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
2021-2022 навчальний рік

КИЇВ, 2022

УДК 004.9
П78

Рекомендовано до друку вченою радою факультету комп’ютерних наук та кібернетики Київського національного університету імені Тараса Шевченка (Протокол № 9 від 17 травня 2022 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. – Київ, 2022. – 155 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

ЗМІСТ

ІНТЕГРАЦІЯ НАВЧАЛЬНИХ КУРСІВ НА ОСНОВІ ПРОЄКТНОГО ПІДХОДУ ТА ГНУЧКИХ МЕТОДОЛОГІЙ УПРАВЛІННЯ

Людмила Омельчук, Олексій Ткаченко, Олена Шишацька

Дана робота висвітлює процес реалізації другого 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 року (2020/2021 навчальний рік) реалізації проєкту оприлюднені в [1]. Результати першого року проведення проєкту вплинули на внесення змін до освітньо-професійної програми вцілому та на зміст задіяних освітніх компонент зокрема.

Актуальність роботи.

Станом на початок 2022 року попит на нових ІТ-фахівців значно перевищував можливості українських закладів вищої освіти (ЗВО). Згідно дослідження ІТ-індустрії, оприлюдненого ІТ-асоціацією 20.01.2022 [2], в київському регіоні на 1000 осіб припадало 99,2 ІТ-фахівців, що складало 35% від загальної кількості фахівців. Робота із сучасними інформаційними технологіями вимагає поглибленого рівня знань, умінь та соціальних навичок. Згідно з результатами досліджень [2, 3] ЗВО України щороку готують 16-17 тисяч бакалаврів ІТ-спеціальностей. Разом з тим, лише 44% випускників після отримання диплому працюють за фахом [3], решта не відповідають рівню своєї кваліфікації. Зазначене свідчить про наявність проблем у професійній підготовці ІТ-фахівців, зокрема, у відповідності рівня фахової підготовки випускників сучасним потребам ІТ-ринку.

Як свідчать опитування стейкголдерів освітнього процесу, при підготовці фахівців для ІТ-галузі основними проблемами є [1, 2]:

(1) існування невідповідності між очікуваннями роботодавців в галузі ІТ та практичними уміннями і соціальними навичками випускників ІТ-факультетів і, як наслідок, висока потреба у підвищенні рівня зазначених компетентностей у здобувачів вищої освіти;

(2) недостатній зв'язок між теоретичним і практичним матеріалом навчальних дисциплін;

(3) розмитість міждисциплінарних зв'язків;

(4) недостатнє системне розуміння студентами життєвого циклу розробки програмного забезпечення.

З погіршенням економічної ситуації в країні, яка пов'язана в 2020-2022 роках з глобальною епідемією COVID-19, а в 2022 році – з повномасштабною агресією росії проти України, зазначені проблема мають тенденцію до поглиблення.

Інтеграція навчальних курсів на основі проєктного підходу та гнучких методологій управління повинна сприяти усуненню зазначених проблем. Розробка та впровадження 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 33 студенти);
- "Коректність програм та логіки програмування" (вибіркова дисципліна, 4 рік навчання, викладач: к.т.н. Ткаченко О.М., задіяно 33 студенти);
- "Інструментальні середовища та технології програмування (обов'язкова дисципліна, 2 рік навчання, викладач: к.ф.-м.н. Омельчук Л.Л., задіяно 6 студентів).

В проєкті передбачалося досягнення таких цілей:

(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).

Другий рік реалізації проєкту було організовано у другому семестрі 2021/2022 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 IT-проєктами Ви маєте**;
- **в якій системі управління IT-проєктами (методології) Ви б хотіли працювати в проєкті**;
- н**а 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.

Чи маєте ви досвід роботи в реальних колективних проєктах (не навчальних)?

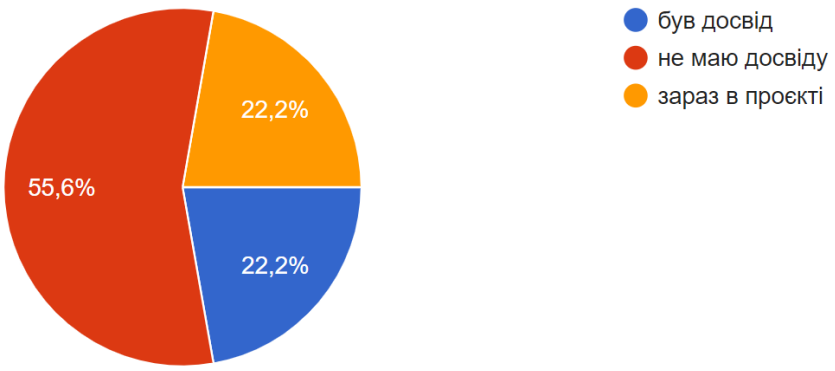


Рисунок 1. Результати анкетування щодо досвіду участі в колективних проєктах

Які компетентності ви би хотіли розвинути в рамках майбутнього проєкту?

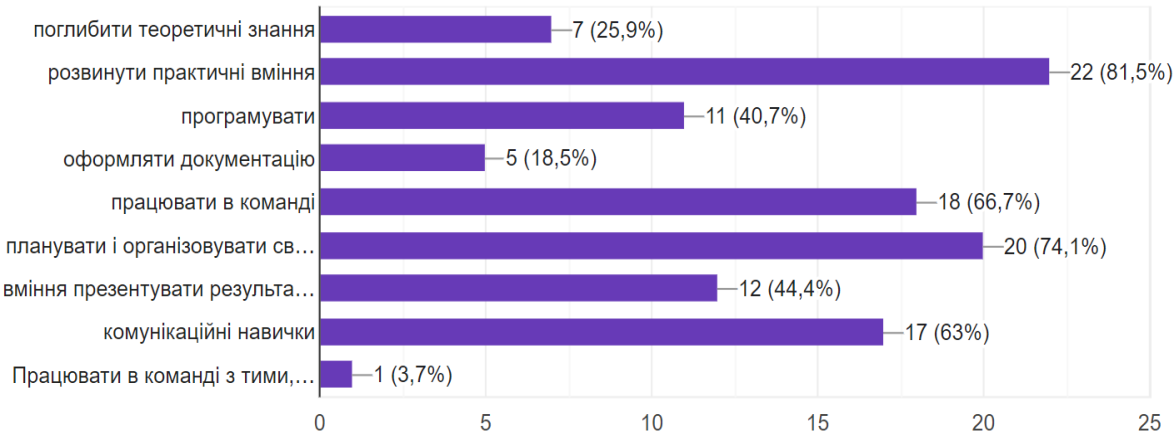


Рисунок 2. Результати анкетування щодо бажаних компетентностей

На вашу думку, використання якої мови для вас є

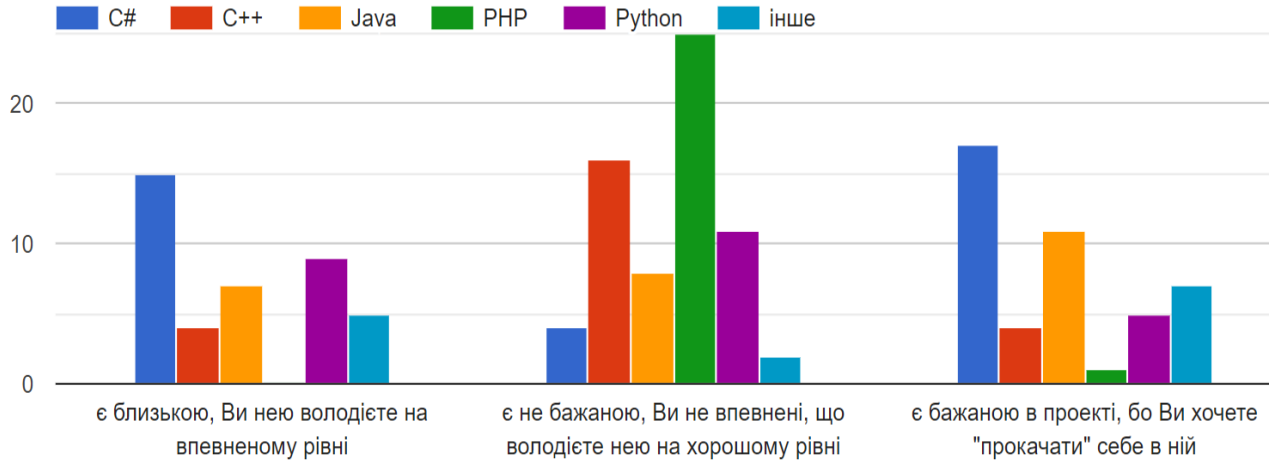


Рисунок 3. Результати анкетування щодо рівня володіння мовами програмування

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. Зазначимо, що 6 здобувачів четвертого року навчання не заповнили дану анкету.

З метою експерименту 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 взодили здобувачі другого року навчання. При цьому враховувалася місце зазначеної категорії учасників в побудованому соціометричному графі. Автори вважають, що 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.

Було сформовано сім команд, які усі успішно виконали проєкт. До складу трьох команд входило по два другокурсника.

