвенції REST API, а для передачі даних використовувався формат JSON.

```
bp = Blueprint("companies", __name__, url_prefix="/companies")


@bp.route("", methods=["POST"])
@token_required
def create_company(user):
    data = request.get_json()
    name = data['name']
    email = data['email']
    address = data['address']
    phone_number = data['phoneNumber']
    employees_num = data['employeesNum']
    location = data['location']
    description = data['description']
    company = Company(name=name, email=email, address=address, phone_number=phone_number,
                      employees_num=employees_num, location=location,
                      description=description, user_id=user.id)
    db.session.add(company)
    db.session.commit()

    response = company.get_info()
    return jsonify(response)
```

Рисунок 3. Ендпоінт створення компанії

ТЕСТУВАННЯ

Напрям розробки. Бекенд частина додатку була протестована з використанням модифікованого методу сендвіча.

Обґрунтування вибору методології тестування. Ключовими факторами при виборі методології тестування слугували міра паралелізму та можливість якнайшвидше почати збирати модулі і скелетні версії програми. Спочатку як кандидат розглядався метод великого стрибка, адже він має високий ступінь паралелізму, що дозволило б учасникам працювати над модулями окремо, а в кінці проєкту розпочати інтеграцію. Проте перешкодою для використання даного методу стало сповільнення збірки модулів до моменту готовності всіх компонентів програми. Через це було обрано модифікований метод сендвіча. Було розглянуто основні недоліки даного методу: необхідність написання як драйверів, так і заглушок, а також дещо ускладнена можливість планування. Дані недоліки не перешкоджали плану розробки проєкту, а також дозволяли отримати нові знання в тестуванні, тому модифікований метод сендвіча було затверджено.

Результати тестування

Для проведення тестування було написано заглушки та драйвери. Було створено unit, інтеграційні та end to end тести. Для тестування було використано бібліотеку pytest. Для конфігурації стану програми перед кожним тестом створювалася тимчасова БД без даних, налаштовувалися змінні оточення і створювався окремий екземпляр додатку.

На Рисунку 6.3.1 наведений приклад end to end тестування функціоналу адміністратора для доступу до компаній. Після створення тестових даних у базі даних до серверу відправляється HTTP запит. Відповідь сервера співставляється з очікуваною, також перевіряється статус HTTP відповіді сервера - він має бути 200.

Інші end to end тести побудовані аналогічно.

Якщо запустити тести, побачимо звіт представлений на Рисунку 6.3.2. З рисунку видно, що всі наявні тести проходять успішно.

```python
def test_all_companies(test_client, app):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/admin/companies' page is requested (GET)
    THEN check the response is valid
    """
    __insert_companies_to_db__(app)
    response = test_client.get("/admin/companies")
    assert response.status_code == 200
    response_data = json.loads(response.data.decode("utf-8"))
    assert len(response_data["companies"]) == 4


def test_banned_companies(test_client, app):
    """
    GIVEN a Flask application configured for testing
    WHEN the '/admin/companies?statuses=banned' page is requested (GET)
    THEN check the response is valid
    """
    __insert_companies_to_db__(app)
    response = test_client.get("/admin/companies?statuses=banned")
    assert response.status_code == 200
    response_data = json.loads(response.data.decode("utf-8"))
    assert len(response_data["companies"]) == 2
```

Рисунок 4. End to end тести функціоналу для доступу до компаній

```
============================================================================================================ test session starts =============
platform win32 -- Python 3.9.2, pytest-7.2.2, pluggy-1.0.0
rootdir: C:\Users\dmitriy\PycharmProjects\backend
collected 21 items

tests\functional\test_admin_companies.py ...
tests\functional\test_admin_requests.py ....
tests\functional\test_admin_users.py ......
tests\functional\test_projects.py ...
tests\functional\test_users.py ....
tests\unit\test_models.py .

============================================================================================================ 21 passed in 2.57s =============
```

Рисунок 5. Результат виконання тестів

ВИКОРИСТАННЯ СИСТЕМИ

При переході на стартову сторінку сайту користувач бачить опис основного функціоналу застосунку, а також основні кнопки для початку роботи з застосунком. Користувач може обрати сторінки входу (Sign in), реєстрації (Sign up) або сторінку з пошуком компаній (Companies).

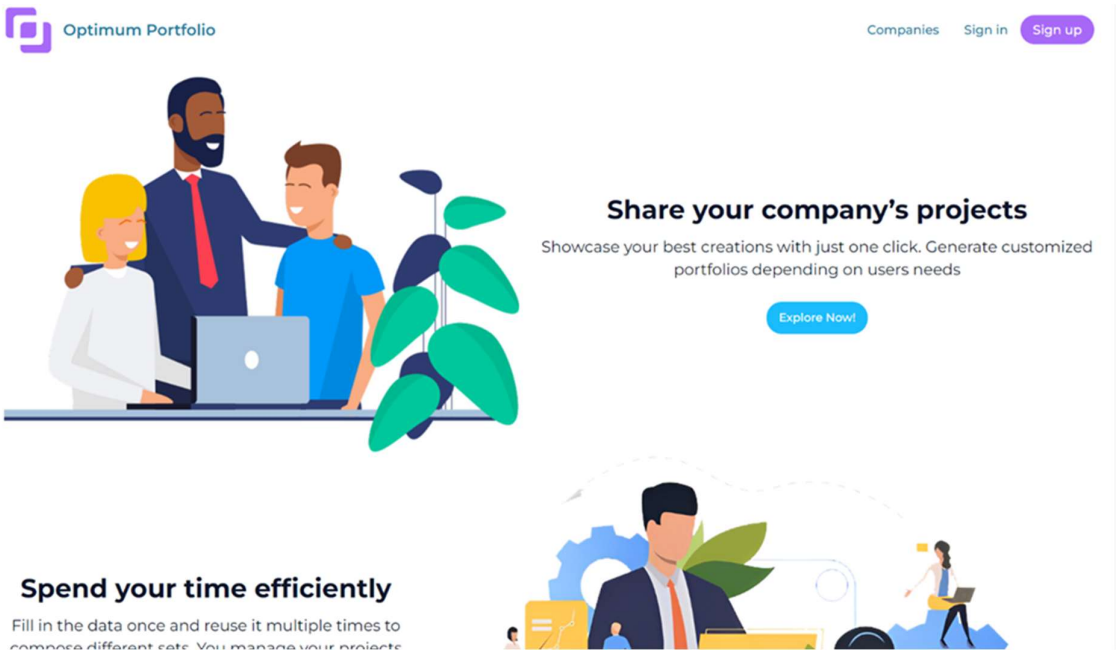


Рисунок 6. Стартова сторінка застосунку

Щоб отримати доступ до основного функціоналу користувач повинен зареєструватися на сайті. Без реєстрації користувачеві надається можливість переглядати компанії та проєкти, але створення власних компаній, а також написання приватних повідомлень буде недоступним.

Для реєстрації на першій сторінці потрібно вказати валідну адресу поштової скриньки та пароль (має бути не коротшим за 6 символів, містити в собі велику та малу літери, цифру і спеціальний символ).

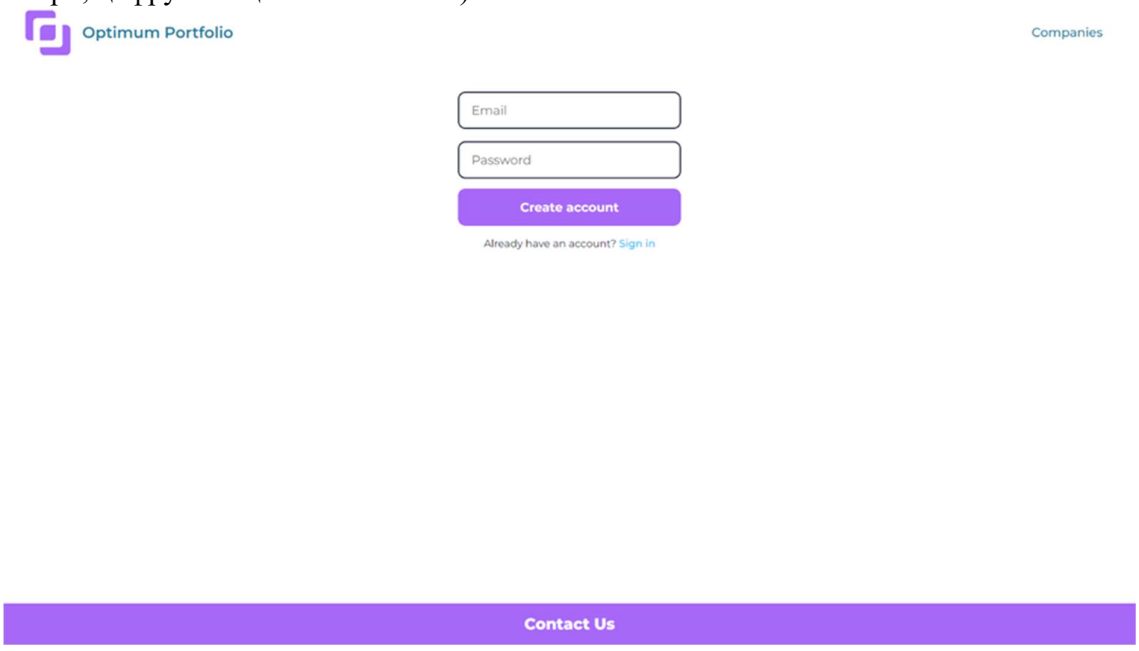


Рисунок 7. Сторінка реєстрації користувача

На наступній сторінці реєстрації потрібно буде ввести ім’я, прізвище, номер телефону, а також за бажанням інформацію про користувача.

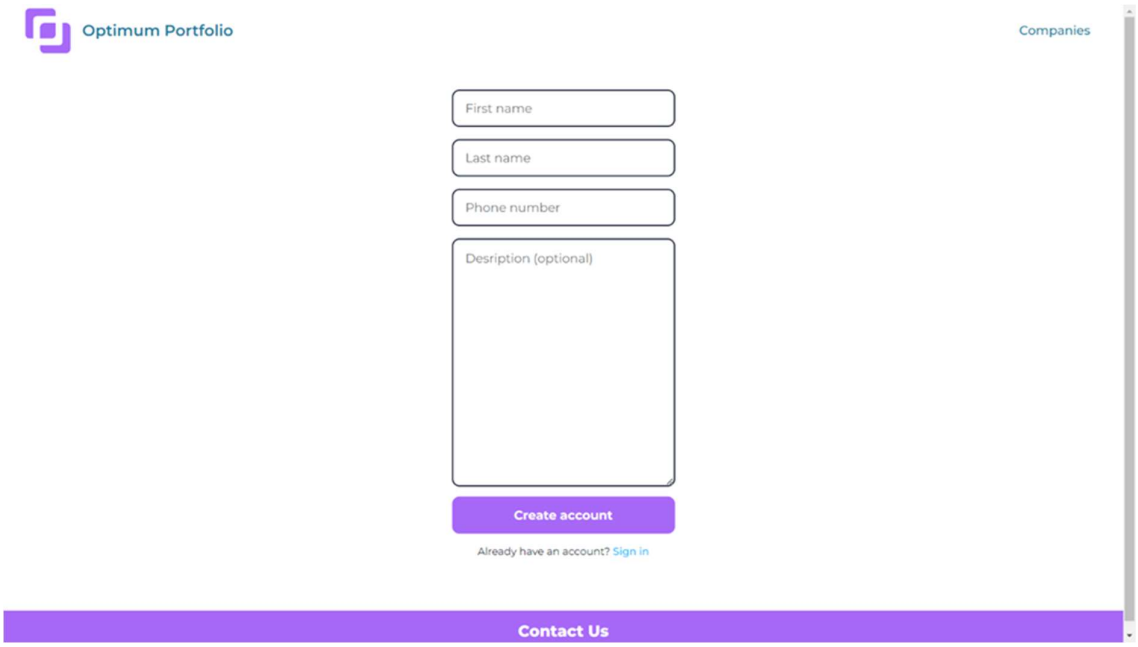


Рисунок 8. Друга сторінка реєстрації

Якщо всі поля були коректно заповнені, то буде створено користувача. Інакше буде висвітлена помилка з поясненнями.

Використовуючи email та пароль, що були вказані при реєстрації, можна увійти в особистий обліковий запис при подальшому використанні сайту.

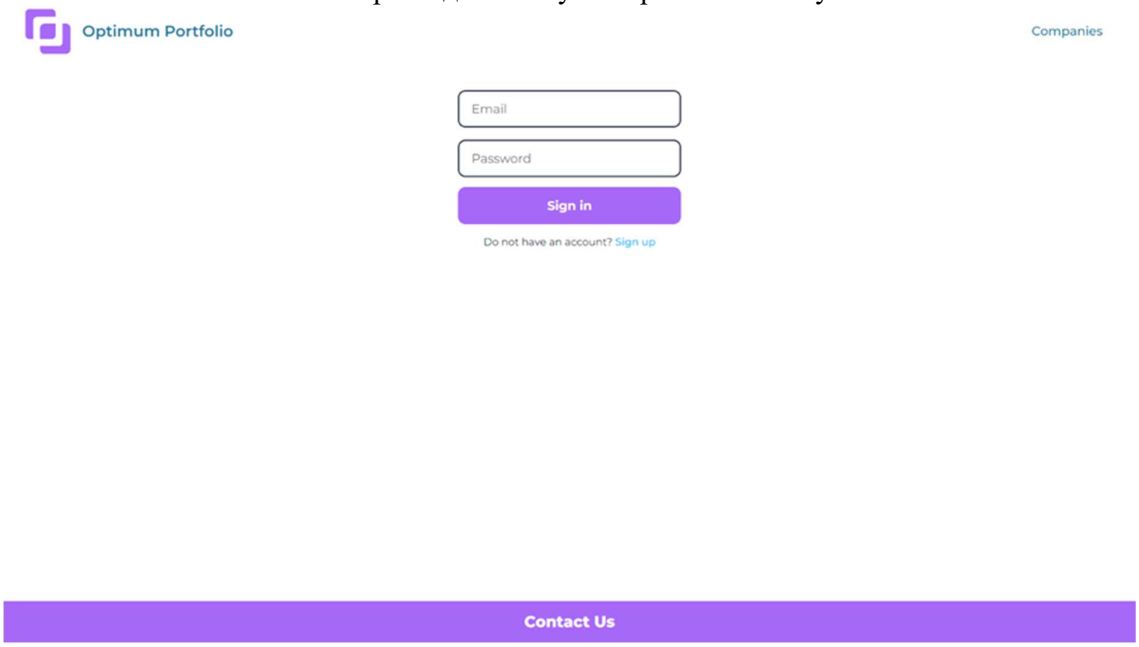


Рисунок 9. Сторінка логіну користувача

При успішній реєстрації або вході користувач буде перенаправлений на головну сторінку сайту (рисунок 10), де будуть відображатися посилання на основні сторінки сайту: чати (Chats), пошук проєктів (Projects), керування компаніями (Own Companies) та пошук компаній (Search Companies).

На сторінці пошуку компаній користувач може знайти потрібну компанію за її назвою або знайти список компаній, вказавши потрібну йому індустрію або технологію.

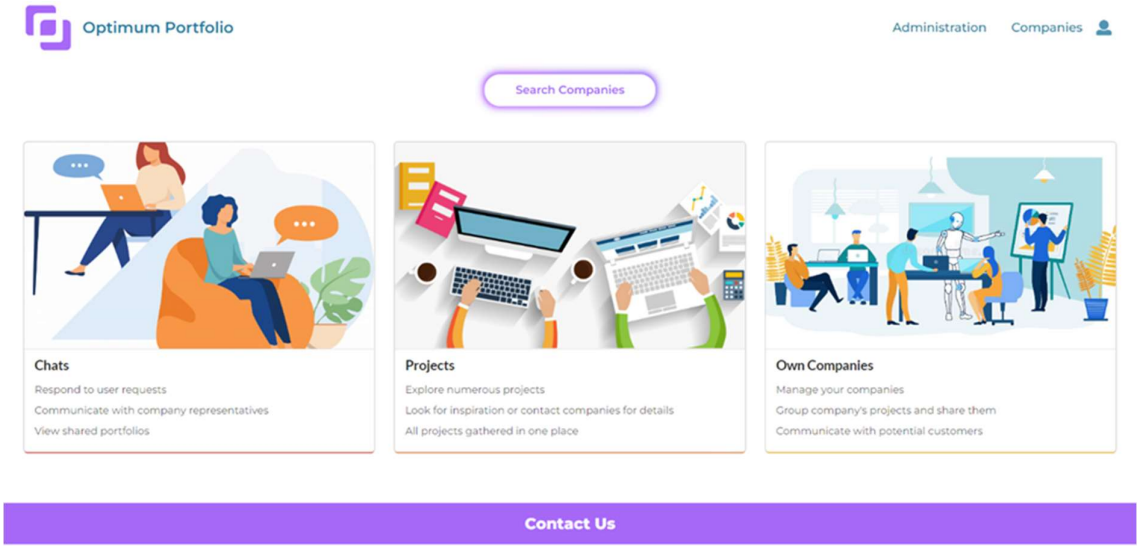


Рисунок 10. Головна сторінка користувача

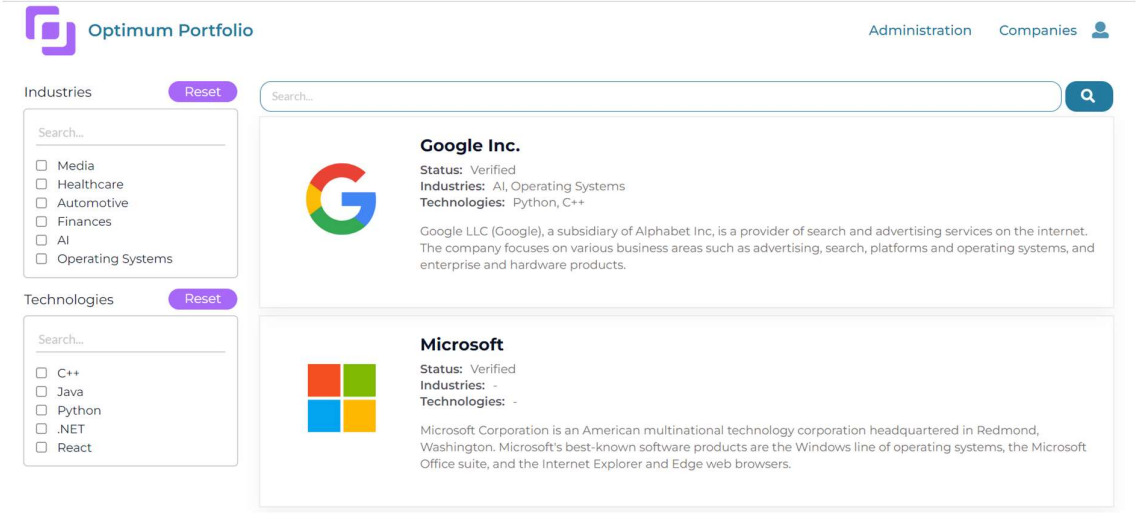


Рисунок 11. Сторінка компаній

Якщо користувач натисне на картку однієї із компаній, які зареєстровані в системі, то він опиниться на сторінці компанії. При потребі можна переглянути створені цією компанією проєкти або сконтактувати з представником компанії за допомогою внутрішнього чату (рисунок 12).

На сторінці чату, зображеній на рисунку 13 користувач зможе написати користувачу або представнику компанії, які його зацікавили.

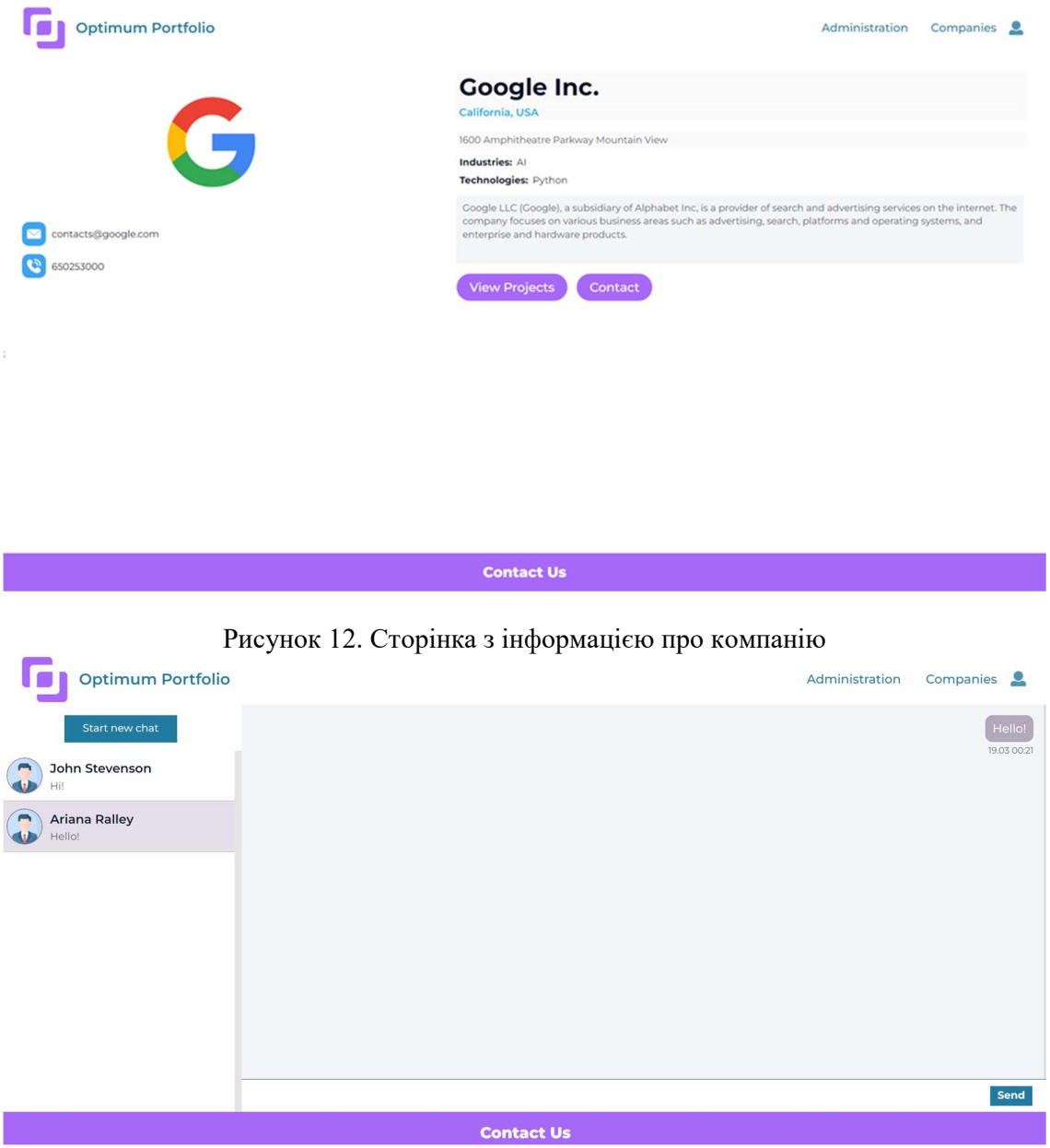


Рисунок 12. Сторінка з інформацією про компанію

Рисунок 13. Сторінка з чатами

Сторінка пошуку проєктів та сторінка проєктів певної компанії є подібними до таких же сторінок для компаній. Тут можна вказати потрібні фільтри або знайти проєкт за іменем (рисунок 14).

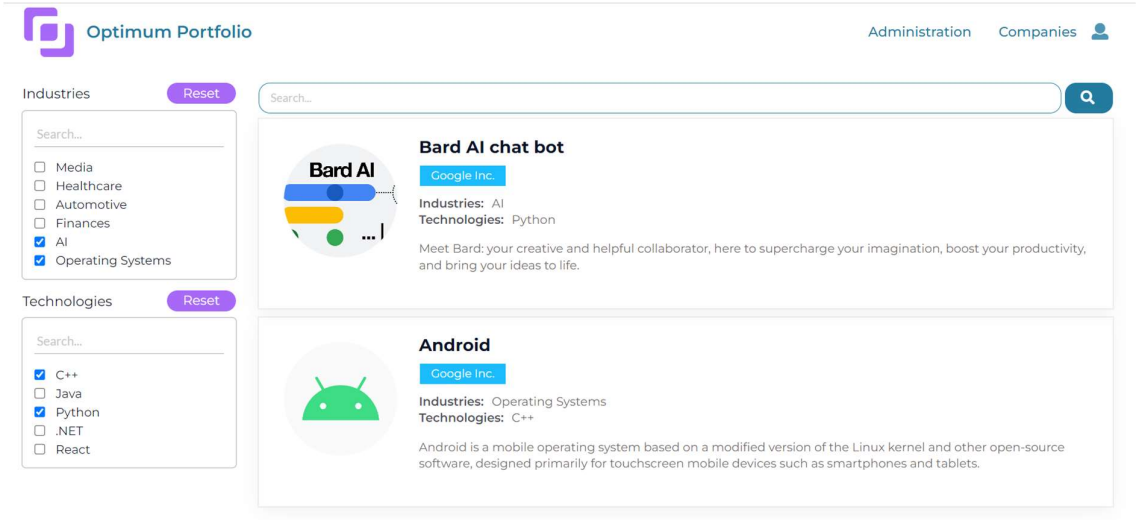


Рисунок 14. Сторінка зі списком проєктів

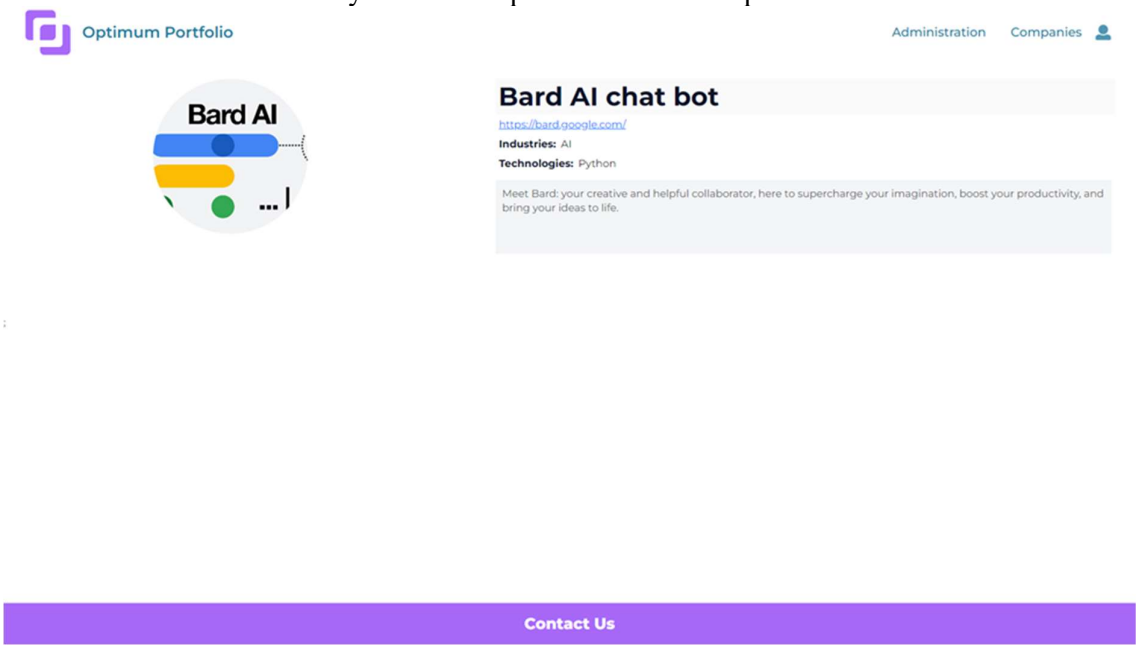


Рисунок 15. Сторінка з описом проєкту

Користувач може переглянути компанії, представником яких він є. Для того, щоб створити нову компанію, треба зайти у розділ з власними компаніями та натиснути ‘+’ у правому нижньому кутку (рисунок 16).

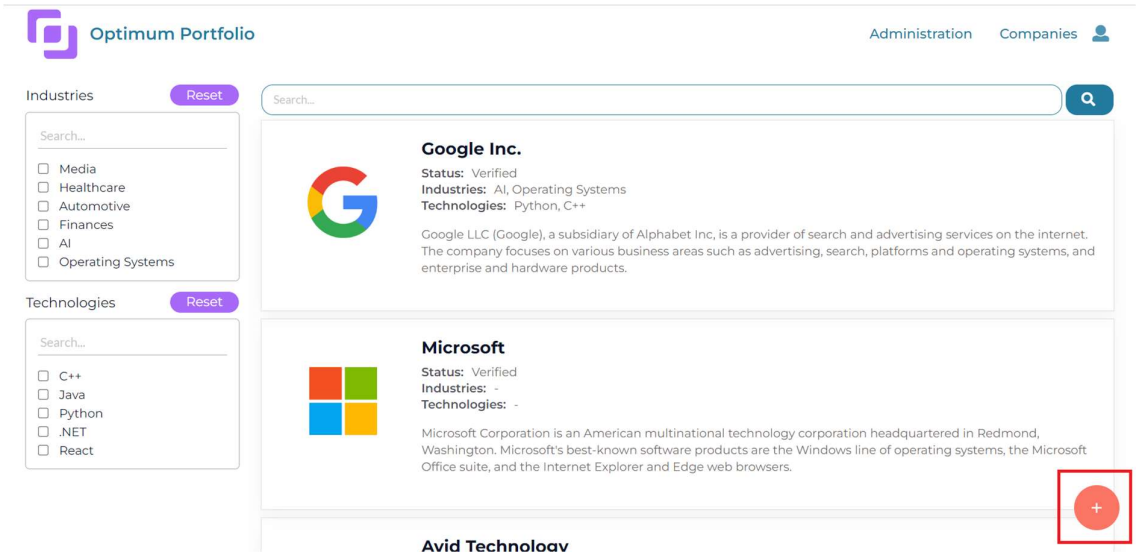


Рисунок 16. Кнопка для додавання нової компанії

Для створення компанії необхідно заповнити її ім’я, електронну адресу, номер телефону, локацію, адресу головного офісу, кількість співробітників та опціонально опис (рисунок 17).

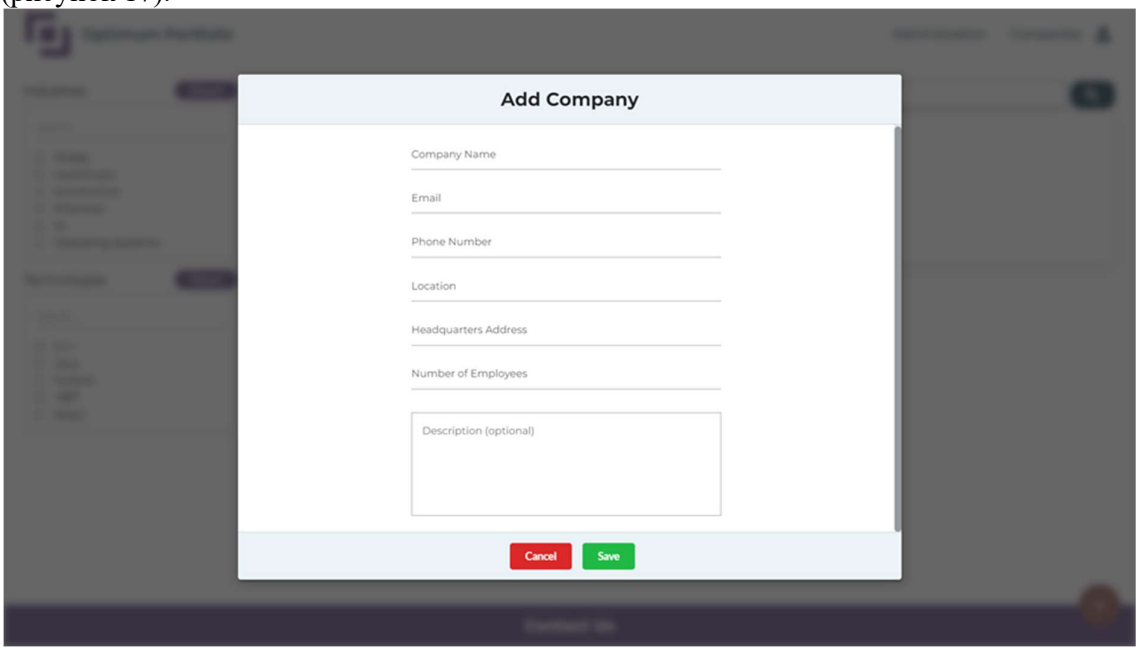


Рисунок 17. Модальне вікно створення нової компанії

Користувач, що є представником компанії, матиме змогу редагувати дані про неї безпосередньо на сторінці компанії (кнопка Edit, рисунок 18).

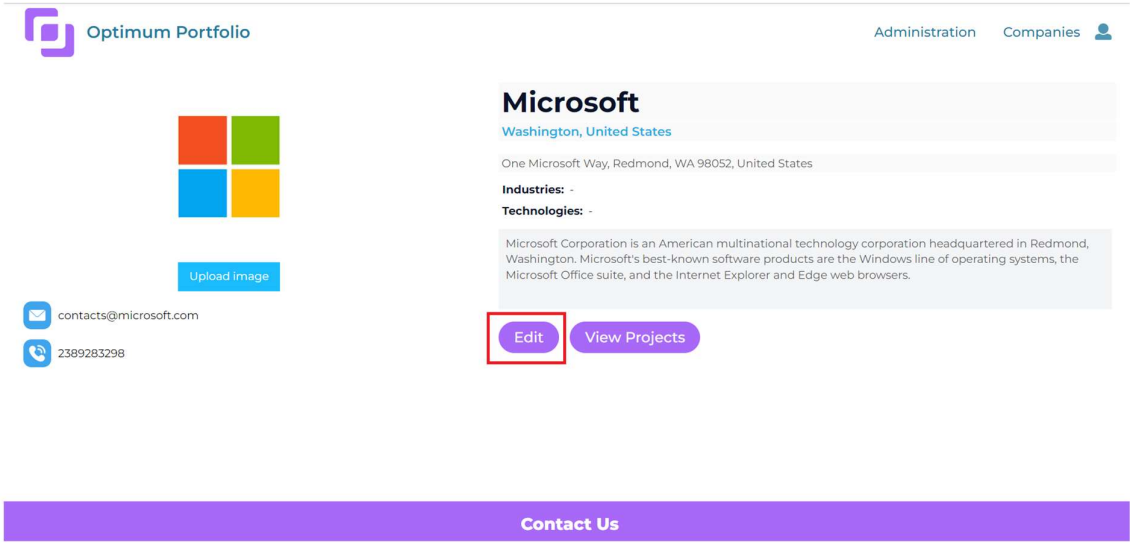


Рисунок 18. Кнопка для редагування компанії

Після натискання на кнопку редагування компанії з'явиться підсвітка полів, доступних для редагування (рисунок 19).

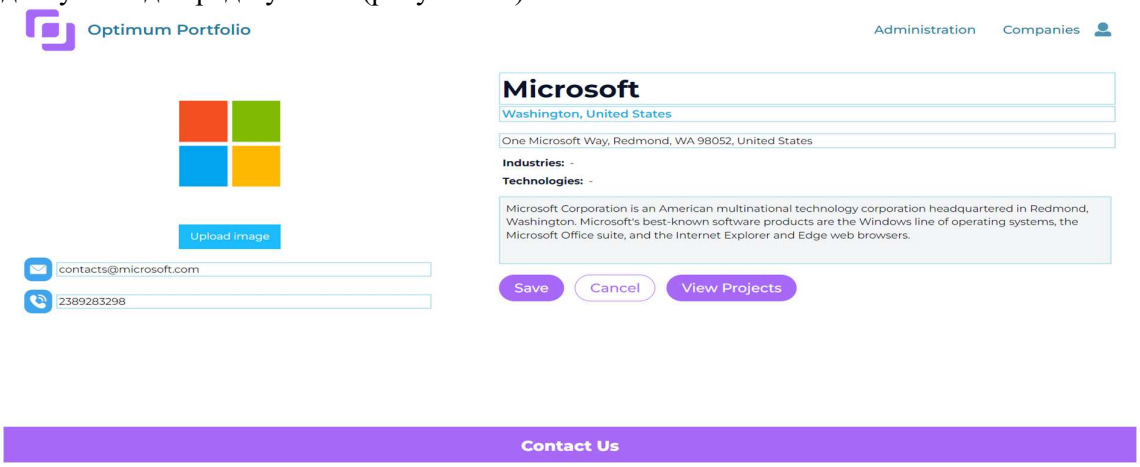


Рисунок 19. Вигляд сторінки під час редагування компанії

ФОРМАЛЬНА СПЕЦИФІКАЦІЯ ТА ВЕРИФІКАЦІЯ

Обґрунтування вибору інструмента специфікації та верифікації.

Специфікацію та верифікацію було проведено з використанням мови Dafny. Dafny - це засіб перевірки програм, що включає мову програмування та специфікацію конструкції. Дану мову було обрано через підтримку програмування зі специфікаціями, що дозволяє писати контракти, які визначають очікувану поведінку коду. Ці специфікації автоматично перевіряються Dafny під час процесу перевірки та дозволяють гарантувати, що код відповідає зазначеним вимогам [11]. Також було використано функцію компіляції Dafny у мову програмування Python, на якій розроблявся основний функціонал проєкту.

Формальна специфікація обраного модуля системи. Для виконання завдання було обрано задачу реєстрації компанії. На рисунку 20 приведено сигнатуру методу add_company для верифікації задачі додавання компанії з вказаними передумовами та

постумовами. Було обрано наступні передумови: назва компанії має бути не менше певної довжини, номер телефону повинен відповідати заданому формату. Постумови: компанія має бути збережена лише у випадку, коли вона є унікальною (тобто її назва) та якщо її електронна пошта валідна.

```
method add_company(
  name: string,
  email: string,
  address: string,
  phone_number: string,
  employees_num: int,
  logo_url: string,
  mini_logo_url: string,
  location: string,
  description: string
)
  returns (company: Company, unique:bool, created: bool, validEmail:bool)
  modifies this
  requires |this.companies|>=0 && |name|>3 && 16>=|phone_number|>=8
  && forall i::1<i<|phone_number|==> '0'<=phone_number[i]<='9'&&phone_number[0]=='+'


  ensures (unique && validEmail)== created
```

Рисунок 20. Передумови та постумови.

Опис результатів формальної верифікації обраного модуля.

Результати роботи верифікатора можна побачити на рисунку 21. Видно, що робота виконана успішно та без помилок. Таким чином було специфіковано та верифіковано функціонал створення компанії у застосунку.

```
Dafny program verifier finished with 7 verified, 0 errors
2
```

Рисунок 21. Результат роботи верифікатора.

ОПИС ПРОЦЕСУ РОЗРОБКИ

Учасники команди та розподіл ролей.

До складу команди “Conic” увійшли: Анна Алєксєєнко, Олександр Лихопуд, Богдан Огородній, Володимир Чучканов, Дмитро Шабанов, Максим Юдкін, Нікіта Орляк і Ярослав Приходько. Розподіл ролей відбувався з урахуванням побажань студентів та був збережений впродовж усієї роботи над проєктом.

Таблиця 19. Розподіл ролей в команді

Учасник	Роль учасника
Анна Алєксєєнко	бекенд, фронтенд, тестувальник, модератор, керівник проєкту
Олександр Лихопуд	бекенд, фронтенд, тестувальник
Богдан Огородній	бекенд, тестувальник
Володимир Чучканов	бекенд, фронтенд, тестувальник, фахівець з документації
Дмитро Шабанов	бекенд, фронтенд, тестувальник

Максим Юдкін	бекенд, тестувальник
Нікіта Орляк	фронтенд, тестувальник, фахівець з документації
Ярослав Приходько	бекенд, фронтенд, тестувальник

Опис обраної методології та інструментарію

Для керування управління проєктом було обрано методологію Kanban. Зустрічі команди проводилися з використанням платформи Google Meetings та включали в себе щотижневі для синхронізації прогресу, а також ретроспективи та окремі сесії планування задач. На першій зустрічі командою був узгоджений час та день тижня, коли будуть проводитися зустрічі. Також було прийнято графік зустрічей задля ефективного виконання завдань. Окрім цього, був створений чат у Telegram для асинхронної комунікації, в якому приймались проміжні рішення та обговорювались питання. Для більшої структурованості та пріоритезації завдань, які необхідно реалізувати в проєкті, та їх розподілу між учасниками команди використовувалася система Trello (рисунок 22). Задачі розроділялися між 5 стовпцями:

- TODO – заплановані задачі, які кожен з учасників команди може взяти собі відповдно до ролі в команді;
- In Progress – задачі, що наразі знаходятья в процесі виконання;
- In Review – виконані задачі, що чекають на відгуки та коментарі членів команди;
- Done – зроблені задачі;
- On Hold – заплановані задачі, які поки що не можна виконати через певні обставини;

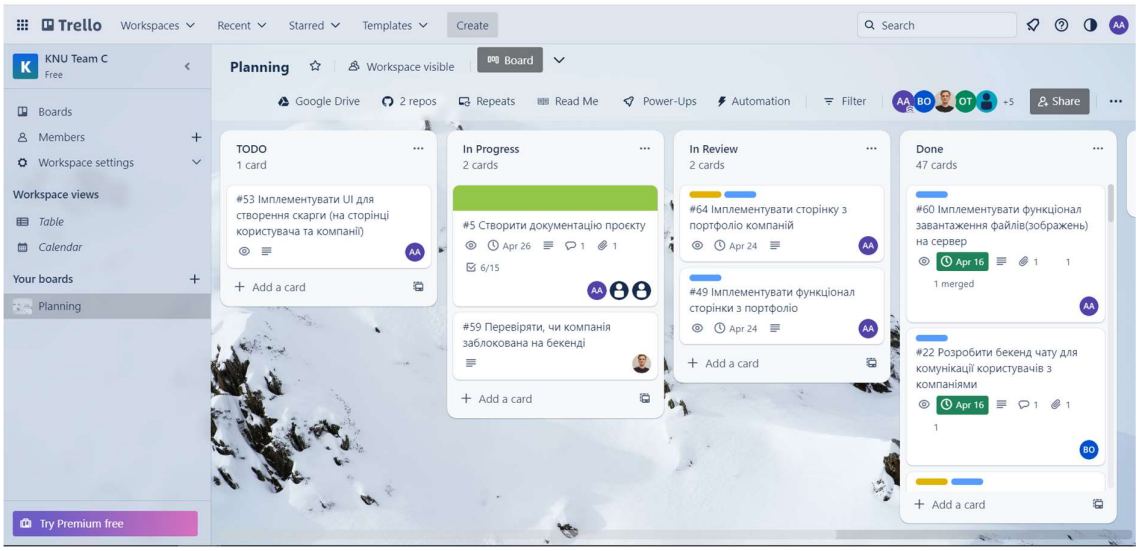
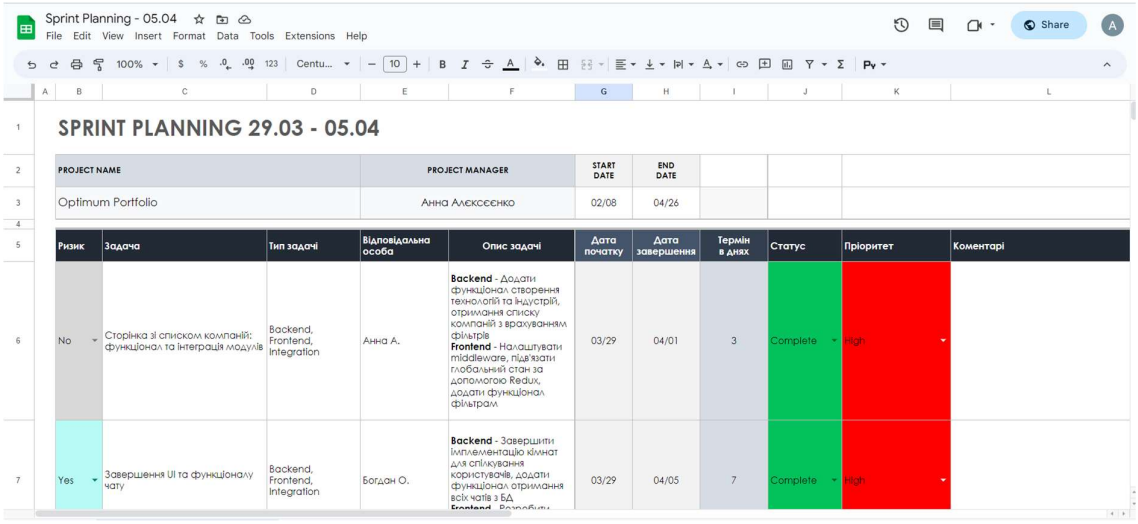


Рисунок 22. Trello дошка команди з задачами

Демонстрація планування спринтів

Планування спринтів проводилося з використанням таблиць в Google Sheets. Було розписано задачі, які треба виконати за тиждень спринта, відповідальні за задачу особи, деталізований опис задачі, а також бажану дату початку та завершення задачі. Таблицю зображено на рисунку 23. У таблиці також враховувалися ризики та пріоритети кожної з задач. Також для зручності створювалися графіки виконання задач (рисунок 24).