Для забезпечення можливості порівняння результатів поєкту усім командам було рекомендовано реалізацію своєї версії системи підтримки управління ІТ-проєктом.

В рамках проєкту здійснювалися щотижневі зустрічі учасників студентських команд з викладачами. та організовано фінальний публічний захист студентських розробок. Звіти про виконання колективних проєктів, розроблених в межах проєкту-експерименту, наведено в даному збірнику. З усіма командами викладачі проводили окремі зустрічі і не розголошували їх результати іншим учасниками проєкту.

По завершенню проєкту було проведено два вихідних опитування: анонімне та авторизоване. Деякі результати цих опитувань наведено на рис. 4-7.

Рисунок 4. Результати вихідного опитування щодо розвинених компетентностей

Уявіть, що ми розпочали аналогічний командний проєкт.
Які компетентності ви хотіли б «прокачати»?

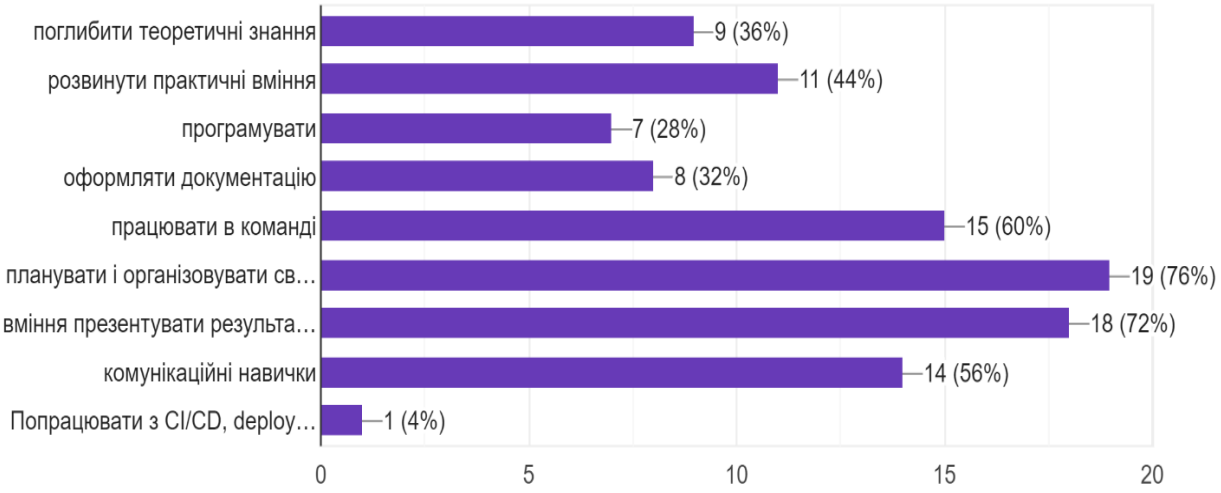


Рисунок 5. Результати вихідного опитування щодо бажаних компетентностей

Уявіть, що ви викладач і формуєте склад команд в майбутньому проєкті
Оцініть важливість критеріїв підбору учасників команд.

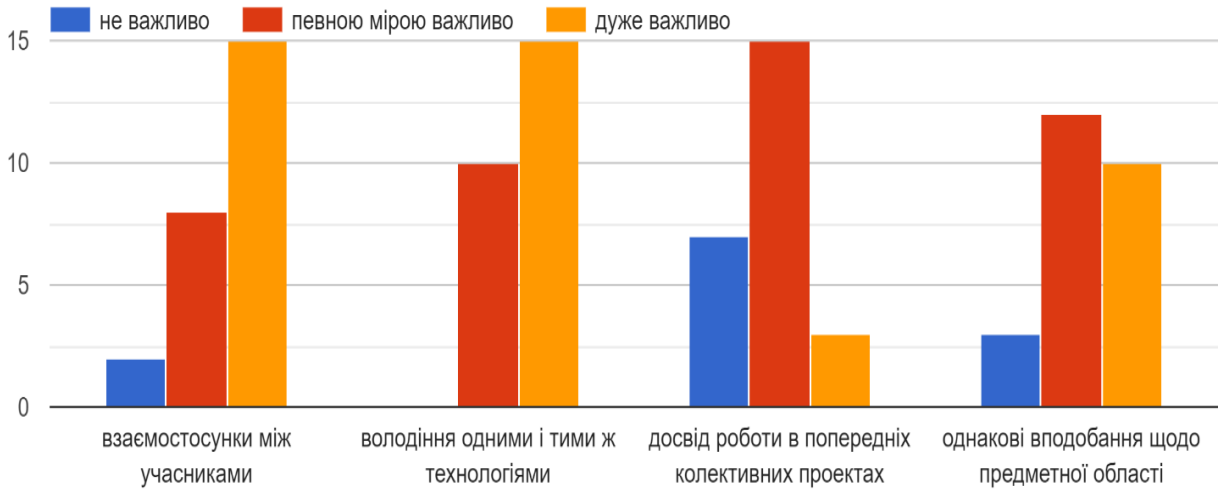


Рисунок 6. Результати вихідного опитування щодо важливості критеріїв формування команд

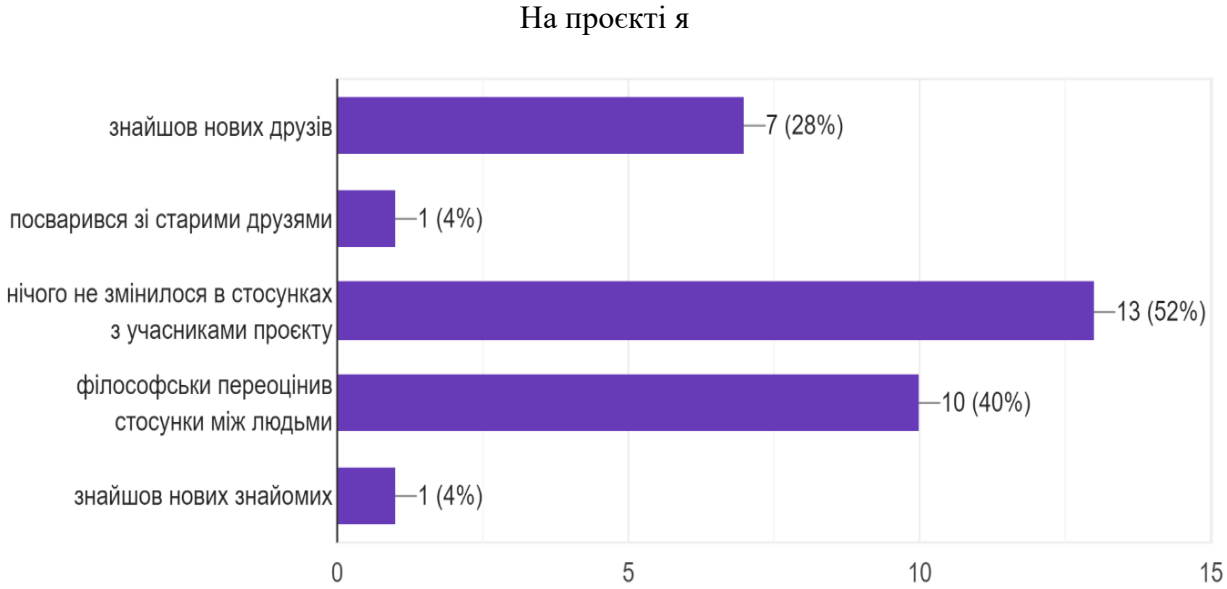


Рисунок 7. Результати вихідного опитування – рефлексія

На жаль, у 2022 році на ефективність організації проєкту крім пандемії COVID-19 вплинуло введення воїнного стану через зовнішню аграсію та окупацію частини території України, що спричинило масові переміщення громадян України, в тому числі студентів і викладачів. Ці обставини відобразились на мотивації студентів. Водночас після відновлення навчального процесу, усі команди прийняли рішення про продовження проєкту.

Висновки. В результаті проведення проєкту досягнуто наступні результати:

(1) розроблено програмні продукти згідно з поставленою задачею; незважаючи на спільну предметну область, проєкт кожної команди містив власні унікальні функціональні можливості;

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

(2) вироблення рекомендацій щодо 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 теоретично орієнтованих курсів;

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

Література:

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. https://reports.itukraine.org.ua)

3. Експерт: 44% випускників вишів працюють не за фахом // Укрінформ https://www.ukrinform.ua/rubric-society/2737000-ekspert-44-vipusknikiv-visiv-pracuut-ne-za-fahom.html, останній доступ: 2021/05/22.

РОЗРОБКА WEB-ДОДАТКУ "GEEKON"

Софія Булейко, Ігор Дуйловський, Олександр Капленко,
Марія Лісова, Сергій Мазаєв, Андрій Стецук

ВСТУП

Менеджмент у будь-якій сфері – складна і неоднорідна діяльність, але, як показує практика, на 30-70% відсотків пов'язана з участю в проектах.

Дійсно, досить велика частина сучасного бізнесу в Україні, Європі та в усьому світі є проектно-орієнтованою. В Україні ця частка наближається до 50% [1]. Це пов'язано з тим, що все більше компаній орієнтуються на створення принципово нових продуктів або послуг, на досягнення нових результатів у відомих сферах. Проектами називають вже не набір технічної документації і кошторисів (як це було раніше), а серйозні заходи, які націлені на реалізацію окремих цілей компанії.