Sprint Planning - 05.04

File Edit View Insert Format Data Tools Extensions Help

Share

SPRINT PLANNING 29.03 - 05.04

PROJECT NAME	PROJECT MANAGER	START DATE	END DATE
Optimum Portfolio	Анна Алексєєнко	02/08	04/26

Ризик	Задача	Тип задачі	Відповідальна особа	Опис задачі	Дата початку	Дата завершення	Термін в днях	Статус	Пріоритет	Коментарі
No	Сторінка зі списком компаній: функціонал та інтеграція модулів	Backend, Frontend, Integration	Анна А.	**Backend** - Додати функціонал створення технологій та індустрій, отримання списку компаній з врахуванням фільтрів **Frontend** - Налаштувати middleware, підв'язати глобальний стан за допомогою Redux, додати функціонал фільтрам	03/29	04/01	3	Complete	High	
Yes	Завершення UI та функціоналу чату	Backend, Frontend, Integration	Богдан О.	**Backend** - Завершити імплементацію кімнат для спілкування користувачів, додати функціонал отримання всіх чатів з БД **Frontend** - Розробити	03/29	04/05	7	Complete	High	

Рисунок 23. Планування задач на спринт

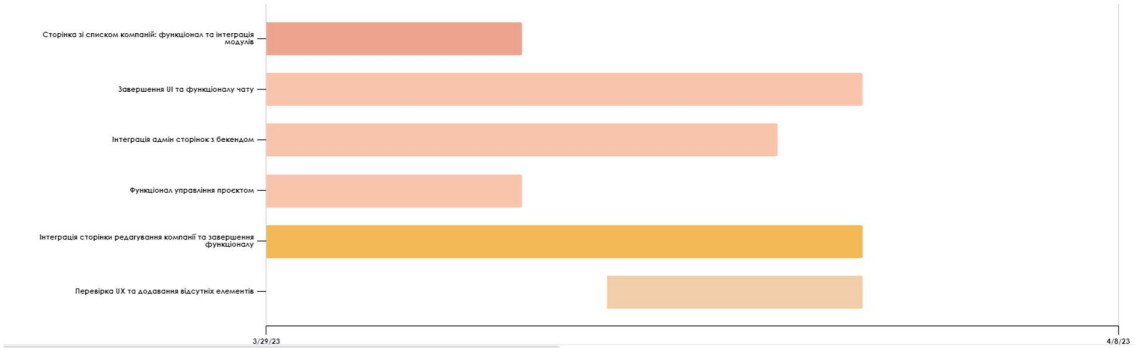


Рисунок 24. Графік виконання задач в спринті

Рефлексія

У ході виконання проєкту учасники навчилися працювати разом у команді, синхронізувати виконану роботу та допомагати один одному з задачами. На початку проєкту команда мала невеликі проблеми з комунікацією – не всі учасники мали змогу доєднуватися до спільних зустрічей, що впливало на швидкість виконання задач. Проте дана проблема була вирішена рядом опитувань, проведених серед учасників з ціллю визначення найкращого часу для зустрічей та затвердження алгоритму асинхронної комунікації у разі відсутності учасника на зустрічі. З часом учасники почали об'єднуватися у групи для вирішення спільних проблем та взаємодопомоги, проводилися зустрічі для ознайомлення учасників з інструментарієм розробки.

Результати командної роботи

У ході виконання проєкту учасники на практиці ознайомилися з життєвим циклом програмного забезпечення від аналізу вимог до обслуговування та підтримки застосунку, покращили технічні навички, а також навички комунікації, ознайомилися з методологіями розробки, використали різноманітні інструменти для планування та ведення спринтів, а також навчилися враховувати ризики для задач.

ВИСНОВКИ

Командою була спроєктована, а потім розроблена система управління портфоліо компаній “Optimum Portfolio”, яка є комплексним рішенням для організації проєктів компаній в портфоліо, якими можна ділитися за допомогою посилань. Під час виконання проєкту командою було проведено аналіз схожих програмних продуктів на ринку, їхніх переваг та недоліків.

Підвищено рівень соціальних та комунікативних навичок у студентів, що брали участь у проєкті. Кожен учасник команди зміг спробувати себе у цікавій йому ролі та отримати цінний досвід. Було отримано або поглиблено знання таких технологій розробки:

- Бекенд: Python, Flask, Dafny, PostgreSQL.
- Фронтенд: HTML, CSS, JavaScript, React, Redux-Saga.
- Деплой: Google Cloud Run, Terraform, Docker.

У результаті було розроблено готовий програмний продукт, що відповідає поставленим технічним вимогам, цілям та реалізує задуманий функціонал.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Python 3.11.3 Документація – [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/
2. Flask (2.2.x) Документація – [Електронний ресурс] – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.2.x/
3. JavaScript - MDN Web Docs – [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
4. React Документація – [Електронний ресурс] – Режим доступу до ресурсу: https://reactjs.org/
5. Google Cloud Документація – [Електронний ресурс] – Режим доступу до ресурсу: https://cloud.google.com/docs
6. Pytest Документація [Електронний ресурс] – Режим доступу до ресурсу: https://docs.pytest.org/en/7.1.x/contents.html
7. Generating API documentation (Postman) – [Електронний ресурс] – Режим доступу до ресурсу: https://learning.postman.com/docs/publishing-your-api/
8. Learn how to diagram using diagrams.net – [Електронний ресурс] – Режим доступу до ресурсу: https://www.diagrams.net/blog/features
9. Figma Best Practices – [Електронний ресурс] – Режим доступу до ресурсу: https://www.figma.com/best-practices/guide-to-developer-handoff/components-styles-and-documentation/
10. Trello – Режим доступу до ресурсу: https://trello.com/uk
11. Dafny Документація – [Електронний ресурс] – Режим доступу до ресурсу: https://dafny.org/
12. Омельчук Л.Л. Формальні методи специфікації програм. – К.: УкрІНТЕІ, 2010. -

78 с. – Режим доступу до ресурсу: http://www.csc.knu.ua/en/library/books/omelchuk-10.pdf

УНІВЕРСИТЕТСЬКИЙ МЕСЕНДЖЕР "TINKER CHAT"

Антон Чернов, Андрій Христофор, Дмитро Голійчук, Назар Лаврентюк, Владислав Бурцевич, Аліна Бабенко, Владислав Таран, Михайло Лапшин

ВСТУП

Однією з найбільш важливих задач університетів є розробка веб-додатку для комунікації студентів та викладачів. Такий месенджер має на меті допомогти користувачам швидко та зручно обмінюватися інформацією, надавати доступ до необхідних ресурсів та полегшувати співпрацю між студентами та викладачами.

Методи дослідження, що використовуються для розробки месенджера університету, включають аналіз вимог користувачів, проектування баз даних, розробку функціональності та інтерфейсу користувача, тестування та підтримку програмного забезпечення.

Основними цілями розробки месенджера університету є:

- забезпечення користувачів швидким та зручним способом обміну інформацією;
- полегшення співпраці між студентами та викладачами; покращення якості освіти та забезпечення доступу до необхідних ресурсів;
- підвищення рівня задоволення користувачів від використання месенджера.

Крім прикладних цілей, розробка месенджера університету має на меті також закріпити знання та вивчити нові технології розробки програмного забезпечення. Для досягнення цієї мети необхідно:

- розробити структуру бази даних, яка дозволяє зберігати інформацію про користувачів;
- розробити функціональність, що дозволяє користувачам обмінюватися повідомленнями;
- розробити функціональність, що дозволяє користувачам інших користувачів та почати спілкування з ними;
- створити зручний та привабливий інтерфейс користувача;
- провести тестування веб-додатку та виправити можливі помилки;
- забезпечити підтримку веб-додатку та вирішення технічних проблем, що можуть виникнути під час експлуатації.

Розробка веб-додатку університетського мессенджера є важливим проєктом, який забезпечує зручний та швидкий спосіб комунікації студентів, викладачів та інших співробітників університету, а також дозволяє покращити процес дистанційного навчання. Крім того, розробка веб-додатку університетського мессенджера є вагомою задачею для фахівців з розробки програмного забезпечення, оскільки дозволяє закріпити та поглибити знання з технологій розробки веб-додатків.

ОГЛЯД НАЯВНИХ НА РИНКУ СИСТЕМ

На основі наданих даних можна виділити кілька конкурентів для додатку "Університетський мессенджер".

Основними конкурентами насамперед є:

- Telegram;
- Google Chat;
- Facebook Messenger;

Але їхнім основним недоліком ми би хотіли назвати ускладнена інтеграція з існуючою екосистемою університету. Основною метою даного проєкту є створення основи для такого веб-додатку, який міг би бути інтегрованим в навчальний процес та централізувати спілкування в рамках університету.

ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Технології розробки: Angular, Angular material, Node.js, express.js, Socket.io; технології проектування: diagram.io, figma; технології тестування, верифікації та специфікації: Karma, Express-contracts, OCL.js.

Angular є однією з найбільш популярних фреймворків JavaScript для розробки веб-додатків. Він пропонує компонентний підхід до створення складних користувацьких інтерфейсів, що спрощує процес розробки та забезпечує високу ефективність. Angular Material - це колекція готових компонентів для Angular, які забезпечують консистентний дизайн та зручний UX.

Node.js - це середовище виконання JavaScript на стороні сервера, що дозволяє розробляти швидкі та масштабовані веб-додатки. Express.js - це фреймворк для Node.js, який дозволяє легко створювати веб-сервери та API.

Socket.io - це бібліотека JavaScript для реалізації двостороннього зв'язку між клієнтом та сервером за допомогою WebSocket-протоколу. Це дозволяє розробникам створювати веб-додатки з реальним часом та зменшувати затримки у спілкуванні між клієнтом і сервером.

Для проектування інтерфейсу додатку ми використовуємо декілька інструментів. Першим з них є diagram.io, який дозволяє створювати діаграми та схеми різного рівня складності.

Другим інструментом є Figma - інструмент для дизайну інтерфейсів, який ми використовуємо для розробки дизайну додатку. За допомогою Figma ми створюємо макети інтерфейсу, дизайн іконок та інші елементи, які згодом інтегруються в додаток.

Для верифікації додатку ми використовуємо Express-contracts - бібліотеку для верифікації Express додатків за допомогою контрактів. Для тестування Angular додатків ми використовуємо Karma - фреймворк для запуску тестів на різних браузерах та платформах. Karma дозволяє створювати тести для Angular компонентів, сервісів та інших елементів додатку, а також забезпечує покриття тестами всього додатку. Для специфікації – OCL.js.

ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

Призначенням університетського мессенджеру є створення зручного та безпечного інструменту для студентів та викладачів для спілкування та обміну інформацією в рамках університетської спільноти. Система буде забезпечувати можливість особистого та групового чатування, матеріалами для навчання, а також можливість створення групових проектів та спільної роботи над ними.

Крім того, система повинна забезпечувати високий рівень захисту даних та приватності користувачів, забезпечувати можливість налаштування рівнів доступу до чатів та інформації, а також забезпечувати можливість зв'язку з адміністрацією університету для розв'язання проблем та отримання допомоги. В цілому, система має стати необхідним інструментом для ефективного навчання та спілкування в університетській спільноті.

Таким чином ми намагалися створити максимально зручний ресурс відповідно до побажань користувачів.

Робота передбачає:

- Аналіз потреб користувачів
- Проектування функціональності - на основі аналізу потреб користувачів необхідно розробити дизайн та функціональні вимоги до месенджера. Це включає в себе визначення типів повідомлень, можливостей для спільної роботи над документами, календарних подій та інше.

- Розробка месенджера - на основі визначених вимог, необхідно розробити месенджер та забезпечити його функціональність.
- Тестування та налагодження - після створення месенджера, необхідно провести тестування та налагодження, щоб переконатись, що всі функції працюють коректно та месенджер задовольняє потреби користувачів.
- Розгортання та впровадження - після успішного тестування месенджер готовий до розгортання та впровадження.

Цілі створення системи:

- забезпечення зручного спілкування між студентами, викладачами та іншими учасниками університетської спільноти;
- забезпечення можливості обміну актуальною інформацією та документами між користувачами;
- забезпечення можливості створення груп для спільної роботи над проектами та завданнями;
- забезпечення зручного інтерфейсу та можливості персоналізації користувачами свого профілю.

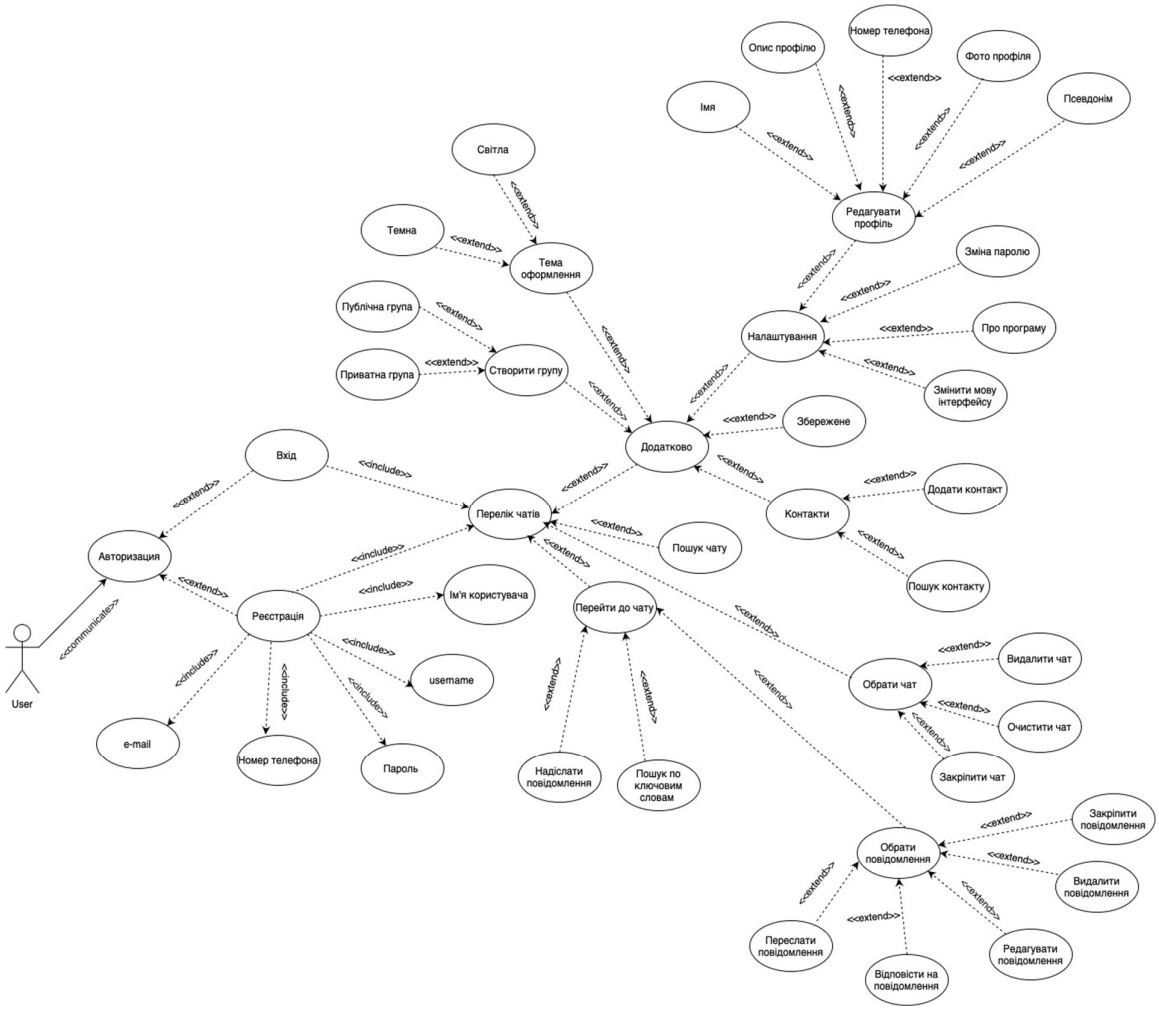


Рисунок 1. Use-Case diagram

Загальні вимоги до системи

Система "Університетський месенджер" повинна бути простою та корисною в застосуванні. В Системі передбачається виділити наступні функціональні підсистеми:

- **конфігураційна** — призначена для збереження, обробки та оновлення інформації , представленій на сайті та пов'язаною з даними користувачів , а також забезпечення функціонування веб-застосунку;
- **підсистема користувача —** призначена для забезпечення можливості авторизуватися в системі, перегляду наявних чатів та груп, створення нових, обмінюватись повідомленнями.

Вимоги до функцій, які виконуються системою

Конфігураційна підсистема.

Таблиця 1. Перелік функцій, задач що підлягають автоматизації

Функція	**Задача**
Конфігурація даних користувача	Редагування та видалення інформації про користувача
	Налаштування веб-додатку (вибір мови інтерфейсу, теми).

Підсистема користувача

Таблиця 2. Перелік функцій, задач що підлягають автоматизації

Функція	**Задача**
Реєстрація	введення інформації про себе
	видалення інформації про себе
Робота з чатами	перегляд наявних чатів
	пошук контактів та створення чатів з ними
	створення групових чатів
	Обмін повідомленнями в обраному чаті
	перехід до підсистеми конфігурації

Системні вимоги

Браузери: Google Chrome - версія 79 або новіша Mozilla Firefox - версія 72 або новіша Apple Safari - версія 13 або новіша Microsoft Edge - версія 79 або новіша Opera - версія 66 або новіша

На довільні некоректні дії користувача, пов'язані з введенням невірних даних, не заповненням обов'язкових полів введення в формах та інші, які можуть бути оброблені системою, генеруються відповідні повідомлення про помилки обраною мовою інтерфейсу, в межах загального дизайну сайта.

Операційна система: немає вимог щодо операційної системи, сучасні технології дають можливість уникати відмінностей різних ОС. Наприклад, на проекті працювали люди, які мали Windows та Linux (Ubuntu, Debian), при цьому жодного разу не виникло проблем.

Джерелом даних для Системи повинна бути інформаційна система. Було вирішено використовувати Mongo СУБД. Для взаємодії з базою даних використовувалась бібліотека Mongoose.

ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

Логічна структура бази даних

У нашому проекті використовується система контролю базами даних MongoDb. Це одна із найпопулярніших нереляційних баз даних, яка легко інтегрується з Node.js за допомогою бібліотеки Mongoose.

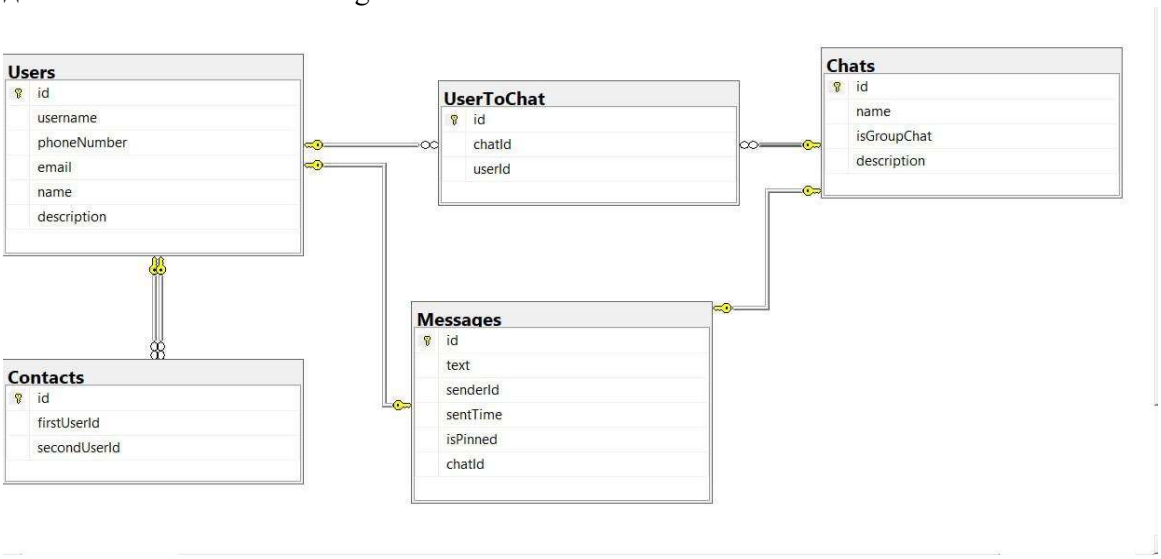


Рисунок 2. Діаграма бази даних сервісу

Users, Chats, Messages - основні структурні таблиці сутностей. UserToChat, Contacts - допоміжні таблиці.

Таблиця 3. Перелік таблиць бази даних

Номер	Таблиця	Опис
1	Users	Таблиця, в якій містяться всі зареєстровані користувачі та дані про них.
2	Chats	Таблиця, в якій містяться чати усіх користувачів.
3	Messages	Таблиця усіх надісланих повідомлень.
4	UserToChat	Таблиця зв’язків між користувачами та чатами(належність користувача до певного чату).
5	Contacts	Таблиця зв’язків між користувачами(чи встановлено контакт між ними).

Опис таблиць

Таблиця 4. Таблиця Users

Атрибут	Тип	Опис
_id	string	Ідентифікатор.
email	string	Email користувача.
password	string	Хеш пароля користувача.
name	string	Повне ім’я користувача.
username	string	Ім’я користувача для використання у системі.

Таблиця 5. Таблиця Chats

Атрибут	**Тип**	**Опис**
_id	string	Ідентифікатор.
name	string	Ім’я чату(у випадку діалогу 2 користувачів - конкатенація їх ідентифікаторів).
isGroupChat	boolean	Показник чи є чат груповим.
messages	Array<string>	Список повідомлень у чаті.

Таблиця 6. Таблиця Messages

Атрибут	**Тип**	**Опис**
_id	string	Ідентифікатор.
text	string	Текст повідомлення.
senderId	string	Ідентифікатор відправника.
chatId	string	Ідентифікатор чату, в який відправлене повідомлення.
isPinned	boolean	Показник чи є повідомлення закріпленим.
sentTime	DateTime	Час відправки повідомлення у форматі UTC.

Таблиця 7. Таблиця UserToChat

Атрибут	**Тип**	**Опис**
_id	string	Ідентифікатор.
chatId	string	Ідентифікатор чату, до якого належить користувач.
userId	string	Ідентифікатор користувача.
isPinned	boolean	Показник чи є чат закріпленим у користувача.

Таблиця 8. Таблиця Contacts

Атрибут	**Тип**	**Опис**
_id	string	Ідентифікатор.
firstUser	string	Ідентифікатор першого користувача.
secondUser	string	Ідентифікатор другого користувача.

РЕАЛІЗАЦІЯ СИСТЕМИ

Backend

Для реалізації сервісу було вирішено використовувати 2 технології взаємодії клієнту та серверу: REST API для авторизації у системі та websocket для взаємодії з основною частиною у режимі реального часу (відправка повідомлень, створення чатів і т.д.).

При плануванні проекту було обрано фреймворк .Net та СУБД MsSQL для написання серверної сторони продукту, проте в процесі розробки команда зіткнулась з проблемою реалізації обох вищезгаданих технологій в одному проекті та розгортання його на віддаленому сервері. Крім того, в багатьох учасників команди був досвід розробки з використанням Node.js, тому було прийняте рішення перейти на нього. Фінальний продукт використовує Node.js та Express.js для backend-у та нереляційну СУБД MongoDb.

Повний опис бази даних, таблиць та зв’язків між ними наведено вище.

Далі розглянемо певні реалізації безпосередньо backend.

app.js

app.js > <unknown>

```
1   const express = require('express')
2   const mongoose = require('mongoose')
3   const bodyParser = require('body-parser')
4   const passport = require('passport')
5   const keys = require('./config/keys')
6   const app = express()
7
8   const AuthRouter = require('./routes/auth')
9
10  app.use(passport.initialize())
11  require('./middleware/passport')(passport)
12
13  app.use(bodyParser.urlencoded({ extended: true }))
14  app.use(bodyParser.json())
15  app.use(require('cors')())
16  app.use(require('morgan')('dev'))
17
18  mongoose.connect(keys.mongoURI)
19    .then(() => console.log('MongoDB connected.'))
20    .catch(error => console.log(error))
21
22  app.use('', AuthRouter)
23
24  module.exports = app
```

Рисунок 3. Головний файл проекту з бібліотеками

Для авторизації користувачів використовуємо REST API.

models > User.js > ...

```
1   const mongoose = require('mongoose')
2   const Schema = mongoose.Schema
3
4   const userSchema = new Schema({
5       email: {
6           type: String,
7           required: true,
8           unique: true
9       },
10      password: {
11          type: String,
12          required: true
13      },
14      name: {
15          type: String,
16          required: true
17      },
18      username: {
19          type: String,
20          required: true
21      }
22  })
23
24  module.exports = mongoose.model('users', userSchema)
```

Рисунок 4. Схема користувача з використанням Mongoose

```
module.exports.signup = async function (req, res) {

  const candidate = await User.findOne({ email: req.body.email })

  if (candidate) {
    res.status(409).json({
      message: 'Email is already taken. Try to use another one.'
    })
  }
  else {
    const salt = bcrypt.genSaltSync(10)
    const password = req.body.password
    const user = new User({
      email: req.body.email,
      password: bcrypt.hashSync(password, salt),
      name: req.body.name,
      username: req.body.username
    })

    try {
      const savedUser = await user.save()

      const accessToken = jwt.sign({
        email: savedUser.email,
        id: savedUser._id,
        name: savedUser.name,
        username: savedUser.username
      }, keys.jwt, { expiresIn: 3600 })

      res.status(201).json({ accessToken })
    } catch (e) {
      // errorHandler(res, e)
      console.log(e)
    }
  }
}
```

Рисунок 5. Endpoint для реєстрації нового користувача в системі

Для модуля авторизації наявна валідація та обробка помилок(email уже зайнятий, не здайдено зареєстрованого користувача, паролі не співпадають і тд).

Подібний функціонал для CRUD операцій з використанням Mongoose реалізовано для усіх інших сутностей(Chats, Messages і тд).

```
const User = require('../models/User')
const Contact = require('../models/Contact')
const keys = require('../config/keys')


module.exports.addContact = async function (firstUser, secondUser) {
    const firstUserId = firstUser;
    const secondUserId = await User.findOne({ username: secondUser })

    const contactFirstToSecond = new Contact({
        firstUser: firstUserId, secondUser: secondUserId
    })

    const contactSecondToFirst = new Contact({
        secondUser: firstUserId, firstUser: secondUserId
    })

    try {
        await contactFirstToSecond.save()
        await contactSecondToFirst.save()
    } catch (e) {
        console.log(e)
        // errorHandler(res, e)
    }
}

module.exports.getContacts = async function (user) {
    const contacts = await Contact.find({ firstUser: user })

    const fullContacts = await Promise.all(contacts.map(async (contact) => await User.findById(contact.secondUser.toString())));

    return fullContacts
}
```

Рисунок 6. Приклад деяких CRUD операцій для сутності контактів

Для реалізації вебсокетів використовується бібліотека Socket.io. Це дозволяє миттєво відображати будь-які необхідні зміни на усіх підключених пристроях.

```
    socket.on('createDialog', async (data) => {
        const firstUser = data.firstUser
        const secondUser = data.secondUser

        const chatCandidate = await Chat.findOne({ name: `${firstUser.id}:${secondUser._id}` })

        console.log('CHAT CANDIDATE', chatCandidate)

        if (!chatCandidate) {
            const chat = await chats.createChat({
                name: `${firstUser.id}:${secondUser._id}`,
                isGroupChat: false
            })

            await userToChats.addUserToChat(secondUser._id, chat._id)

            socketIO.sockets.to(connectionId).emit('createDialog', await userToChats.addUserToChat(firstUser.id, chat._id))
        }
        else {
            socketIO.sockets.to(connectionId).emit('createDialog', await userToChats.getUserToChat(firstUser.id, chatCandidate._id))
        }
    })

```

Рисунок 7. Приклад взаємодії вебсокету

Кожному користувачу при авторизації виділяється “кімната”, щоб повідомлення транслювались за конкретною адресою. При входженні користувача у чат також виділяється “кімната” для усіх учасників. Це забезпечує безпеку передачі повідомлень.

```
    socket.on('getChatInfo', async (data) => {
        socket.join(data.chatId)
        socketIO.sockets.to(connectionId).emit('getChatInfo', await chats.getChatInfo(data))
    })

    socket.on('sendMessage', async (data) => {
        const messageToSave = new Message({
            text: data.message,
            senderId: data.senderId,
            chatId: data.chatId,
            isPinned: false
        })

        console.log(messageToSave)
        try {
            const savedMessage = await messageToSave.save()
            const chat = await Chat.findById(data.chatId)
            chat.messages.push(savedMessage)
            await chat.save()
            socketIO.sockets
                .to(data.chatId)
                .emit('getMessage', savedMessage)
        }
        catch (e) {
            console.log(e)
        }
    })
```

Рисунок 8. Приклад взаємодії користувача з чатом

Frontend

Angular - це фреймворк для розробки веб-додатків на мові програмування TypeScript, який забезпечує створення високопродуктивних і масштабованих додатків. Він включає в себе багато компонентів та бібліотек, які допомагають розробнику створювати різні частини додатку, включаючи UI компоненти, маршрутизацію, форми та багато іншого.

Angular Material - це набір компонентів, які розроблені з використанням Material Design - дизайн-мови від Google. Angular Material дозволяє швидко створювати привабливі та функціональні користувацькі інтерфейси, які виглядають однаково на різних пристроях.

```css
/* You can add global styles to this file, and also import other style files */

html,
body {
    height: 100%;
}

body {
    margin: 0;
    font-family: 'Poppins', sans-serif !important;
    overflow: hidden;
}

.h2 {
    margin: 0 !important;
    font-family: 'Poppins', sans-serif !important;
    font-style: normal !important;
    font-weight: 400 !important;
    font-size: 20px !important;
    line-height: 30px !important;
}

::-webkit-scrollbar {
    width: 5px;
}

/* Track */
::-webkit-scrollbar-track {
    box-shadow: inset 0 0 5px grey;
    border-radius: 10px;
}

/* Handle */
::-webkit-scrollbar-thumb {
    background: #fafaff;
    border-radius: 10px;
}
```

Рисунок 9. Фрагмент глобальних стилів для проєкту

Для реалізації маршрутизації в Angular використовується вбудований механізм маршрутизації, який дозволяє описати шляхи до різних компонентів у додатку. Для цього використовуються модулі RouterModule та Route, які є частинами фреймворку Angular.

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AppRoutes } from './types/enums/app-routes.enum';
import { RootRoutes } from './types/enums/root-routes.enum';

const routes: Routes = [
  {
    path: AppRoutes.EMPTY,
    redirectTo: `${AppRoutes.ROOT}/${RootRoutes.MAIN}/${RootRoutes.MESSENGER}`,
    pathMatch: 'full',
  },
  {
    path: AppRoutes.ROOT,
    loadChildren: () =>
      import('./root/root.module').then((m) => m.RootModule),
  },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Рисунок 10. Приклад роутер модуля

Для спілкування з сервером було створено 2 сервіси з усіма необхідними функціями. Один для спілкування за допомогою Rest API, інший за допомогою webSocket.

```typescript
@Injectable({
  providedIn: 'root'
})
export class WebSocketService {

  socket!: io.Socket;

  constructor() {

    this.startConnectionForCurrentUser();
  }

  startConnectionForCurrentUser() {
    console.log(this.socket);
    if(this.socket){
      this.socket.close();
      this.socket.disconnect();
    }
    const token = localStorage.getItem('authToken');
    if (token){
      const user = jwt_decode(token);
      this.socket = io.connect('165.22.85.146:3000', {
// @ts-ignore
        query: {'userId': user?.id as string}
      });
    }
  }

  listen(eventname: string): Observable<any> {
    return new Observable((subscriber) => {
      this.socket.on(eventname, (data: any) => {
        subscriber.next(data);
      })
    })
  }

  emit(eventname: string, data: any) {
    this.socket.emit(eventname, data);
  }
}
```

Рисунок 11. Сервіс для спілкування за допомогою webSocket

```
submitData() {
    this.signUpForm.markAllAsTouched();

    if (this.signUpForm.valid) {
        const valueToSend = JSON.parse(JSON.stringify(this.signUpForm.value));
        delete valueToSend['repeatPassword'];
        valueToSend.password = hashPassword(valueToSend.password);
  this.isLoading = true;
        console.log(valueToSend);

        this.http.post('api/register', {
            ...valueToSend
        })
            .pipe(catchError(error => of(error)))
            .subscribe((response: any) => {
                if (response.error) {
        console.log(response.error);
        this.isLoading = false;
        this.signUpError = (typeof response.error.message === "string") ? 'email is already in use' :  response.error.message.details[0].message.includes('/^.+@knu.ua$/') ? 'Email domain should be knu.ua' :
          'username should start with @'
                    return;
                }
                localStorage.setItem('authToken', response?.accessToken);
      this.wsService.startConnectionForCurrentUser();
      this.isLoading = false;
                this.router.navigateByUrl('/root/main/messenger');
            });
    }
}
```

Рисунок 12. Обробка помилок авторизації

Можна бачити, що також присутня обробка помилок, якщо відповіді з сервера не буде або вона буде негативною, то буде показано відповідне повідомлення з описом помилки. У випадку позитивного результату буде створено нового користувача.

Для основної частини додатку було створено окремий модуль, який включає в себе розбиті за функціональністю компоненти та пайпи.

```
@NgModule({
  declarations: [
    MessengerComponent,
    LeftPanelComponent,
    ChatsListComponent,
    ChatComponent,
    ChatPlaceholderComponent,
    SettingsComponent,
    ChatInListComponent,
    MessageComponent,
    ContactsComponent,
    ChatCreationComponent,
  ],
  imports: [
    CommonModule,
    MessengerRoutingModule,
    MatButtonModule,
    ReactiveFormsModule,
    MatChipsModule,
    TranslateModule,
    ReactiveFormsModule,
    MatSelectModule,
    MatIconModule,
    FormsModule,
    MatProgressSpinnerModule,
    SharedModule,
  ],
  providers: [
    {provide: ColorsService}
  ]
})
export class MessengerModule { }
```

Рисунок 13. Messanger Module

Для реалізації форм було використано Reactive forms в Angular, які дозволяють зручно та гнучко керувати як станом форми, так і її відображенням на веб-інтерфейсі. Reactive forms дозволяють реагувати на зміни введених даних та проводити їх валідацію.

```
signUpForm: FormGroup = new FormGroup({
    email: new FormControl('', [Validators.required, Validators.email]),
    name: new FormControl('', Validators.required),
    username: new FormControl('', Validators.required),
    password: new FormControl('', Validators.required),
    repeatPassword: new FormControl('', Validators.required)
});
```

Рисунок 14. Приклад ініціалізації форми

ТЕСТУВАННЯ

В цьому розділі ми розглянемо результати тестування, виконаних на проекті. Наведено детальний огляд процесу тестування, від підготовки та планування до аналізу результатів. Також ми розглянемо, як отримані результати впливають на проект як в цілому, так і на рівень його придатності.

Етап підготовки почався з порівняння технологій для тестування, які можуть забезпечити повноцінне тестування фронтенд-частини додатку. Тестування на стороні бекенду проводилось до міграції за допомогою TestTools.UnitTesting, а після міграції ми обмежились верифікацією. Тому в цьому розділі ми покажемо детальніше тестування зі сторони клієнтської частини та методів Manual QA.

Для тестування було вирішено користуватись стандартним для Angular фреймворком тестуванням Karma. Karma - це інструмент для запуску юніт-тестів для JavaScript-коду в різних браузерах та на різних платформах. Він дозволяє автоматизувати процес тестування та забезпечує гарантію того, що ваш код працює однаково на різних пристроях і браузерах.

Переваги Karma для юніт-тестування включають: Автоматизація процесу тестування:

- Karma дозволяє автоматично запускати тести у різних браузерах та на різних платформах, що дозволяє економити час розробника.
- Зручність та швидкість: Karma дозволяє швидко виконувати тести, просто запустивши їх командою у терміналі. Крім того, він надає зручний інтерфейс для перегляду результатів тестування.
- Надійність: Karma забезпечує надійність тестів, оскільки він дозволяє виконувати їх у різних браузерах та на різних платформах, що дозволяє перевірити, що код працює правильно у різних умовах.
- Гнучкість: Karma підтримує різні фреймворки для написання тестів, такі як Mocha, Jasmine, і Jest, що дозволяє розробникам використовувати свій улюблений фреймворк для тестування.
- Інтеграція з іншими інструментами: Karma може легко інтегруватися з іншими інструментами, такими як Grunt або Gulp, для автоматизації різних процесів у проекті.

У загальному, Karma дозволяє забезпечити якість вашого коду та впевнитися в тому, що ваші юніт-тести працюють на різних платформах та браузерах. Це допомагає зменшити кількість багів та помилок.

Також для перевірки якості коду було використано SonarCloud - інструмент для автоматичної перевірки коду на наявність можливих багів, дуплікації коду, можливості витоку чутливої інформації користувачів. Цей інструмент було додано до наших репозитаріїв у github, та активовано за допомогою Github Actions. Ці перевірки запускалась для кожного пулл реквеста при його створенні і\або додаванні змін.

Далі наведені приклади юніт тестів для клієнтської частини додатку, та результати роботи Sonar Cloud

```
it('should create', () => {
  expect(component).toBeTruthy();
});

it('should validate required fields', () => {
  component.signUpForm.controls['email'].setValue('');
  component.signUpForm.controls['name'].setValue('');
  component.signUpForm.controls['username'].setValue('');
  component.signUpForm.controls['password'].setValue('');
  component.signUpForm.controls['repeatPassword'].setValue('');

  expect(component.signUpForm.valid).toBeFalsy();

  component.signUpForm.controls['email'].setValue('test@gmail.com');
  component.signUpForm.controls['name'].setValue('test');
  component.signUpForm.controls['username'].setValue('test');
  component.signUpForm.controls['password'].setValue('password');
  component.signUpForm.controls['repeatPassword'].setValue('password');

  expect(component.signUpForm.valid).toBeTruthy();
});

it('should check if password and repeat password are the same', () => {
  component.signUpForm.controls['password'].setValue('password');
  component.signUpForm.controls['repeatPassword'].setValue('wrongPassword');

  expect(component.signUpForm.get('repeatPassword')?.hasError('notSame')).toBeTruthy();

  component.signUpForm.controls['repeatPassword'].setValue('password');

  expect(component.signUpForm.get('repeatPassword')?.hasError('notSame')).toBeFalsy();
});

it('should send data when form is valid', () => {
  spyOn(component.http, 'post').and.returnValue(of({}));

  component.signUpForm.controls['email'].setValue('test@gmail.com');
  component.signUpForm.controls['name'].setValue('test');
  component.signUpForm.controls['username'].setValue('test');
  component.signUpForm.controls['password'].setValue('password');
  component.signUpForm.controls['repeatPassword'].setValue('password');

  component.submitData();

  expect(component.isLoading).toBeTruthy();
```

Рисунок 14. Фрагмент юніт тесту

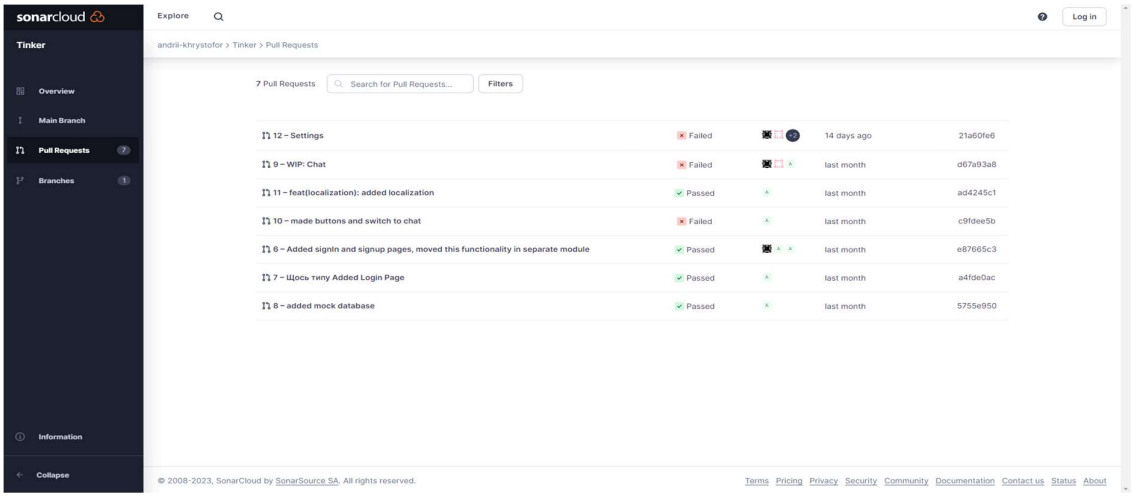


Рисунок 15. Перевірка Sonar Cloud для front-end репозиторія

ВИКОРИСТАННЯ СИСТЕМИ

При переході на сайт за посиланням користувач потрапляє на сторінку авторизації

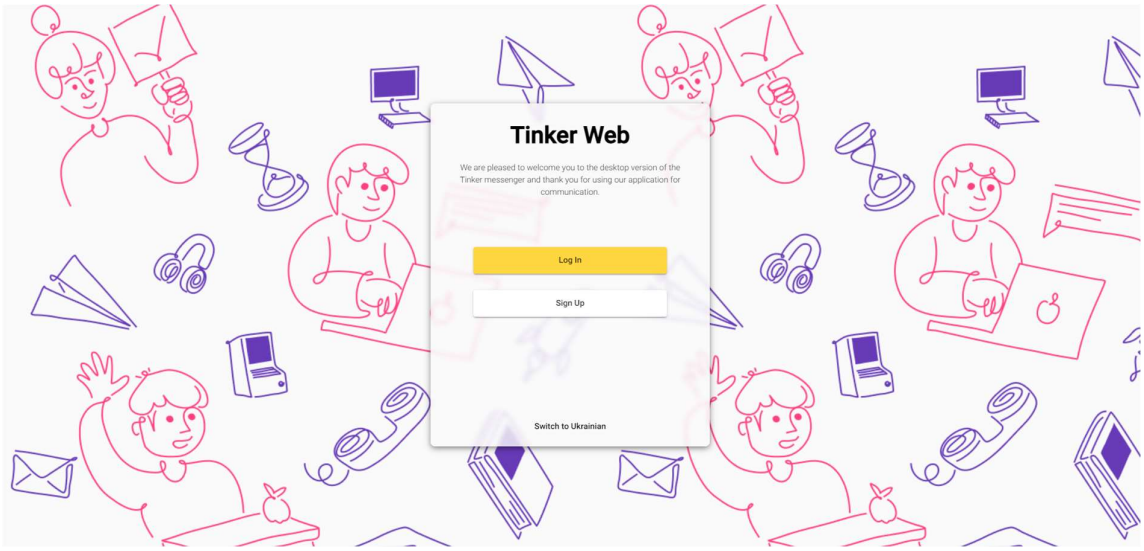


Рисунок 16. Сторінка авторизації

При натисканні на Log In користувач потрапляє на сторінку логіну, де може ввести необхідні дані та увійти до системи, як існуючий користувач.

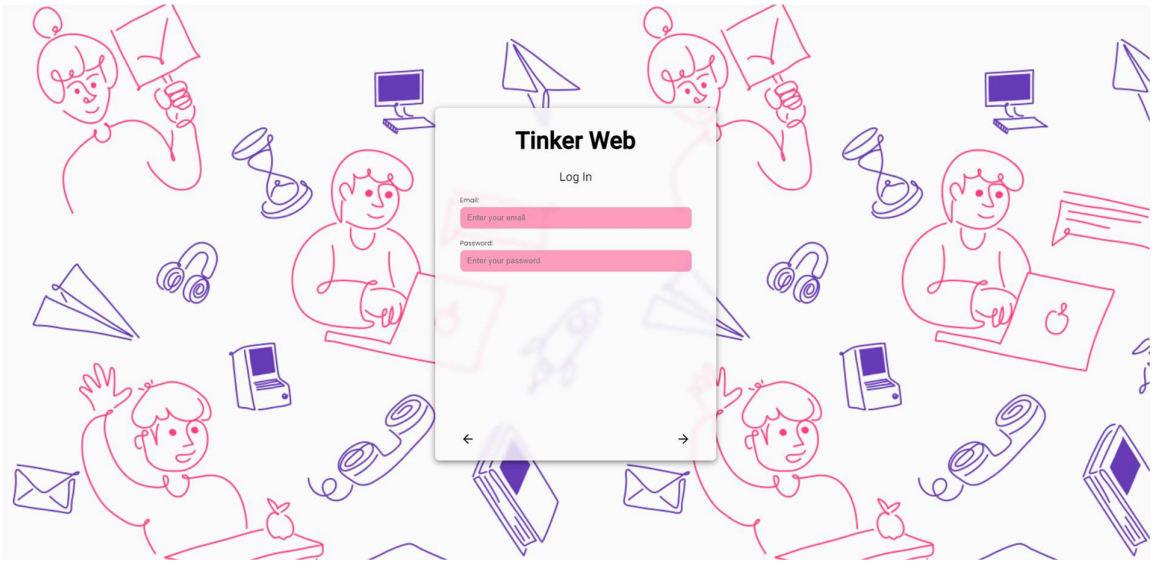


Рисунок 17. Log In

При натисканні на Sign up користувач потрапляє на сторінку реєстрації, де може ввести необхідні дані та зареєструватись, як новий користувач

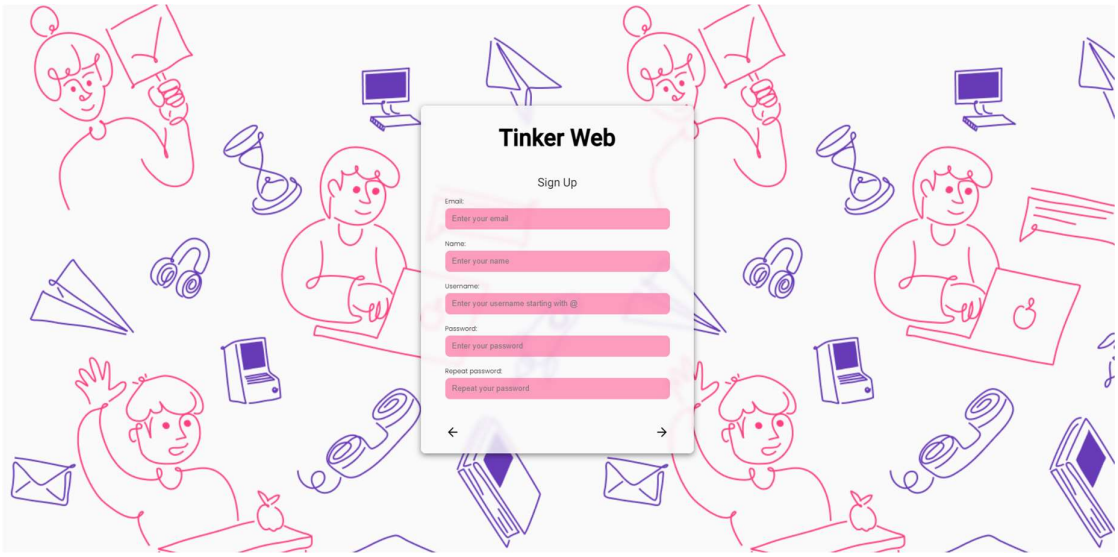


Рисунок 18. Sign up

Після успішної авторизації користувач потрапляє на основну сторінку додатку, де може переглянути список існуючих сайтів та взаємодіяти з ними

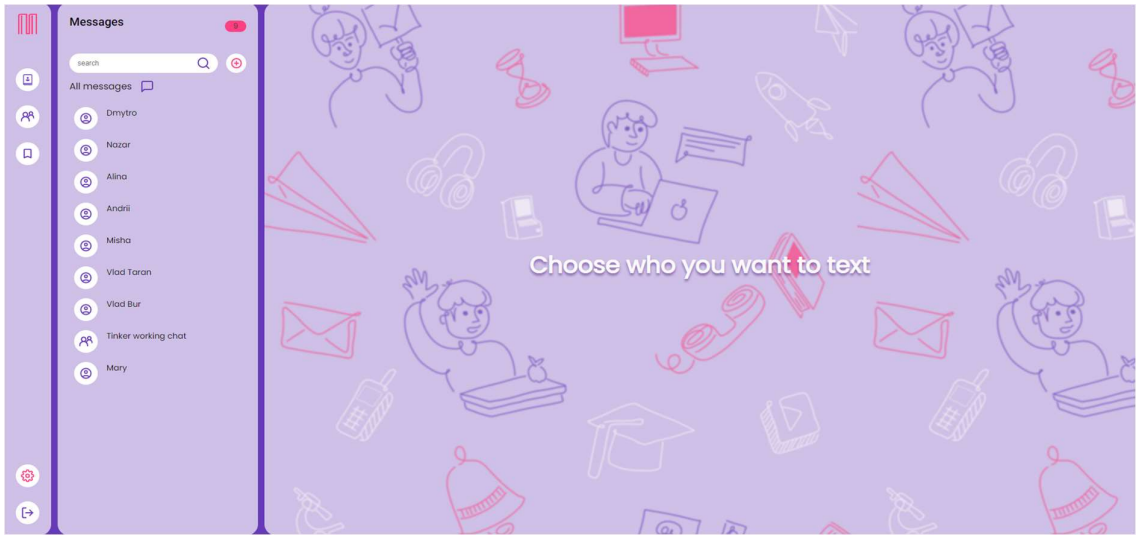


Рисунок 19. Основна сторінка

Для зміни налаштувань користувач може натиснути на відповідну кнопку та відкрити діалогове вікно з наявною інформацією та можливістю її зміни.

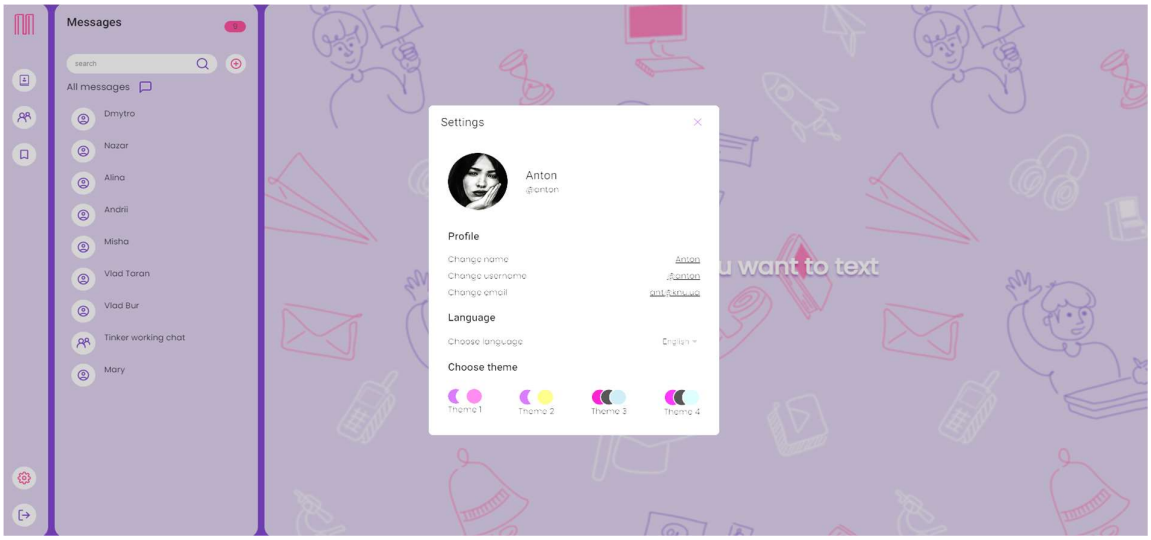


Рисунок 20. Сторінка налаштування

Для того, щоб потрапити у діалог, користувачу достатньо натиснути на один з доступних у списку.

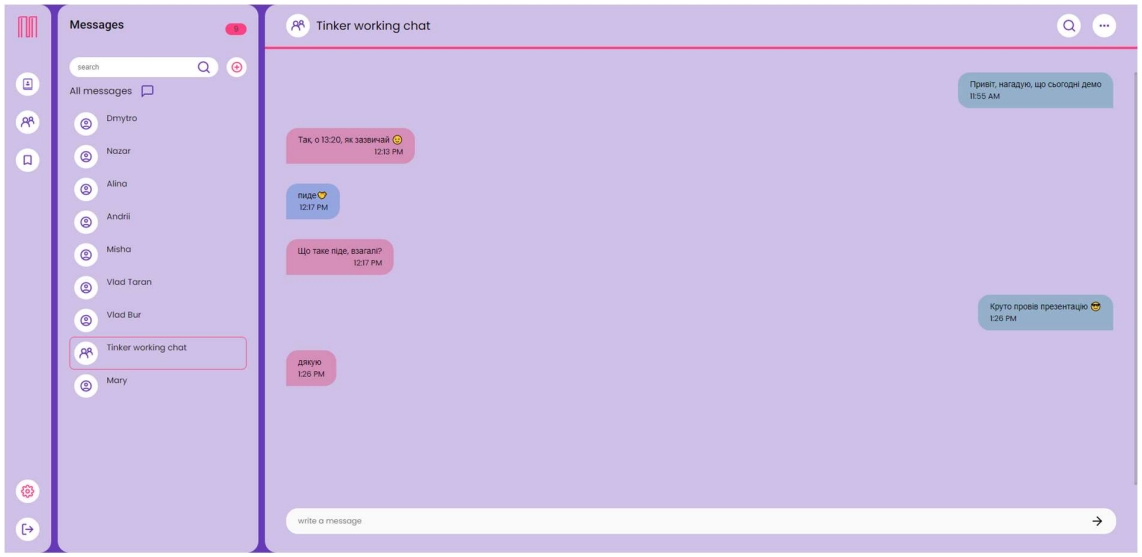


Рисунок 21. Обрання чату

На сторінці з’являється чат, зі списком повідомлень. відображеним у правій частині сторінки.

Для створення нового діалогу достатньо додати нового користувача до своїх контактів. Для цього необхідно натиснути на необхідну кнопку, а у відкритому діалозі ввести юзернейм нового контакту та натиснути кнопку “додати контакт”

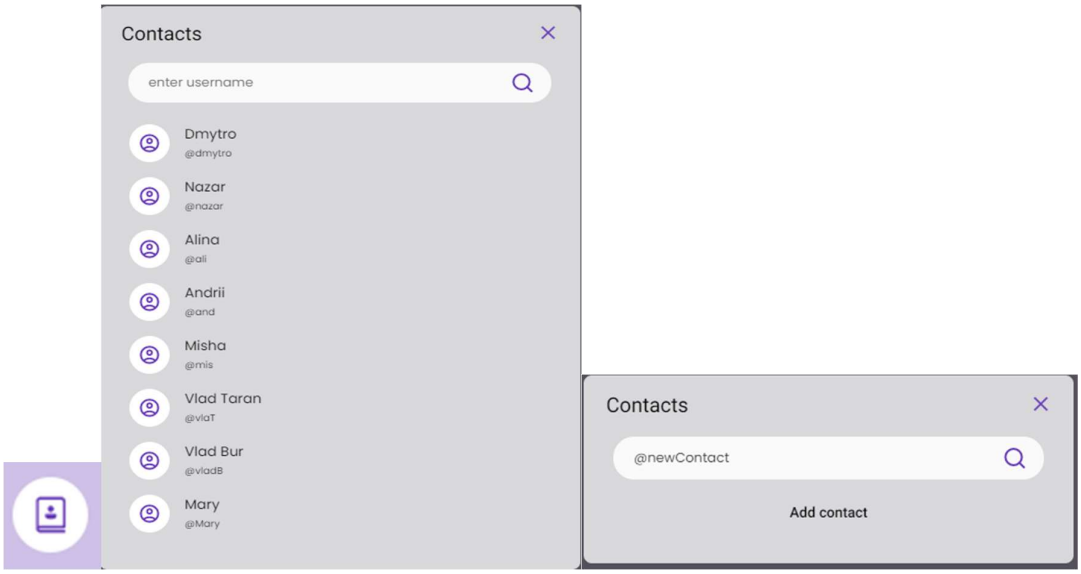


Рисунок 22. Контакти

Для створення групового чату необхідно натиснути кнопку та обрати користувачів зі списку контактів у діалоговому вікні.

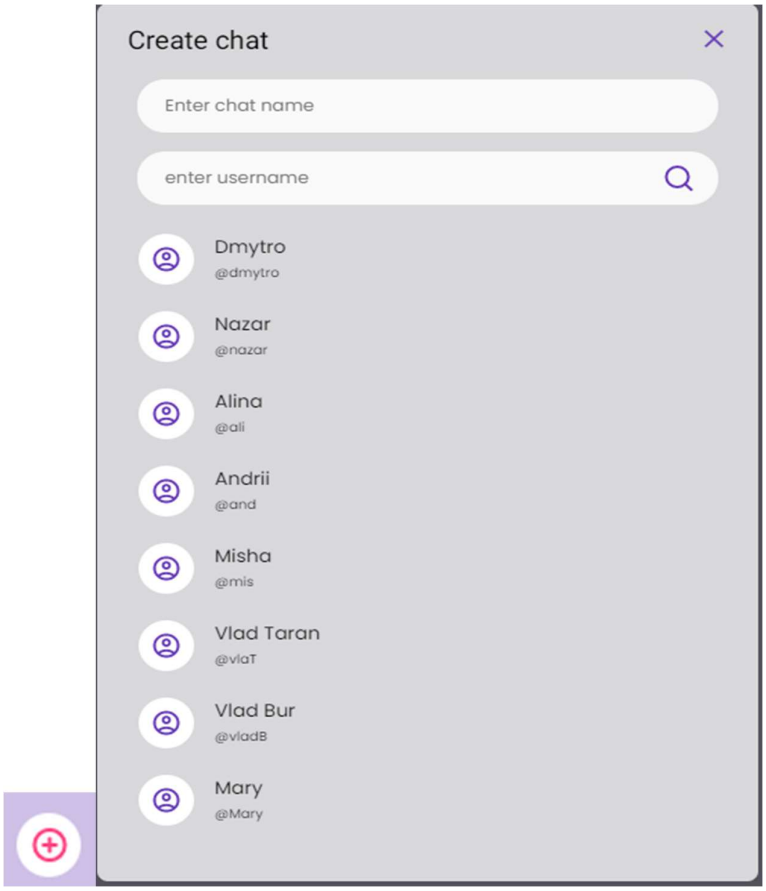


Рисунок 23. Створення групового часу

ФОРМАЛЬНА СПЕЦИФІКАЦІЯ ТА ВЕРИФІКАЦІЯ

Специфікацію було вирішено розробляти з використанням мови OCL (Object Constraint Language). Цей спосіб перевірки програм дозволяє визначати обмеження на дані та взаємодію між об'єктами. OCL підтримує конструкції для формулювання передумов та післяумов методів, а також може бути використаний для визначення інваріантів класів. Мова є об’єктивною, послідовною та декларативною, що підходить для нашої системи.

Для специфікації ми використовуємо клас “User” з полями email, username, name та password.

```
class User  {
    email;
    password;
    name;
    username;
}

const OclEngine = require("@stekoe/ocl.js").OclEngine;

// Define OCL rule
const myOclExpression = `
    -- User email should be unique
    context User
        inv: self.email->isUnique(self.email)

    -- Username should starts with @
    context User
        inv: self.username.substring(0,1)="@"

    -- Email should be a knu.ua
    context User
        inv: self.username.indexof("knu.ua") > 0


`;
```

Рисунок 24. OCL вираз

```
user OclResult {
  namesOfFailedInvs: [],
  evaluatedContexts: [],
  result: true
}
OclResult {
  namesOfFailedInvs: [],
  evaluatedContexts: [],
  result: true
}
OclResult {
  namesOfFailedInvs: [],
  evaluatedContexts: [],
  result: true
}
```

Рисунок 25. Результат OCL для різних класів

Верифікація за допомогою Express-contracts проводилась на 2х ендпоінтах, оскільки бібліотека не має можливості працювати з webSocket, тому цю частину необхідно було створити самим

```
const loginShema = Joi.object().keys({
    email: Joi.string().min(3).max(30).required().pattern(new RegExp('^.+@knu\.ua$')),
    password: Joi.string().required(),
  });
const signupSchema =  Joi.object().keys({
    email: Joi.string().min(3).max(30).required().pattern(new RegExp('^.+@knu\.ua$')),
    password: Joi.string().required(),
    name: Joi.string().min(3).max(30).required(),
    username: Joi.string().min(3).max(30).required().pattern(new RegExp('^@.*')),
  });
```

Рисунок 26. Схема для верифікації

```
router.post('/login', contract(loginSchema), controller.login)

router.post('/register', contract(signupSchema), controller.signup)
```

Рисунок 27. Додавання верифікації до ендпоінтів

Отже, нам вдалося успішно верифікувати створення нового користувача. Виконати валідацію на кожне поле базового класу та переконатися, що наш код працює.

ОПИС ПРОЦЕСУ РОЗРОБКИ

З самого початку наша команда вирішила працювати за методологією Scrum, адже вона дозволяє:

- Scrum передбачає чітку розподіл ролей та відповідальності в команді, що дозволяє кожному учаснику розуміти свої завдання та внести свій внесок у проект.
- Ітераційний підхід: Scrum базується на ітераційному підході до розробки, що дозволяє команді постійно вдосконалювати продукт, враховуючи відгуки клієнтів та змінні умови проекту.
- Стійке покращення: Scrum передбачає постійне покращення процесів розробки через перегляди та ретроспективи, що дозволяє команді постійно вдосконалюватись та підвищувати якість продукту.
- Керування продуктом: Scrum дозволяє команді керувати продуктом через backlog та вимоги клієнта, що дозволяє зосередитись на створенні продукту, який відповідає потребам клієнта.

Що найбільше з даної технології мало вплив на успішне виконання запланованих задач:

- регулярні Scrum активності. Це дало нам змогу чітко розуміти на якому етапі знаходится кожна задача. Звісно, отримали можливість оперативно допомагати один одному у разі виникнення труднощів.
- ведення всіх задач в trello. Це дозволило нам візуалізувати прогрес роботи та уникнути ситуації, коли завдання залишаються невиконаними до кінця спринту. Кілька днів до завершення спринту ми аналізували стан завдань та спільно обговорювали, як можна виправити ситуацію з невиконаними завданнями.

Для ведення задач було обрано Trello, оскільки цей інструмент зручний у використанні, безкоштовний та інтуїтивно зрозумілий. Приємною несподіванкою стала можливість автоматично прикріпити пуллреквести до задач, що дозволило нам простіше орієнтуватися в процесі розробки. Ось так виглядала наша дошка для одного із спринтів, тут можна побачити що до кожної задачі прикріплені виконавці, та теги, які позначають до чого саме відносилась конкретна задача.

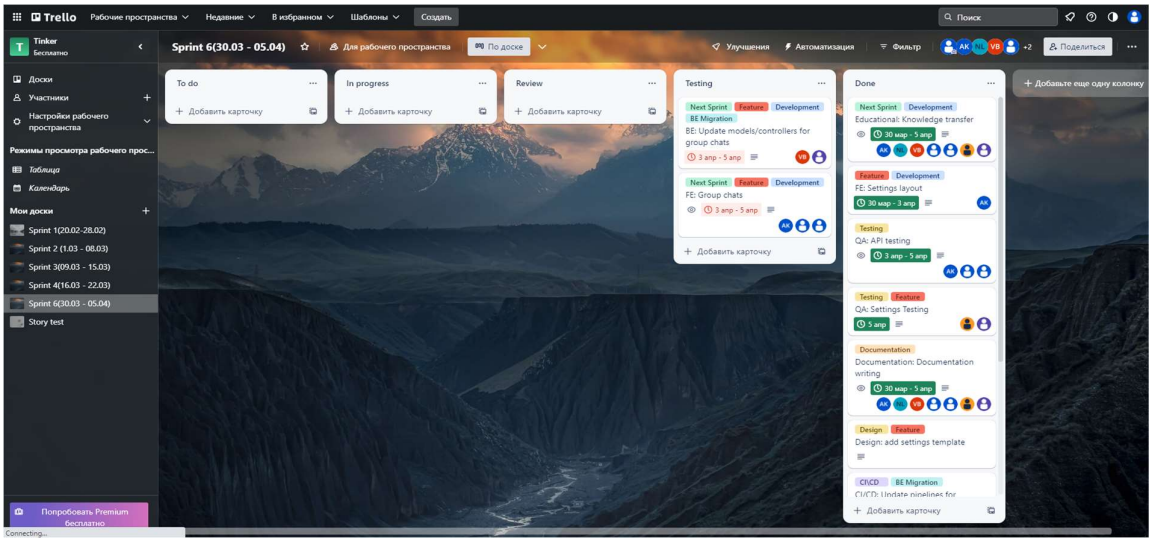


Рисунок 28. Дошка Trello

Планування спринтів проводилось щотижня з метою розподілу завдань між членами команди таким чином, щоб в кінці спринту не залишалось багато невиконаних задач. Нам вдалося знайти баланс між тим, щоб всі члени команди мали достатньо задач для виконання, але ніхто не був перевантажений. Кожному члену команди призначалися задачі, враховуючи його навички та здібності.

Труднощі та задачі, з якими зіштовхнулись.

Недовершене планування. На нашу думку основною причиною цієї проблеми був недостатній досвід в питанні оцінки задач, та іноді проблеми з нерівномірністю часу, який члени команди виділяли на проєкт, це породжувало проблему з блокуванням задач, адже у кожного в команді був свій темп.

Способи вирішення даної проблеми:

1) Розбиття задач на більш дрібні.
2) Детальне обговорення задачі.
3) Поширення знань.

Міграції Back-end'у. Як було зазначено вище, у ході розробки проєкту було прийняте рішення зміни стеку розробки серверної частини додатку. Ця проблема потребувала великої кількості часу не лише на вивчення нової технології, але й на перенесення існуючої функціональності.

На цьому проєкті ми здобули досвід співпраці та роботи в команді, вивчили планування, ефективну комунікацію та управління часом. Нам вдалося працювати в одному напрямку, щоб разом досягти поставлених цілей, що давало нам мотивацію та зацікавленість. Крім того, ми навчилися ефективно обмінюватися інформацією в проекті, висловлювати свої думки та контролювати виконання власних задач, що допомогло нам успішно основну частину завершити проєкт. Можна впевнено сказати, що проект завершився вдало, оскільки команда досягла поставлених на початку цілей. В майбутньому планується додати до проєкту інтеграцію з іншими частинами екосистеми університету.

Кожен учасник проекту мав можливість зіткнутись з новими технічними завданнями та відчути професійний розвиток у своїй галузі. Додатково, ми здобули досвід використання гнучких методів розробки.

ВИСНОВКИ

Команда успішно завершила розробку університетського месенджера. Кожен учасник отримав нові виклики, що дозволило розвиватись як фахівцю та поглиблювати свої знання в галузі програмування та розробки. Команда працювала продуктивно та максимально ефективно використовувала свої навички та знання для досягнення мети проекту. Працювати в команді було захоплююче та стимулювало до спільної відповідальності, що в результаті дозволило справитися з усіма задачами і викликами.

Ось список тих навичок, які ми змогли покращити під час роботи над проєктом:

1. Backend-розробка: ми вдосконалили свої знання Node.js, Express, що дозволило створити динамічний масштабовану та гнучку серверну частину проєкту.

2. Frontend-розробка: ми вдосконалили свої знання Angular, що дозволило створити динамічний та інтерактивний інтерфейс користувача, а також покращити швидкість роботи веб-сайту.

3. Дизайн: ми вдосконалили навички роботи з Figma, що допомогло нам створити більш привабливий та зручний для користувачів дизайн.

4. Управління проектами: ми навчились ефективно використовувати Trello для організації роботи, стеження за прогресом та спілкування в команді.

Цей проект покращив наші навички у програмуванні та дизайні, а також навчив нас працювати у команді та ефективно спілкуватися один з одним. Ми пишаємося результатами нашої роботи та з нетерпінням чекаємо на подальші виклики та можливості для розвитку.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Angular Material. Документація. [Електронний ресурс]. Режим доступу: https://material.angular.io/.
2. Angular. Документація. [Електронний ресурс]. Режим доступу: https://angular.io/.
3. Node.js. Документація. [Електронний ресурс]. Режим доступу: https://nodejs.org/en/.
4. Express. [Електронний ресурс]. Режим доступу: http://expressjs.com/.
5. Socket.io. Документація. [Електронний ресурс]. Режим доступу: https://socket.io/.
6. Figma. Офіційний веб-сайт. [Електронний ресурс]. Режим доступу: https://www.figma.com/.
7. GitHub. [Електронний ресурс]. Режим доступу: https://github.com/.
8. Official Git Documentation. [Електронний ресурс]. Режим доступу: https://git-scm.com/doc/.
9. OCL.js. Документація [Електронний ресурс]. Режим доступу: https://ocl.stekoe.de/.
10. Express-contracts. Документація [Електронний ресурс]. Режим доступу: https://github.com/Dallas62/express-contract/.
11. MongoDB. Офіційний веб-сайт. [Електронний ресурс]. Режим доступу: https://www.mongodb.com/.
12. Postman. Офіційний веб-сайт. [Електронний ресурс]. Режим доступу: https://www.postman.com/.
13. UML diagram tutorial. Стаття. [Електронний ресурс]. Режим доступу: https://medium.com/@smagid_allThings/uml-class-diagrams-tutorial-step-by-step-520fd83b300

СИСТЕМА ПІДТРИМКИ ВОЛОНТЕРСЬКОЇ ДОПОМОГИ "HELPER"

Валерія Кандиба, Максим Марчишак, Максим Микитюк, Ігор Селезень, Марія Ануфрієва, Катерина Мовчанюк, Олена Бондарєва, Ольга Проценко

ВСТУП

Актуальність обраної теми. Війна в нашій країні показала, які ми відкриті та небайдужі: люди готові допомагати один одному. Масштаб потреб варіюється від допомоги з покупками продуктів для старших сусідів, до збору коштів на амуніцію або електрогенератори. Щодня у Телеграм-каналах та Фейсбук-спільнотах з'являються сотні прохань про допомогу чи звістку про можливість допомогти. Деякі запити іноді виконують кілька людей паралельно, дублюючи роботу, тоді як інші залишаються поза увагою та "губляться" між новинами, особистими повідомленнями та десятками інших повідомлень.

Застосунок під назвою Helper може допомогти різним категоріям користувачів, зокрема людям з обмеженими можливостями, літнім людям, а також всім тим, хто шукає швидку та ефективну допомогу.

Методи дослідження.Для розробки такого застосунку методи дослідження включають аналіз потреб та проблем, з якими стикаються різні категорії користувачів, аналіз конкурентів, опитування та спостереження, тестування та оцінка ефективності застосунку.

Цілі і завдання. Основна ціль полягає в тому, щоб створити комфортний, легкий у використанні для людей будь-якого віку, корисний застосунок для волонтерства.

Для досягнення цієї цілі поставлено такі завдання:

- узагальнення інформації про волонтерство;
- дослідити наявні застосунки з подібною тематикою в Україні;
- ознайомлення з наявними технологіями та програмами для ефективної організації робочого процесу;
- розробити інтерфейс та дизайн застосунку;
- поглиблення практичних навичок роботи в команді;
- власне створення та тестування платформи "Helper"**;**

ОГЛЯД НАЯВНИХ НА РИНКУ СИСТЕМ

Наразі на ринку представлено дуже мало подібних проєктів. Командою було проаналізовано наступні конкурентні рішення:

Бандеролька - він дозволяє кожному користувачеві бути як в ролі людини, яка шукає будь-якої допомоги, так і в ролі волонтера, який може її надати. "Бандеролька" створили, аби дати можливість структурувати запити про допомогу різних напрямів та масштабів й уникнути ситуації, коли одні звернення ігнорують, а інші виконують паралельно кілька осіб.

Переваги:

- Вдалий логотип.
- Існує форма реєстрації.
- Окрема сторінка з запитами, можна обрати фільтрування, наприклад: надати допомогу чи потребую допомогу, також категорії: закупка, доставка, перевезення людей, продукти, гігієна, медикаменти, амуніція, шукаю дім, техніка, людські ресурси, збір коштів. Також можна обрати регіон.
- Окрема сторінка з запитом з яким готовий допомогти, окрема зі своїми запитами, і запити які можна зберегти, щоб не загубити.

- Сторінка профіль: змінити особисті дані, сповіщення в залежності від типу: потребую допомогу чи надаю, і які категорії, і регіон, так можна не загубити запити які потенційно можеш закрити, або викласти запит в якому регіону чи який тип продукції готовий надати.

Недоліки:

- Не можна підтвердити особистість, тобто будь-який користувач може взяти будь-які контактні дані.
- Немає політики конфіденційності.
- Не можна додавати фото до запиту.
- Незручний інтерфейс, немає поняття типографіки, чіткого розмежування запитів.

Ераунд - це інтерактивна мапа для волонтерів та людей, які потребують допомоги у цей складний час. Новий волонтерський застосунок від команди, яка створила додаток єТривога, який з перших днів війни сповіщає про повітряну тривогу в областях та містах України.

Переваги:

- Швидка авторизація (через фейсбук, або apple).
- Існує мапа зі значками потребую допомогу чи надаю допомогу.
- Сторінка Список – фото користувача, надаю допомогу чи потребую, опис, контактні дані (телефон, інстаграм, категорія запиту(їжа, притулок, діти, військові, місто).
- Сторінка Додати – можна на мапі додати запит.
- Сторінка Профіль – змінити ім’я, ваші публікації, вийти з акаунта, скасування.

Недоліки:

- Не зрозумілий інтерфейс і назва(важко знайти у додатку якщо не знаєш назви додатку).
- Не можна змінити особисті дані.
- Немає фільтра запитів.
- Немає додаткового опису політики конфіденційності.
- Не можна додавати фото до запиту.

Turbota - Платформа взаємопомічі для тих хто хоче допомогти та для тих хто потребує допомоги. Якщо ти волонтер — створи об'яву про допомогу та чекай коли з тобою зв'яжуться ті, кому ти потрібен у такий важкий час. Якщо тобі потрібна допомога — зайди у додаток та вибери волонтера який може тобі допомогти. (Під час написання звіту про проєкт - даний застосунок вийшов з ринку, тобто завантажити його тепер неможливо).

Переваги:

- Зрозумілий логотип і назва.
- Існує вибір: надати допомогу і надаю допомогу.
- Можна написати розробнику додатка особисто в Телеграм.
- Авторизація через номер телефону, додатково підтвердження (код приходить за номером телефону).
- Створити Добро: ваше ім’я, адреса, перелік категорії: транспорт, вода, ліки, їжа, діти, притулок, тварини, одяг, інше. Опис допомоги: в якій кількості, одразу можна написати умови. Варіанти зв’язку: Телеграм або Вайбер.
- Коли запит створений можна видалити запит, варіанти: повернутися на головну, вийти з акаунту, скасувати дію. Також створити новий запит.
- Якщо обрати Надаю добро: можна обрати локацію і категорію допомоги.
- Можна переглянути детально запити інших людей з описом і контактними даними.

- Можна подивитися фото людини що потребує.

Недоліки:

- Не можна змінити дані створеного запиту (лише видалити).
- Немає мапи.
- Не можна змінити особисті дані (лише створити новий акаунт).
- Не можна додавати фото до запиту.

Висновок аналітика. Запозичено з інших додатків:

- Зрозумілий логотип і назва.
- Форма реєстрації.
- Запити окрема сторінка, можна обрати фільтрування: надати допомогу чи потребую допомогу, категорії:закупка, доставка, перевезення людей, продукти, гігієна, медикаменти, амуніція, шукаю дім, техніка, людські ресурси, збір коштів.
- Окремо сторінка з запитом який збираєшся допомогти, окремо зі своїми запитами.

Додали:

- При реєстрації підтвердження через пошту.
- Простий та зручний дизайн для використання.
- Додавання фото в пости.
- Згода з політикою конфіденційності.

ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Dart - це об'єктно-орієнтована мова програмування, яка була розроблена компанією Google. Dart може використовуватись для розробки вебдодатків, мобільних додатків та серверних додатків.

Переваги:

- Продуктивність: Dart має вбудовану систему збирання сміття, яка дозволяє автоматично вивільняти пам'ять, коли об'єкти більше не потрібні. Це дозволяє зменшити споживання ресурсів та забезпечити високу продуктивність програмного забезпечення.
- Гнучкість: Dart дозволяє розробникам легко змінювати та доповнювати додаток за допомогою вбудованих фреймворків та бібліотек. Це забезпечує гнучкість та розширюваність додатка.
- Швидкість розробки: Dart має простий та лаконічний синтаксис, що дозволяє розробникам швидко створювати програмне забезпечення.
- Відкритість: Dart має відкритий вихідний код та активну спільноту розробників, яка допомагає у розв’язанні питань та розвитку мови.

Flutter - технологія розробки мобільних додатків з відкритим кодом, створена компанією Google. Вона базується мовою програмування Dart і дозволяє розробляти крос-платформні додатки для Android та iOS.

Flutter пропонує швидкість розробки та високу продуктивність. Основна перевага полягає в тому, що Flutter використовує власний рушій для малювання і рендерингу, що дозволяє досягти високої продуктивності, а також зменшити час розробки. Завдяки цьому Flutter є ідеальним вибором для розробки високопродуктивних мобільних додатків.

Flutter також дозволяє швидко розробляти додатки з гарним дизайном, завдяки вбудованій бібліотеці Material Design та Cupertino, які забезпечують стильний та сучасний інтерфейс користувача. Бібліотека Material Design, створена Google, містить набір віджетів, які дозволяють втілити всі елементи дизайну Material Design. Бібліотека Cupertino дозволяє створювати інтерфейс, що схожий на iOS.

Flutter також дозволяє легко збирати та використовувати власні віджети, що дозволяє розробникам створювати власний дизайн та функціональність, які відповідають їх потребам та потребам користувачів.

За допомогою Flutter можна також створювати додатки з анімацією та ефектами, що забезпечують високу інтерактивність та привабливість.

Golang **-** Мова Go розроблялась, як мова для створення різних високоефективних програм, однак більшість програмістів сходяться на думці, що найкраще вона підходить для створення вебдодатків (в якості back-end). При цьому Go дає можливості писати й інші проєкти, наприклад, Docker, InfluxDB і Kubernetes. По суті, застосування мови Go обмежується трьома основними напрямками: мережеве програмне забезпечення, консольні утиліти та бекенд.

Однією з відмінних рис мови є оригінальна система типів: в мові відсутнє наслідування (один з принципів об'єктно-орієнтованого програмування). Також в Go використовується скорочений синтаксис визначення змінних і синтаксис анонімних функцій.

Ще одна особливість цієї мови - паралелізм, який полягає в тому, що будь-яка функція може бути виконана одночасно з іншою.

Через те, що Go є однією з молодих мов програмування, регулярно виникають обговорення доцільності її використання.

Чому обрали Go? Бо вона швидка, проста, потужна мова програмування яка компілюється.

Бібліотека Gin **-** використовує стандартну бібліотеку http, але по суті просто спрощує це все, і скорочує код шляхом перевикористання.

Бд mongodb **-** проста швидка NoSQL база даних, яка дозволяє гнучко та швидко підіймати проєкти.

Для зручності роботи над проєктом було використано вебсервіс для спільної розробки програмного забезпечення GitHub. Посилання на репозиторій знаходиться в Додатку Б.

ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ.

Призначення системи. Призначенням системи застосунку "Helper" є автоматизація процесу створення запитів для допомоги в будь-якій частині України. Така система дає можливості користувачам створювати запити, шукати певні речі, змінювати їх статус, бути корисним на благо країни.

Робота передбачає:

- аналіз схожих програмних продуктів на ринку;
- визначення функціональних можливостей для майбутньої системи;
- проєктування діаграм згідно з поставленими вимогами;
- програмна реалізація системи.

Цілі створення системи. "Helper" була створена з метою:

- можливість надання/отримання допомоги в будь-якій частині України;
- 100% виконання запитів;
- допомога військовим, їх родичам, переселенцям, літнім людям і дітям;

В тому числі застосунок призначений для волонтерів. Волонтерам часто треба шукати по запитах деякі речі, часто волонтери працюють не за свої особисті кошти, а за донати звичайних людей, тому застосунок допоможе волонтерам у пошуках речей, які допоможуть закрити запит.

Вимоги до системи

1) Стилістичне оформлення системи має відповідати дизайн-макету, погодженого з командою та замовниками.

2) Використання колірних схем, графічних елементів, шрифтів погодженому дизайн-макету в процесі створення системи є обов'язковим.

3) Вимоги до графічного дизайну сайту.

4) Сайт та усі його компоненти не повинні бути графічно перенасиченими. Інтерфейс має бути зручним у навігації, інтуїтивно зрозумілим, більшість ключових елементів мають бути доступні через дві-три прості дії.

5) Дизайн сайту має передбачати вдале поєднання стилістики, кольорів, шрифтів, логотипу.

6) Використанням сучасних технологій розробки (модернізації) прикладного програмного забезпечення та забезпеченням якісного його тестування.

7) Застосуванні сучасних інформаційних технологій.

8) Підтримці актуальності, повноти, несуперечності, цілісності та доступності інформації.

9) Крім того, програма має передбачати авторизацію користувача. Система повинна надавати можливість авторизуватися за допомогою електронної адреси.

Вимоги до функцій, які виконуються системою

Підсистема користувача.

Функція	**Задача**
Реєстрація	Введення ім'я користувача
	Введення коректної пошти
	Введення паролю який надійшов на пошту
	Придумати пароль
Авторизація	Введення коректної особистої пошти
	Введення паролю, який був записаний на початку реєстрації
Сторінка запитів	Переглянути всі актуальні запити
	Подивитися особисті запити
	Зберегти запит
Створити запит	Ввести заголовок запиту
	Обрати пункт "потребую допомогу" або "надаю допомогу"
	Вибрати категорію товару
	Обрати місце розташування
	Написати детальний опис товару
	Завантажити фотографію
	Зробити фотографію
	Опублікувати запит

ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

Для розробки бекенду використовується СУБД MongoDB.