Отже, саме від уміння реалізувати проект в компанії залежить успіх всього підприємства. В зв'язку з цим, управління проектами стає **актуальною** і важливою темою для менеджерів будь-якої ланки. На даний момент управління проектами – цілий напрям досліджень і практики, який включує систему знань, правил і стандартів. Аби швидко і ефективно досягати поставлених цілей, команді необхідні легкі, зрозумілі інструменти керування та ресурси для організації роботи.

Метою даної роботи є розробка web-додатку "GeekOn" для управління проектами**.** Досягнення поставленої мети обумовлює вирішення таких **завдань**: узагальнення інформації про менеджмент та управління; ознайомлення з наявними технологіями та програмами для ефективної організації робочого процесу; поглиблення практичних навичок роботи в команді; власне створення та тестування платформи "GeekOn" .

ОГЛЯД НАЯВНИХ НА РИНКУ СИСТЕМ

Одним з перших етапів нашої роботи був детальний аналіз ринку. У таблиці 1 наведено короткий опис наявних аналогів, на які ми орієнтувались, з їх перевагами та недоліками.

Таблиця 1. Аналіз ринку

Назва застосунку	**Переваги**	**Недоліки**
ClickUp [2]	Багатофункціональний Достатньо зрозуміла реєстрація з підказками	Занадто складний для індивідуальних потреб Погана підтримка Не інтуїтивна у використанні Багато часу на навчання персоналу Низька швидкість роботи Поганий додаток для андроїда
monday.com [3]	Багатофункціональний Зручний інтерфейс	Погана підтримка Багато часу на навчання персоналу Мобільна версія набагато гірше веб-версія Малофункціональні механізми звітності

		Незручне Відстеження часу у великих проєктах Не всі функції гарно пропрацьовані
Asana [4]	Прості інструментами керування завданнями Інтуїтивна у використанні Функціональна безкоштовна версія	Незручний при роботі з зображеннями та файлами Регулярні оновлення мають проблеми з синхронізацією з іншими системами, що призводить до проблем з інтеграцією Погана підтримка Одне завдання для одного користувача і ніяк інакше
Trello [5]	Дуже простий у використанні	Малофункциональный Незручне відстеження часу Незручне керування кількома проектами Незручний інтерфейс Погана підтримка Важко працювати з великими проектами
Jira [6]	Зручна інтеграція з іншими продуктами Багатофункціональність Зручний інструментарій для звітів та аналітики по проектам та задачам	Незручна структура дозволів для адміністратора Низька швидкість роботи Багато часу на навчання персоналу Незручна робота з користувачами не з організації Не інтуїтивна реєстрація

При розробці сайту були враховані вищеперераховані недоліки та переваги. Дещо було узагальнено та запозичено, дещо покращено (як, наприклад, проста та інтуїтивна реєстрація з підказками, стильний дизайн та зрозумілий інтерфейс).

ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Проект розроблено в інтегрованому середовищі розробки програмного забезпечення Visual Studio та Visual Studio Code, використовуючи один з фреймворків, що стрімко набирає популярності, ASP.NET Core.

ASP.NET Core – це остання версія популярного фреймворку для написання веб орієнтованих додатків від Microsoft. ASP.NET Core - це абсолютно новий підхід в розробці і розгортанні веб-додатків, при якому пропонується незалежна від хостингу інфраструктура і високопродуктивна модель. Це означає, що ASP.NET Core додатки відрізняються високою продуктивністю і при цьому можуть працювати під управлінням різних операційних систем. З даною платформою є можливість створювати як прості веб-ресурси, так і дуже складні сайти, здатні до обробки багатотисячного потоку користувачів.

Дизайн проекту було створено за допомогою онлайн-сервісу розробки інтерфейсів Figma та відтворено завдяки використанню стандартних технологій веб-розробки HTML5 та CSS3.

HTML – мова розмітки гіпертекстових документів, стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Документ HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді. Попри те, що HTML – штучна комп'ютерна мова, вона не є мовою програмування. HTML разом із каскадними таблицями стилів та вбудованими скриптами — це три основні технології побудови веб-сторінок [7]. HTML впроваджує засоби для:

• створення структурованого документа шляхом позначення структурного складу тексту: заголовки, абзаци, списки, таблиці, цитати та інше;

• отримання інформації із Всесвітньої мережі через гіперпосилання;

• створення інтерактивних форм;

• включення зображень, звуку, відео, та інших об'єктів до тексту.

Документ HTML 5.0 складається з трьох частин:

• Декларація типу документа (англ. Document type declaration, Doctype), на початку документа, в якій визначається тип документа (DTD).

• Шапка документа (знаходиться в межах елемента head), в якій записано загальні технічні відомості або додаткова інформація про документ, яка не відтворюється безпосередньо в браузері;

• Тіло документа (може знаходитися в елементах body або frameset), в якому міститься основна інформація документа.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки — розділення змісту сторінки (даних) та їхньої візуальної презентації. CSS використовується авторами та відвідувачами веб-сторінок, щоб визначити кольори, шрифти, верстку та інші аспекти вигляду сторінки.

Також для відображення та динамічного оновлення певних даних на сторінці було використано бібліотеку JQuery. А для реєстрації користувачів було застосовано Google API та Facebook API.

Сервіс BoxIcons[8] було використано для пошуку різних іконок, використаних в дизайні.

Для зручності роботи над проектом було використано вебсервіс для спільної розробки програмного забезпечення GitHub. Посилання на репозиторій знаходиться в Додатку А.

ПРИЗНАЧЕННЯ І ВИМОГИ СИСТЕМИ

Призначення системи. Система "GeekOn" повинна забезпечити можливість зручного доступу користувачів до управління та організації роботи проектів, формування єдиної бази користувачів, які можуть як створювати власні проекти, так і приєднатися до вже існуючих в ролі учасників.

Також необхідна реалізація наступних завдань:

- можливість доступу до інформації та документів, розміщених на проєктах;
- можливість самостійно створювати проєкт(и);
- можливість бути приєднаним до проєкту(ів);

Цільова аудиторія представлена наступними групами користувачів:

- користувачі, які бажають створити проєкт та приєднати інших користувачів;
- користувачі, які бажають створити проєкт для особистого користування

Вимоги до системи.

Вимоги до стилістичного оформлення системи.

Стилістичне оформлення Системи має відповідати дизайн-макету, погодженого з командою та замовниками. Використання колірних схем, графічних елементів, шрифтів погодженого дизайн-макету в процесі створення Системи є обов’язковим.

Вимоги до графічного дизайну сайту.

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.

Дизайн повинен бути адаптивним: склад, розміри і взаємне розташування елементів всіх шаблонів динамічно змінюються в залежності від розміру вікна браузера, в якому відображається інформація.

Дизайн сайту має передбачати вдале поєднання стилістики, кольорів, шрифтів, логотипу.

Присутня мінімальна кількість графіки, більшість елементів повинні бути реалізовані засобами CSS.

Вимоги до верстки і програмування

Система повинна бути оптимізованою і зверстаною для роботи в різних стабільних версіях браузерів (без помилок). Для правильного відображення інтерфейсу Системи на різних пристроях і браузерах повинна бути коректна, адаптивна верстка.

Система повинна бути виконана за шаблоном MVC та містити в собі всі необхідні компоненти (моделі, контролери та представлення) для сутностей проєкту, завдання та підзавдання.

Сутність "Проєкт" має містити його назву, список завдань.

Сутність "Завдання" має містити назву, список підзавдань.

Сутність "Підзавдання" має містити назву, статус.

Крім того програма має передбачати авторизацію користувача. Система повинна надавати можливість авторизуватися за допомогою електронної адреси, та соц. мереж, таких як Facebook.

Система повинна містити інформацію про права інтелектуальної власності, яка: розташована внизу на кожній сторінці із вмістом та помітна.

Типографія Системи повинна відповідати наступним вимогам:

- поля вводу інформації повинні бути підписаними, мати короткі, інформативні та однозначні назви, які розташовуються над полем. Поля вводу інформації повинні бути вирівняні одне під одним, по одному в рядок, ширина поля повинна відповідати вмісту, який вводиться;
- під час введення інформації користувачам необхідно надавати помітні та зрозумілі підписи та/або інструкції;
- помилки введення повинні виявлятися автоматично і, якщо відомо, як їх виправити, то користувачеві повинні надаватися підказки щодо їх виправлення;

Вимоги до вікна Системи

Система повинна забезпечувати коректне відображення даних в наступних браузерах актуальних версій:

- Google Chrome;
- Opera;
- Microsoft Edge;
- Safari.

Система повинна забезпечувати коректне відображення даних у веб-браузерах на мобільних пристроїв.

ОПИС ОРГАНІЗАЦІЇ ІНФОРМАЦІЙНОЇ БАЗИ

Логічна структура бази даних

Використана СУБД : MySQL 2018. Для детальнішого ознайомлення і роботи з sql-запитами було опрацьовано "Microsoft SQL Server 2012 T-SQL Fundamentals"[9]. На

рисунку 1 зображено діаграму бази даних GeekOnDB, рисунку 4.2 діаграма класів для реєстрації та авторизації користувача.