Номер	**Таблиця**	**Опис**
1	users	Таблиця для збереження інформації про користувачів які зареєстровані в додатку
2	adverts	Таблиця для збереження інформації про ооголошення та тих хто ці оголошення створив
3	attachments	Таблиця для збереження інформації про фото файли для оголошень

Номер	**Таблиця**	**Опис**
Таблиця users Атрибут	**Тип**	**Опис**
_id	string	Ідентифікатор
full_name	string	Повне ім’я користувача
email	string	Електронна поштова скринька користувача
password	byte[]	Захешований пароль користувача
created_at	int32	Unix date stamp створення користувача
edited_at	int32	Unix date stamp редагування користувача
Таблиця adverts Атрибут	**Тип**	**Опис**
_id	string	Ідентифікатор
user_id	string	Ідентифікатор користувача
name	string	Ім’я оголошення
body	string	Опис оголошення
type	string	Тип оголошення
category	string	Категорія оголошення
location	sting	Локація оголошення
attachments	string[]	Ідентифікатори фото файлів оголошення
created_at	int32	Unix date stamp створення користувача
edited_at	int32	Unix date stamp редагування користувача

Таблиця attachments Атрибут	**Тип**	**Опис**
_id	string	Ідентифікатор
name	string	Назва фото
data	byte[]	Бінарні дані файлу
created_at	int32	Unix date stamp створення користувача
edited_at	int32	Unix date stamp редагування користувача

РЕАЛІЗАЦІЯ СИСТЕМИ

Опис загальної структури, можливі невеликі принципові фрагменти коду.

Для розробки бекенду була використана мова програмування Golang та фреймворк Gin з використанням onion архітектури. Концепція onion архітектури передбачає поділ програми на шари, де залежно від рівня, відповідального за бізнес-логіку, дані та взаємодію з користувачем, використовуються відповідні структури даних та інтерфейси.

Модель (англ. model): описує дані, що використовуються в додатку, а також логіку, яка пов'язана безпосередньо з даними. У onion архітектурі моделі поділяються на два типи: моделі уявлень, що використовуються уявленнями для відображення та передачі даних, та моделі домену, які описують логіку управління даними. Модель може містити дані та логіку управління цими даними, але не повинна містити логіку відображення даних у поданні.

Представлення (англ. view): відповідають за візуальну частину або інтерфейс користувача. У onion архітектурі уявлення не містять логіку обробки запиту користувача або управління даними.

Контролер (англ. controller): представляє центральний компонент бекенду, який забезпечує зв'язок між користувачем та додатком, представленням та сховищем даних. Він містить логіку обробки запиту користувача, а також взаємодії з моделями та сховищем даних. У onion архітектурі контролер взаємодіє з інтерфейсом репозиторію для отримання та збереження даних у базі даних mongodb.

Таким чином, використання мови програмування Golang та фреймворку Gin у поєднанні з onion архітектурою дозволяє розробляти бекенд за більш сучасними принципами проєктування програмного забезпечення. Onion архітектура базується на принципах інверсії залежностей та розділенні відповідальності між компонентами програми. Вона дозволяє розробляти масштабовані та легко тестирувані системи, оскільки різні компоненти можуть бути замінені або модифіковані без впливу на інші частини програми.

Golang є мовою програмування з високою продуктивністю та низьким рівнем абстракції, що дозволяє отримати швидку роботу програм та ефективне використання ресурсів. Gin є легковаговим фреймворком для веб-розробки, який надає широкий набір функцій для швидкої розробки RESTful API. Використання Gin дозволяє прискорити розробку та полегшити підтримку проєкту.

Компонент моделі в onion архітектурі відповідає за роботу з даними та бізнес-логікою додатка. Використання бази даних MongoDB дозволяє забезпечити високу продуктивність та масштабованість зберігання даних. Контролер в onion архітектурі відповідає за обробку запитів користувачів та взаємодію з моделлю. Завдяки використанню Gin та Golang, контролер може бути розроблений швидко та ефективно.

У цілому, використання Golang, Gin та onion архітектури дозволяє створювати масштабовані, продуктивні та легко тестирувані веб-додатки.

Наведемо деякі приклади коду моделі та контролера:

```
 9  type IHandler interface {
10          // users
11          CreateUser(c *gin.Context)
12          DeleteUser(c *gin.Context)
13          UpdateUser(c *gin.Context)
14          GetUser(c *gin.Context)
15
16          // auth
17          Login(c *gin.Context)
18          Register(c *gin.Context)
19          CheckToken(c *gin.Context)
20          Middleware(c *gin.Context)
21
22          // adverts
23          CreateAdvert(c *gin.Context)
24          DeleteAdvert(c *gin.Context)
25          GetAdvert(c *gin.Context)
26          GetAdverts(c *gin.Context)
27
28          // get advert attachments
29          GetAdvertAttachments(c *gin.Context)
30
31          // email validation
32          NewEmailCode(c *gin.Context)
33          CheckEmailCode(c *gin.Context)
34  }
35
```

Рисунок 1. Код інтерфейсу Handler

```
type IMongoClient interface {
	// user
	CreateUser(ctx context.Context, email, fullname string, password []byte) (*dbmodels.User, error)
	DeleteUser(ctx context.Context, id string) error
	UpdateUser(ctx context.Context, id, email string, password []byte) error
	GetUser(ctx context.Context, id string) (*dbmodels.User, error)
	GetUserByEmail(ctx context.Context, email string) (*dbmodels.User, error)

	// auth
	NewToken(ctx context.Context, userID string) (string, error)
	CheckToken(ctx context.Context, token string) (bool, string)

	// adverts
	CreateAdvert(ctx context.Context, name, body, atype, category, location, userID string, attachments [][]byte) (*dbmodels.Advert, error)
	LinkAttachments(ctx context.Context, advertID string, attachIDs []string) error
	DeleteAdvert(ctx context.Context, id string) error
	GetAdvert(ctx context.Context, id string) (*dbmodels.Advert, error)

	// attachments
	CreateAttachment(ctx context.Context, name string, data []byte) (*dbmodels.Attachment, error)
	DeleteAttachment(ctx context.Context, id string) error
	GetAttachment(ctx context.Context, id string) (*dbmodels.Attachment, error)
	GetAdvertAttachments(ctx context.Context, ids []string) ([]*dbmodels.Attachment, error)
	GetAdverts(ctx context.Context, filter models.AdvertsFilter, limit, offset int) ([]*dbmodels.Advert, error)

	// email codes
	NewEmailCode(ctx context.Context, email, code string) error
	CheckEmailCode(ctx context.Context, email, code string) (bool, error)
}
```

Рисунок 2. Код інтерфейсу СУБД

```
type IService interface {
	// user
	CreateUser(ctx context.Context, password, email, fullName string) (*dbmodels.User, error)
	DeleteUser(ctx context.Context, id string) (*string, error)
	UpdateUser(ctx context.Context, id, password, email string) (*dbmodels.User, error)
	GetUser(ctx context.Context, id string) (*dbmodels.User, error)

	// auth
	NewToken(ctx context.Context, userID string) (token string, err error)
	CheckToken(ctx context.Context, token string) (userID string, err error)
	Login(ctx context.Context, email, password string) (userID, token string, err error)
	Register(ctx context.Context, email, fullName, password string) (userID, token string, err error)

	// adverts
	CreateAdvert(ctx context.Context, name, body, atype, category, location, userID string, attachments [][]byte) (*dbmodels.Advert, error)
	DeleteAdvert(ctx context.Context, id string) error
	GetAdvert(ctx context.Context, id string) (*dbmodels.Advert, error)
	GetAdverts(ctx context.Context, filter models.AdvertsFilter, limit, offset int) ([]*dbmodels.Advert, error)

	// attachments
	GetAdvertAttachments(ctx context.Context, advertID string) ([]*dbmodels.Attachment, error)

	// email validation
	NewEmailCode(ctx context.Context, email string) error
	CheckEmailCode(ctx context.Context, email, code string) error
}
```

Рисунок 3. Код інтерфейсу бізнес логіки

ТЕСТУВАННЯ

Тестування програмного забезпечення є невід'ємною частиною створення якісного продукту, воно є одним з найважливіших етапів в процесі розробки програмного продукту. Мета тестування полягає у перевірці функціональності, якості та безпеки програми перед її випуском на ринок.

Етап тестування починається ще з зародження ідеї проєкту, адже тестувальник може заздалегідь продумати плюси та мінуси, допомогти компанії не втратити більше грошей та перемістити фокус з непотрібного на те, що може дійсно зробити проєкт успішнішим. Технології тестування постійно вдосконалюються. Створюються різні допоміжні системи, за допомогою яких, можна знайти більше багів, швидше їх фіксувати та більш структуровано описувати помилку, для того щоб розробникам було зручно та легше.

Тому тестування ПЗ допомагає забезпечити високу якість та надійність програмного продукту. Відсутність тестування може призвести до серйозних проблем, таких як втрата даних, некоректна робота програми та збій системи. Тестування ПЗ є важливим елементом у процесі розробки програмного продукту, оскільки допомагає виявляти та виправляти помилки на ранніх етапах розробки, що значно зменшує витрати на подальше усунення проблем.

Ручне тестування є більш гнучким, оскільки тестувальник може швидко змінювати напрямок тестування, зосереджуючись на тих аспектах програми, які вважає важливими. Команда обрала напрямок ручного тестування. Також обрали стратегію Bottom-up, тому що саме ця стратегія складається з тестування задач по зростанню їх складності, а також вона підходить для послідовного навчання засобів та технологій.

API-тестування. API - це інтерфейс, який дозволяє взаємодіяти між компонентами програмного забезпечення, або між програмою та зовнішнім сервісом. Тестування API дозволяє визначити, чи відповідає API специфікації, чи працює він швидко та ефективно, та чи забезпечує він безпеку даних.

API тестування в Postman є дуже потужним інструментом, що дозволяє проводити ефективне тестування API, зберігати та повторювати тестові сценарії, автоматизувати процес тестування та отримувати точну та коректну інформацію про відповіді на запити.

Test Case #	Test Case Description	Test Data	Expected Result	Actual Result	Pass/Fail
1	Check response when valid email is entered during registration	"email": "token57@mail.com"	The information about user should be displayed(user id, token)	The information about user is displayed(user id, token)	Passed

Рисунок 4. Тестовий приклад “ Valid email. Registration”

3	Check response when valid password is entered during registration	"password": "123412qw86hg"	The information about user should be displayed(user id, token)	The information about user is displayed(user id, token)	Pass

Рисунок 5. Тестовий приклад “Valid password. Registration”

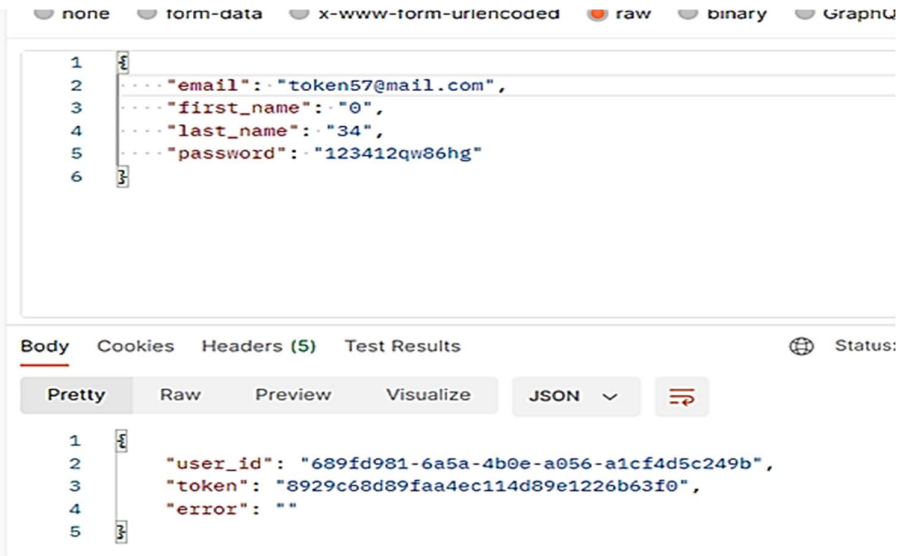


Рисунок 6. Перевірка на успішність реєстрації

Check response when valid token is entered in "check token"	"token": "f208f25b2ee0ed93cd0d001db93ba889"	The information about user should be displayed(user id, ok)	The information about user is displayed(user id, ok)	Pass

Рисунок 7. Тестовий приклад “Valid token”

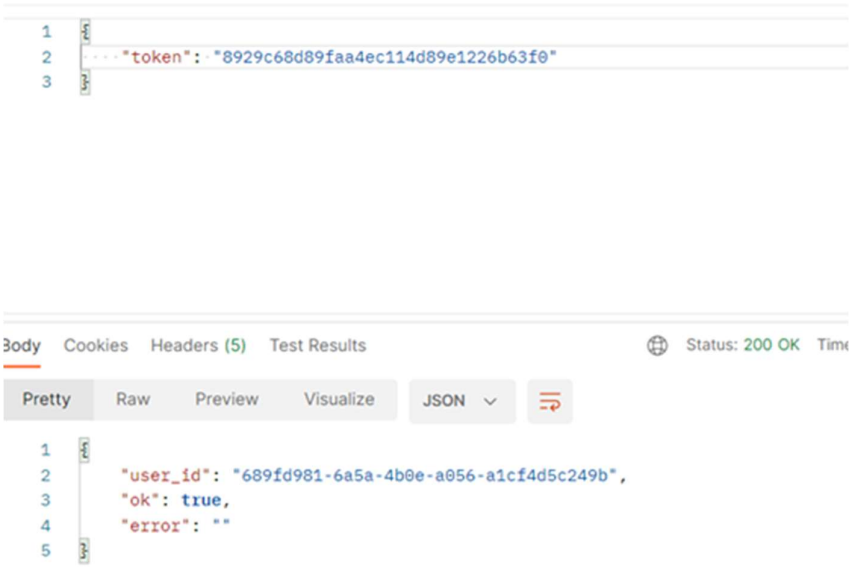


Рисунок 8. Перевірка на успішність отримання інформації користувача за вказаним токеном

Check response when valid email is entered during logging in	"email": "token57@mail.com"	The information about user should be displayed(user id, token)	The information about user is displayed(user id, token)	Pass

Рисунок 9. Тестовий приклад “Valid email. Logging in”

Check response when valid password is entered during logging in	"password": "123412qw86hg"	The information about user should be displayed(user id,	The information about user is displayed(user id, token)	Pass

Рисунок 10. Тестовий приклад “Valid password. Logging in”

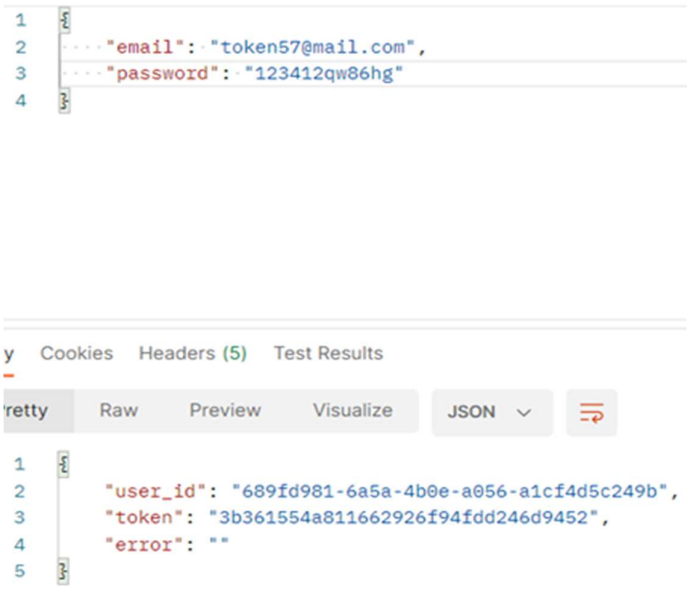


Рисунок 11. Перевірка на успішність входу

ВИКОРИСТАННЯ СИСТЕМИ

Розглянемо варіант неактивованого користувача. При відкритті, кожен користувач потрапляє на головну сторінку.

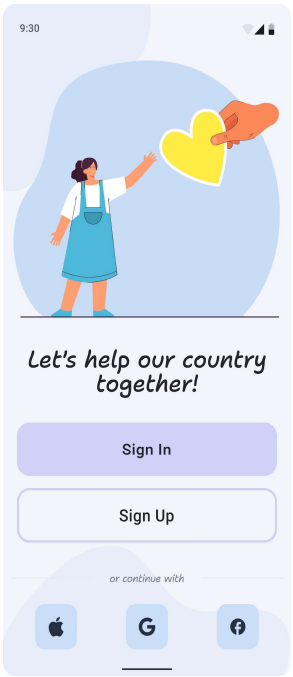


Рисунок 12. Головна сторінка

Далі реєстрація, вводимо ім’я та пошту, на пошту прийде пароль який треба ввести, і останній етап – придумати пароль.

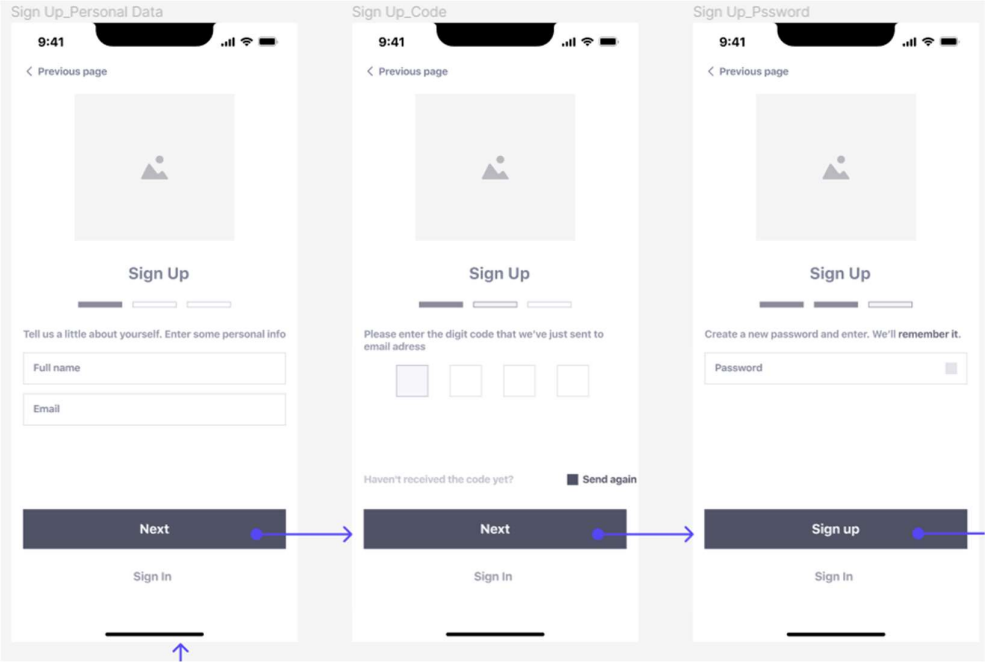


Рисунок 13. Реєстрація в системі.

Після реєстрації потрапляємо на сторінку із запитами. Бачимо фільтрацію, знак “додати запит”. Окремо можна дивитися лише свої публікації, або збережені інших користувачів. Відразу бачимо інформацію по запитам : фотографію користувача, опис, місто, збереження публікації та написати особисто автору запиту.

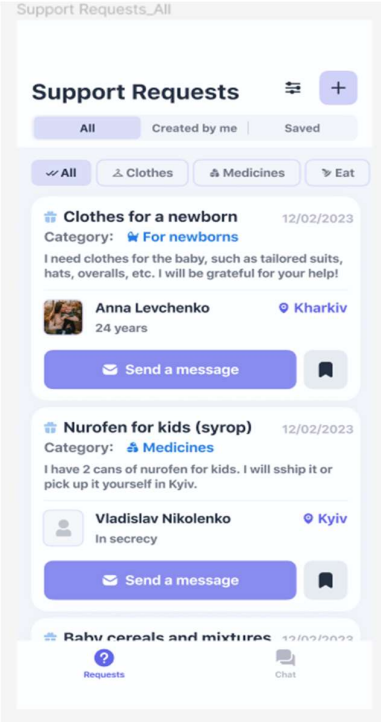


Рисунок 14. Всі запити

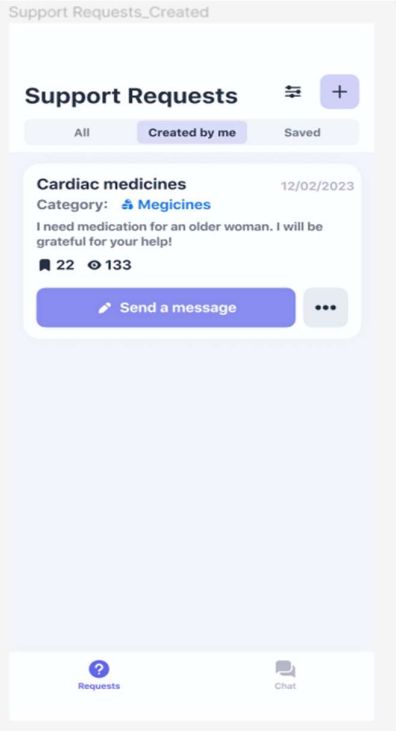


Рисунок 15. Особисті запити

Можемо створити особистий запит, для цього треба ввести:

Request name – заголовок (короткий та гучний опис).

I need/I help – обрати надає чи потребує користувач допомогу.

Choose the category - обрати категорію речі (їжа, ліки, їжа, одяг, для малюків, для вагітних, засоби інтимної гігієни, текстиль).

Enter your location – обрати місто/село в якому знаходиться користувач

Write short description for the ad. – написати опис товару.

Додати Фотографії.

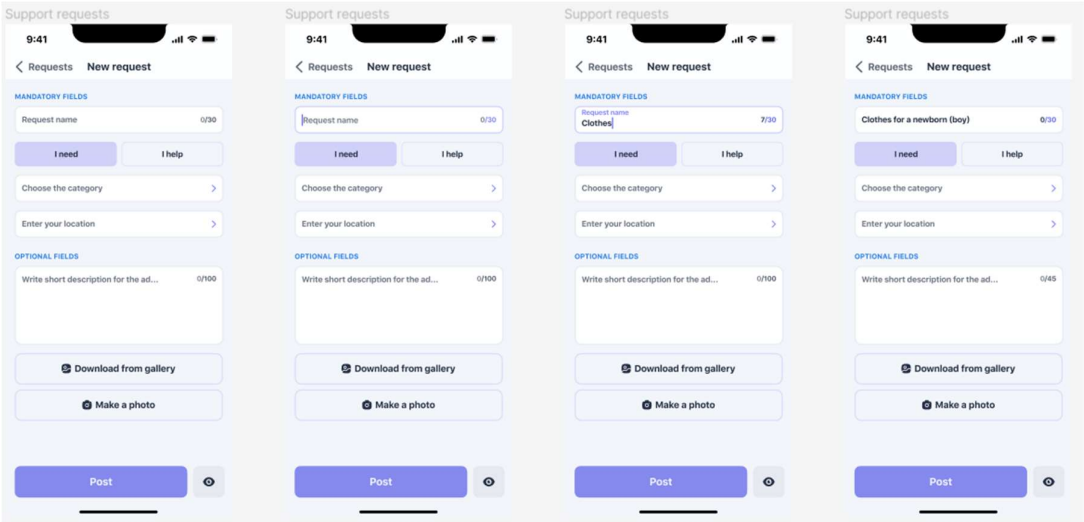


Рисунок 16. Заголовок запиту

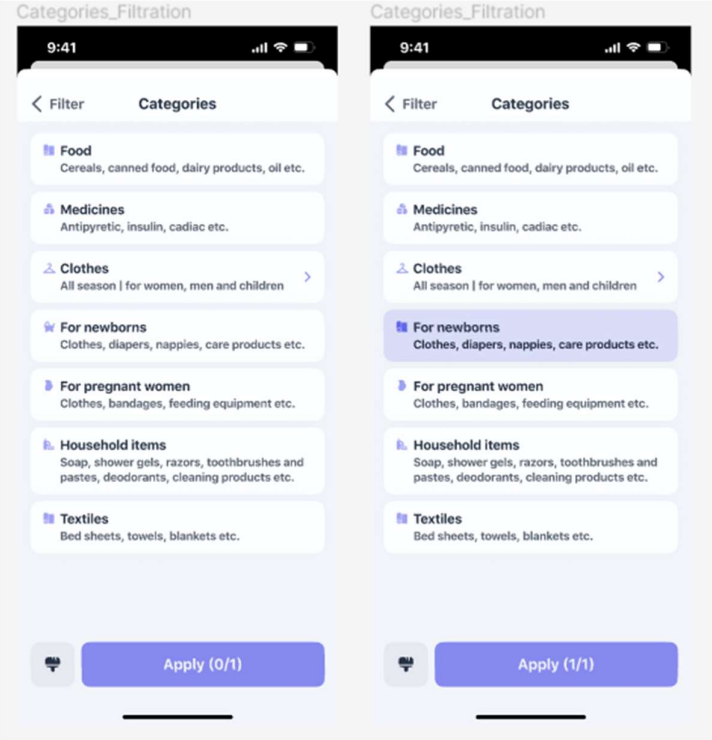


Рисунок 17. Обрання категорії товару

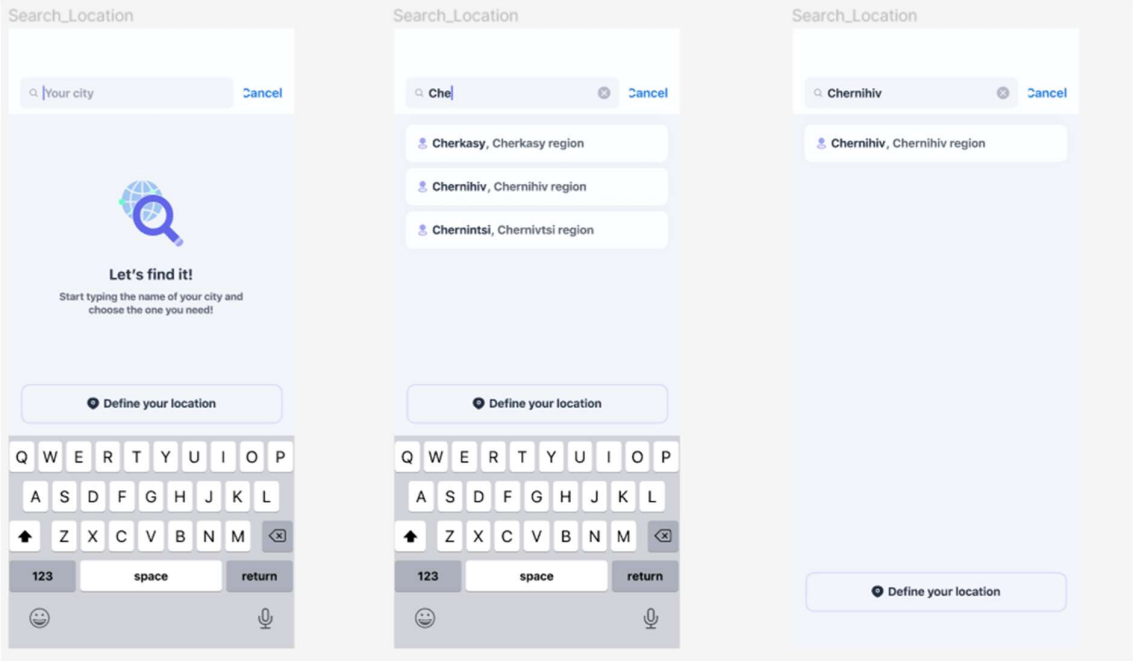


Рисунок 18. Обрати місто

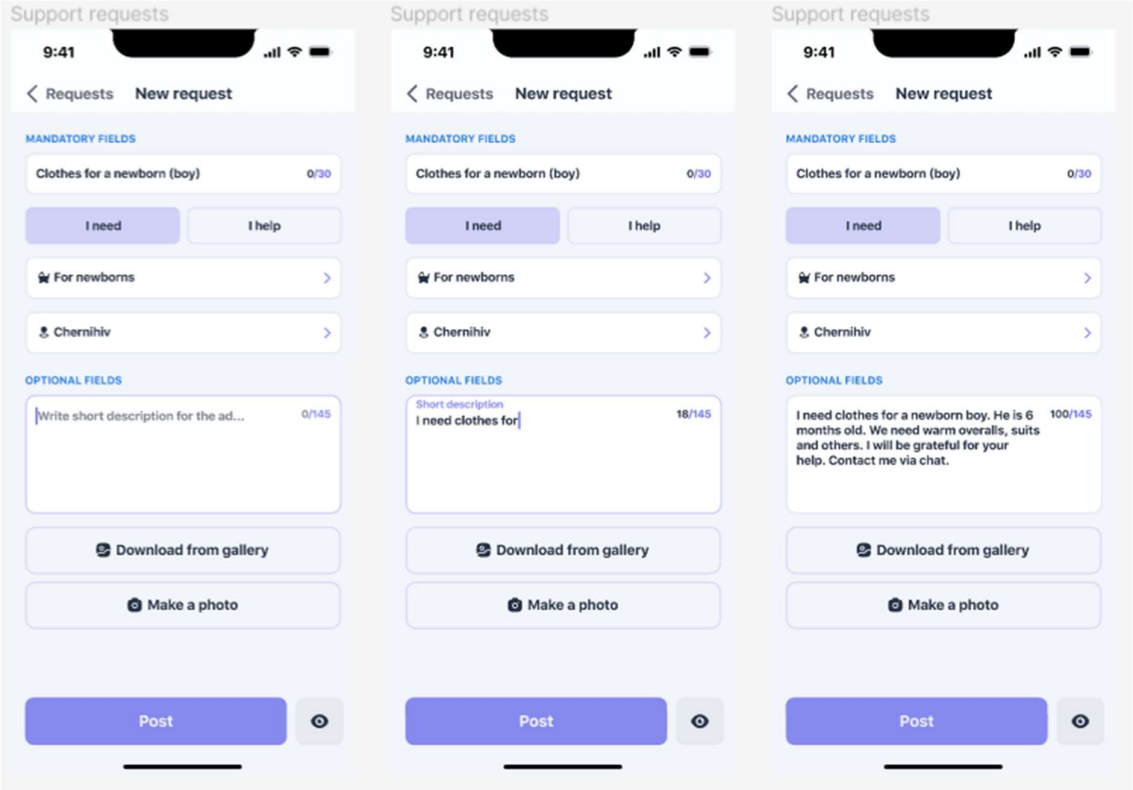


Рисунок 19.Додати опис товару

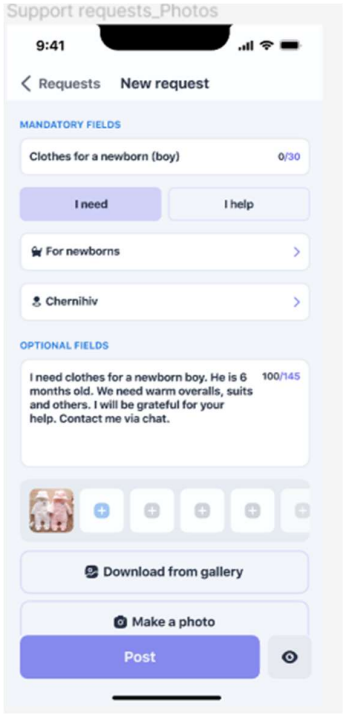


Рисунок 20. Додати фотографію товару

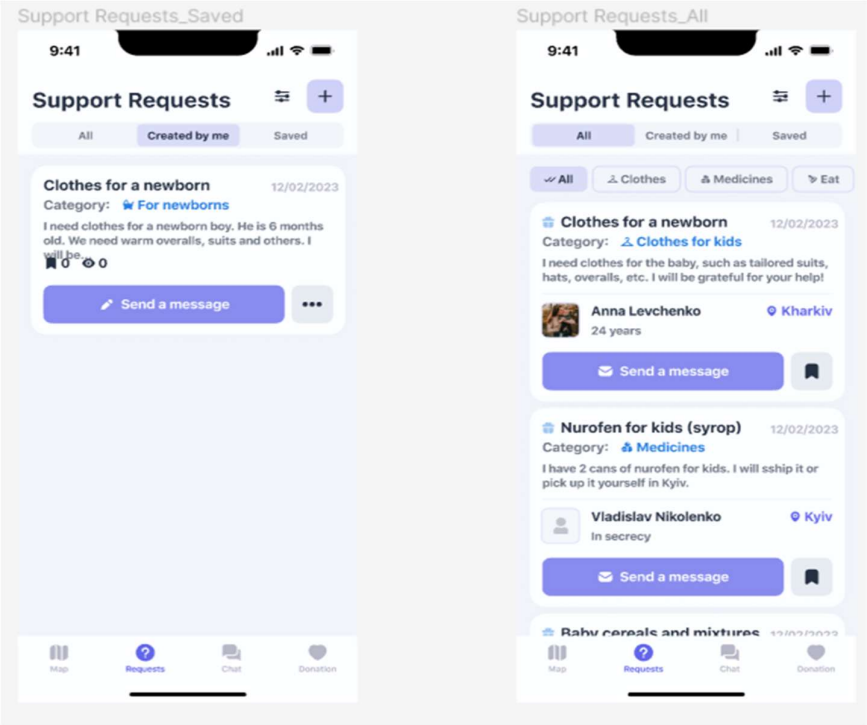


Рисунок 21. Успішна публікація запиту

ФОРМАЛЬНА СПЕЦИФІКАЦІЯ ТА ВЕРИФІКАЦІЯ

Розробити специфікацію було вирішено для модуля перевірки електронної пошти, написаного на C#. Як інструмент обрано бібліотеку Contracts із System.Diagnostics.Contracts, яка дозволяє вводити перед- та постумови, інваріанти, здійснювати перевірки коду в процесі. Найбільшу увагу було приділено перевірці змінних на null та коректності коду, який генерується для валідації пошти.

```
[ContractAbbreviator]
5 references
private void EnsureStringNotNull(string str)
{
    Contract.Requires(str != null);
}


[ContractVerification(true)]
[HttpPost(Name = "CheckEmail")]
0 references
public string Post(string email)
{
    // check validity
    EnsureStringNotNull(email);
    try
    {
        var eAddress = new MailAddress(email);
    }
    catch
    {
        return "";
    }

    // generate code
    string pr_code = GenerateCode();
    Contract.Requires(!string.IsNullOrEmpty(pr_code));

    Contract.Ensures(Contract.Result<string>() != null);
    // send email
    if (!SendEmail(email, pr_code)) return "";

    // send response
    Contract.Assert(!string.IsNullOrEmpty(email));
    return pr_code;
}
```

Рисунок 22. Контракти методу Post

```
[ContractVerification(true)]
1 reference
private string GenerateCode()
{
    Random _random = new Random();
    var builder = new StringBuilder(8);

    char offset = 'A';
    const int lettersOffset = 26;

    for (var i = 0; i < 8; i++)
    {
        var @char = (char)_random.Next(offset, offset + lettersOffset);
        Contract.Assert(char.IsLetter(@char));
        builder.Append(@char);
    }

    Contract.Ensures(Contract.Result<string>() != null);
    return builder.ToString();
}
```

Рисунок 23. Контракти для генерації коду

ОПИС ПРОЦЕСУ РОЗРОБКИ

Опис обраної методології та інструментарію. Під час розробки додатка було використано наступні технології. Для організації та планування роботи:

· Google Docs&Sheets – зручний інструмент для ведення та зберігання документації, дозволяє декільком користувачам одночасно працювати над документом.

· Trello – інструмент для планування та трекінгу виконання завдань, канбан-дошка проєкту.

· Figma – інструмент для створення макета майбутнього додатка, дуже зручний у використанні, не потрібно витрачати час на написання коду, достатньо перемістити основні структурні елементи на шаблон.

·Android Studio -інтегроване середовище розробки для платформи Android.

·Postman - це API-платформа, на якій розробники можуть розробляти, створювати, тестувати та повторювати свої API.

·Flutter - це програмний каркас із відкритим кодом для створення додатків для платформ Android та iOS, а також на вебі, розроблений компанією Google.

·Go - компільована мова програмування із вбудованими засобами для паралельних обчислень і засобами віддаленого керування пакунками. Цю мову програмування розробив Google як частину проєкту з розробки операційної системи Inferno.

MongoDB - документо-орієнтована система керування базами даних з відкритим вихідним кодом, яка не потребує опису схеми таблиць. MongoDB займає нішу між швидкими й масштабованими системами, що оперують даними у форматі ключ/значення, і реляційними СКБД, функціональними та зручними у формуванні запитів.

Демонстрація планування спринтів. Після кожної зустрічі з викладачами команда збиралась ввечері для обговорення всіх питань, які наступні задачі кожного з члена команди, які труднощі наразі є, які завдання встигаємо зробили, які ні, також відразу приділяли увагу на те, що рекомендували зробити викладачі, де і які проблеми в нас наразі існували – щоб якомога швидше все виправити. Приділяли увагу на самопочуття кожного члена команди або сімейні обставини, не турбували один одного без вагомої причини в таких випадках.

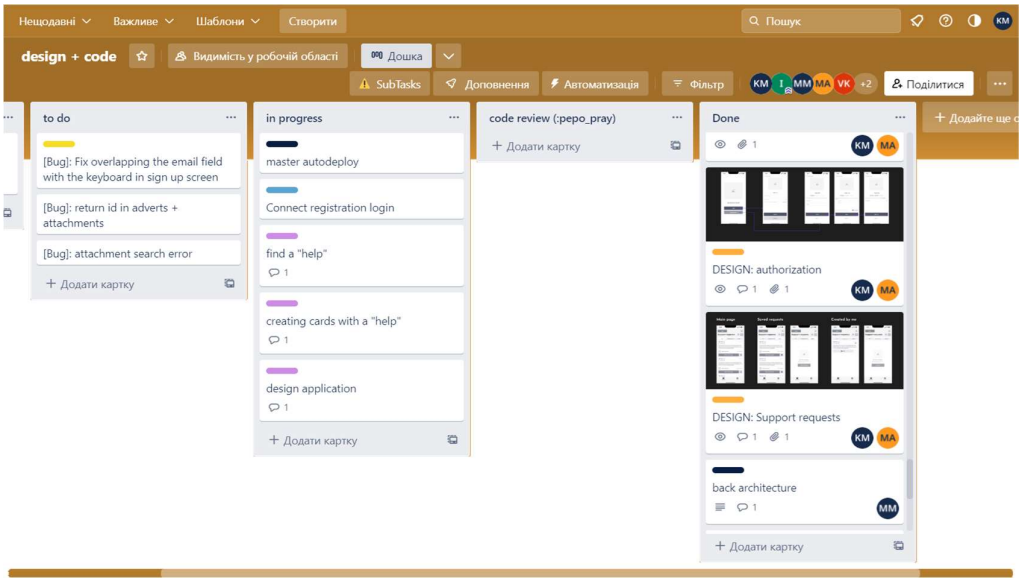


Рисунок 24. Дошка Trello

Наші плани на кожен тиждень:

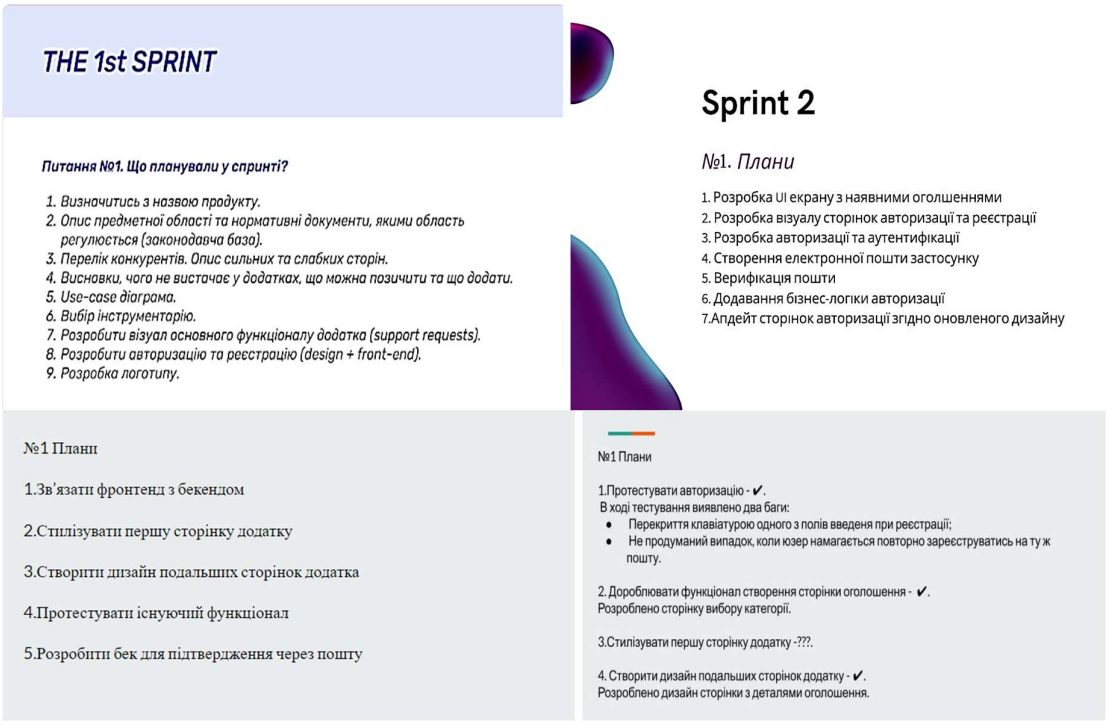


Рисунок 25. Приклад планування на тиждень

Рефлексія. Проблема була лише на початку спільного проєкту: не зрозуміли як працювати один з одним, треба було звикнути, які кому ролі треба взяти, який саме проєкт треба реалізувати. Після першої зустрічі з викладачами – нам допомогти у вирішенні всіх питань, та змоги направити на вірний шлях, в той же день ввечері влаштували спільну онлайн зустріч, і після цього робота пішла достатньо швидко. Всі комунікували один з один без проблем, ніяких сутичок не виникало. Були проблеми з

тим, що могли недооцінити масштаби завдань – в такому випадку робили все можливе, щоб на наступну зустріч з викладачами все зробити.

ВИСНОВКИ

В результаті роботи було розроблено застосунок “Helper”. Студенти отримали цінний досвід роботи в команді з використанням реальних методологій розробки, на зразок Trello. В ході виконання роботи команда провела аналіз схожих систем на ринку та визначили головні риси майбутньої системи. Було поглиблено знання формальних методів специфікації програм та перевірки якості створюваного продукту. Кожен учасник проєкту мав змогу спробувати себе у цікавій йому ролі та отримати потрібний досвід. Було отримано або поглиблено знання таких технологій розробки: Бек-енд: Flutter, Go, MongoDB, Фронт-енд: Dart, Flutter, Android Studio, Дизайнери: Figma, Тестування: Postman, Manual testing. В кінці роботи було отримано застосунок, над яким в подальшому плануємо працювати.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Trello Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/ru/software/trello.
2. Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/
3. Google Sheets [Електронний ресурс]. – Режим доступу: https://www.google.com/intl/uk_ua/sheets/about/
4. Figma [Електронний ресурс]. – Режим доступу: https://www.figma.com/
5. Google Classroom [Електронний ресурс]. – 2023. – Режим доступу до ресурсу: https://classroom.google.com/ .
6. Flutter [Електронний ресурс]. – Режим доступу : https://flutter.dev/
7. Dart [Електронний ресурс]. – Режим доступу : https://dart.dev/
8. Golang [Електронний ресурс]. – Режим доступу : https://go.dev/

ДОДАТКИ

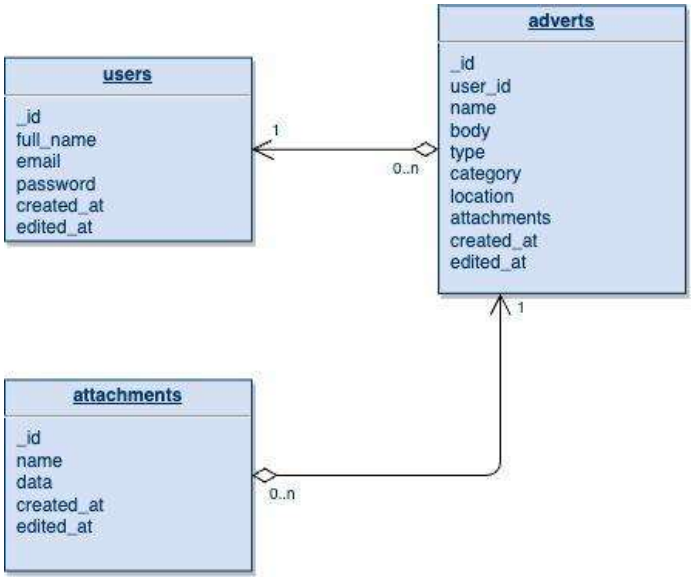
Додаток А. Use-case діаграма

Додаток Б

Посилання на репозиторій проекту на GitHub: https://github.com/supperdoggy/helper_backend

Додаток В

Схема бази даних



GYM MANAGEMENT SYSTEM

Yevhen Yeris, Hlib Petruk, Vasyl Hryts, Mykhailo Shvets, Oleksandr Holovach, Ihor Butenko, Kiryl Ampa Giovanni Atasse Johnson

INTRODUCTION

Sports complex networks have become increasingly popular as people across society acknowledge the significance of engaging in physical activity. These complexes serve as convenient hubs for individuals to prioritize their well-being and lead healthier lives, even amidst sedentary lifestyles. By subscribing to these complexes, people gain the freedom to select training times, locations, sports, exercises, coaches, and create personalized programs tailored to their needs. The implementation of efficient management systems streamlines essential processes like tracking attendance, handling subscription payments, and organizing coach schedules. This ensures that users and coaches can easily access their schedules, while administrators effectively manage locations, time slots, and coach assignments.

Goals and Objectives. The goal of this project is to create a web application for managing a sports complex, allowing online management of the facility. The team members aim to enhance their skills in application design, development, and testing, while also improving their soft skills in teamwork and time management. The project will be carried out through various stages of work to accomplish the following tasks:

- Analyze the market by identifying the strengths and weaknesses of real systems.
- Design the future application and anticipate its application.
- Develop a design using Figma.
- Create a relational database, REST API, and React client.
- Support development by testing.
- Establish a continuous process of team development.

Methods and Tools. The object of development is the process of managing a sports complex, and the subject is a web application for automating online management.

Tools utilized for backend development are mainly Microsoft technologies, such as .NET and SQL Server. The frontend part is developed using React and its extensions. The development process follows an Agile methodology, ensuring an iterative and collaborative approach to the project.

MARKET ANALYSIS

Through market analysis conducted by the team's Business Analyst (BA), an evaluation of the pros and cons of existing competitors was carried out, as outlined in Table 1.

Based on the system features mentioned earlier, the primary goal was to develop an application that simplifies the management of a sports complex while offering the flexibility to add new sports activities and gym facilities. This capability sets our application apart from the competing solutions currently available in the market.

USED TECHNOLOGIES

While choosing technologies and tools our team considered two key factors: the need to develop the application in the short term and the desire to acquire new tech skills. As a result, the backend was developed using technologies that most team members were already familiar with, ensuring efficiency and speed in development. On the other hand, for the frontend part, React JS was chosen, even though none of the team members had prior experience with it.

Table 1. Comparing analogues

Name	Website	Price	Advantages	Disadvantages
Glofox	https://www.glofox.com/products/	-	easy administration easy calendar setup predefined types of sports option to choose number of sports centers mobile version available - option to add a shop	inability to adjust the number of centers during operation inability to add new activities - price/quality ratio not suitable for different price categories of the application
Virluagym	https://business.virtuagym.com/gym-software/	140$/mo	clear navigation and management easy setup - ability to add a new role and configure access	- inability to add a new type of sport that is not provided by the developers

The project was developed using **Visual Studio** and **Visual Studio Code** as the integrated software environment, utilizing C# and JavaScript as the programming languages, along with the ASP.NET Core and Entity Framework Core 6. ASP.NET Core is a versatile framework for web-oriented applications, offering high performance and cross-platform compatibility. AutoMapper was used to configure business models mapping.

The database was organized using **SQL Server** Management Studio. On the frontend, JavaScript was used, specifically **React JS**, along with HTML, CSS, and Material UI 5 technology. React JS is a powerful library for building user interfaces, enabling efficient updates without page reloading.

To facilitate API development, **Swagger** tools were utilized for development, API interaction, and documentation. Unit testing was conducted using **xUnit.net**, an open-source testing tool for .NET, along with **Moq** for object imitation.

Design work was performed using **Diagrams.net**, **Figma**, and **WireframeSketcher**. Collaborative work and communication took place through Google Classroom, shared Google Drive, Discord meetings, and project management was facilitated by Trello, following the Kanban board methodology.

GitHub, a collaborative software development service, was used for efficient version control and collaboration throughout the project.

PURPOSE AND OBJECTIVES OF SYSTEM CREATION

The purpose of the sports management system is to provide customers of a sports complex with convenient access to available training options and management of their schedules, as well as to facilitate the management of the sports complex.

The work involves:

– analyzing methods, techniques, and models used for developing web applications

– designing and implementing the "SportManager" system

– implementing the ability to access a personal account with a personal schedule for both the client and the trainer, selecting a type of sport for training, adding new locations and types of sport, and editing schedules.

The objective of creating the "SportManager" system is to provide multifunctional capabilities for managing sporting events in a sports complex.

System requirements

The system includes the following functional subsystems:

– **administrative**, intended for creating and editing locations and types of sport, setting schedules for specific types of sport.

– **client and trainer**, intended for attending the sports complex, such as selecting a type of sport for training, creating a training schedule, and booking sports classes.

Administrator subsystem.

Table 2. Admin subsystem functions

Function	**Task**
Manage the work of system users	Editing and deleting information about system users
	Changing the role of a user from visitor to coach
Working with sports complex locations	Viewing the list of locations
	Adding a new location
	Editing and deleting information about a location
Working with activities (sports)	Viewing the list of activities
	Adding a new activity
	Editing and deleting information about an activity
	Associating a location with an activity
Working with the schedule of classes/workouts	Viewing the schedule of classes/workouts
	Adding and deleting time slots from the schedule of classes/workouts

Subsystem for user.

Table 3. User subsystem functions

Function	**Task**
Registration	Entering personal information
	Updating registration
	Deleting personal information
Schedule management	Viewing own class schedule
	Booking a visit: 1. Selecting a sport/activity 2. Viewing the schedule on the calendar (each calendar day consists of one-hour time slots) 3. Selecting a time slot 4. Selecting a location 5. Confirming the registration
	Canceling a visit: 1. Viewing own booked visits on the calendar 2. Selecting a visit 3. Deleting a visit

System requirements.

The system should provide correct data display in the following up-to-date browsers: Google Chrome, Opera, Microsoft Edge.

On any incorrect user actions related to entering incorrect data, not filling in required input fields in forms, and others that can be processed by the system, corresponding error messages in Ukrainian language are generated within the overall website design.

Operating system: Windows 10 and above.

DESCRIPTION OF DATABASE

Logical structure of the database/ Microsoft SQL Server Management Studio was used as a DBMS. Figure 1 shows the database diagram.

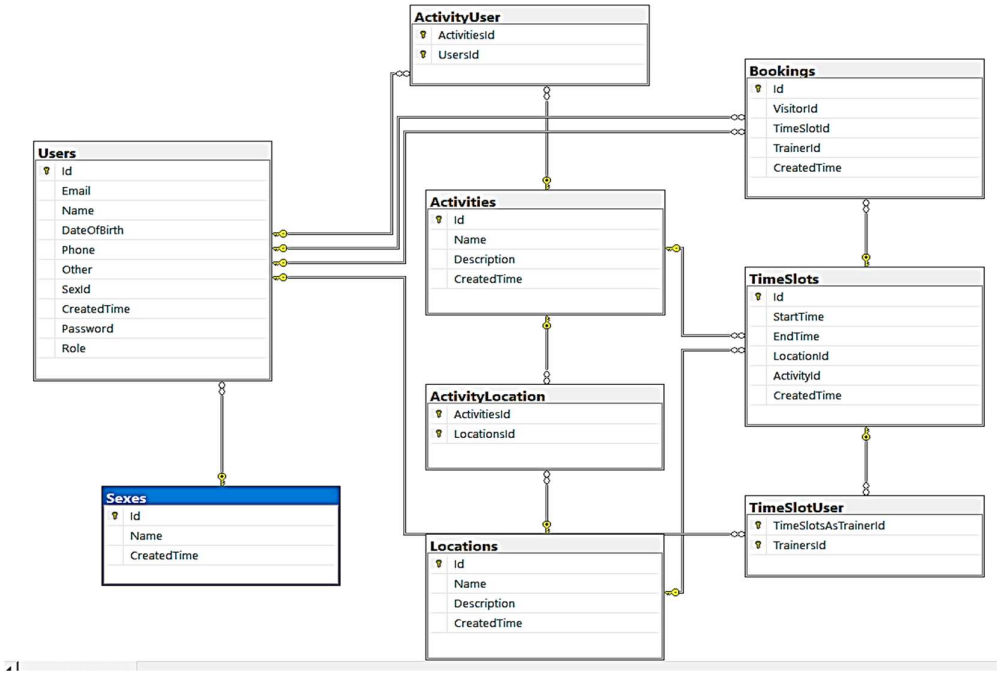


Figure 1. Database Diagram

List and brief description of the database tables are given in Table 5.

Table 5. Tables

Number	Table	Description
1	ActivityUser	An auxiliary table to link activities and users
2	Bookings	A table to store information about class bookings
3	TimeSlots	A table to store information about slots in the class schedule
4	Activities	A table to store information about activities
5	ActivityLocation	An auxiliary table to link activities and locations
6	TimeSlotUser	An auxiliary table to link slots in the schedule and users
7	Locations	A table to store information about locations
8	Sexes	A table to store information about genders
9	Users	A table to store information about users

Description of tables. An auxiliary table to link activities and users.

Table 6. Identifiers of activities and users

ActivityUser Table Attribute	Type	Description
ActivitiesId	int	Identifier of activities
UsersId	int	Identifier of users

Table for storing information about class bookings

Table 7. Bookings

Bookings Attribute	Type	Description
Id	int	Booking ID
VisitorId	int	Visitor ID
TimeSlotId	int	Time slot ID
TrainerId	int	Trainer ID
CreatedTime	datetime2	Creation date/time

Table for storing information about schedule slots.

Table 8. TimeSlots

TimeSlots Attribute	Type	Description
Id	int	Id of time slot
StartTime	datetime2	Start time
EndTime	datetime2	End time
LocationId	int	Id of location
ActivityId	int	Id of activity
CreatedTime	datetime2	Creation date

Table for storing information about activities.

Table 9. Activities

Activities Attribute	Type	Description
Id	int	Activity Id
Name	nvarchar	Name
Description	nvarchar	Description
CreatedTime	datetime2	Created Time

Auxiliary table to link activity and location.

Table 10. ActivityLocation

ActivityLocation Attribute	Type	Description
ActivitiesId	int	Identifier of activities
LocationsId	int	Identifier of locations

Auxiliary table to link schedule slots and users

Table 11. TimeSlotUser

TimeSlotUser Attribute	Type	Description
TimeSlotsAsTrainerId	int	Identifier of time slots assigned to a trainer
TrainersId	int	Identifier of trainers

Table for storing information about locations.

Table 12. Locations

Locations Attribute	Type	Description
Id	int	Location ID
Name	nvarchar	Name
Description	nvarchar	Description
CreatedTime	datetime2	Creation date

Table for storing information about genders.

Table 13. Genders

Таблиця Sexes Attribute	Type	Description
Id	int	Identifier of gender
Name	nvarchar	Name
CreatedTime	datetime2	Creation date

Table for storing information about users.

Table 14. Users

Users Table Attribute	Type	Description
Id	int	User ID
Email	nvarchar	Email
Name	nvarchar	Name
DateOfBirth	datetime2	Date of birth
Phone	nvarchar	Phone
Other	nvarchar	Other information
SexId	int	Gender ID
CreatedTime	datetime2	Registration date
Password	nvarchar(MAX)	Password
Role	int	Role

SYSTEM IMPLEMENTATION

The developed system (Figure 2) comprises a combination of three key components:

1. Data storage in the form of a relational database.
2. Three layered web-application.
3. A separate frontend client.

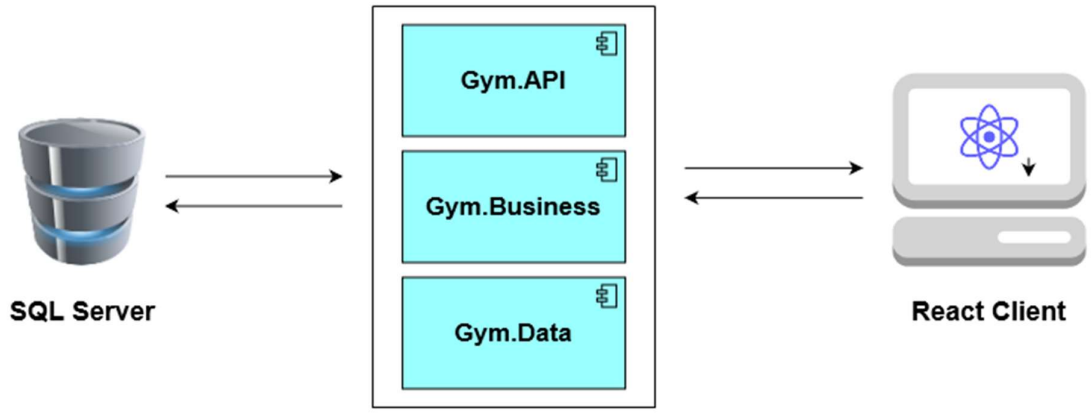


Figure 2. Common system architecture

The system architecture creation was mainly guided by the intention to develop an easily testable and flexibly upgradeable application.

Gym.Data is a .NET project containing definitions and configurations of the business entities, Entity Framework context and data migration history used for the Code First approach.

The Gym.Business project holds the core logic of the system. It contains service classes abstractions and implementations, as well as model definitions. The Figure 3 shows IScheduleService as the example of a service class abstraction. The services perform data manipulation and extraction tasks involving the required validation. Separation of this layer facilitated its testability as it was easier to abstract from data source and outer interface while designing automated tests.

```
public interface IScheduleService
{
    2 references
    IEnumerable<TimeSlotModel> GetTimeSlots(ScheduleRequestModel request);

    2 references
    Task<IEnumerable<TimeSlot>> CreateTimeSlots(AddSlotsRequest addSlotsRequest);

    2 references
    Task DeleteTimeSlot(int id);

    2 references
    IEnumerable<Booking> GetBookings(ScheduleRequestModel request);

    2 references
    Task<Booking> CreateBooking(Booking booking);

    2 references
    Task DeleteBooking(int id);

    2 references
    Task DeleteBooking(BookingModel model);
}
```

Figure 3. IScheduleService

Application interface is implemented in the Gym.API component, representing an ASP.NET WebApi project. Positioning the business logic and data access separate layers allowed to create lightweight controllers (Figure 4).

```
[HttpGet("{id}")]
0 references
public async Task<ActionResult<UserModel>> Get(int id)
{
    try
    {
        var entity = await _userService.GetEntity(id);
        var viewModel = _mapper.Map<UserModel>(entity);
        return Ok(viewModel);
    }
    catch (ArgumentNullException ex)
    {
        return NotFound(ex.Message);
    }
}

[HttpGet("me")]
0 references
public async Task<ActionResult<UserModel>> Me()
{
    try
    {
        var entity = await _userService.GetEntity(_clientAuthorizationData.UserId);
        var viewModel = _mapper.Map<UserModel>(entity);
        return Ok(viewModel);
    }
    catch (ArgumentNullException ex)
    {
        return NotFound(ex.Message);
    }
}
```

Figure 4. Data access

TESTING

The Bottom-up approach was suitable choice for our small group project for several reasons. First, as the project is small, the amount of code to be tested might not be too large. This means that the tester can focus on the code and architecture to develop tests.

Second, since this approach involves testing the code, it can help identify any issues with the application's internal workings that might not be evident at the higher levels of testing. This can help the team to ensure that the application works as expected and improve its quality.

Third, the Bottom-up approach allows for a more detailed and thorough check of the application's operation. This is especially important in a small group project, where there might not be many resources available for testing. By using the Bottom-up approach, the team can ensure that the application works correctly, and there are no hidden issues that might impact the project's success.

Param choosing

1. Time to module build - In the Bottom-up approach, testing starts at the code level, which means that modules are built as testing progresses. As a result, time to module build can be shorter than in other approaches.

2. Time to create initial program "skeleton" versions - Bottom-up approach allows for creating an initial program "skeleton" quickly, as testing can begin at the code level and then move upward.

3. Need for drivers and other testing tools - The Bottom-up approach requires drivers and testing tools to simulate higher-level components, but the use of these tools is reduced compared to other approaches.

4. Need for placeholders - The Bottom-up approach does not require placeholders since testing starts at the code level.

5. Degree of parallelism - The Bottom-up approach allows for a higher degree of parallelism since testing can be done at the individual module level concurrently.

6. Ability to test individual paths - The Bottom-up approach allows for testing individual paths since it starts with testing at the code level.

7. Ability to plan and control sequence - The Bottom-up approach requires careful planning and coordination since testing starts at the code level and moves upward. However, it also allows for more control over the testing sequence since modules are built and tested one by one.

Table 15. Test Cases

Id	Test scenario	Test Case	Test Steps	Test Data	Expestation
1	Registration	Test registration of a new user	1. Open the web-page 2. Feel text fields according to rules 3. Push the button 4. Check the user list	User name: Tony Stark Email: tonyStark_avenger@fmail.com Password: tony13 Repassword: tony13	See new user named Tony Stark
2	Registration validation	Test registration validation	1. Open the web-page 2. Push the registration button on every error case (blank fields, inappropriate values, password not equal to second password)	User name: Email: something Password: tony13 Repassword: ton	Every time we push the button we got new error or warning
3	Add activity	Test creation of new activity	1. Open the web-page 2. Push “Add Activity” button 3. Fill the Form 4. Click “Save” button. 5. Check activities list	Name: Swimming Info: Any kind of swimming and diving	New line about Swimming appear in list

4	Activity adding activity validation	Check activity adding validation	1. Open the web-page. 2. Push the add button on every error case 3. Try to add existing	Name: Info:	Nothing new appear in the list
5	Add location	Create new location	1.Open the web-page 2.Push “Add Location” button 3.Fill the Form 4.Click “Save” button. 5.Check activities list	Name: Pool Info: Moderm sweeming pool for any kind pf water activity	New line appears
6	Activity adding location validation	Check location adding validation	1. Open the web-page. 2. Push the add button on every error case 3. Try to add existing	Name: Info:	Progtam don`t add the wrong location

An example of tests designed using xUnit and Moq.

```
[Fact]
public async Task Authenticate_ValidCredentials_ReturnsUserAccessToken()
{
    // Arrange
    var email = "test@test.com";
    var password = "password";
    var userId = 1;
    var hashedPassword = password.GetHash();
    var user = new User { Id = userId, Email = email, Password = hashedPassword };
    var expectedToken = new UserAccessToken { Token = "token", UserId = userId };

    _userServiceMock.Setup(x => x.GetAllEntities())
        .Returns(new[] { user }.AsEnumerable());

    _oAuthTokenProviderMock.Setup(x => x.RegisterToken(userId))
        .Returns(expectedToken);

    // Act
    var result = await _authService.Authenticate(email, password);

    // Assert
    Assert.Equal(expectedToken, result);
}

[Fact]
public async Task Authenticate_InvalidEmail_ThrowsInvalidDataException()
{
    // Arrange
    var email = "test@test.com";
    var password = "password";

    _userServiceMock.Setup(x => x.GetAllEntities())
        .Returns(Enumerable.Empty<User>());

    // Act and Assert
    await Assert.ThrowsAsync<InvalidDataException>(() => _authService.Authenticate(email, password));
}
```

```
[Fact]
public async Task AddEntity_AddNewActivity_ActivityAdded()
{
    // Arrange
    var testList = new List<Activity>();

    var mockSet = new Mock<DbSet<Activity>>();
    mockSet.Setup(d => d.Add(It.IsAny<Activity>())).Callback<Activity>((s) => testList.Add(s));

    var gymContextMock = new Mock<GymContext>();
    gymContextMock.Setup<IQueryable<Activity>>(x => x.Activities).Returns(mockSet.Object);

    var newActivity = new Activity
    {
        Name = "Test 123",
        Description = "Test 123",
    };

    // Act
    var service = new ActivityService(gymContextMock.Object);
    var addedActivity = await service.AddEntity(newActivity);

    // Assert
    Assert.Equal(newActivity, addedActivity);
    Assert.Single(testList);
    gymContextMock.Verify(x => x.SaveChangesAsync(default), Times.Once);
}
```

Figure 5. An example of tests designed using xUnit and Moq

USING OF THE SYSTEM

To log in the system, users need to enter their email and password. A new member account can be registered as well.

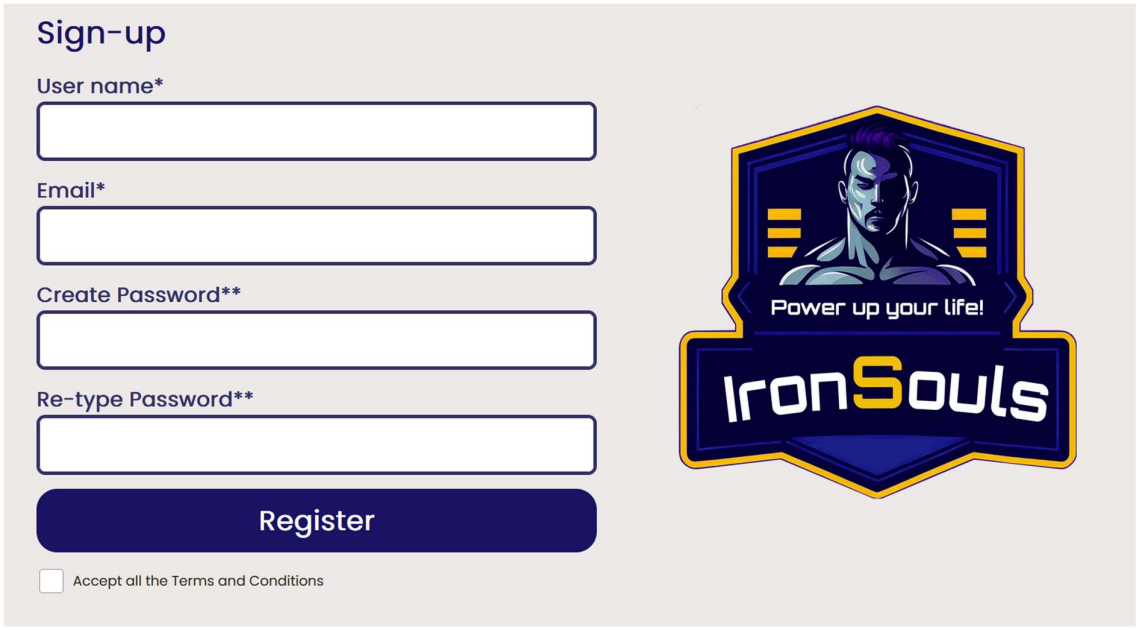


Figure 6. Logging

After signing in the user gets redirected to the main page, consisting of an introductory quote, short description and a schedule block.

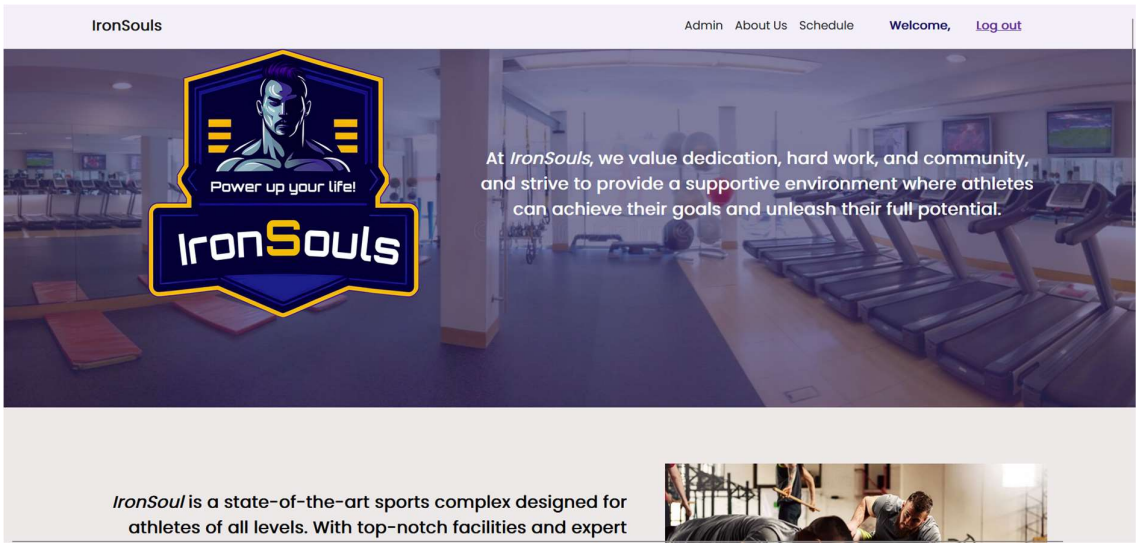


Figure 7. Main page

The schedule block comprises a calendar displaying booked sports classes and available time slots. The calendar is capable of navigating back and forth, allowing users to book classes in advance as well as to view the history of bookings.

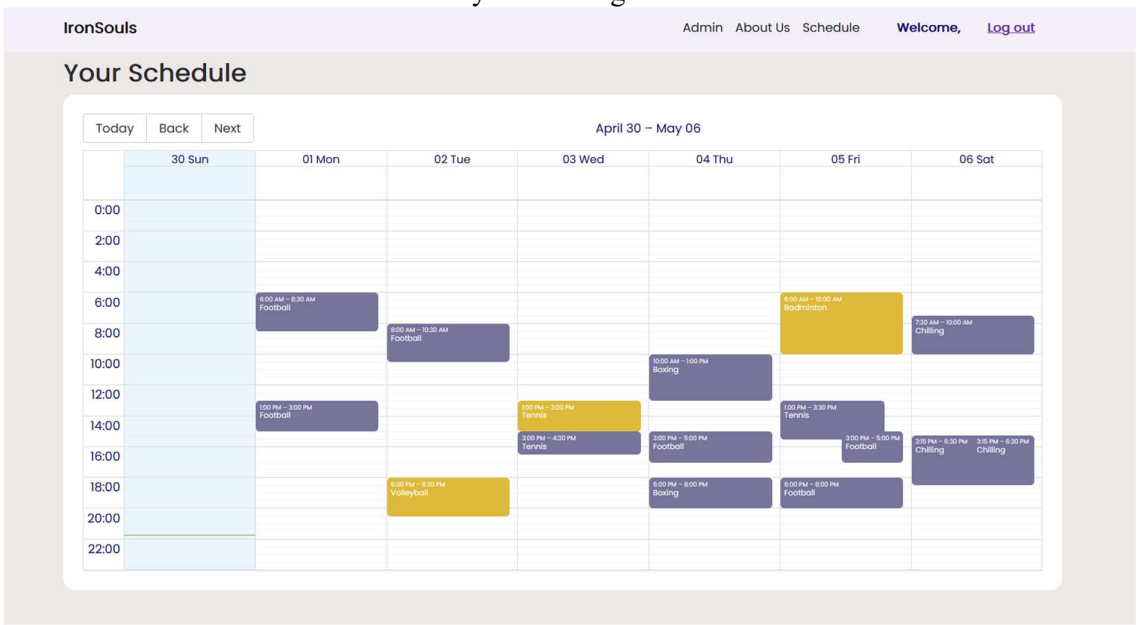


Figure 8. Scheduler

Administrator is capable of viewing a full list of gym entities, such as activities or locations. It is possible to create, update or delete these entities on dedicated pages.

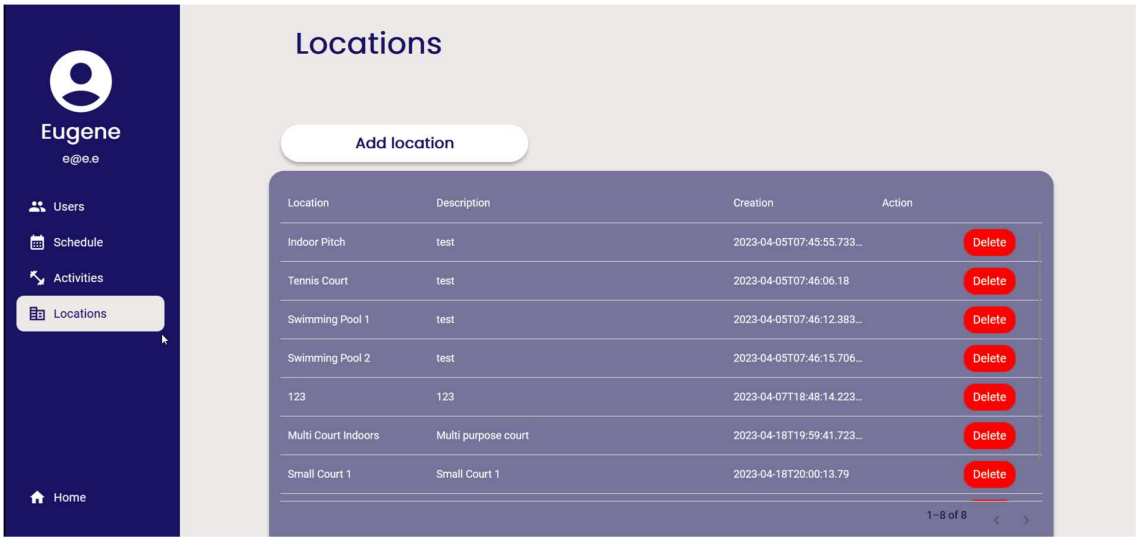


Figure 9. Locations

By navigating to the Activities page and clicking on a position in a list administrator can edit a sports activity and assign it to any of existing locations. This allows to modify any current gym configuration.

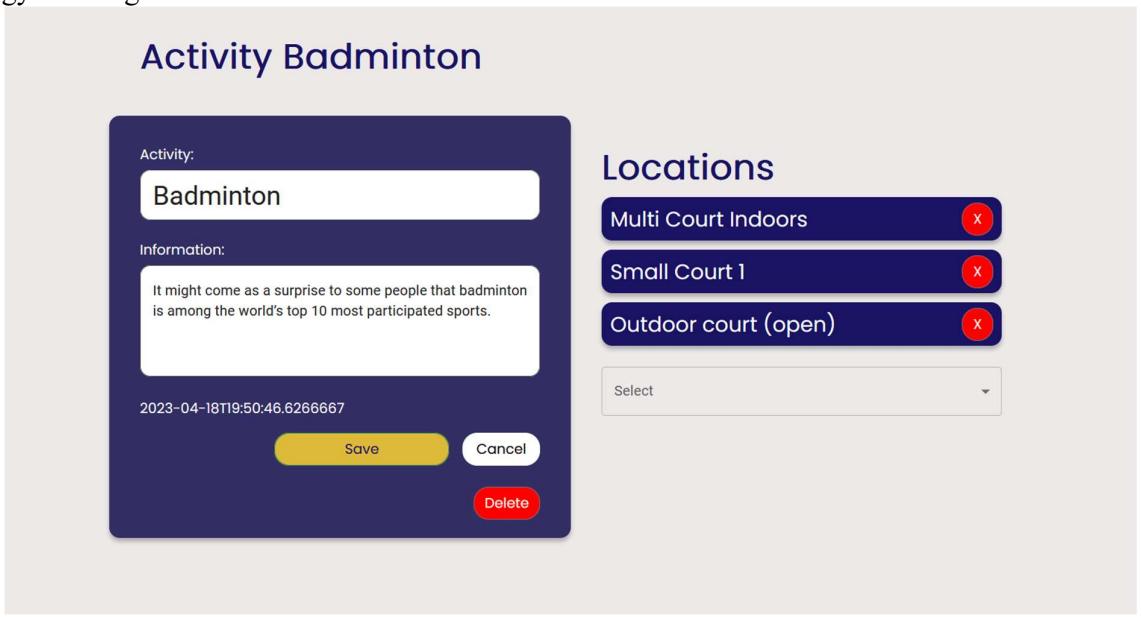


Figure 10. Activity example

Schedule configuration is performed via the dedicated page. Administrators can create a series of planned classes in advance by selecting a number of repetitions and a starting day. Figure 1, for instance, shows the creation of 10 Tennis classes at the Small Court 1 scheduled from 12:30 to 15:00 every Wednesday starting from 04/26/2023.

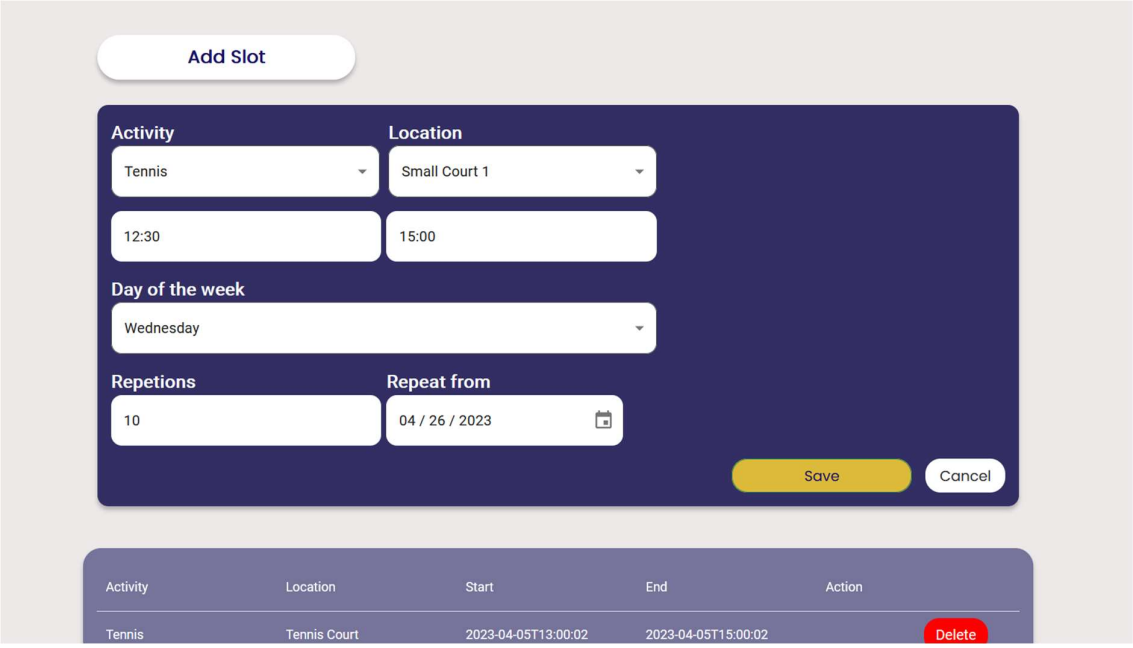


Figure 11. Slot adding

Administrator also has the rights of viewing and modifying users' personal training schedules.

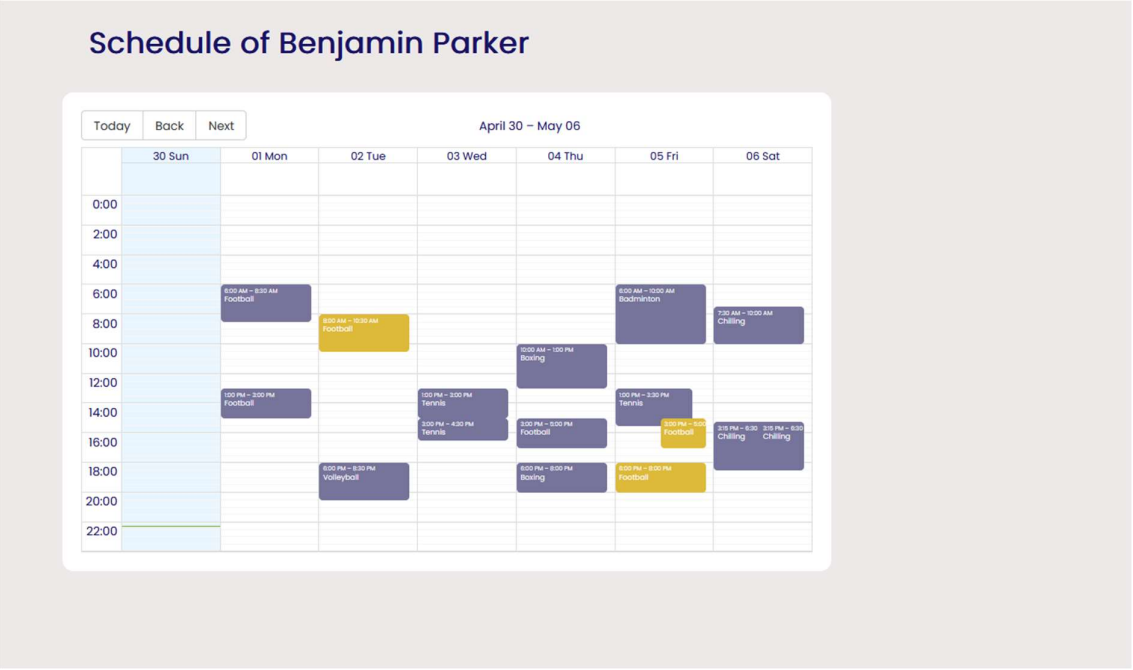


Figure 12. Example of schedule

FORMAL SPECIFICATION

The key application process of sports classes booking involves several steps, including trainer assignment, creation of a booking record, and verification and alteration of the trainer and facility volume. This process has been specified using Dafny, a programming language and

verification tool. By using Dafny, the application process can be defined through classes, methods, and invariants, allowing for precise and formal specification of the desired behavior. Dafny's language constructs and verification capabilities can help ensure correctness and reliability in the implementation of the sports classes booking system.

Dafny was chosen for specifying the key application process of sports classes booking due to its formal verification capabilities, allowing for rigorous correctness checks. The language's expressive constructs, strong static typing, and tool support aid in accurately specifying system behavior and catching errors early. The active Dafny community and available resources further contribute to its suitability for the task, providing valuable support and knowledge sharing opportunities.

```
datatype Role = Visitor | Trainer

class User
{
        var Id: int;
  var Role: Role;

  constructor(id: int, role: Role)
        {
        this.Id := id;
        this.Role := role;
        }
}

class Booking
{
        var Id: int;
  var VisitorId: int;
  var TimeSlotId: int;
  var TrainerId: int;
  var IsConflict: bool;

  constructor(id: int, visitorId: int, trainerId: int, timeSLotId: int, IsConflict: bool)
        {
        this.Id := id;
        this.VisitorId := visitorId;
        this.TrainerId := trainerId;
        this.TimeSlotId := timeSLotId;
        this.IsConflict := IsConflict;
        }
}

class TimeSlot
{
        var Id: int;
  var StartTime: int;
  var EndTime: int;
  var AvailableCount: int;
```

```
  constructor(id: int, startTime: int, endTime: int, availableCount: int)
        {
        this.Id := id;
        this.StartTime := startTime;
        this.EndTime := endTime;
        this.AvailableCount := availableCount;
        }
    }

    class BookingService
    {
        var Users: set<User>;
      var TimeSlots: set<TimeSlot>;
      var Bookings: set<Booking>;

      constructor(users: set<User>, timeSlots: set<TimeSlot>, bookings: set<Booking>)
        {
        Users:= users;
        TimeSlots:= timeSlots;
        Bookings:= bookings;
        }

        predicate userExists(userId: int)
        reads this
        reads this.Users
      {
        exists user: User::user in this.Users && user.Id == userId
      }

        predicate userInRoleExists(userId: int, role: Role)
        reads this
        reads this.Users
      {
        exists user: User::user in this.Users && userExists(userId) && user.Role == role
      }

        predicate timeSlotExists(timeSlotId: int)
        reads this
        reads this.TimeSlots
      {
        exists timeSlot: TimeSlot::timeSlot in this.TimeSlots && timeSlot.Id == timeSlotId
      }

        predicate isSlotVacant(timeSlotId: int)
        reads this
        reads this.TimeSlots
        {
        exists timeSlot: TimeSlot::timeSlot in this.TimeSlots && timeSlot.Id == timeSlotId
&& timeSlot.AvailableCount > 0
        }
```

```
        predicate conflictsExist(userId: int, timeSlotId: int)
        reads this
        reads this.TimeSlots
        reads this.Bookings
        reads this.Users
    {
        userExists(userId)
        && timeSlotExists(timeSlotId)
        && !exists booking: Booking::booking in this.Bookings // booking
                    && (booking.VisitorId == userId || booking.TrainerId == userId)
// user is visitor or trainer of the booking
                    && exists timeSlot: TimeSlot :: timeSlot in this.TimeSlots // time
slot
                                && timeSlot.Id == booking.TimeSlotId // time
slot of the booking
                                && exists bookingConflict: Booking ::
bookingConflict in this.Bookings && bookingConflict.Id == booking.Id // our booking
                                                                && exists
timeSlotConflict: TimeSlot :: timeSlotConflict in this.TimeSlots // the timeSlot of our booking

    && ((timeSlotConflict.StartTime > timeSlot.StartTime &&
timeSlotConflict.StartTime<timeSlot.EndTime)

    || (timeSlotConflict.EndTime > timeSlot.StartTime &&
timeSlotConflict.EndTime<timeSlot.EndTime)) // time range intersects
     }

    method BookTimeSlot(visitorId: int, trainerId: int, timeSlotId: int)
        returns(booking: Booking?)
        modifies this
     {
        var dataExists := userInRoleExists(visitorId, Visitor)
            && userInRoleExists(trainerId, Trainer)
            && timeSlotExists(timeSlotId)
            && isSlotVacant(timeSlotId);

    var isConflict := conflictsExist(visitorId, timeSlotId);

    booking := new Booking(0, visitorId, trainerId, timeSlotId, isConflict);

        this.Bookings := this.Bookings + { booking };
     }
    }
```

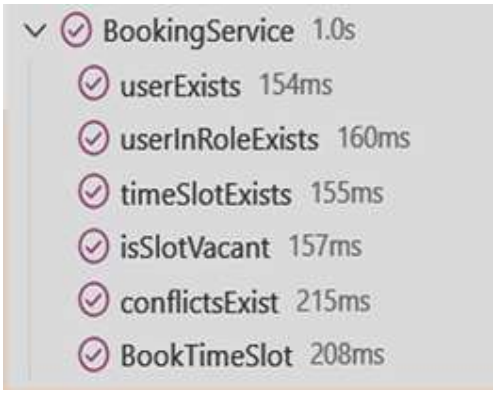


Figure 13. Successful verification

DEVELOPMENT PROCESS

The selection of a development methodology was a crucial decision, so we held a meeting to evaluate our options. After careful deliberation, we collectively agreed to adopt Scrumban as our chosen approach. We believed that Scrumban would provide us with the flexibility of Kanban and the structure of Scrum, enabling us to effectively manage our project's evolving requirements.