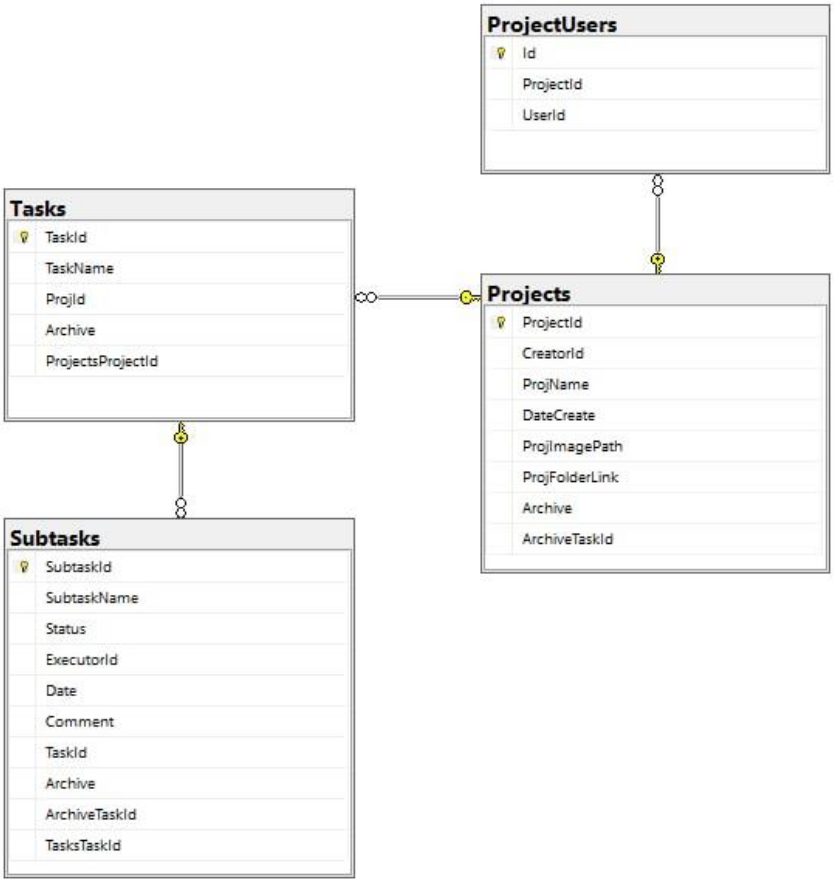


Рисунок 1. Діаграма бази даних GeekOnDB

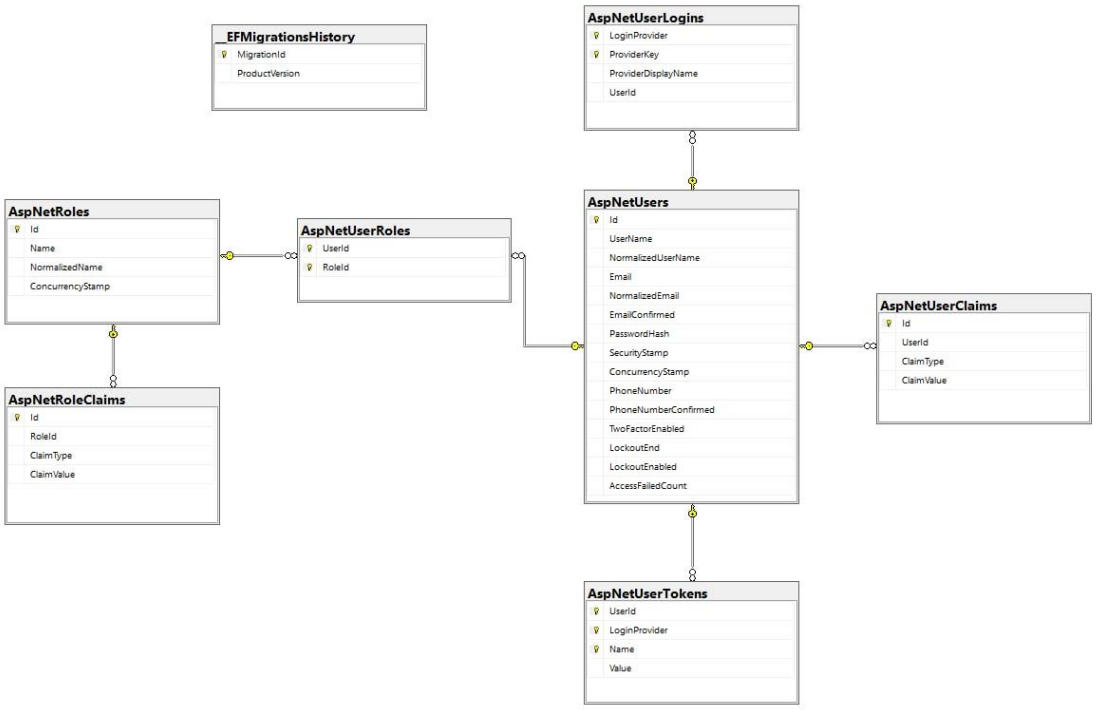


Рисунок 2. Діаграма бази даних GeekOnDBIdentity

Перелік та короткий опис таблиць наявних в базі наведено в таблиці 4.1. Описи атрибутів кожної таблиці з бази наведені в таблицях 2-5.

Таблиця 2

Номер	**Таблиця**	**Опис**
1	Projects	Таблиця для збереження інформації про проекти (детальніше в таблиці 4.2)
2	Tasks	Таблиця для збереження інформації про категорії завдань (детальніше в таблиці 4.3)
3	Subtaks	Таблиця для збереження інформації про завдання (детальніше в таблиці 4.4)
4	ProjectUsers	Таблиця для реалізаціїї відношення "багато до багато" для таблиць проектів на користувачів (детальніше в таблиці 4.5)

Опис таблиць

Таблиця для збереження інформації про проекти (таблиця 3).

Таблиця 3. Таблиця Projects

Таблиця Projects Атрибут	**Тип**	**Опис**
ProjectId	integer	Ідентифікатор проекту
CreatorId	integer	Ідентифікатор користувача, що створив проект
ProjName	nvarchar(max)	Назва проекту
DateCreate	dateTimeOffset (7)	Дата створення (з урахуванням часового поясу)
ProjImagePath	nvarchar(max)	Шлях до зображення, обраного фоном проекту
ProjFolderLink	nvarchar(max)	Посилання на папку проекту на гугл диску
Archive	bit	Вказує, чи заархівовано проект
ArchiveTaskId	integer	Ідентифікатор категорії, в яку будуть додаватись заархівовані завдання

Таблиця для збереження інформації про категорії завдань(таблиця 4).

Таблиця 4.Таблиця Tasks

Таблиця Tasks Атрибут	**Тип**	**Опис**
TaskId	integer	Ідентифікатор категорії завдань
ProjId	integer	Ідентифікатор проекту, якому належить категорія завдань
TaskName	nvarchar(max)	Назва категорії
Archive	bit	Вказує, чи заархівовано проект

Таблиця для збереження інформації про завдання(таблиця 5).

Таблиця 5. .Таблиця Subtaks

Таблиця Subtaks Атрибут	**Тип**	**Опис**
SubtakId	integer	Ідентифікатор завдання
ExecutorId	integer	Ідентифікатор виконавця завдання
TaskId	integer	Ідентифікатор категорії, до якої належить це завдання зараз
SubtaskName	nvarchar(max)	Назва завдання
Status	nvarchar(max)	обраний один з чотирьох статусів виконання завдання (To do, In Progress, Finished, Bugs)
Date	dateTimeOffset(7)	Фінальна дата (дедлайн)
Comment	nvarchar(max)	Коментар до завдання щодо його виконання, чи будь-який інший
Archive	bit	Вказує, чи заархівовано проект
ArchiveTaskId	integer	ідентифікатор категорії, якій буде належати це завдання після його архівації (або розархівації)

Таблиця для реалізаціїї відношення "багато до багато" для таблиць проектів на користувачів(таблиця 6)

Таблиця 6. Таблиця ProjectUsers

Таблиця ProjectUsers Атрибут	**Тип**	**Опис**
Id	integer	Ідентифікатор
ProjectId	integer	Ідентифікатор проекту
UserId	integer	Ідентифікатор користувача

РЕАЛІЗАЦІЯ СИСТЕМИ "GEEKON"

Реалізація системи

Виконання роботи було розділене на кілька етапів. В таблиці 7 відображений календарний план виконання проекту.

Таблиця 7. Календарний план проєкту

№ з/п	**Назва послуг за етапами етапу**	**Термін**	**Результат**
1	**Аналіз ринку, вибір методики**	02.02-09.02	Таблиця з порівнянням, недоліками та перевагами існуючих систем. Обрана методологія роботи в команді.
2	**Створення технічного завдання на розробку**	09.02-16.02	Технічне завдання
3	**Створення логотипу та назви**	09.02-16.02	Зразок зображення логотипу, виконаний в Adobe Illustrator

4	**Розробка дизайну**	09.02-16.02	Інтерактивний зразок програмного забезпечення у figma.com
5	**Розробка початкових основ програмного забезпечення**	16.02-23.02	Зразок програми з готовою реєстрацією (авторизацією) користувача, БД та контролерами.
6	**Подальша робота з ПЗ**	23.02-09.03	Зразки програми з уточненими деталями бекенду та реалізованим фронтендом
7	**Загальна модернізація програмного забезпечення та створення робочої документації**	09.03-23.03	Дослідний зразок сайту. Програма та методика попередніх випробувань Протокол попередніх випробувань
8	**Тестування програмного забезпечення**	16.03-30.03	Програма та методика приймальних випробувань; Опис системи; Посібник користувача; Посібник адміністратора ; Завершене програмне забезпечення

Для реалізації було обрано архітектурний шаблон MVC, та створений проект ASP.NET MVC Web Application. На рисунку 3 відображені згенеровані класи моделей, для кожного з яких були створені також контроллери та представлення.

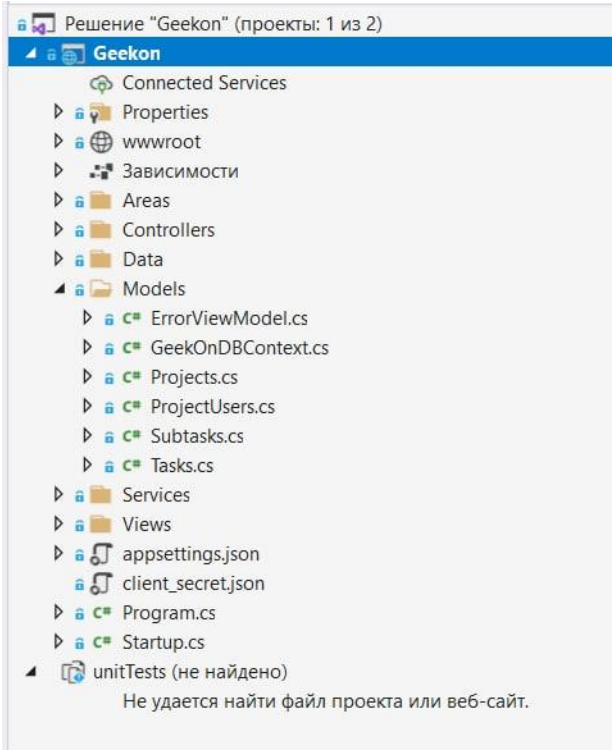


Рисунок 3. Класи моделей.

Як вже зазначалось вище, для представлень було використано стандартні технології веб-розробки HTML5 та CSS3. А також застосовано бібліотеку JQuery при написанні ajax-запитів для видалення, архівацію, редагування більшості елементів, а також для відображення категорій, завдань, меню та форм. Приклади коду для редагування завдання та видалення проекту зображені на рисунка 4 та 5 відповідно. Для зручного пролистування екрану було використано сучасний слайдер Swiper [10]. Приклад застосування в js та html коді наведено на рисунках 6-7.

```
function SUBMIT_MODAL_SUBTASK_EDIT(event) {
    event.preventDefault();
    var data = $("#edit-subtask-form").serialize();
    $.ajax({
        type: "POST",
        url: "/Subtasks/Edit",
        data: data,
        success: function (res) {
            $("#editSubtaskPlaceholder").find('.modal').modal('hide');
            $("#editSubtaskPlaceholder").html(res);
            $("#editSubtaskPlaceholder").find('.modal').modal('show');
            $.ajax({
                type: "GET",
                url: "/Tasks",
                data: {
                    projId: @ViewBag.projId,
                },
                success: function (res) {
                    $("#placeholderForTasks").html(res);
                    PROJ_STAT(@ViewBag.projId);
                }
            })
        }
    })
}
```

Рисунок 4. Код редагування завдання

```
function PROJ_DELETE(projId) {
    if (confirm("Confirm delete?")) {
        $.ajax({
            type: "POST",
            url: "/Projects/Delete",
            data: {
                id: projId,
            },
            success: function (res) {
            window.location = "/ProjectUsers/Index";
            }
        })
    }
}
```

Рисунок 5. Код для видалення проекту

```
<script>
    var swiper = new Swiper('.swiper', {
        loop: true,
        spaceBetween: 60,
        slidesPerView: 3,
        centeredSlides: true,
        slideToClickedSlide: true,

        pagination: {
            el: '.swiper-pagination',
            clickable: true,
        }
    });
    swiper.on('activeIndexChange', function () {
        swiper.updateSlidesClasses();
        document.getElementById('ProjImagePath').value = document.getElementsByClassName('swiper-slide-active')[0].id;
    });
</script>
```

Рисунок 6. Swiper

```
<div class="swiper" style="padding:60px 0 60px 0;">
    <div class="swiper-wrapper">
        @foreach (var slides in ViewData["projBack"] as List<string>)
        {
            <div id="@slides" class="swiper-slide" style="background-image:url('../projBack/@slides'); background-size:cover;"></div>
        }
    </div>
    <div class="swiper-pagination"></div>
</div>
```

Рисунок 7. Swiper (html)

Інтерфейс користувача

На рисунку 8 зображено вигляд головної сторінки веб-ресурсу для незареєстрованого користувача .

Рисунок 8. Головна сторінка веб-ресурсу "GeekOn"

При натисканні на анімаційну стрілку відвідувачеві пропонується вхід (можливий вхід через аккаунти Google/Facebook) або реєстрація(з підтвердженням через пошту). Вигляд відповідних вікон відображено на рисунках 9-10, перехід між якими відбувається рухом мишки зліва-направо і навпаки.