Scrum vs Kanban vs Scrumban

	Scrum	Kanban	Scrumban
Time base	1-4 weeks sprints	No time base - Kanban is event-driven	1-year, 6-months and 3-months buckets
Rules	Complete constrained process	few constraints mostly flexible process	Slightly restricted process
Roles	Product owner, Scrum master, scrum team and stakeholders	No specific roles required	No specific roles required
Event-Based	No - Once started sprints cannot be modified	Yes - On going work can react to the workflow	Yes - On going work can react to the workflow and cause On-Demand Planning
Board	Defined/resets each sprint	Persistent - the Kanban board	Persistent - the Scrumban board
Prioritization	Through backlog	Optional	Recommended on each planning
Work routines	The product owner manages tasks and assigns them to team members	Team members choose and pull tasks	The project manager push tasks in the To-Do column and team members choose and pull from there
Scope limits	Sprint limits the work amount	Work in progress limits current on going work amount	Work in progress limits and optional To-Do limit
Task size	What can be delivered in a single sprint	Any size	Any size
New items in an iteration	Not allowed	Allowed whenever the queue allows it (WIP limits)	Allowed whenever the queue allows it (To-Do & WIP limits)
Meetings	Sprint planning, daily stand-ups, sprint reviews and retrospectives	Avoidable	On-Demand Planning
Estimation	Has to be done before sprint has started	Optional	Optional
Planning routines	Sprint planning	Release/iteration planning, demand planning	Planning on demand for new tasks
Performance metrics	Burndown	Cumulative flow diagram, lead time cycle time	Average cycle time
Performance feedback	Sprint retrospective	Optional	Improvement events are an option

Figure 14. Scrum vs Kanban

Utilization of Scrumban allowed us to focus on developing and testing the application and not being bound by strict Scrum requirements while designing and implementing the system.

The team agreed on the need of conducting strictly scheduled meetings **three times a week**. These 15-minutes-long meetings were meant to perform status updates, let everyone share their problems and plans. In addition, it was a common practice to plan additional meetings with some of the team members in order to conduct planning sessions, working on bugs, writing code and sharing knowledge.

13.02.2023
Обравши голосуванням тематику проєкту, зібралися для планування майбутнього додатку. Спланували структуру сутностей додатку, ролі користувачів на передбачений функціонал.

24.02.2023
Задачі, що перейшли в Code Review: Розробити структуру з підсистеми адміністратора
Через виникнення проблеми з локальним запуском клієнта була додана задача Налаштувати SPA App в ASP.NET Core
Задачі, що перейшли в Doing: Дослідити конкурентів, Налаштувати SPA App в ASP.NET Core
Задачі, що перейшли в Done: Створити базову структуру проєктів, Розробити базову специфікацію додатку, Створити git репозиторій, Ознайомитися з Figma
Підняли питання про вибір бібліотеки компонентів для фронтенда

27.02.2023
Додані задачі Робота з документацією та Сформулювати очікування від проєкту
По задачі Налаштувати SPA App в ASP.NET Core є прогрес, але поки що залишається в Doing
Створені та сконфігуровані доменні сутності відповідно до діаграми. Додані класи сутностей та GymContext, Fluent Api конфігурація сутностей, оновлена локальна БД.
Реалізовані операції CRUD для сутності Activity, створений контролер ActivitiesController
Відповідні задачі перейшли в Code Review
Досліджені конкуренти, відповідна задача перейшла в Done
– – – –
За результатами перших зроблених задач по фронтенду і бекенду, Василь і Михайло провели короткий knowledge transfer з колегами

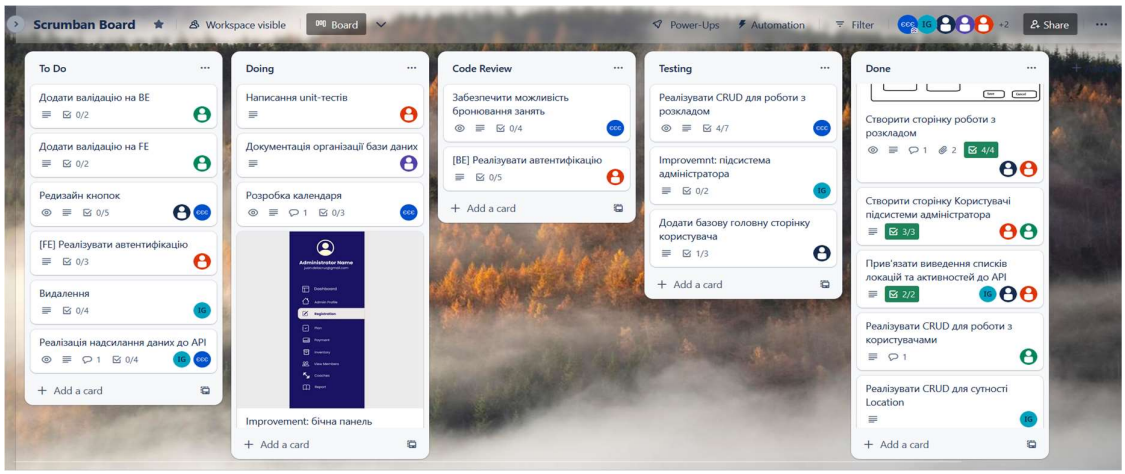


Figure 15. Scrumban board

Each task added to the backlog (To Do column) contained requirements description and a verification checklist for the testing engineer.

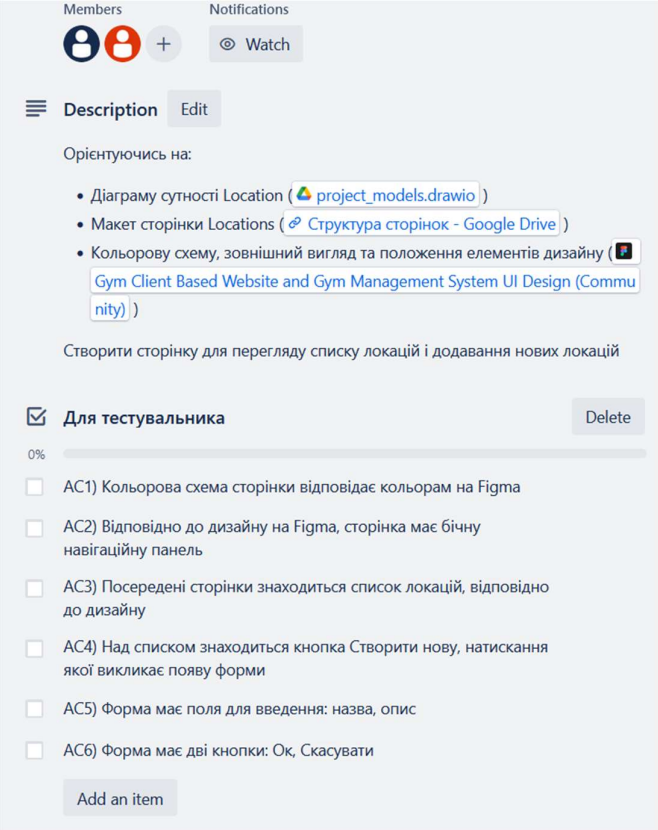


Figure 16. Adding the task

ONLINE RESTAURANT RESERVATION SERVICE "RESERVEAT"

Sofiia Bilinska, Tamara Volcharenko, Oleksandra Ordak, Denys Rudenko, Mykhailo Semitkin, Bohdan Stukalo, Oleksandra Timofieieva

INTRODUCTION

In today's fast-paced world, people are increasingly looking for convenient and fast ways to solve their needs.

One of these ways is to make restaurant reservations online. This allows you to secure a seat at the table without any hassle or waiting. Thanks to online services such as restaurant reservations, people can find and book a table at any time, wherever they are.

The development of an online restaurant reservation service is an important and **relevant** step in the improvement of the gastronomy industry. Such reservation services are useful and convenient for many customers, allowing them to get information about menus, prices and promotions, which helps them make an informed choice and enjoy a pleasant dinner.

In this context, the development of an online restaurant reservation service is an interesting and promising area of development in the gastronomy industry. It allows to ensure convenience and accessibility for customers, efficient use of restaurant resources.

The aim of this work is to develop a web application "ReservEat" for booking tables in restaurants. Achieving this goal requires solving the following tasks:

- Familiarization with existing reservation systems and their analysis;
- Familiarization with the methods of program specification;
- Deepening practical teamwork skills;
- Develop technical specifications for the software product;
- Develop a convenient and beautiful design;
- Development of the backend, frontend and testing of the online booking service "ReservEat".

REVIEW OF SYSTEMS AVAILABLE ON THE MARKET

One of the first steps in our work was a detailed market analysis. Table 1 provides a brief description of the existing analogues we focused on, with their advantages and disadvantages.

Table 1. Market analysis

Application name	Advantages	Disadvantages
OpenTable	User-friendly interface, worldwide table reservations, a large number of filters, mobile application	Charges the restaurant for each reservation through the app
TheFork	User-friendly interface, mobile app, discounts for regular users who book through the app, a large number of filters	Has worse functionality for a restaurant than OpenTable, not available in all countries
DinnerBooking	Convenient interface, integration with social networks, which allows you to share reservations among friends	Only available in Denmark, limited search filters
RestOn	Offers discounts and loyalty programs for regular users, mobile application	Available only in Kyiv, limited search filters, not a user-friendly and beautiful interface

OVERVIEW OF USED TECHNOLOGIES

Java 17 - is an object-oriented programming language. It is known as a high-level language because its code is easy for programmers to read and write. This language was chosen as the most popular programming language, and also because it is the language of choice for most of our backend developers.

Kotlin - statically-typed programming language that runs on the Java Virtual Machine. Kotlin is designed to be concise, safe, and interoperable with Java.

Spring Boot 3.0 - is an open-source framework for building standalone, production-grade applications that are based on the Spring framework. Spring Boot provides a streamlined way to develop and deploy Spring-based applications by providing a pre-configured set of dependencies and a simplified configuration model. This means that developers can quickly create and deploy Spring-based applications without having to spend a lot of time on manual configuration. We use this framework due to its ease of use and powerful features.

Gradle - is an open-source build automation tool that is used primarily for building Java and JVM-based projects. Gradle is a powerful and flexible build automation tool that can help developers streamline their build process and improve productivity.

Docker - is an open-source containerization platform that enables developers to create, deploy, and run applications in containers. Containers are lightweight, standalone executables that include everything needed to run an application, including code, libraries, and dependencies. Docker allows developers to package their applications into containers and deploy them anywhere, whether it's on a local machine, in a data center, or in the cloud.

Swagger - is an open-source framework for designing, building, documenting, and consuming RESTful APIs (Application Programming Interfaces). Swagger provides tools and specifications that allow developers to define API endpoints, parameters, request and response payloads, authentication methods, and other details in a standardized way using YAML or JSON. We have chosen this OpenApi due to its ease of use, extensibility, and support for a wide range of programming languages and frameworks.

PostgreSQL - is a popular open-source relational database management system (RDBMS) that is known for its robustness, reliability, and scalability. PostgreSQL is released under the PostgreSQL License, which is a permissive open-source license that allows for free use, modification, and distribution of the software, that is why we have chosen it.

React - is a JavaScript library used for building user interfaces, particularly for web applications. React follows a declarative approach, where developers define what should be rendered based on the current state of the application, and React takes care of efficiently updating the user interface when the state changes. We have chosen it, because it is widely used by front-end developers.

HTML – is a standard markup language used for creating web pages and web applications.

CSS - is a styling language used in web development to control the layout, appearance, and design of web pages.

Figma - is a cloud-based design tool used for user interface (UI) and user experience (UX) design, prototyping, and collaboration. It allows designers and design teams to create and share interactive design mockups, prototypes, and design systems.

Jira - is a popular project management and issue tracking tool used by software development teams to plan, track, and manage tasks, projects, and software development workflows. It provides a collaborative platform for teams to work together, organize and prioritize tasks, track progress, and communicate about work items in a centralized location. We have chosen it because of a wide range of features.

PURPOSE AND GOALS OF THE SYSTEM DEVELOPMENT

The purpose of the "ReservEat" system is to automate the process of reservation tables in the restaurants, which gives customers to conveniently book tables without wasting time, and restaurants to track the number of visitors. The work includes:

- design and software implementation of the "ReservEat" system
- verification and testing "ReservEat" system
- analysis of existing software tools in the industry of online reservations

The **target audience** of this system is people looking for a convenient and fast way to book a table in a restaurant or other catering establishment, as well as interact with it during booking and service. Among the potential ReservEat users include:

- Customers who want to book a table in a restaurant for a specific time and date, in particular, for special events such as birthdays, anniversaries, various events, etc.
- Tourists looking for places to eat in new places and cities where they are unfamiliar with local restaurants and catering establishments.
- Business people looking for places to hold business meetings and dinners.
- Couples looking for cozy and romantic places for lunch and dinner.
- People who want to learn more about restaurants and catering establishments restaurants and catering establishments, their menus, prices.
- Managers of restaurants and other catering establishments who want to use ReservEat to control the flow of visitors and make the restaurant convenient for booking;

Goals of the system.

"ReservEat" system is created in order to:

- Restaurant search - users can search for restaurants by various parameters, such as location, cuisine, rating, prices, etc.
- Table reservations - users can book tables at restaurants through ReservEat, choosing the time and number of people.
- Review restaurants - users can view information about restaurants, including menus, prices, photos, and reviews
- Menus - users can view restaurant menus and other details about the food, such as dietary restrictions, allergens, etc

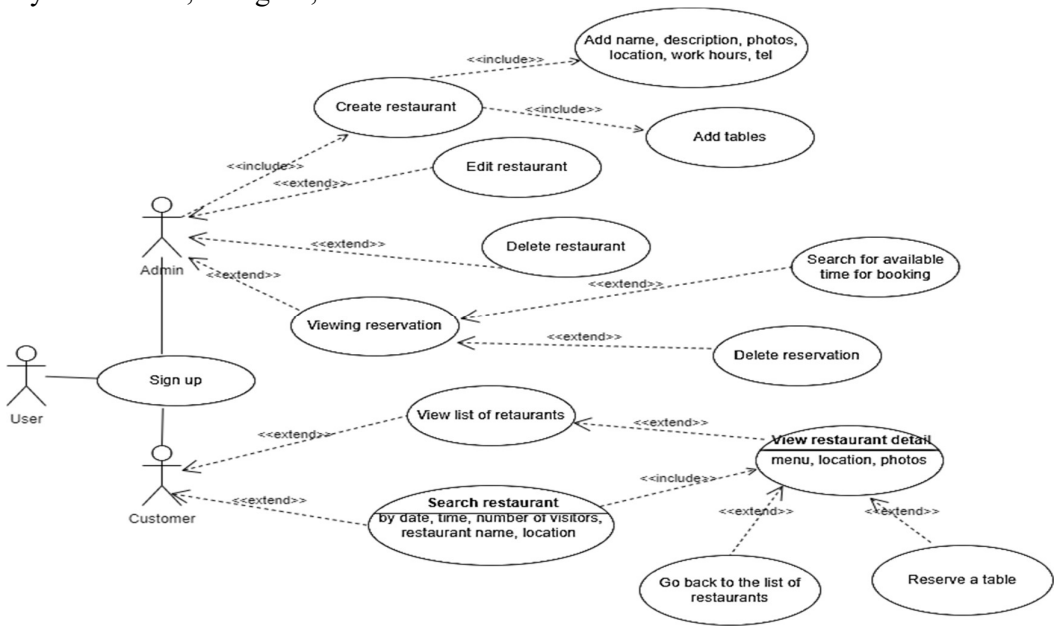


Figure 1. Use-Case Diagram

SYSTEM REQUIREMENTS

General requirements for the system.

Requirements to the structure and functioning of the system. The software product shall have an architecture based on modern technologies for storing, processing, analyzing and accessing data; provide simultaneous work of several users. Functionally, the development should be a web application and unified in terms of software and hardware platform. The development should provide a unified and comfortable, as simple and intuitive user interface.

The following functional subsystems are expected to be allocated in the System:

- administrative, assigned to create restaurants and add information about it;
- user subsystem, assigned to simple customers, who want search restaurant and book table in it;

Requirements for functions performed by the system.

Administrator subsystem.

Table 1. List of functions and tasks to be automated

Function	**Task**
Restaurant management	Add new restaurant
	Edit restaurant data
	Delete restaurant
	Add new photos of the restaurant
	Add new locations for restaurant
Reservation management	Viewing reservation
	Search for available time for booking
	Delete reservation
	Add new tables in restaurant

User subsystem

Table 2. List of functions and tasks to be automated

Function	**Task**
Search for table in the restaurant	Look for list of the restaurants
	Search restaurant by name
	Search table in the restaurant by number of visitors
	Search a table by suitable hours
	Look for restaurants details
Reservation	Make a reservation for a table in a specified location

System requirements.

The software product must ensure the correct display of data in the following browsers of current versions:

- Google Chrome;
- Internet Explorer;
- Opera;
- Mozilla Firefox;
- Safari.

Any arbitrary incorrect user actions related to entering incorrect data, not filling in mandatory input fields in forms and others that can be processed by the by the system, appropriate error messages are generated in Ukrainian, within the the overall design of the website.

Table 3. System hardware requirements

Software	**Version**
Operating system	Windows
Programming language of web application	Java, Kotlin, JavaScript, HTML, CSS
Database management system	PostgreSQL

Requirements for graphic design of ReservEat.

1. Simplicity and clarity - the main page and other pages of ReservEat should be simple and clear for users. The main interface should be easy to read, and colors, backgrounds and fonts should be easy on the eyes.
2. Consistency - all ReservEat pages should use the same colors, fonts, and other design elements to ensure consistency and user experience.
3. Ease of navigation - menus and other navigation elements should be easily and accessible to users. It is also important to provide simple and intuitive navigation between pages.
4. Brand alignment - ReservEat's design should reflect its brand by using colors, fonts, and other elements that reflect the values and features of the platform.
5. Use of images - images of restaurants and food should be of high quality and appetizing to grab the attention of users and and show them what they can expect from visiting a restaurant.

Figure 2. ReservEat Logo

DESCRIPTION OF DATABASE

Logical structure of the database. Used database management system is PostgreSQL.

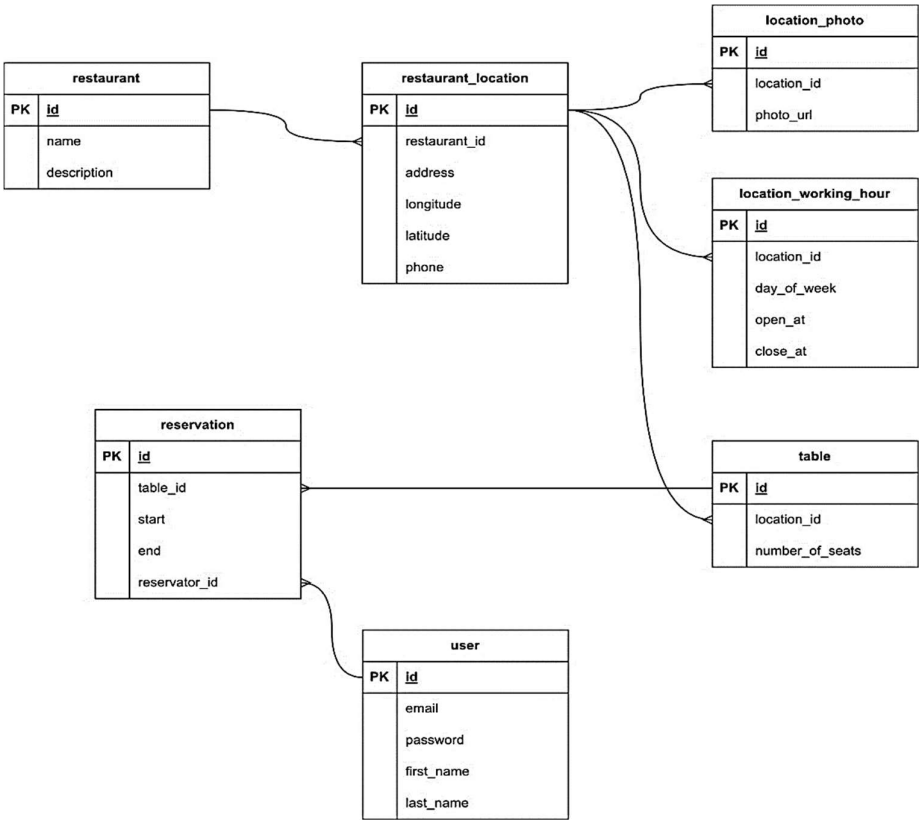


Figure 3. Database Diagram

Table 4. DB tables

Number	Table	Description
1	user	Table for storing information about the user
2	reservation	Table for storing information about reservations
3	restaurant	Table store data of the restaurant
4	restaurant_location	Table store address of the restaurant
5	location_photo	Table store all photos of the restaurant
6	table	Store information for each table in the restaurant
7	location_working_hours	Table store information about restaurants working hours

Description of tables.

Table 5. Table User

Attribute	Type	Description
id	Integer	Identifier
email	Varchar	Email
password	Varchar	User password
first_name	Varchar	User first name
fast_name	Varchar	User last name

Table 6.Reservation Table

Attribute	Type	Description
id	Integer	Identifier
table_id	Integer	Table identifier
start	Timestamp	Start of reservation
end	Timestamp	End of reservation
reservator_id	Integer	Identifier of user, who make reservation

Table 7. Restaurant Table

Attribute	Type	Description
id	Integer	Identifier
name	Varchar	Restaurant name
description	Varchar	Restaurant description

Table 8. Restaurant_location Table

Attribute	Type	Description
id	Integer	Identifier
restaurant_id	Integer	Restaurant identifier
address	Varchar	Restaurant address
longitude	Real	Longitude
latitude	Real	Latitude
phone	Varchar	Restaurant phone

Table 9. Location_photo Table

Attribute	Type	Description
id	Integer	Identifier
location id	Integer	Restaurant location identifier
photo_url	Varchar	Photo url address

Table 10. Table Table

Attribute	Type	Description
id	Integer	Identifier
location_id	Integer	Restaurant location identifier
number_of_seats	Integer	Number of seats for this table

Table 11. Location_working_hours Table

Attribute	Type	Description
id	Integer	Identifier
location_id	Integer	Restaurant location identifier
day_of_week	Varchar	Day of week
open_at	Time	Start time of working
close_at	Time	End time of working

IMPLEMENTATION OF THE SYSTEM

Back-end. For the development of the server side, we used Java and Kotlin language. A three-tier architecture was chosen to implement the system because of its scalability. This architecture includes the following layers:

Web layer, business logic layer, and database layer.

The work began with the design of OpenAPI contracts to enable further simultaneous development of the server and client parts. At the same time, the database scheme was designed.

```yaml
openapi: 3.0.0
info:
  title: Reserveat API
  version: 1.0.0
  description: API for reserving tables in the restaurants using Reserveat
servers:
  - url: http://localhost:8000/api/v1
    description: Local development server
paths:
  /restaurants:
    post:
      tags:
        - Restaurant
      summary: Create a new restaurant
      operationId: createRestaurant
      requestBody:
        description: Restaurant details
        required: true
        content:
          application/json:
            schema:
              $ref: './components/schemas/RestaurantInput.yaml'
      responses:
        '201':
          description: Restaurant created successfully
          content:
            application/json:
              schema:
                $ref: './components/schemas/RestaurantOutput.yaml'
        '400':
          $ref: './components/responses/BadRequest.yaml'
```

Figure 4. DB scheme

The next step in creating a server application is to write database migrations using Flyway and configure the application to work with the PostgreSQL database. To do this, it was necessary to connect the database driver and write the necessary configurations.

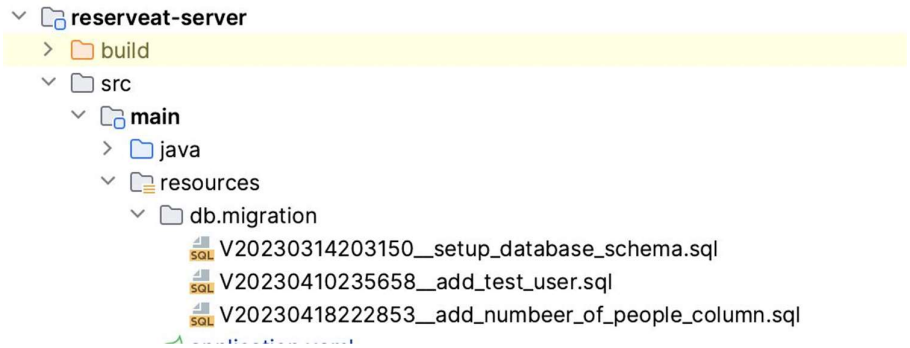


Figure 5. DB migration

The next step was to write the business logic layer and the controller layer, while implementing SQL queries to save and retrieve data from the database.

V20230314203150__setup_database_schema.sql

```sql
create table restaurant
(
    id          serial primary key,
    "name"      varchar not null,
    description varchar
);

create table restaurant_location
(
    id            serial primary key,
    restaurant_id int not null references restaurant (id),
    address       varchar,
    latitude      real,
    longitude     real,
    phone         varchar
);

create table location_photo
(
    id          varchar primary key,
    location_id int     not null references restaurant_location (id),
    photo_url   varchar not null
);
```

Figure 6. SQL queries

Front-end. To develop the client side, we used React,js, Google Maps API, Sass preprocessor and Dom structure.

ReservEat_Front > my-app > src > JS App.js > ...

```
Bohdan Stukalo, 2 weeks ago | 1 author (Bohdan Stukalo)
import './App.scss';       Bohdan Stukalo, 4 weeks ago • Added react app …
import './components/Home/Home';
import { Routes, Route } from 'react-router-dom';
import Home from './components/Home/Home';
import LogIn from './components/LogIn/LogIn';
import Register from './components/Register/Register';
import Restaurants from './components/Restaurants/Restaurants';
import Map from './components/Map/Map';
import Details from './components/Details/Details';
import Reservation from './components/Reservation/Reservation';
import Footer from './components/Footer/Footer'

function App() {
  return (
    <>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/login" element={<LogIn />} />
        <Route path="/register" element={<Register />} />
        <Route path="/details" element={<Details />} />
        <Route path="/restaurants" element={<Restaurants />} />
        <Route path="/reservation" element={<Reservation />} />
      </Routes>
    </>
  );
}

export default App;
```

```

  return (
      <div className="map">
        <div className="google-map">
        <GoogleMapReact
          bootstrapURLKeys={{ key: 'AIzaSyBt20vvUDL_zgFi0W4dIpiVprZ3rifU8lU' }}
          defaultCenter={defaultProps.center}
          defaultZoom={defaultProps.zoom}
        >
          <LocationPin
          lat={lat}
          lng={long}
          text={address}
          />
        </GoogleMapReact>
        </div>
      </div>
  );
}
```

Figure 7. Client side

As for the site's functionality, the user can browse and select restaurants that meet his or her criteria and make a reservation. The site was routed using the React Router DOM library.

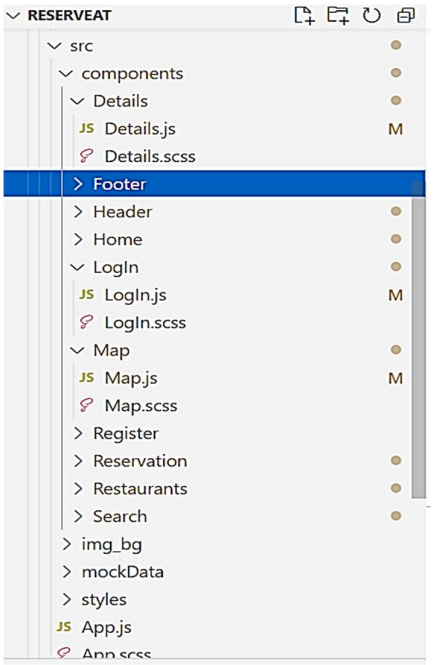


Figure 8. Routing

TESTING

During the development and testing of the system, we used unit testing - this is a software testing method that consists in testing each module of the program code separately of each module of the program code. For each method of the available controllers, we created a separate Unit Test method was created.

```
@RunWith(MockitoJUnitRunner::class)
class PhotoRepositoryTest {

    @Mock
    private lateinit var jdbcTemplate: NamedParameterJdbcTemplate

    private lateinit var photoRepository: PhotoRepository

    @Before
    fun setUp() {
        photoRepository = PhotoRepository(jdbcTemplate)
    }

    @Test
    fun testAddLocationPhoto() {
        // Arrange
        val uuid = "123"
        val locationId = 456
        val url = URL("http://example.com/image.jpg")

        // Act
        val photo = photoRepository.addLocationPhoto(uuid, locationId, url)

        // Assert
        assertEquals(uuid, photo.id)
        assertEquals(url.toString(), photo.url)

        verify(jdbcTemplate).update(anyString(), eq(mapOf(
            "id" to uuid,
            "location_id" to locationId,
            "photo_url" to url.toString()
        )))
    }
```

Figure 9. Unit tests examples

USING OF THE SYSTEM

"ReservEat" it is a web platform, which can help users search for a restaurant and book a table. All users have access to the main page. On this page, the user sets the criteria by which he or she searches for a restaurant and a table in it. We have the following criteria:

- date
- time
- number of visitors
- and you can also search by restaurant name

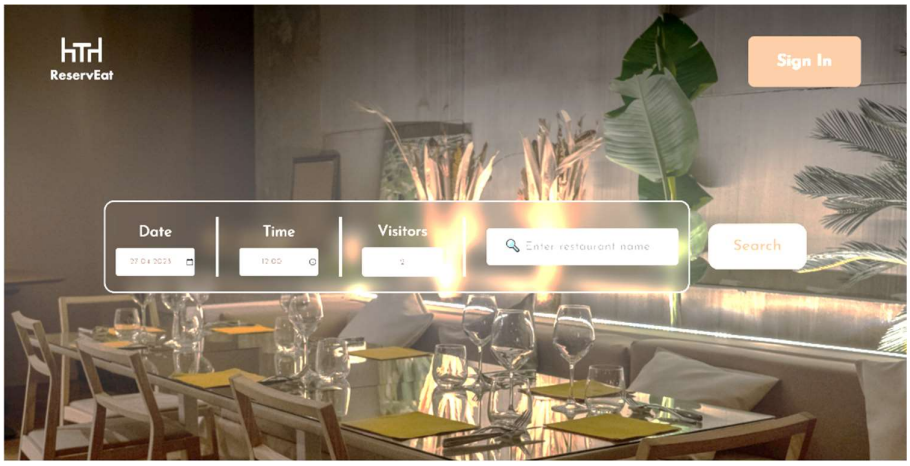


Figure 10. Main page

After user click on the search button, user will see all available restaurants according to the specified filters.

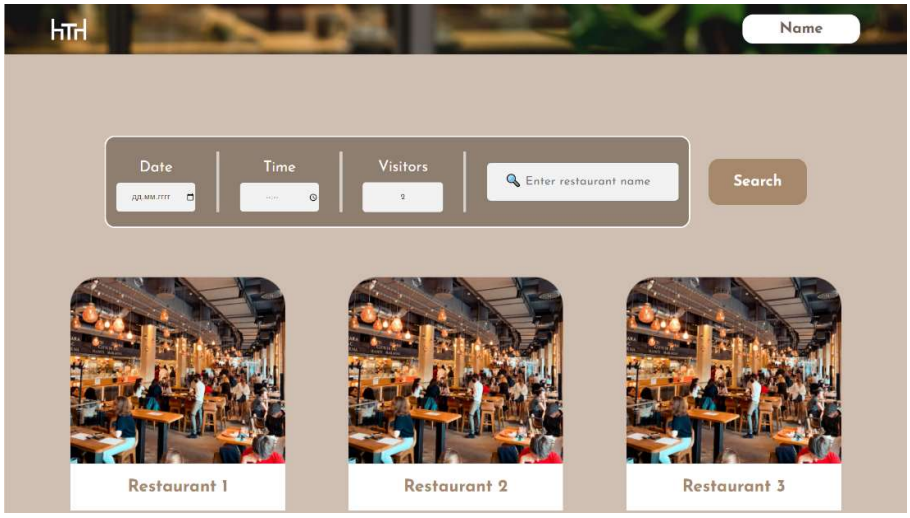


Figure 11. Searching

After this user can view detais about choosen restaurant. In this page user can watch location of the restaurant on the map, photos of the place and working hours and useful information.

Figure 12. Locations

After looking for the restaurant, user can reserve table in this restaurant, he o r she need to click on the button Reserve. After click we have received this formwhich user must to fill.

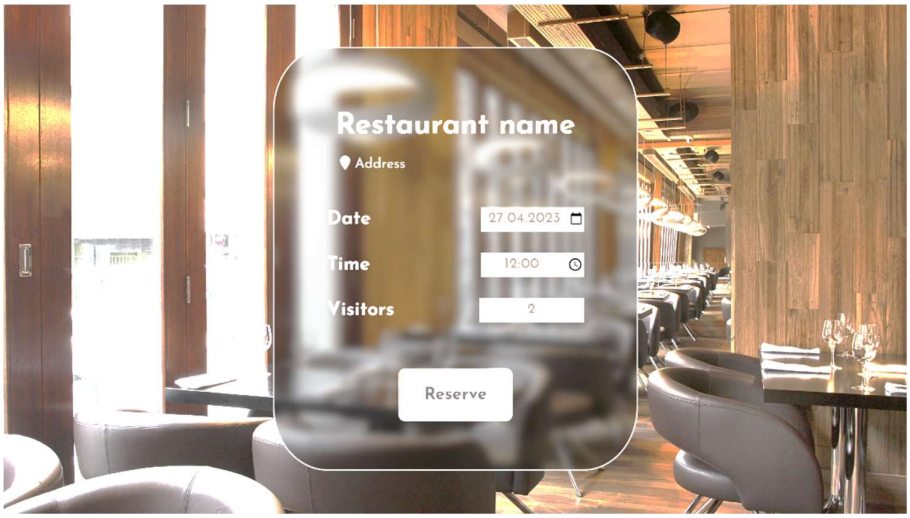


Figure 13. Reserve

FORMAL SPECIFICATION AND VERIFICATION

The specification and verification were implemented in Dafny for the table reservation logic. For this purpose, we developed the Table class, the Reservation class, and the MakeReservation class.

The isReservationPossible class contains a set of tables, and its method checks whether a table is available for booking, depending on the number of people entered for the reservation.

```
class Table {
  var maxSeats: int;
  var numOfTables: int;

  constructor(seats: int, num: int)
    ensures maxSeats == seats && numOfTables == num
  {
    maxSeats := seats;
    numOfTables := num;
  }
}
```

```
class MakeReservation{
  var tables : seq<Table>

  constructor(t: seq<Table>)
    ensures tables == t
  {
    tables := t;
  }

  method isReservationPossible (seats: int) returns (res:bool)
  requires |tables| > 0
  requires forall i :: 0 <= i < |tables| ==> tables[i] != null
  ensures res == true || res == false
  ensures res ==> exists i :: 0 <= i < |tables| && tables[i].maxSeats >= seats && tables[i].numOfTables > 0
  ensures res ==> if exists i :: 0 <= i < |tables| && tables[i].maxSeats >= seats && tables[i].numOfTables > 0 then true else false
  {
    var r := false;
    for i := 0 to |tables|
    invariant 0 <= i <= |tables|
    invariant !r ==> forall j :: 0 <= j < i ==> seats > tables[j].maxSeats || tables[j].numOfTables == 0
    {
      if (seats <= tables[i].maxSeats && tables[i].numOfTables != 0){
        break;
        r := true;
      }
    }
    res := r;
  }
}
```

Figure 14. Examples of requirements formal specification

The Reservation class contains methods that check whether a table can be reserved for an already booked table. The user cannot make a reservation for already pre-booked hours, and the number of guests must not exceed the number of allowed seats.

```
class Reservation {
  var capacity: int;
  var bookingStart: int;
  var bookingEnd: int;

  constructor(c: int, s: int, e: int)
    ensures capacity == c && bookingStart == s && bookingEnd == e
  {
    capacity := c;
    bookingStart := s;
    bookingEnd := e;
  }

  method checkTime(start: int, end: int) returns (r:bool)
    requires start <= end
    ensures r == (end <= bookingStart || start >= bookingEnd)
  {
    r := end <= bookingStart || start >= bookingEnd;
  }

  method isTableAvailable(numGuests: int) returns (r:bool)
  ensures r == (numGuests <= capacity)
  {
    r := numGuests <= capacity;
  }

  method canSeat(numGuests: int, start: int, end: int) returns (result:bool)
  requires start <= end
  requires start >= 0 && start < 24
  requires end >= 0 && end < 24
  ensures result == ((end <= bookingStart || start >= bookingEnd) && (numGuests <= capacity));
  {
    var available := isTableAvailable(numGuests);
    var time := checkTime(start, end);
    result := available && time;
  }
}
```

Figure 15. Reservation specification

DESCRIPTION OF THE DEVELOPMENT PROCESS

To control and manage the System development process, the following methodology was chosen Kanban methodology, as a program system for control we have chosen Trello. But after a while, we realised that this methodology did not comfortable for us and changed it to Scrum methodology and changed system for Jira, because it has a wider functionality. It is example of one of the sprints.

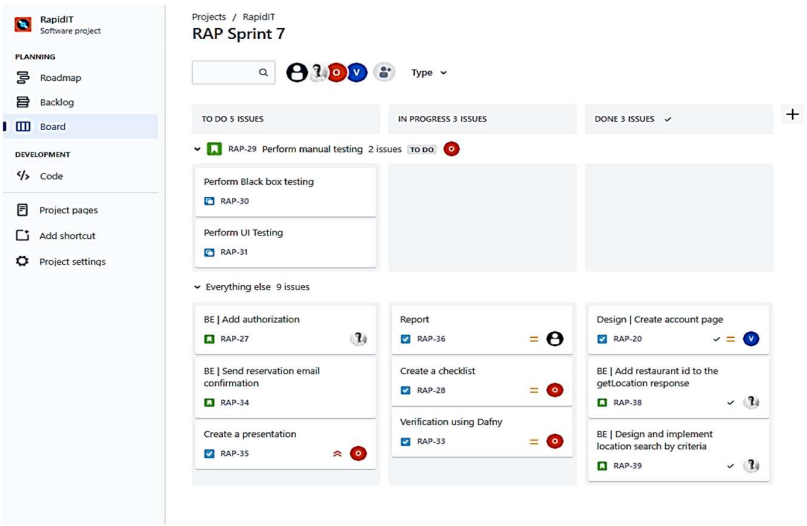


Figure 16. Kanban table

The main idea of this methodology is broken down project development into a series of smaller, manageable tasks. These tasks are then prioritized and completed over a series of short

iterations, called sprints, typically lasting 1-4 weeks. Our sprint lasted 1 week and we have meetings two times a week, additionally, we create telegram chat for everyday communication.

For convenient development for all members of team and version control we create GitHub repository, where we store our code and made code reviews. It is example one of code reviews, that we have done.

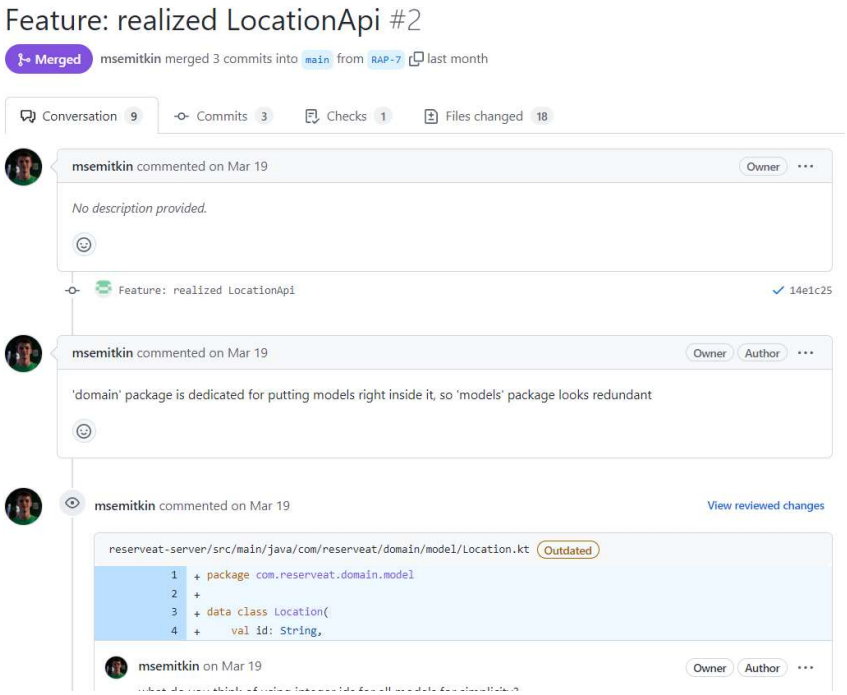


Figure 17. Frgment of code on the GitHub

CONCLUSION

As a result of this training project, we have primarily gained or deepened our teamwork skills and experience. We gained a lot of new knowledge in the field of management and IT project management. We developed our own system for table reservation “ReservEat”.