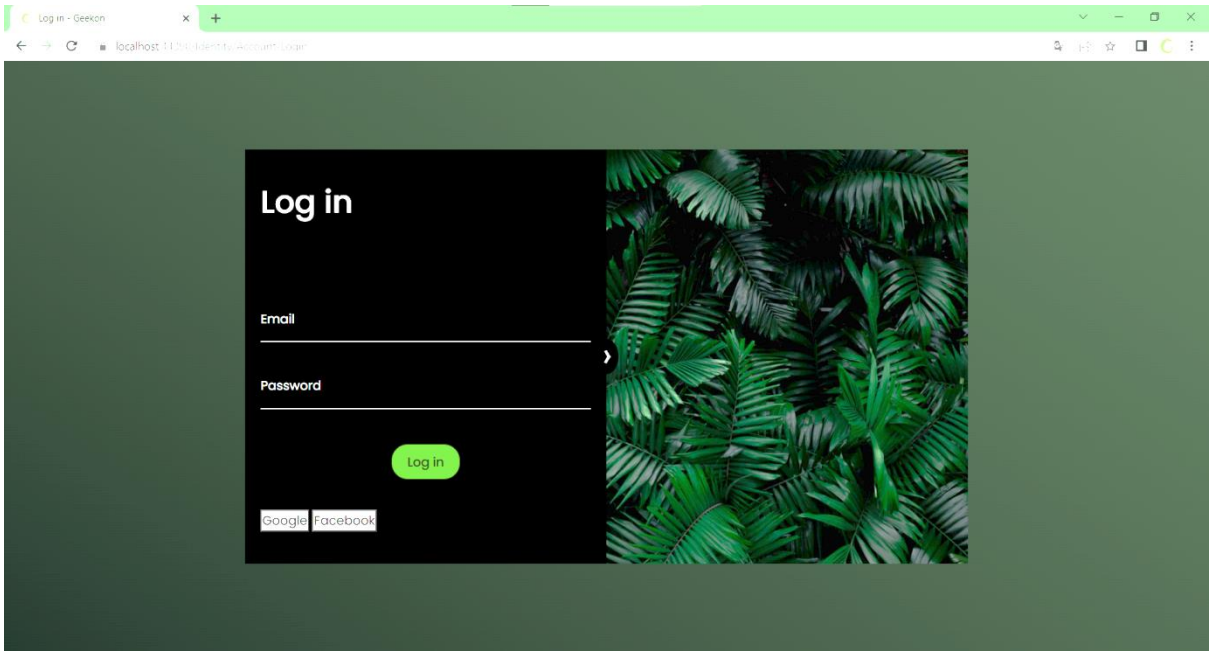


Рисунок 9. Сторінка входу

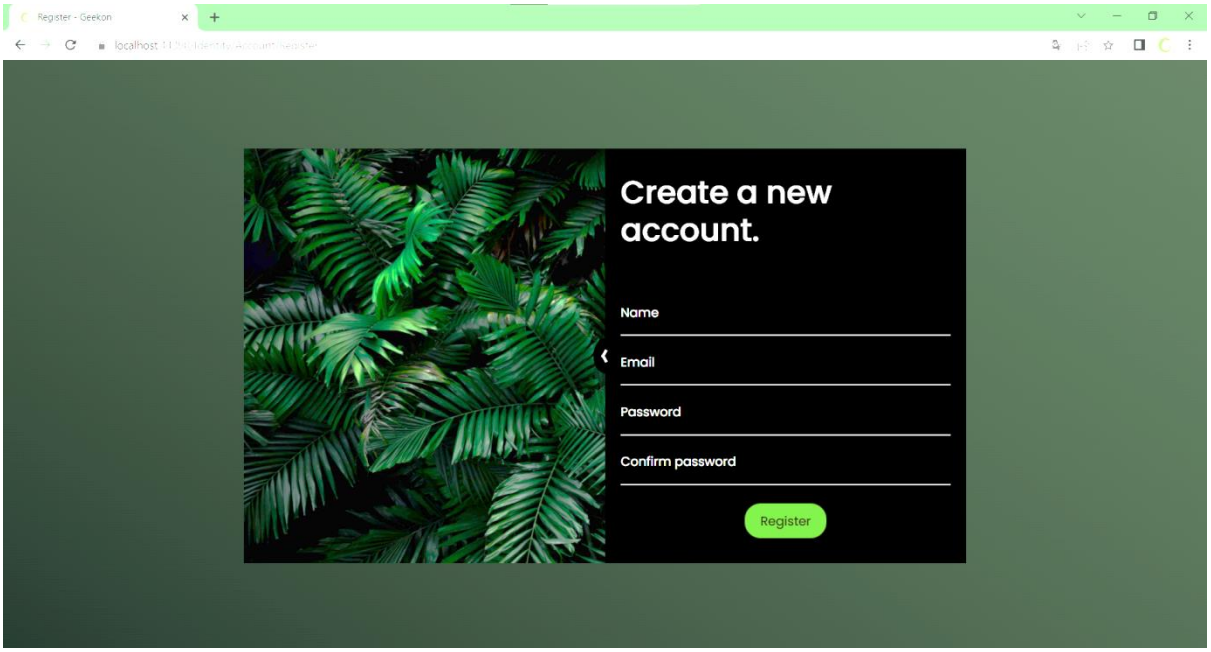


Рисунок 10. Сторінка реєстрації

Після входу користувача головна сторінка містить зліва спливаюче меню. Як відображено на рисунках 11-12.

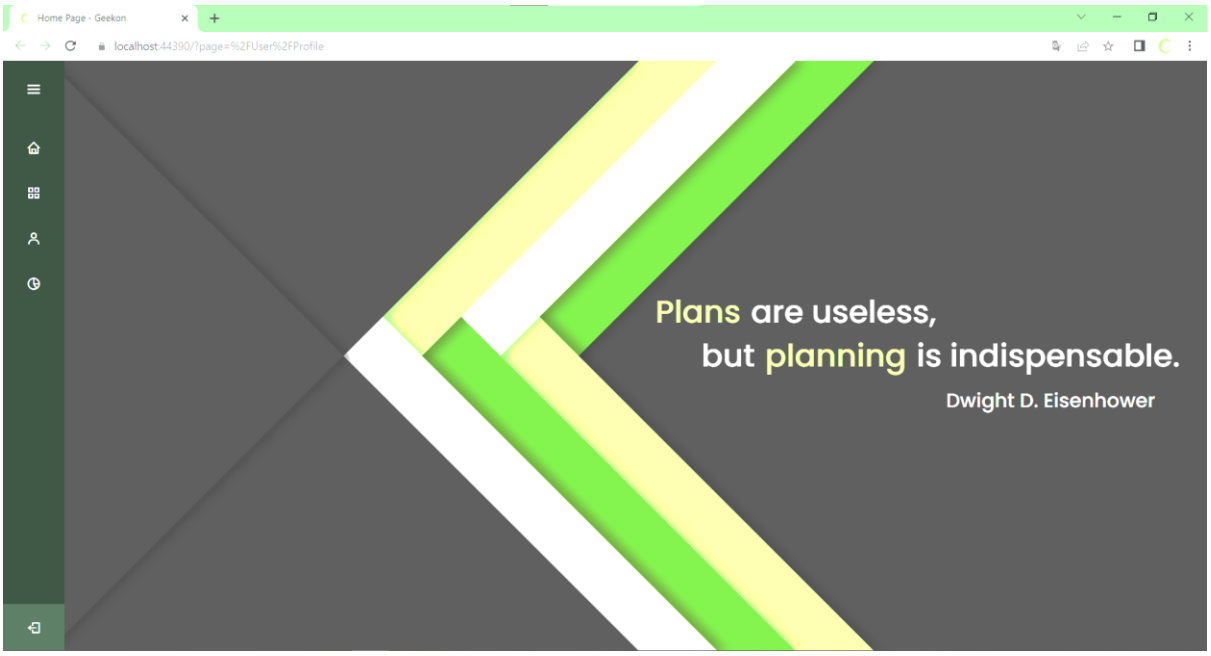


Рисунок 11. Головна сторінка користувача

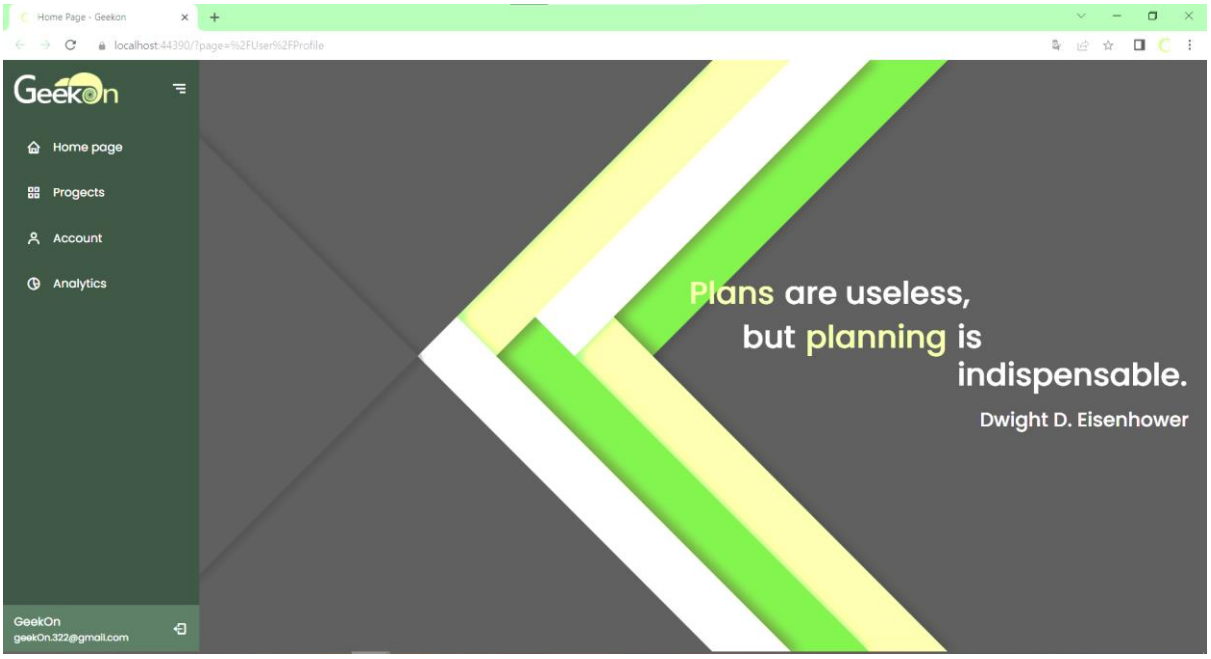


Рисунок 12. Головна сторінка користувача

Обравши в меню "Project", користувач переходить на сторінку з відображенням всіх проектів. А саме його власних, тих в які його додано і архівованих (рисунок 13). Також тут можна додати новий проект. Сторінка створення зображена на рисунку 14, як бачимо можна вибрати з переліку запропонованих зображень фон для проекту. А на рисунку 15 можна побачити початковий вигляд новоствореного проекту.