REFERENCES

1. Jira. URL: https://www.atlassian.com/software/jira
2. Trello. URL: https://trello.com/uk
3. PostgreSQL. URL: https://www.postgresql.org/
4. Java. URL: https://www.java.com/
5. GitHub. URL: https://github.com/
6. React. URL: https://uk.reactjs.org/

AREA MONITORING SYSTEM "GREENZONE"

Serhii Syrota, Serhii Peshko, Mykhailo Bubka, Mikhail Syvachenko, Eduard Andrashchuk, Maksym Rasakhatskyi, Dmytro Ruban

INRODUCTION

The need for a strong, adaptable, and sustainable software solutions has grown due to the quickly changing world of technology. Enterprises across various sectors must remain at the forefront by embracing advanced technologies and effective development techniques. This endeavor emphasizes the significance of utilizing multiple technologies, such as microservices, Go, C#, React, AWS, Docker, and others, to address the demands of modern software creation.

A microservices architecture is essential for handling intricate applications as it facilitates the autonomous deployment and scaling of individual components. The subject's importance originates from demonstrating the benefits of a microservices structure and underlining the need for expertise in various languages and frameworks to develop versatile applications.

To comprehend the finest practices in software creation and the benefits of particular technologies, this endeavor utilized a range of research techniques, including literature reviews, case studies, and hands-on experimentations.

The main objectives of this endeavor were as follows:

1. Create a scalable and efficient software solution.
2. Enhance comprehension of current technologies.
3. Investigate the challenges and advantages of incorporating these technologies.
4. Illustrate the application of the best practices in software creation.

To accomplish these objectives, the endeavor conducted a literature assessment, examined successful case studies, developed and executed a microservice-based application, tested and validated its functionality, scalability, and maintainability, and documented and developed process, challenges faced, and lessons learned.

This endeavor aimed to achieve practical outcomes and meet professional objectives, such as consolidating knowledge, familiarizing oneself with new technologies, and deepening the comprehension of the tools and methodologies used in contemporary software creation. The insights obtained from this endeavor will improve the professional skills of the development team and benefit the wider software development community.

COMPARATIVE OVERVIEW OF MARKER SOLUTIONS

It is essential to overview existing systems on the market that work with geodata before going through our system's details. It will help us to understand each system's advantages and disadvantages and see our solution's unicity. Table 1 provides an overview of two systems: Google Maps and OpenStreetMap. We will review their features, pros and cons.

Table 1. Market solutions overview

Name	Advantages	Disadvantages
Google Maps [1]	Reliability and accuracy of the geodata. Integration with other Google services such as Google Street View and Google Earth. Wide range of functions, including navigation and route search. API support for creating own applications.	Service depends on the Internet connection. Service is collecting data about users and their locations. Need for specialized functionality for polluted and mine-contaminated areas. Some limitations to free usage.
OpenStreetMap [2]	Open-source and free. Community support facilitates constant updating of geodata. Ability to work offline. API support for creating own applications.	Lower accuracy of the geodata compared to Google Maps. Need for specialized functionality for polluted and mine-contaminated areas. Less flexible.

Advantages of our system:

1. Specialized functionality to reflect polluted and mine-contaminated areas.
2. Autonomy: the ability to work offline in areas without or with a weak Internet connection.
3. Integration with MS Excel: import and export of the data for comfortable analysis and processing of the data.
4. Object visualization on the map: each category of objects automatically determines the icon on the map, making navigation and working with the data much more straightforward.
5. Filtration: our system allows users to filter objects by various parameters, facilitating data analysis.
6. Automatic transform between coordinate types: our system allows users to introduce only one type of coordinates (latitude and longitude, sk_42, mgrs), which then transforms automatically to other types.
7. Automatic searching for the closest settlement: if user does not specify region, district, or city, the system automatically determines it based on provided coordinates. It provides accuracy and saves time when working with geodata.
8. Complex validation: our project ensures data reliability by complex and custom validation of each field, which provides accuracy and completeness of the information.
9. Summary comment: our system automatically generates summary comments for the objects, providing qualitative and concise information about things on the map.

Therefore, our system has a couple of advantages compared to other services such as Google Maps and OpenStreetMap. The main asset is that our service has functionality for working with polluted and mine-contaminated areas. This makes our application more attractive to users who operate in such areas than other solutions on the market.

OVERVIEW OF USED TECHNOLOGIES

Our team picked a tools based on their demonstrated ability for growth, dependability, and ease of use.

We adopted a microservices design [3] that splits a comprehensive application into smaller, self-sufficient services, enabling individual development, testing, and deployment for each service. This method streamlines problem identification and resolution, as well as component scaling, while encouraging collaboration among team members who can focus on specific services.

In addition, we implemented a component-based method [4], which is rooted in a microservices architecture. This approach encourages modular design by diving a service into reusable elements with well-defined interfaces that can be effortlessly integrated into various application segments. This method increased design flexibility and enhanced the effectiveness of code maintenance and reusability. Moreover, the component-based method provides a transparent division of tasks, allowing developers to work on separate components without impacting the system's overall functionality. This speeds up development and simplifies component updates without affecting the entire system.

We used Docker [5], a containerization platform, to ensure consistent and repeatable deployment across different environments. Docker creates lightweight containers for each microservice, encapsulating all required dependencies and configurations. This reduces environment-specific problems, lowers resource usage, and speeds up deployment.

We selected Amazon Web Services (AWS) [6] as our cloud service provider due to its extensive variety of offerings and worldwide infrastructure. AWS enabled us to rapidly deploy and oversee our microservices while utilizing robust storage, security, and monitoring solutions. Furthermore, AWS's pay-as-you-go pricing structure allowed us to efficiently manage expenses based on actual consumption.

For building backend services, we utilized Go [7] and .Net [8]. The Go programming language was preferred for its straightforwardness, solid concurrency capabilities, and swift execution times, enabling the team to construct high-performing, scalable backend services. In contrast, the .Net framework provided a seasoned, well-supported platform with an all-inclusive library, potent tools, and community backing, facilitating the development of dependable, maintainable, and secure services that integrated flawlessly with other system elements.

Regarding frontend development, we employed React [9], a popular JavaScript library that streamlines the development of reusable UI components and simplifies application state handling. Moreover, React's performance enhancements, such as virtual DOM, guarantee a smooth user experience.

We leveraged OpenStreetMap (OSM) data as a critical source of geographic information for our project. This data was downloaded and our own tile server was deployed using open-source mapping tools for optimal performance and customization. Hosting our own tile server granted us a complete control over the map rendering process, enabling us to adapt map styles and features to our specific needs. This strategy also reduced reliance on external services, enhancing our solution's overall reliability and stability.

MySQL [10] was our choice for a relational database management system (RDMS) due to its performance, scalability, and ease of use. MySQL provides a robust solution for storing and managing the application's structured data, ensuring consistency and integrity. Additionally, its widespread adoption and community support make it a trustworthy choice.

We utilized Git [11], a commonly used distributed system of version control, to manage our source code and promote collaboration. Git allowed team members to work on different features or bug fixed simultaneously, track changes, and merge them into the main codebase without conflicts. This streamlined the development process and ensured a well-maintained, up-to-date codebase.

For efficient task monitoring and management, we employed YouTrack [12]. This agile project management tool enabled us to create, assign, and prioritize tasks and oversee progress and deadlines.

In conclusion, the team carefully picked a mix of modern, reliable technologies to ensure the project met its objectives while maintaining high standards for performance, maintainability, and user experience. Each technology played an essential role in the project's success, and their combined impact resulted in a robust, efficient, and scalable solution.

PURPOSE AND GOALS OF THE SYSTEM

Purpose of the system.

Our system is created to reflect polluted and mine-contaminated areas and aims to simplify the process of monitoring and managing such objects. The primary purpose of our project is to provide users with a tool for the visualization and analysis of the data related to the problems described above.

The system is appointed for different categories of users, especially ecologists, deminers, representatives of state bodies, and other interested groups of people who work in the area of pollution and mine contamination. This solution can be used for planning and coordinating activities aimed at solving problems related to ecological issues and the safety of people.

The central system's operations include reflection of polluted and mined areas, integration with MS Excel for import and export of the data, support of different coordinate types, and searching for the closest settlement based on provided coordinates.

One of the key advantages of our solution is autonomy which allows working with the map offline. This is especially useful for users who have limited Internet access.

The system generally provides users with a reliable and sufficient tool for visualizing, analyzing, and managing objects to improve the ecological situation and ensure people's safety.

Goals of the system.

Our team defined several key objectives aimed at solving relevant tasks in the area of spatial data management. Key objectives are as follows:

1. Versatility and flexibility: develop a system that can work with different geodata formats and satisfies the requirements of a wide range of users regardless of their activities.
2. User-friendly interface: create a convenient UI that facilitates quick system use without additional learning.
3. Efficiency: develop a service that can process geodata as fast as possible and contains functions that automate this process to save users' time.
4. Reliability and safety: create a system that guarantees users' data protection and provides service stability.
5. Maintenance: provide a permanent ability to update and improve the service in order to adapt to new and modern technologies.

Figure 1 shows Use Case diagram [13] of our system. This UML diagram was created using service *plantuml* and its markup language.

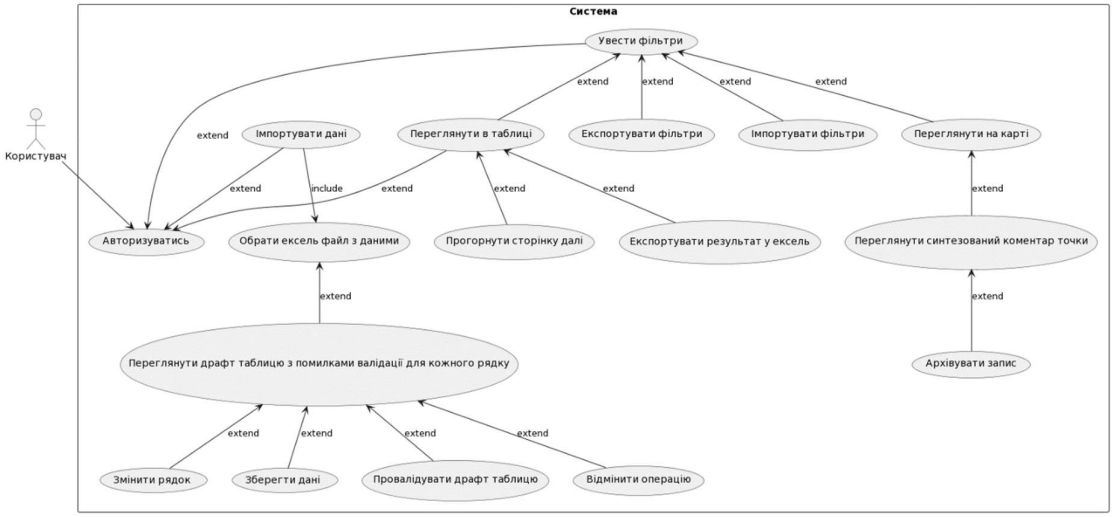


Figure 1. Use Case diagram

System requirements

General system requirements

System "GreenZone" must meet the following general requirements:

1. Data protection: the system must protect users' data from unauthorized access and ensure confidentiality.
2. Scalability: the system must be easy-scalable to work with more data and users.
3. Compatibility: the system must work as expected with different devices and support various operating systems.

There must be two types of users in the system: registered and unregistered users.

Registered users must have access to all the data about polluted and mine-contaminated areas, filters, and other system parameters. They can import and export data as well as visualize it on the map. Registered users can also review detailed information about each object on the map.

Nothing must be accessible to users who is not registered. Authorization is required to gain access to the data and operations. By adding a username and password to the "users" list, the administrator can add new users.

Requirements for the system's functions

Our team determined key functions and tasks that should be automated. This list is provided in the Table 2.

Table 2. List of functions and tasks that should be automated

Functions	**Tasks**
Managing data about polluted and mine-contaminated areas	Data import and export
	Updating and deleting information about objects
	Data validation
Data visualization	Reflect objects on the map
	Reflect objects as a table
	Automatically determine icons based on object category
Filtration	Apply various filters to show objects accordingly to given parameters
Working with different coordinate types	Support of three types of coordinates
	Automatic transformation between different types
Search for the closest settlement	Searching for the closest settlement based on given coordinates
	Working offline
Registration and authorization	User registration
	User authorization with login and password

System requirements

1. Browsers: "GreenZone" must work in the next browsers: Google Chrome (version 58 and above), Mozilla Firefox (version 53 and above), Microsoft Edge (version 79 and above), and Safari (version 10 and above).
2. Errors handling: within the overall design of the website, error messages in Ukrainian must be generated for the user's arbitrary incorrect actions, such as not filling in mandatory input fields on forms or entering invalid data.
3. Operating systems: "GreenZone" should be supported by next operating systems: Windows, macOS, Linux (Ubuntu, Fedora, Debian).
4. MySQL database must be a source of the data.

The primary technical characteristics and supported platforms for working with the "GreenZone" system are outlined in these system requirements.

DATABASE DESCRIPTION

Database logical structure

Our team used MySQL database for the project. Figure 2 shows our database schema.

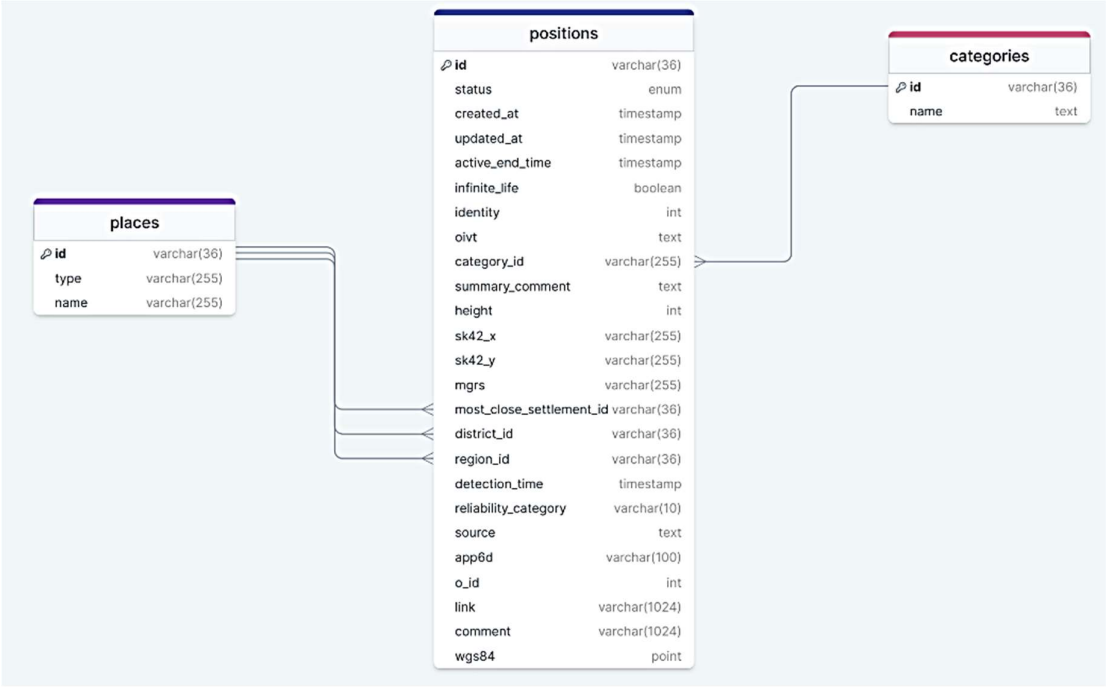


Figure 2. Database schema

Table 3 shows list of tables with description to each table.

Table 3. List of tables with descriptions

Table	**Description**
categories	Category of pollution or mine (Ex. biohazards, water pollution, anti-personnel mine, etc.)
places	Place which can be region, district, city, country, etc.
positions	Contains detailed information about each object (mine or polluted area)

Tables description. Tables 4-6 show detailed description of each table in the database.

Table 4. Table categories

Attribute	**Type**	**Description**
id	varchar(36)	Category unique identifier
name	text	Name of the category

Table 5. Table places

Attribute	**Type**	**Description**
id	varchar(36)	Place unique identifier
type	varchar(255)	Type of the location. For instance, region, district, city, etc.
name	varchar(255)	Name of the location. For instance, city name, country name, etc.

Table 6. Table positions

Attribute	Type	Description
id	varchar(36)	Position unique identifier
status	enum	Status of the object. Can be “ACTIVE”, “ARCHIVE” or “DESTROYED”
created_at	timestamp	The date and time when record was created
updated_at	timestamp	The date and time when record was updated last time
active_end_time	timestamp	Number of days before object gets the status “ARCHIVE”
infinite_life	boolean	If true, the object will be always in status “ACTIVE”
identity	int	Type of the object. For instance, enemy, friend, probably friend, neutral, etc
oivt	text	Technical name of the mines and their amount
category_id	varchar(255)	Identifier of category to which current object belongs
summary_comment	text	Synthesized concise comment about object
height	int	At what height above sea level object is located
wgs84	point	Coordinates of the object in “wgs84” standard
sk42_x	varchar(255)	*x* coordinate of the object in “sk42” standard
sk42_y	varchar(255)	*y* coordinate of the object in “sk42” standard
mgrs	varchar(255)	Coordinates of the object in “mrgs” standard
most_close_settlement_id	varchar(36)	Identifier of the place which is the closest city, town or village to the object
district_id	varchar(36)	Identifier of the place which is a district where the object is located
region_id	varchar(36)	Identifier of the place which is a region where the object is located
detection_time	timestamp	Date and time when the object was detected
reliability_category	varchar(10)	Determines the level of reliability of the source that detected the object
source	text	Person or thing that detected the object
app6d	varchar(100)	Military code of the NATO standard for indicating targets
o_id	int	The number of the record in the database
link	varchar(1024)	Link to the evidence of the object existence. For instance, it can be a photo of the object
comment	varchar(1024)	Additional comment to the object

SYSTEM IMPLEMENTATION

Overall structure. The system is divided into a few components, which are equal to the docker images in our case:

1. Core – backend component, which is the facade to all backend, is also considered as the main logic component.
2. Excel-adapter – backend component, which is the center of Excel manipulation and also provides some geo-mapping logic.
3. Tbsm FE – web frontend of system, responsible for the UI operations.
4. OSM tile server [14] – a third party tool that we leverage for the drawing map UI.
5. Nominatim [15] – a third party tool used for reverse geocoding purposes.
6. GitHub actions runner [16] – a third party, self-hosted CI/CD runner for DevOps purposes.

Core. The main purpose of the service is to glue an excel adapter with main logic and provide a the stable interface for UI. It is written in the Golang language, chosen for its simplicity, security and scalability. For database interconnection, we used the ORM library "gorm" [17]. The team did not have much time to write boilerplate code and manually write and verify SQL queries; hence decided to use ORM. Manually verifying that the application does not have an SQL query leak as "N+1 problem" [18] or other most common problems with ORM usage [19].

Excel adapter. Service responsible for excel marshaling and unmarshalling, written in C#. Also, for reasons, that backend team mostly have expertise in C# service implements coordinates mapping and "smart" reverse geocoding. "smart"– means that it uses custom algorithm wrapper for "Nominatim" service. Sometimes original geocoder cannot find place name by presented coordinates, so excel adapter slightly changes coordinates to apply search in imagine circuit. Algorithm is quite simple:

1. make "search" and "reverse" query to Nominatim (they are quite similar, but not equal, though can both return data, which we need)
2. if desired data is presented, return this data
3. else – change coordinates by shifting them in 8 directions – by direction degree and by the distance(500m) and make 8*2 parallel queries for search, wait for all.
4. if desired data is presented, return it
5. else – increase shifting distance 3 times and go to the third step, loop length is limited to the 6 iterations
6. if 6 iterations passed and data not retrieved, return empty strings

Also, since the reverse geocoding is a bit complex operation, we made caching for the Nominatim responses with 1000 records size limit. Thus, the last performance tuning option was to establish capacity of the Nominatim service. Hardware options was 4 virtual CPU and 8 Gb of RAM. This setup can serve 40 parallel queries with latency lower than 150ms, which matches our purposes. Having a greater number of concurrent queries results in increased latency: 60 – 400ms, 100 – 1s. To leverage concurrency limiting for general performance team used counting semaphore to limit concurrency for 40 queries at the point of time, this was the most hard and the most beautifully solved engineering problem in the team. Code of the reverse geocoder can be found in the ANNEX B.

Tbsm FE. Web-UI of the "Green Zone" is written with React and Redux [20] for the state management. Figure 3 and figure 4 show the UI of the table and the map on the analytics page. Also, component supports simple authorization with "basic auth" approach [21]. For the displaying tiles on map view used an open-source JavaScript library leaflet[22].

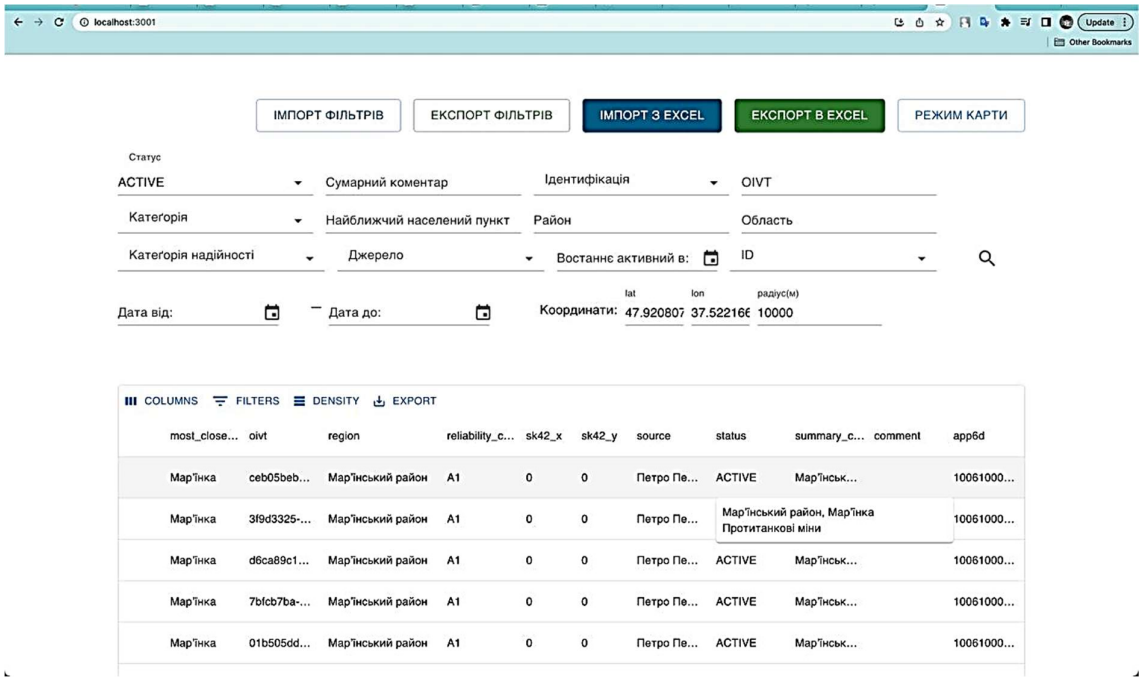


Figure 3. Table UI of the analytics page

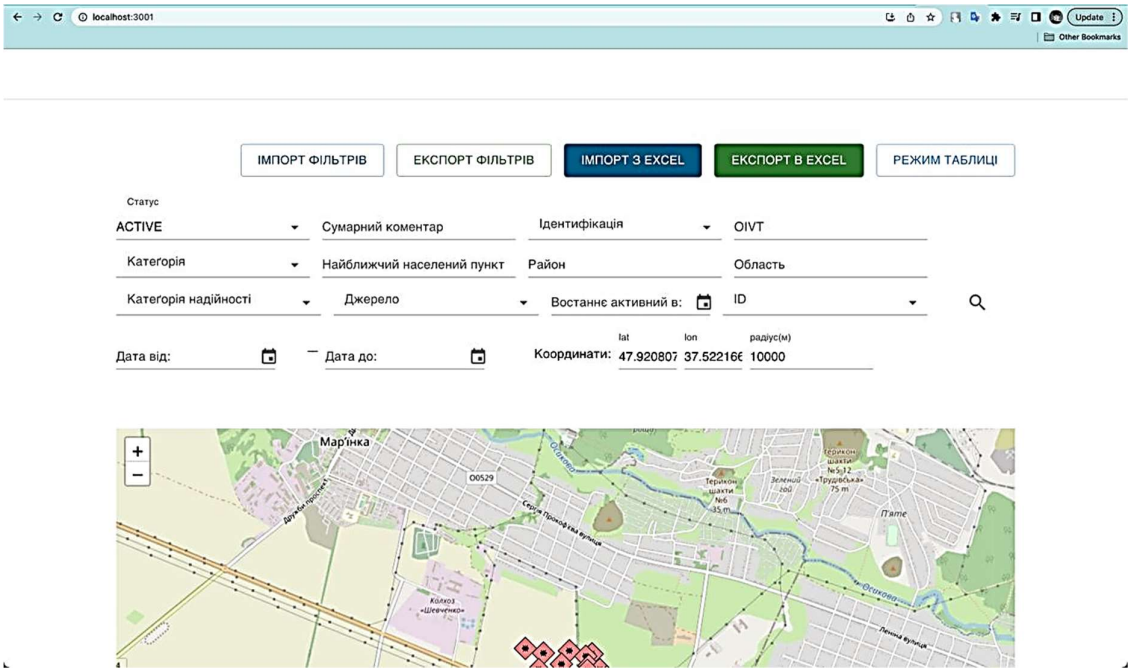


Figure 4. Analytics map view

OSM tile server.

The system must have an independently locally-deployable tile server according to the offline availability requirements. Tile size for Ukraine, Republic of Belarus and parts of russia (~700km from Ukraine boundaries) is near 90GiB disk space, which is a large size but acceptable for our case. For the performance customization it's possible to deploy only the Ukraine map locally, which takes only 20GiB of the disk space. Compressed osm.pbf [23] files

were taken from the “geofabric” resource [24]. Tile server is provisioned by official docker image.

Nominatim. Nominatim uses OpenStreetMap data to find locations on Earth by name and address (geocoding). It can also do the reverse, find an address for any location on the planet [25]. Reverse geocoding – is the process of converting a location as described by geographic coordinates, wgs84 mostly, to a human-readable address or place name, in our case: region, district, and the closest settlement.

TESTING

Direction of the testing.

As of the 5th sprint, the verification situation in the project was as follows:

a) frontend was tested manually;

b) Core part had automated unit tests for the app6d module;

c) ExcelAdapter had automated unit tests and integration tests;

1) unit tests covered the following modules: coordinates transformation, reverse geocoding, and import from excel;

2) integration tests covered integration between import and reverse geocoding.

Overall, the picture resembles upward step-by-step testing with drivers. Currently, the verification of functional requirements in the project is relatively low. The team plans to improve the quality because it is intended to pass the project to the other team for further development after the group project ends.

Small development resources can explain the absence of automated tests in the frontend part. Furthermore, the central part of the logic is implemented on the backend parts. Therefore, the frontend requires only manual testing to verify that UI is displayed correctly and data is correctly passed to the backend.

The situation is more complicated on the backend. We have two services:

1. Core contains most of the business logic and serves as a facade for the entire backend. This service is developed using Golang.

2. ExcelAdapter contains the parsing logic and initial processing of the data imported from excel, and works with spatial data. This service is developed using C#.

Justification of the choice of testing methodology

Our team used the criteria for choosing a testing method suggested by our professors.

1. **Time until the modules assembles.** The project’s code base is relatively small, and team uses languages where time of module assembly counts in seconds; hence this criterion is not essential.

2. **Time to create the first “skeleton” version of the system.** The team considers this criterion and methods allowing iterative testing a priority because the team already has the “skeleton” version of the program.

3. **The need for drivers and other testing tools.** This criterion is essential for us to have the ability to use tools for testing specific branches of the program more accurately.

4. **The need for stubs.** This criterion is important because we used component-based architecture; hence it is essential to have the ability to use stubs for the components.

5. **The degree of parallelism.** This is an unnecessary criterion for us because the speed of our application is enough to run tests sequentially.

6. **Ability to test individual paths.** This criterion is crucial for us to have the ability to use tools to test specific program branches more accurately.

7. **The ability to plan and control the sequence.** This criterion is optional for our team because there will not be long-term planning due to the allocated time for the project.

One of the most important criterion is the possibility of creating functional tests for already existing high-level policies. When viewed from this perspective, the issue necessitates

the use of tools that permit but do not mandate the use of stubs for components whose response is uncertain.

We chose the Sandwich Method [26] for backend testing, taking into account the project's specific circumstances and criteria analysis because it allowed for simultaneous downward and upward testing and early system integration. For a low-complexity project, the point at which the upward tests meet the downward tests is pretty simple to determine:

- unit tests with the appropriate drivers are used to check low-level details like the validation of input data and the mapping of structures;
- functional tests cover high-level module policies that integrate low-level details and make partial use of stubs for third-party components.

As referenced, the frontend improvement aptitude asset is limited, so frontend testing is restricted to manual tests.

Results of testing.

The rules for choosing the usefulness for testing are the criticality of the function, the expected frequency of use, and the potential risks of errors. The following list enumerates things that were thoroughly tested, ranked by importance:

1. Import in ExcelAdapter. This includes initial validation, data parsing, filling in the blanks, and converting data to JSON.
2. Import in Core. This includes integration with ExcelAdapter, filling in the blanks, and final validation.
3. Data filtration in the Core service.
4. Automatic archiving of the positions. This process is based on the time since detection and the expected lifetime of the position.
5. Creating the correct payload of the filtering request to the backend from the frontend components.
6. Export to Excel in the Core service.

Figure 5 shows the results of testing Core service.

```
ok      github.com/knu-tech-stuff/core/src/app6d        (cached)        coverage: 95.7% of statements
ok      github.com/knu-tech-stuff/core/src/oivt (cached)        coverage: 100.0% of statements
ok      github.com/knu-tech-stuff/core/src/routes/excel (cached)        coverage: 69.2% of statements
ok      github.com/knu-tech-stuff/core/src/synth        (cached)        coverage: 100.0% of statements
```

Figure 5. Results of Core tests

Figure 6 shows the results of testing ExcelAdapter service.

```
467
468  Test Run Successful.
469  Total tests: 67
470       Passed: 67
471   Total time: 30.7676 Seconds
```

Figure 6. Results of ExcelAdapter tests

USING OF THE SYSTEM

Authorization. To access the frontend functionality user needs to pass basic authorization as showed on figure 7.

Figure 7. Frontend authorization

The home page is the analytics table page (see figure 8), used to view data in the table and update it if needed. To apply the filters, change desired filters and press the "search" button. To export filters, press the button "Експорт фільтрів" and fill in the file name; after confirming export – the system will download filters in JSON format to the Downloads folder. To import filters, press the button "Імпорт фільтрів" and choose the JSON file with previously exported filters. For export data to Excel, press the button "Експорт в Excel". To change the table view, press the appropriate buttons at the table toolbar, which is placed at the top of the table. To perform the import from Excel, press button "Іморт з Excel".

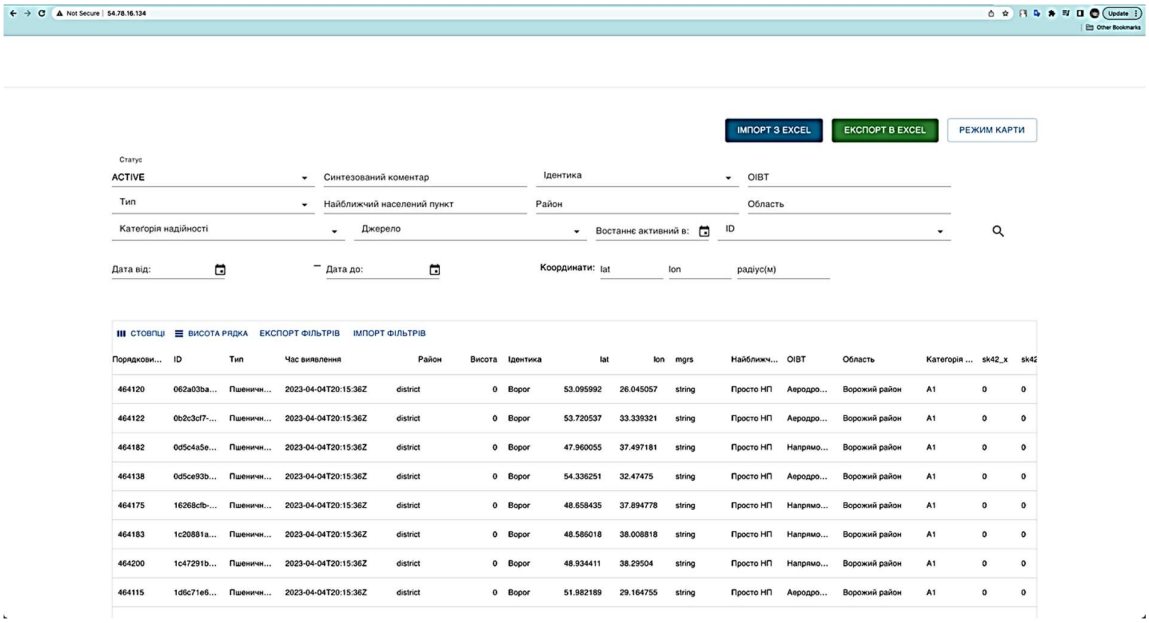


Figure 8. Analytics page, table view

Export to Excel takes all data about filtered positions and maps it to the Excel structure. Example of exported data can be seen on figure 9. It's possible to configure the naming of Excel columns in the ExcelAdapter config file.

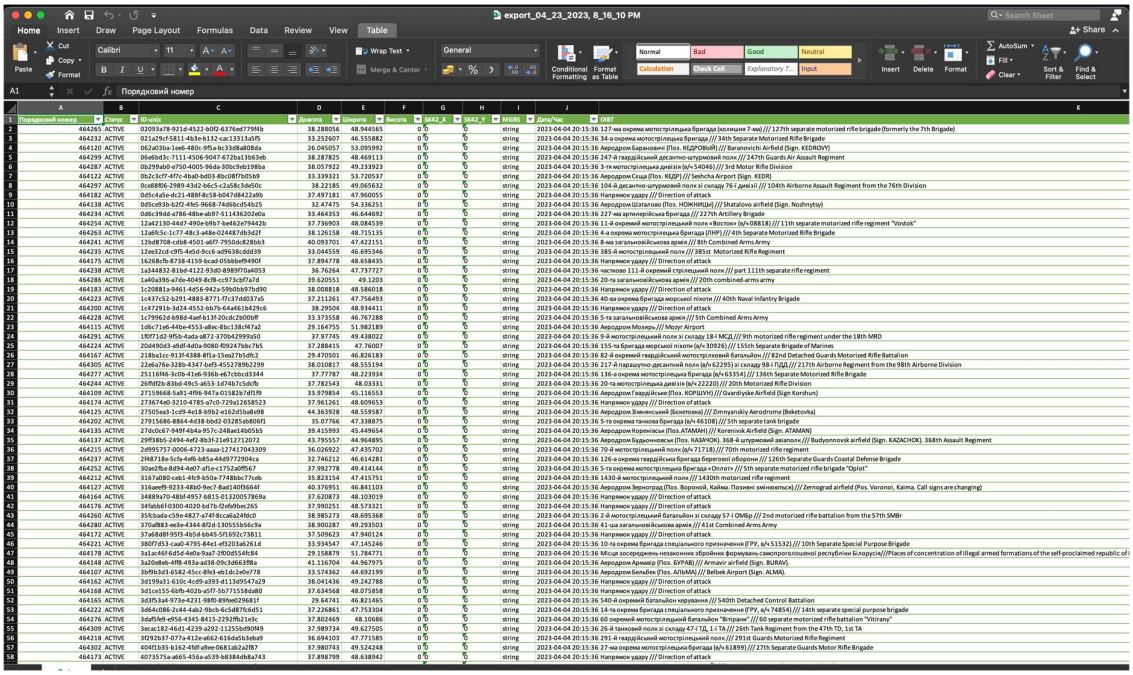


Figure 9. Exported data to excel

To see the interactive map view (see figure 10), press the button “Режим карти”, common recommendation is not to apply map view when filters contain “ARCHIVE” status because there may be thousands of archived positions, and the map may throttle at large volume data, approximately 3k+ positions start to throttle. Also, filters are persistent when changing layouts.

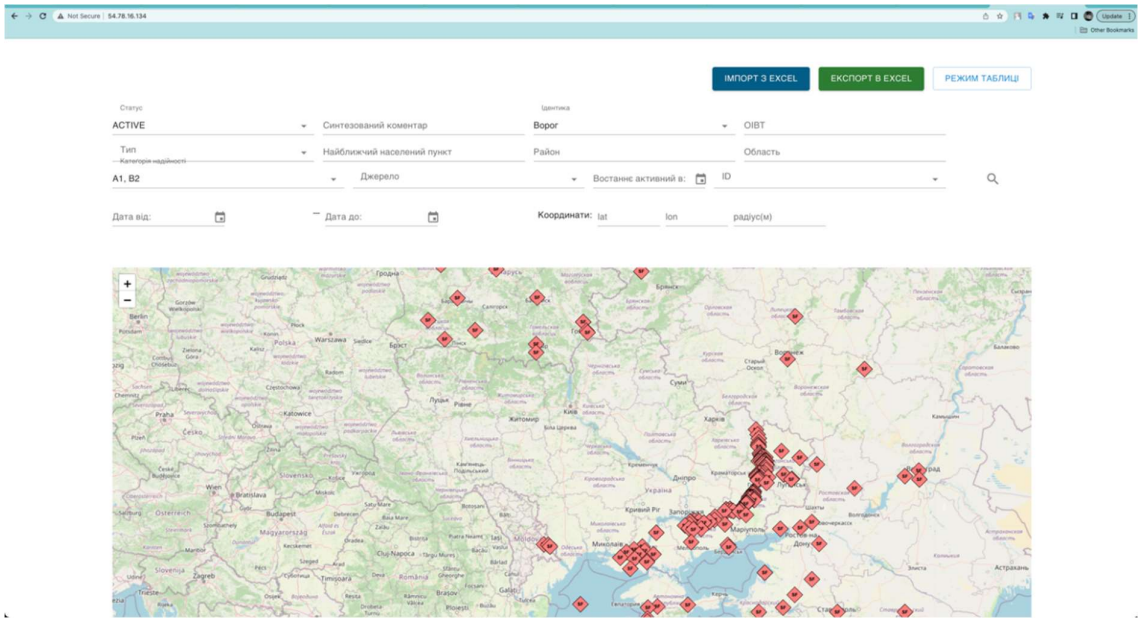


Figure 10. Analytics page, map view

Also, there is a possibility to see details about the position and to archive or mark the position as “destroyed” in map view (see figure 11). Press the appropriate buttons to see the

result. Also, when applying the “ACTIVE” filter, if position becomes archived or destroyed, it disappears from the map view.

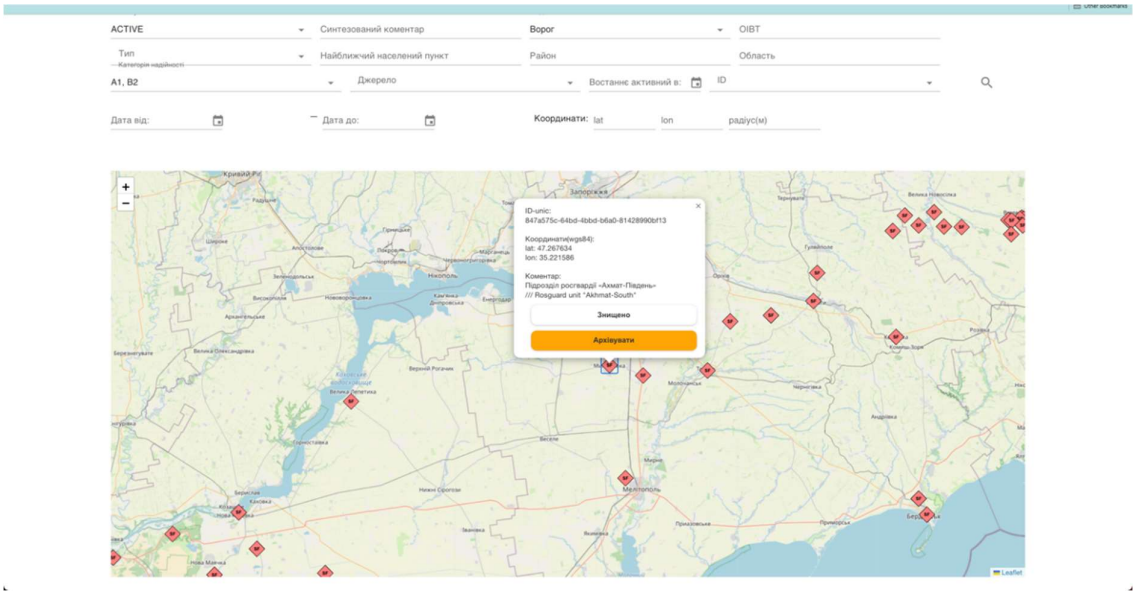


Figure 11. Map view, position managing

Importing from Excel can fill missed fields automatically. For example, if one coordinates type is presented, others will compute automatically. Another example is parsing object type from “oivt” (озброєння і військова техніка). Also, the system can generate synthesized comment and perform reverse geocoding to detect region, the closest settlement, and district, if coordinates are presented. The import page shows errors on data validation, providing user ability to edit data, revalidate it, and save. The import page is shown on figure 12.

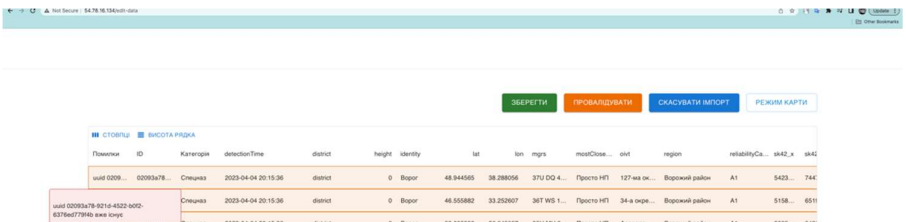


Figure 12. Import page view

The system supports visual validation, and although the system supports preventing duplicates by an unique id, it is useful to avoid unexpected duplication via a visual control.

Imported data is presented as black symbols, and active positions with their original colors are displayed on the map (see figure 13).

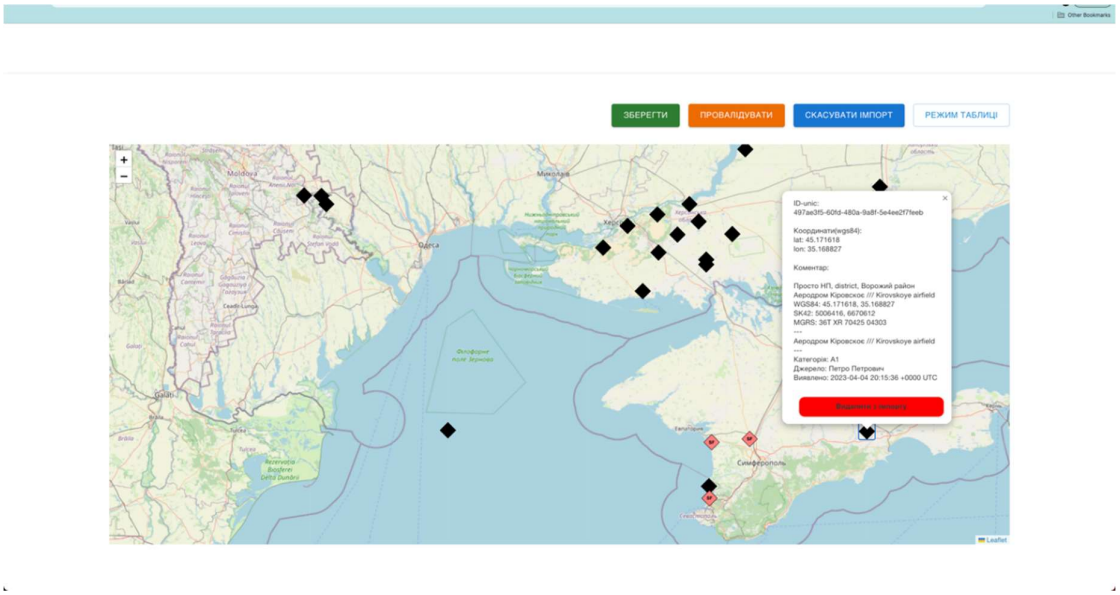


Figure 13. Import page interactive map view

Inserting a single position is available and combines interactive view and input form on one page as shown on figure 14. The system can fill in missed data the way it behaves in the Excel import.

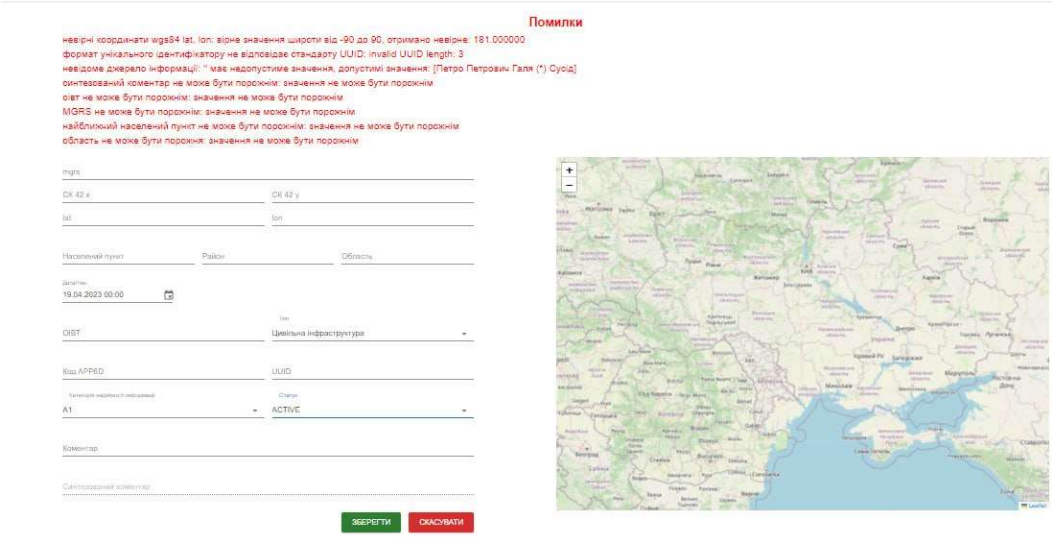


Figure 14 – Manual input of the single position

FORMAL SPECIFICATION AND VERIFICATION

Specification and verification tools. There are a lot of tools for formal specification and verification. Some of them are TLA+, Coq, Isabelle, Z3, Alloy, Agda and Dafny. Our team decided to use Dafny [27] after a thorough analysis of all tools. The main advantages of Dafny are the following:

1. Easy to learn and use: Dafny has a straightforward syntax, simplifying learning and using this tool.

2. Integration with IDEs: Dafny integrates with popular IDEs such as Visual Studio or Visual Studio Code, making it easy for developers to use it.

3. Automatic verification: Dafny automatically verifies preconditions, postconditions, and invariants at compilation time, which helps to detect errors in the early stages of the development process.

Formal specification of the chosen module.

The specification was made for the algorithm of dynamic formation of the synthesized comment based on the template string specified in the user's project settings.

Two methods have been created with preconditions, postconditions, and loop invariants that formally describes their behavior. The *formSummaryComment* method receives a template string and a list of key-value pairs (data) as an input (see ANNEX A). It returns a synthesized comment formed by replacing the keys in the template string with corresponding values. The *subSring* method finds the starting index of a given substring in a larger string (see ANNEX A).

Results of the formal verification.

Verification results showed that our module meets all determined requirements. Figure 15 shows the results of the verification.

```
Dafny program verifier finished with 2 verified, 0 errors
```

Figure 15. Results of the verification

Dafny automatically verifies preconditions, postconditions, and invariants, and as a result, we have two successfully verified methods. It affirms that our module dynamically generates synthesized comments accordingly to specified requirements.

Utilizing Dafny for the formal specification and verification of the module helped our team to guarantee the module's correctness and ensure its high quality.

DEVELOPMENT PROCESS

Description of the chosen methodology, tools, justification of the choice.

Methodology. After careful consideration, our team adopted ***Scrum*** [28] as our project management framework over Waterfall. The decision was influenced by several factors, including our team size of 7 people and the presence of a team member who is an educated Scrum Master. Scrum's focus on teamwork, collaboration, and iterative development made it a more suitable choice for our team. We also appreciated the flexibility of Scrum, which allowed us to adapt to changing requirements and feedback from stakeholders. On figure 16 we can see the burndown chart – one of the many instruments in scrum to track the continuous progress. Additionally, Scrum's emphasis on regular check-ins and continuous improvement helped us to stay on track and ensure that the project was progressing smoothly. Overall, Scrum provided us with the tools and framework to manage our project and successfully deliver high-quality results.

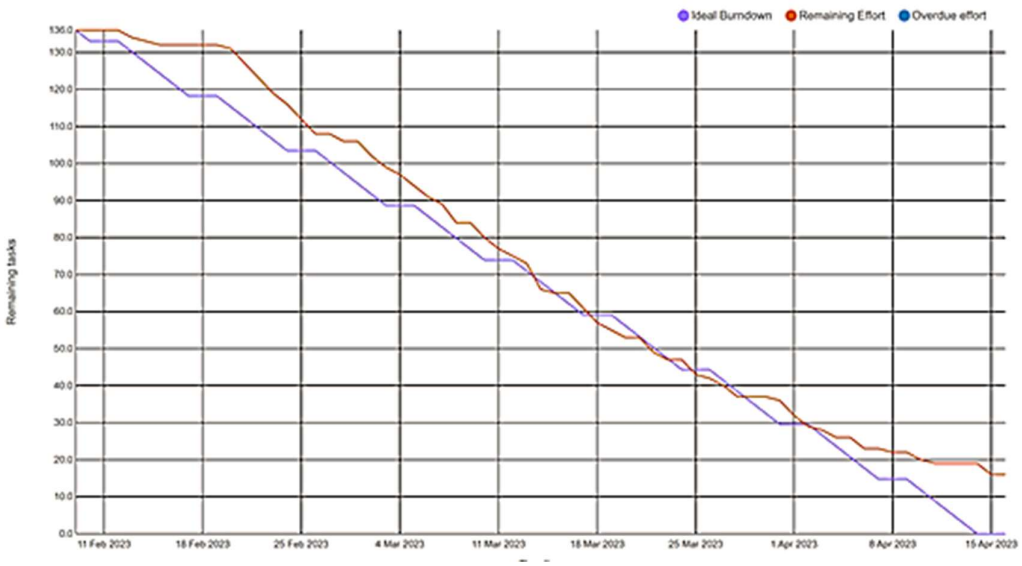


Figure 16. Burndown chart (Scrum)

Team roles. In the Scrum methodology, three key roles are crucial in ensuring the project's success: Product Owner, Scrum Master, and Developers. The Product Owner is responsible for defining the product vision, managing the product backlog, and ensuring that the team is focused on delivering value to the customer. The Scrum Master is responsible for ensuring that the Scrum framework is implemented correctly, facilitating meetings, removing obstacles, and ensuring the team works effectively. The Developers are responsible for delivering the actual product, working together to create the various features and components that make up the final product.

In our project, Serhii Syrota played the role of Product Owner, providing guidance and direction for the team and ensuring that the product met the customer's needs. Mikhail Syvachenko took on the role of Scrum Master, working closely with the team to ensure that they were working effectively and that any obstacles were quickly removed. Additionally, both Serhii Syrota and Mikhail Syvachenko also worked as developers, contributing their technical skills to the project alongside the other team members.

By having these clear roles defined and everyone understanding their responsibilities, we could work more efficiently and effectively as a team. The combination of Serhii Syrota's vision, Mikhail Syvachenko's guidance and support, and the technical skills of the developers allowed us to create a successful project that met the customer's needs.

The rest of the team - Serhii Peshko, Mykhailo Bubka, Eduard Andrashchuk, Maksym Rasakhatskyi, and Dmytro Ruban were taking part as developers.

Issue tracking application. In deciding on the project management tool for our team's project, it was ultimately decided that ***YouTrack*** was the best choice over Jira. While both platforms have their strengths, the simplicity and minimalistic design of YouTrack better suited our team's needs. With YouTrack, we appreciated the streamlined interface that allowed us to quickly create and track tasks without being bogged down by unnecessary features or cluttered menus. Additionally, YouTrack's customizable workflows and issue-tracking capabilities made it easy to adapt to our project's specific needs (figure 17 shows the Agile Board – an interactive board to manage team tasks). Overall, we found YouTrack a more user-friendly and intuitive tool that allowed us to focus on what mattered most: managing our project effectively and efficiently.

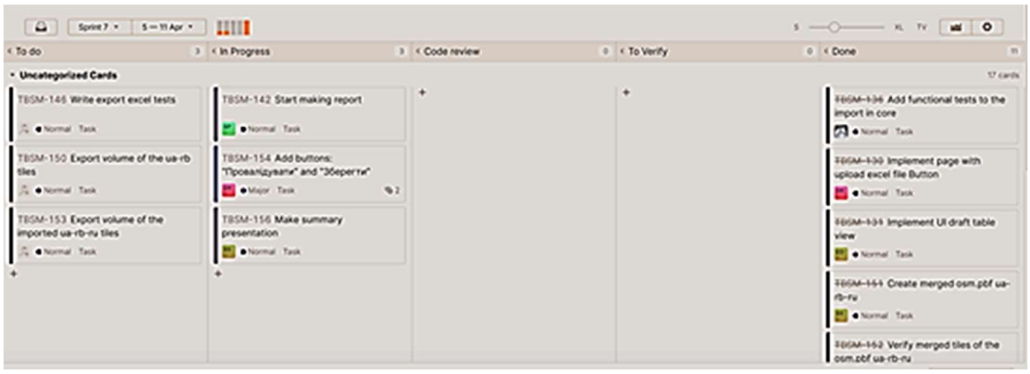


Figure 17. Agile board

YouTrack tools. In managing our project, our team decided to utilize a ***Knowledge Base*** in addition to two separate agile boards for management and development tasks. The Knowledge Base allowed us to compile and organize all relevant project information, making it easily accessible to all team members. This helped to ensure that everyone was on the same page and clearly understood project goals, requirements, and progress. The use of two ***separate agile boards for management and development*** tasks provided us with a more efficient and organized approach to project management. The management board allowed us to track high-level tasks and goals, while the development board focused on the technical details and specific tasks required to achieve those goals. This division helped us to manage both the overall project scope and the individual tasks necessary to complete it. By utilizing a Knowledge Base (figure 18, 1 picture – list of different topics covered by Knowledge Base) and two separate agile boards (figure 18, 2 picture – Agile board for the management), our team was able to stay organized, informed, and focused on delivering a successful project.

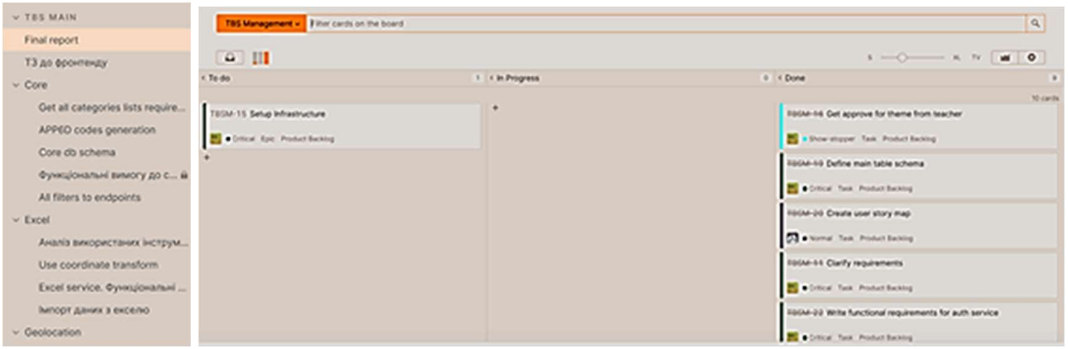


Figure 18. Knowledge Base and Management board

Pull requests and code review. Our team used a pull requests system to ensure code quality (see figure 19 with example of pull requests). Each developer worked on a separate branch and created a pull request after the work was done to merge his branch with a "master" branch. After the pull request was created, other developers conducted a code review. This helped to reveal potential problems, bugs, code style violations, or other problems related to merging branches. Comments and suggestions for improving the code were left to the developer if necessary. Figure 20 shows the example of comments to the pull request.

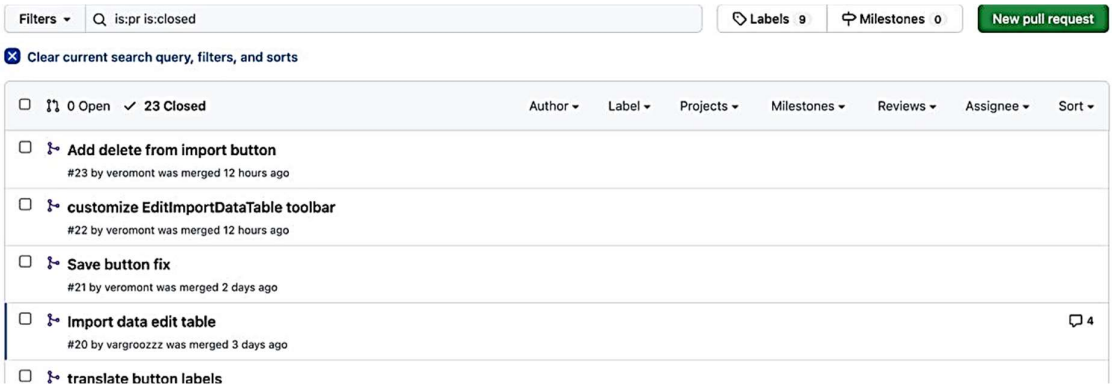


Figure 19. List of pull requests example

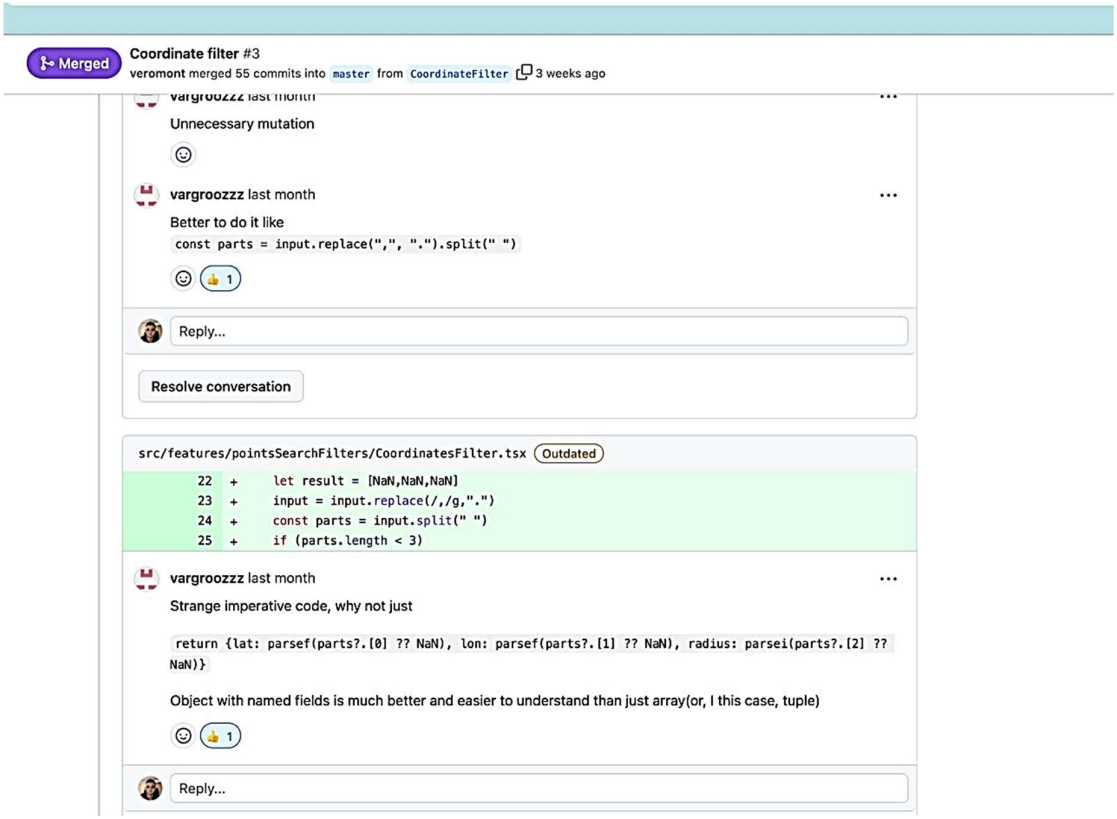


Figure 20. Example of comments to the pull request

Automatic deployment. We created CI/CD pipelines to simplify deploying our services to the server. The deployment process was started automatically after each push to the “master” branch. This process includes compiling and pushing the docker image to the server. If some tests fail, new version will not be pushed. The server receives a new docker image and runs it instead of old one. This helped us to make changes to the production quickly. Figure 21 shows an example of a pipeline that was successfully run.

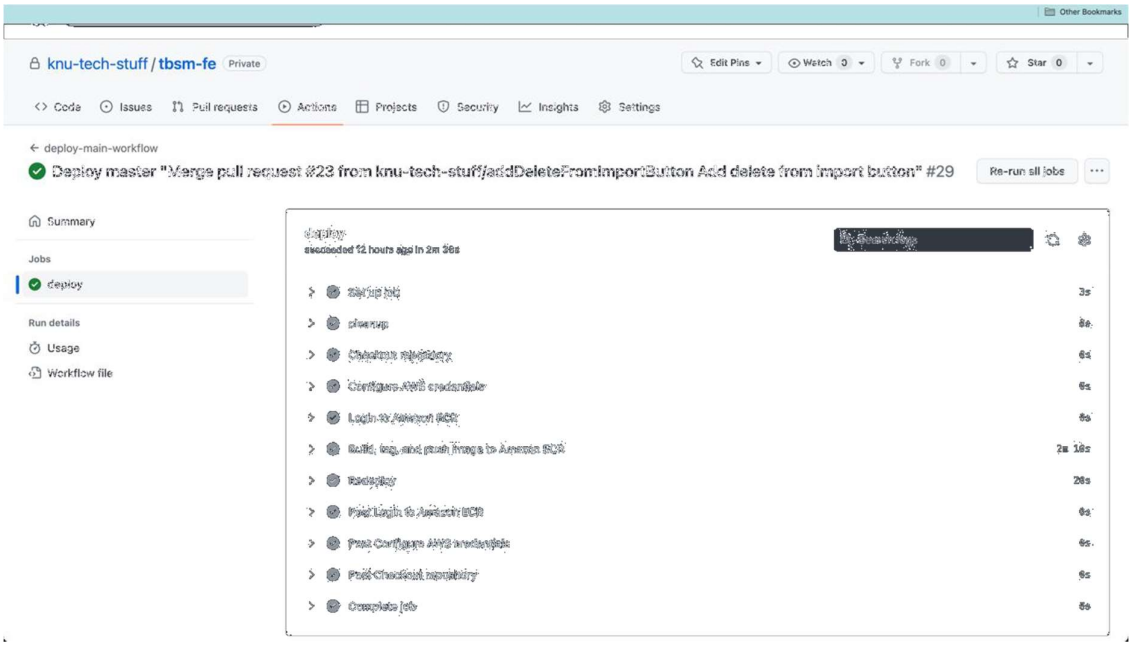


Figure 21. Example of CI/CD pipeline

Pair programming. Our team practiced pair programming [29] during the project implementation. Serhii Peshko was using Golang for the first time; hence, pair programming was used to help him complete his first task. He was programming with Serhii Syrota, who is an experienced Golang developer. This method helped Serhii acquire the required practical skills and knowledge quickly. Pair programming helped improve code quality, enhance soft skills, and share knowledge with the team. As a result, our team became more cohesive and productive.

Sprint planning.

Our past experiences, both successes and failures, heavily influenced our team's approach to sprint planning. Before each sprint, we conducted a thorough sprint review with our supervisors to evaluate the progress made in the previous sprint and identify areas for improvement. By reflecting on our past performance, we learned from our mistakes, celebrated our successes, and adjusted our approach accordingly. We used this valuable feedback to set realistic goals and objectives for the upcoming sprint, making sure to focus on areas where we needed improvement. This approach allowed us to continuously improve our processes and methodologies, and ensure that each sprint (figure 22 shows a list of sprints, the team finished on the way to finish the implementation) was better than the last. Using our past experiences as a guide, could plan and execute each sprint with greater efficiency, leading to better results for the team and the project as a whole.

Figure 22. Sprints list

At the start of the project the team created a user story map to arrange user stories into a successful model of understanding the functionality of a system. Figure 23 shows our user story map.

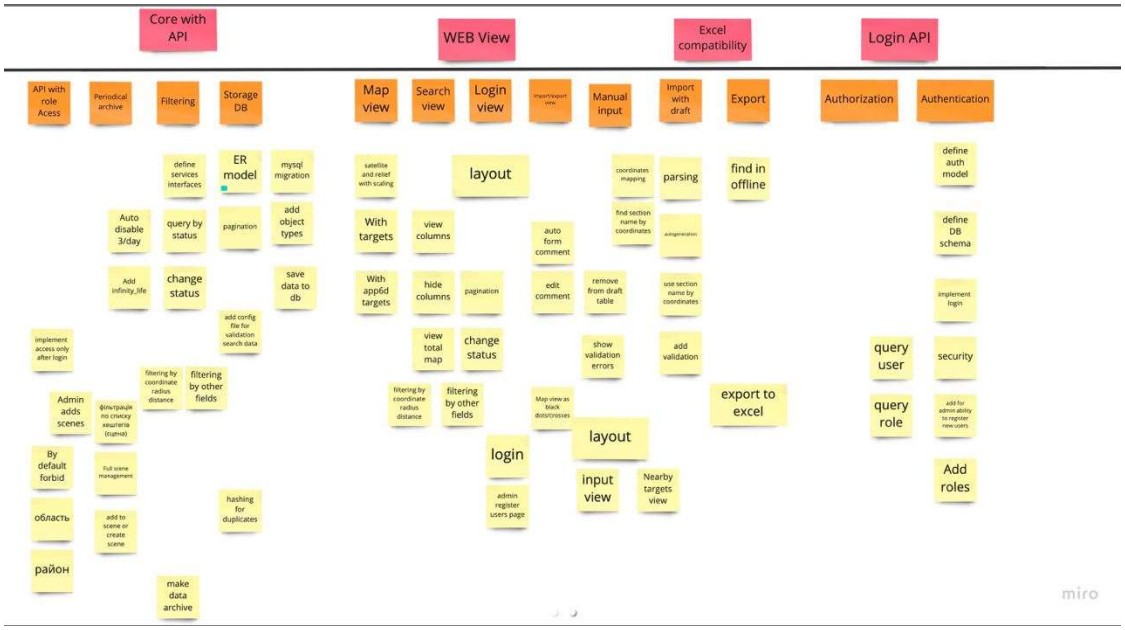


Figure 23. User story map

Table 6 shows a detailed description of each sprint.

Table 6. Sprints description

Sprint	Goals	Success	Fails
1	1. Set up the infrastructure. 2. Define basic infrastructure. 3. Gain understanding of project domain	Implemented vital flows: 1. Data filtration. 2. Map on FE. 3. Coordinate system mapping. 4. Tile server as 3rd party docker image investigated	1. Insufficient architecture disclosure on the sprint planning, which led to miscommunication and unjustified expectations. 2. Uncertainty at the beginning of the project on what to work on for less advanced participants.
2	1. Prepare the deployment environment for release. 2. Create the design for frontend. 3. Extend the features of application. 4. Deploy the Core and ExcelAdapter services.	1. Deployment CI-pipelines and Amazon ECR created. 2. Created design for FE, APP6D icons of FE implemented. 3. Investigated the reverse-geocoding docker image. 4. ExcelAdapter partially covered with tests. 5. Conducted a Sprint Retrospective.	Lost a team member.
3	1. Full text search and pagination for Core service. 2. APP6D categories mapping. 3. Data filtering. 4. Implementation of coordinate conversion in different systems on import. 5. Implementation of reverse geocoding prototype for Ukrainian regions. 6. Basic implementation of search page on FE according to design	1. Pagination for Core services. 2. APP6D categories mapping. 3. Data filtering. 4. Implementation of coordinate conversion on import. 5. Implementation of reverse geocoding prototype for Ukrainian regions. 6. Basic implementation of search page on FE.	1. Full text search for Core service. 2. The implementation goals were changed on fly because of unexpected blocker. 3. Unexpected efforts changed the outcome of Sprint (small part of implementation was not finished).
4	1. Implement reverse geocoding based on import. 2. Reverse geocoding integration – handle corner cases (implement the district determination based on the location).	1. Implemented reverse geocoding based on import. 2. Handled corner cases for reverse geocoding. 3. Implemented advanced search (full text, time series).	1. Not every implementation got the appropriate testing. 2. Due to time pressure the codebase quality was suffering, which led to code being harder to be

Sprint	Goals	Success	Fails
	3. Enrich the FE implementation with the endpoints interaction.	4. Enriched the FE with the endpoints interaction.	changed, a need to refactor it arised.
5	1. Complete analytics features. 2. Continue implementing the data import functionality. 3. Add new field "comment" to the db schema, import and export. 4. Add the "summary comment" formalized synthesizing. 5. Continue to implement the offline maps. 6. Review our verification approach.	1. Completed analytics features. 2. Continued implementing the data import functionality. 3. Added new field "comment" to the db schema, import and export. 4. Added the "summary comment" formalized synthesizing. 5. Continued to implement the offline maps. 6. Reviewed our verification approach.	
6	1. Implement export to excel on FE. 2. Fix date search bug, write tests for it. 3. Complete tile server initialization. 4. Implement auto archive function. 5. Write the functional tests for import on Core. 6. Draw the import functionality design. 7. Add import button to upload excel file on FE.	1. Implemented export to excel on FE. 2. Fixed date search bug, wrote tests for it. 3. Implemented auto archive function. 4. Wrote the functional tests for import on Core. 5. Drew the import functionality design	Tile server was deployed and proved to be working but not yet automated and autonomous.
7			

Reflection on the process.

Our team is proud to report that we have successfully completed our project and created the majority of the value we had planned from the start. Throughout the project, we regularly reviewed our progress and adjusted to ensure we were on track to meet our objectives. As part of this process, we identified specific parts of the project that were initially expected to be implemented but later were deemed to have low value and were canceled. While these decisions were not always easy, they allowed us to focus our efforts and resources on areas of the project that were more likely to generate value.

We also faced unexpected challenges when we lost a team member due to personal reasons. However, rather than viewing this as a setback, we were able to turn it to our advantage. As a result of the team member's departure, we had less effort wasted on unproductive work and could focus more on the core elements of the project. This allowed us to make better use of our remaining resources, and we ultimately completed the project successfully, despite this

setback. We believe that our ability to adapt to changes and make the best use of our resources was a key factor in our success, and we are proud of what we have accomplished as a team.

Results.

Throughout the course of this project, our team learned a great deal about the value of developing with a Scrum approach. We found that the Scrum methodology provided us with a framework that allowed us to work collaboratively, efficiently, and in a way that was focused on achieving our goals. With regular sprint planning, daily stand-up meetings, sprint reviews, and retrospectives, we could stay on track and ensure everyone was clear about their roles and responsibilities.

Another key lesson that we learned was the importance of teamwork. By working closely with one another and supporting each other, we were able to accomplish more than we could have on our own. Each team member brought their unique skills and perspectives to the table, and we learned to appreciate and leverage these differences to create a more robust and effective team.

We also learned a great deal about the importance of understanding our roles and the roles of other team members. We could work more effectively as a team by being clear about what was expected of us and what others were responsible for. This helped to reduce misunderstandings, confusion, and duplicated effort.

Finally, we learned the importance of communication and collaboration. Through regular meetings and open communication, we shared our ideas, concerns, and progress with one another. This allowed us to make informed decisions, identify and resolve issues quickly and ensure we were all working towards the same goals.

Overall, our experience with this project has provided valuable insights into how to work effectively as a team and use the Scrum methodology to achieve our goals. We are proud of our accomplishments and believe that these lessons will serve us well in future projects.

CONCLUSION

Our team fulfilled the tasks and achieved desired results during the project implementation. We created a flexible and effective solution, deepened our understanding of modern technologies, and demonstrated the best practices in software development. Our team reached all the following essential goals:

1. Versatility and flexibility: develop a system that can work with different geodata formats and satisfies the requirements of a wide range of users regardless of their activities.
2. User-friendly interface: create a convenient UI that facilitates quick system use without additional learning.
3. Efficiency: develop a service that can process geodata as fast as possible and contains functions that automate this process to save users' time.
4. Reliability and safety: create a system that guarantees users' data protection and provides service stability.
5. Maintenance: provide a permanent ability to update and improve the service in order to adapt to new and modern technologies.

As a result, the team developed a system that meets all the requirements listed above. Furthermore, we explored, learned, and deepened our knowledge of spatial data processing. In general, the implementation of the tasks is successful and highly effective.

Thus, our group developed a high-quality, dependable, adaptable, and durable solution that significantly simplifies managing polluted and mine-contaminated areas. Due to the experience each member of the team acquired, we can keep improving and fostering our project, considering market changes and clients' feedback.

REFERENCES

1. Google Maps. URL: https://www.google.com/maps
2. OpenStreetMap. URL: https://www.openstreetmap.org/#map
3. Microservice architecture. URL: https://microservices.io/
4. Component Based Architecture. URL: https://medium.com/omarelgabrys-blog/component-based-architecture-3c3c23c7e348
5. Docker. URL: https://www.docker.com/
6. AWS. URL: https://aws.amazon.com/
7. Go. URL: https://go.dev/
8. NET. URL: https://dotnet.microsoft.com/en-us/
9. React. URL: https://react.dev/
10. MySQL. URL: https://www.mysql.com/
11. Git. URL: https://git-scm.com/about
12. YouTrack. URL: https://www.jetbrains.com/youtrack/
13. UML Use Case Diagrams. URL: https://www.uml-diagrams.org/use-case-diagrams.html
14. Osm tile server. URL: https://github.com/Overv/openstreetmap-tile-server
15. Nominatim. URL: https://nominatim.org/
16. GitHub Actions. URL: https://docs.github.com/en/actions/learn-github-actions/understanding-github-actions
17. Gorm. URL: https://gorm.io/
18. N+1 problem. URL: https://stackoverflow.com/questions/97197/what-is-the-n1-selects-problem-in-orm-object-relational-mapping
19. Vietnam of computer science. URL: https://www.odbms.org/wp-content/uploads/2013/11/031.01-Neward-The-Vietnam-of-Computer-Science-June-2006.pdf
20. Redux. URL: https://redux.js.org/
21. Basic auth. URL: https://datatracker.ietf.org/doc/html/rfc7617
22. Leaflet. URL https://leafletjs.com/
23. PBF Format. URL: https://wiki.openstreetmap.org/wiki/PBF_Format
24. Geofabric. URL: https://download.geofabrik.de/
25. Reverse geocoding. URL: https://developers.google.com/maps/documentation/javascript/geocoding#ReverseGeocoding
26. Sandwich Testing | Software Testing. URL https://www.geeksforgeeks.org/sandwich-testing-software-testing/
27. The Dafny Programming and Verification Language. URL: https://dafny.org/
28. What is Scrum. URL: https://www.scrum.org/learning-series/what-is-scrum
29. Pair programming. URL: https://en.wikipedia.org/wiki/Pair_programming

ANNEX A

Dafny code of the specified and verified methods

```
method formSummaryComment(template: string, data: seq<(string, string)>) returns (result: string)
{
  var resultStr := template;

  var i:= 0;
  var subStrIndex := 0;

  while i <|data|
    invariant subStrIndex == -1 || 0 <= subStrIndex <= |template|
    invariant 0 <= i <= |data|
    decreases |data| - i
  {
    var key := "$" + data[i].0;
    var value := data[i].1;
    subStrIndex:= subString(template, key);

    assert subStrIndex != -1 ==> (0 <= subStrIndex + |key| <= |template|);

    i:=i+1;
    if(subStrIndex == -1)
    {
      continue;
    }
    else
    {
      var head := template[..subStrIndex];
      resultStr := head + value + template[(subStrIndex+|key|)..];
    }
  }

  return resultStr;
}
```

Figure A.1. Dafny code of the formSummaryComment method

```
method subString(str: string, subStr: string) returns (startId : int)
  requires  0 < |subStr|
  ensures   startId == -1 || 0 <= startId <= |str| - |subStr|
{
  var i := 0;

  while (i <= |str| - |subStr|)
    invariant 0 <= i <= |str|
    decreases |str| - |subStr| - i
  {
    if(str[i] == subStr[0])
    {
      var j:= 0;
      while j < |subStr| - 1
        invariant 0 <= j < |subStr|
        decreases |subStr| - 1 -j
      {
        if(str[i+j] != subStr[j])
        {
          break;
        }

        assert i+j < |str|;
        assert str[i+j] == subStr[j];

        j:= j+1;
      }
      return i;
    }
    i:= i+1;
  }
  return -1;
}
```

Figure A.2 – Dafny code of the subString method

ANNEX B

Smart reverse geocoding algorithm

```
using Microsoft.Extensions.Caching.Memory;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
using WebApi.Model;

namespace WebApi.Services
{
    public class GeoMappingService
    {
        private string reverse_url;
        private string search_url;
        private MemoryCache cache;
        private SemaphoreSlim semaphore = new SemaphoreSlim(40);

        public GeoMappingService()
        {
            cache = new MemoryCache(new MemoryCacheOptions
            {
                SizeLimit = 1000
            });
        }

        public async Task<(bool, string)> GetNearestPlaceAsync(string reverseGeoUrl, string searchGeoUrl, double latitude, double longitude)
        {
            reverse_url = reverseGeoUrl + "?lat={0}&lon={1}&zoom=10&addressdetails=1&format=json";
            search_url = searchGeoUrl + "?q={0},{1}&addressdetails=1&format=json";
            try
            {
                var goAround = await GoAround(latitude, longitude);
                return (true, JsonConvert.SerializeObject(goAround));
            }
            catch (Exception ex)
            {
                return (false, ex.Message);
            }
        }

        public virtual async Task<string> SendRequestAsync(string url)
        {
            if(cache.TryGetValue(url, out string cachedResponse))
            {
                return cachedResponse;
            }
            using var httpClient = new HttpClient();
            httpClient.DefaultRequestHeaders.Add("accept-language", "uk-UA");
            httpClient.DefaultRequestHeaders.Add("user-agent", "ExcelAdapter");
            httpClient.DefaultRequestHeaders.Add("user-agent", "ExcelAdapter");
            using var response = await httpClient.GetAsync(url);
            if(response.IsSuccessStatusCode)
            {
                string content = await response.Content.ReadAsStringAsync();

                var cacheEntryOptions = new MemoryCacheEntryOptions().SetSize(1).SetSlidingExpiration(TimeSpan.FromHours(1));
                cache.Set(url, content, cacheEntryOptions);
                return content;
            }
            else
            {
                throw new Exception();
            }
        }

        private async Task<Address> TryGetAddress(string url)
        {
            await semaphore.WaitAsync();
            Address res;
            try
            {
                var resp = await SendRequestAsync(url);
                var isRespObj = !resp.StartsWith('[');
                var address = isRespObj ? JObject.Parse(resp)["address"] : JArray.Parse(resp)[0]["address"];
                res = ParseAddress(address);
            }
            finally
            {
                semaphore.Release();
            }
            return res;
        }

        private Address ParseAddress(JToken address)
        {
            string city = string.Empty;
            string district = address["district"] != null ? address["district"].ToString() : (address["borough"] != null ? address["borough"].ToString() : string.Empty);
            string region = address["state"] != null ? address["state"].ToString() : (address["region"] != null ? address["region"].ToString() : string.Empty);
            string country = address["country"] != null ? address["country"].ToString() : string.Empty;
            if(address["city"] != null)
            {
```

```
                city = address["city"].ToString();
            }
            else if(address["town"] != null)
            {
                city = address["town"].ToString();
            }
            else if(address["village"] != null)
            {
                city = address["village"].ToString();
            }
            return new Address { City = city, Region = region, Country = country, District = district };
        }

        private async Task<Address> GoAround(double startLat, double startLon)
        {
            double[] directions = { 0, 45, 90, 135, 180, 225, 270, 315 };
            int steps = 6;
            double distance = 0;
            Address result = new Address();
            for(int i = 1; i <= steps && !result.HasRequiredInfo(); ++i)
            {
                List<Task<Address>> tasks = new();
                foreach(double direction in directions)
                {
                    double newLat = startLat;
                    double newLon = startLon;
                    if(distance > 0)
                    {
                        var newCoordinates = CalculateNewCoordinates(startLat, startLon, distance, direction);
                        newLat = newCoordinates.Item1;
                        newLon = newCoordinates.Item2;
                    }
                    tasks.Add(TryGetAddress(string.Format(reverse_url, newLat, newLon)));
                    tasks.Add(TryGetAddress(string.Format(search_url, newLat, newLon)));
                    if(distance == 0)
                    {
                        --i;
                        break;
                    }
                }
                var results = await Task.WhenAll(tasks);
                foreach(var res in results)
                {
```

```
                foreach(var res in results)
                {
                    if(result.HasRequiredInfo())
                    {
                        break;
                    }
                    result.AssignNotNull(res);
                }
                distance = distance == 0 ? 500 : distance += 3;
            }
            return result;
        }

        private (double, double) CalculateNewCoordinates(double lat, double lon, double distance, double direction)
        {
            double radiusEarth = 6371000;
            double radianLat = ToRadians(lat);
            double radianLon = ToRadians(lon);
            double radianDirection = ToRadians(direction);
            double angularDistance = distance / radiusEarth;

            double newLat = Math.Asin(Math.Sin(radianLat) * Math.Cos(angularDistance) + Math.Cos(radianLat) * Math.Sin(angularDistance) * Math.Cos(radianDirection));
            double newLon = radianLon + Math.Atan2(Math.Sin(radianDirection) * Math.Sin(angularDistance) * Math.Cos(radianLat), Math.Cos(angularDistance) - Math.Sin(radianLat) * Math.Sin(newLat));
            return (ToDegrees(newLat),  ToDegrees(newLon));
        }

        private double ToRadians(double degrees) => degrees * (Math.PI / 180);

        private double ToDegrees(double radians) => radians * (180 / Math.PI);
    }
}
```

НОТАТКИ

НОТАТКИ

НОТАТКИ

Наукове видання

"ПРОГРАМУВАННЯ: ТЕОРІЯ ТА ПРАКТИКА"
ЗБІРНИК МАТЕРІАЛІВ
ЗА РЕЗУЛЬТАТАМИ ІТ-ПРОЄКТУ
МІЖДИСЦИПЛІНАРНОЇ ІНТЕГРАЦІЇ
2022-2023 навчальний рік

"PROGRAMMING: THEORY AND PRACTICE"
MATERIALS OF THE INTERDISCIPLINARY
INTEGRATION IT PROJECT RESULTS
2022-2023 academic year

Підписано до друку 20.06.2023 р.
Формат 64x108/8.
Папір офс. Друк офс. Ум. друк арк. 20,0
Гарнітура Times New Roman. Наклад: 300 прим.,
Замовлення № 20/06/23
Віддруковано з готового оригінал-макета.

Видавництво: Видавничий дім «Гельветика» 65101,
Україна, м. Одеса, вул. Інглезі, 6/1
Телефон: +38 (095) 934 48 28, +38 (097) 723 06 08
E-mail: mailbox@helvetica.com.ua
Свідоцтво суб'єкта видавничої справи
ДК № 6424 від 04.10.2018 р.

Друкарня: Типографія «Айс Принт»
Телефон: +38 (099) 192-00-33, +38 (048) 706-92-82
E-mail:info@ice-print.com.ua