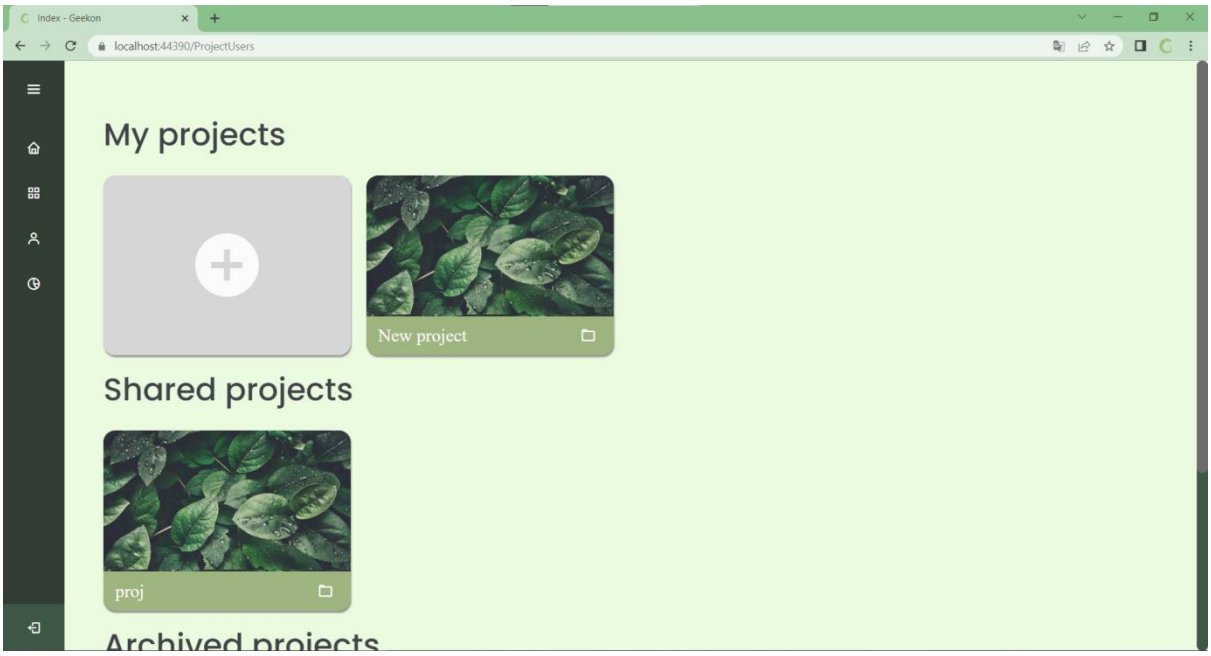


Рисунок 13. Сторінка з проектами

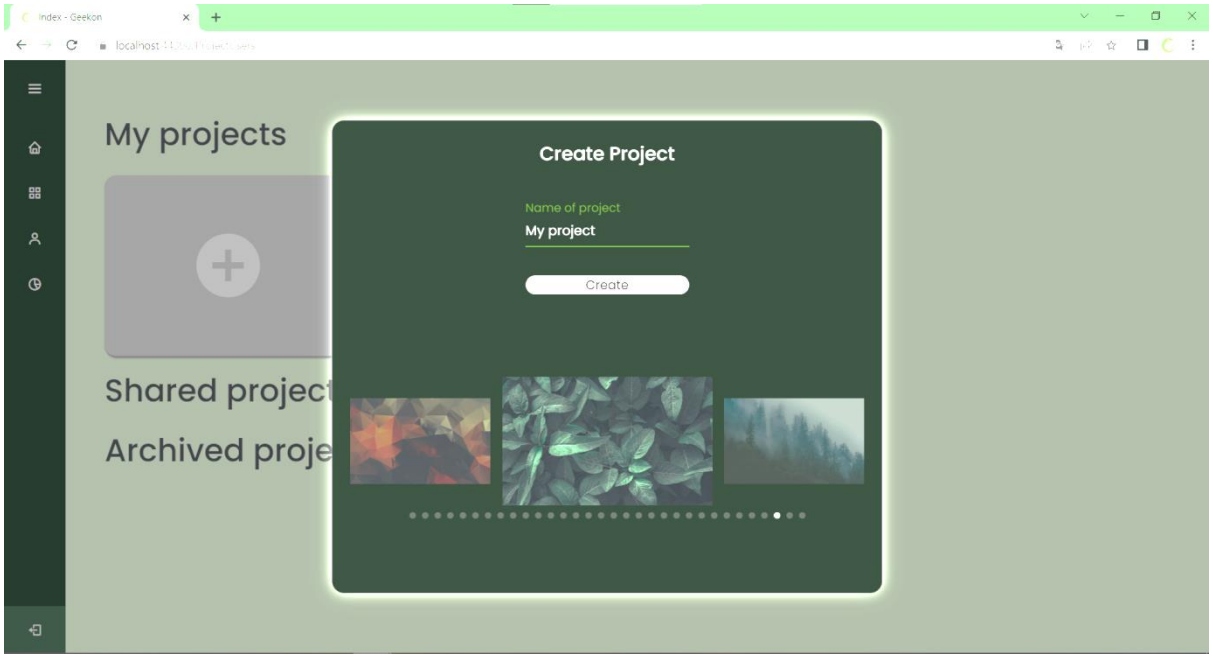


Рисунок 14. Створення нового проекту

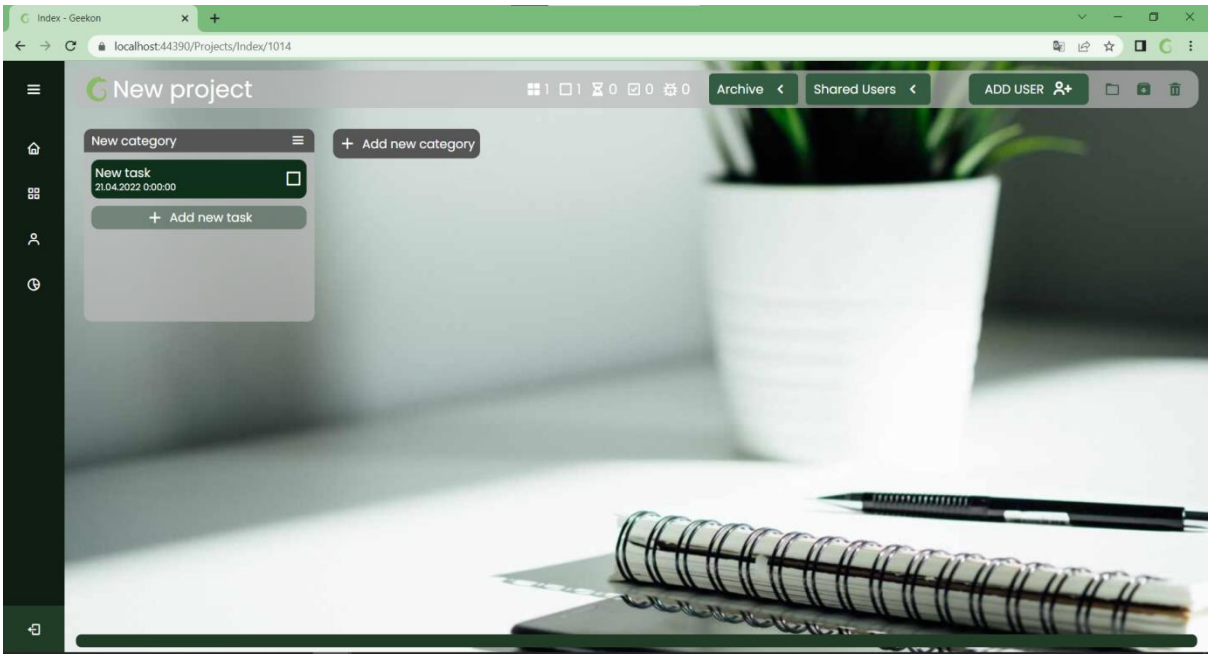


Рисунок 15. Новостворений проект

У проекті є можливість додати нові категорії та завдання, видалити , відредагувати або архівувати(для появи кнопок архівації та видалення необхідно натиснути меню, праворуч від назви категорії). При редагуванні завдання (рисунок 16) можна змінити назву, обрати в календарі дату дедлайну, а також обрати актуальний статус виконання завдання: To do, In Progress, Finished, Bugs, які схематично зображені картинками. На верхній горизонтальній панелі також відображено кількість завдань на проекті по кожному з статусів. Список архівованих завдань та категорій випливає при натисканні на панелі "Archive" (рисунок 17)

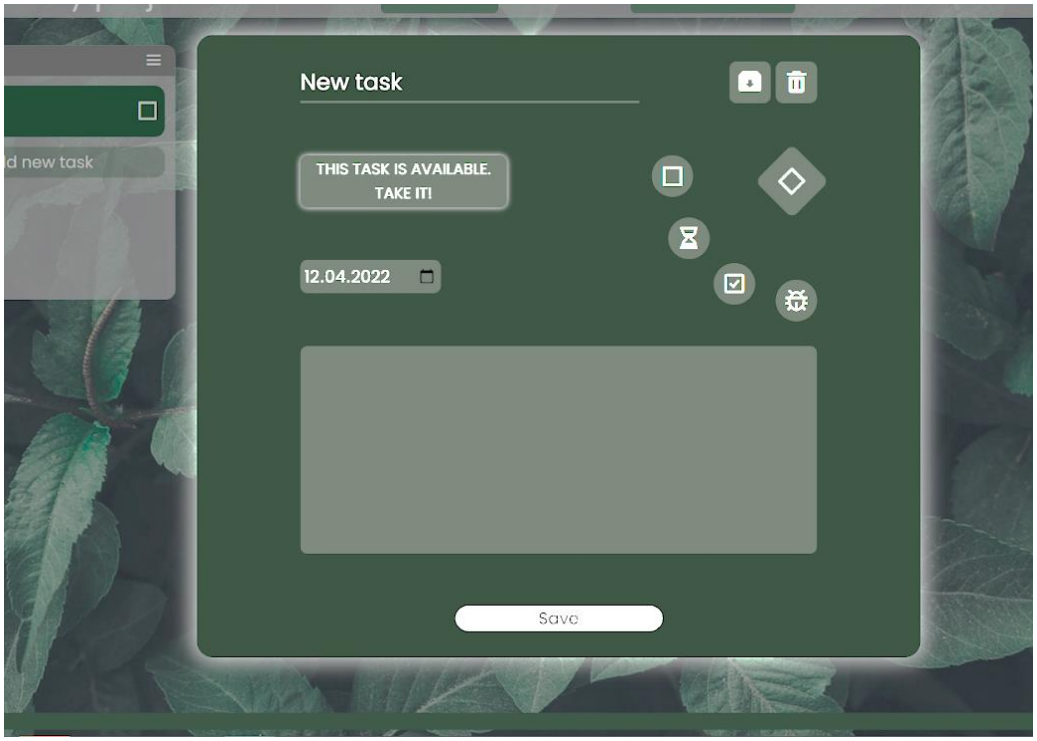


Рисунок 16. Редагування завдання

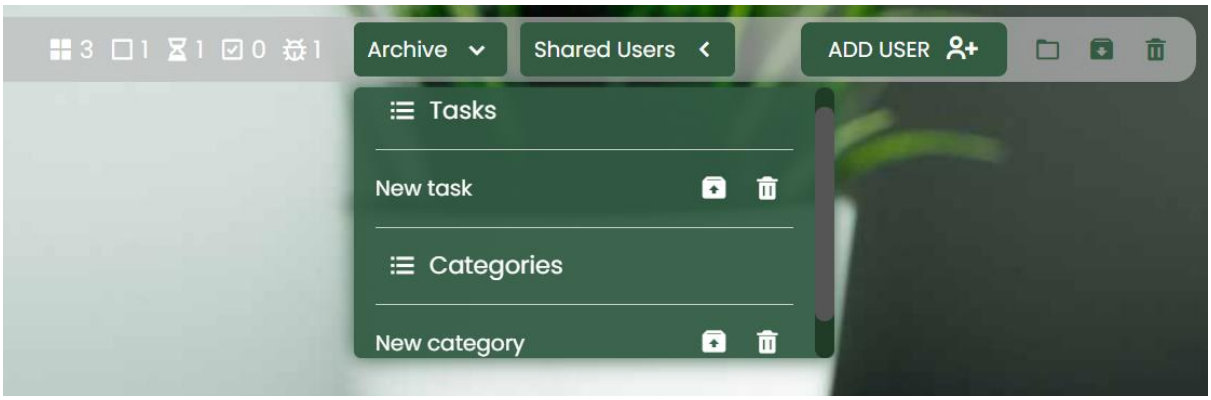


Рисунок 17. Архівовані завдання

Також на верхній панелі маємо поля для додавання нових виконавців на проект (відбувається за електронною поштою користувачів), а також перегляду всіх користувачів, присутніх на даному проекті (рисунок 18)

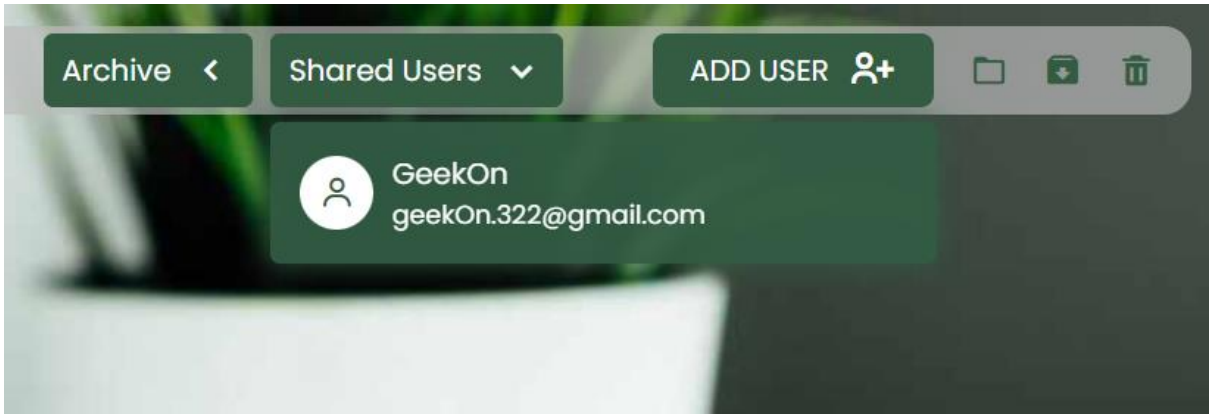


Рисунок 18. Перегляд доданих користувачів

Варто зазначити, що при видаленні будь-яких елементів система запитує підтвердження видалення у спливаючому вікні. В боковому меню присутні також такі розділи як Account та Analytics. В Акаунті користувач може змінювати свої особисті дані, наприклад, свій нікнейм, пошту чи пароль (рисунок 19). А в розділі Аналітика відображено у відсотковому та кількісному співвідношенні завдання користувача в різних можливих статусах. А також відсоткове відображення кількості зданих вчасно та завдань з простроченими дедлайнами(рисунок 20).

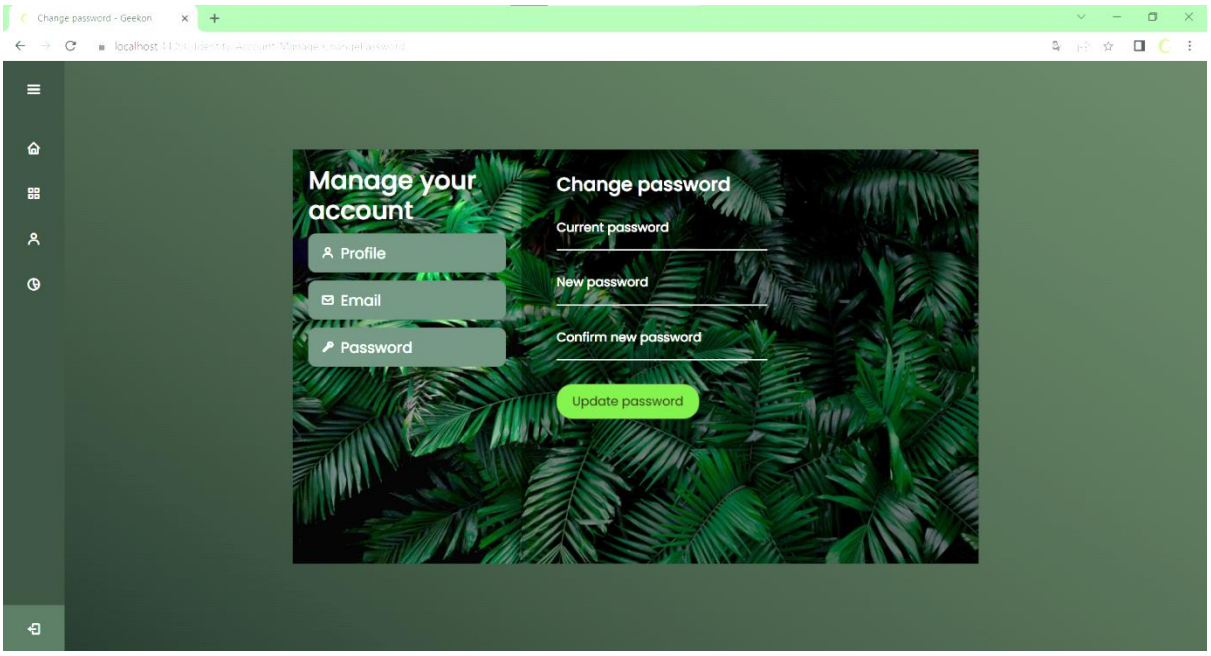


Рисунок 19.Зміна пароля

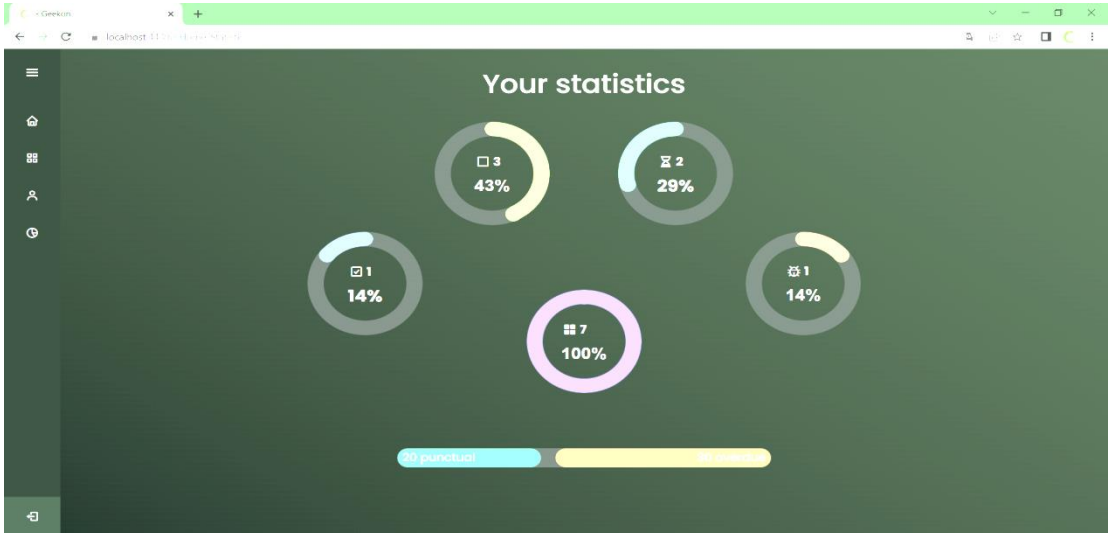


Рисунок 20. Статистика завдань по аканту

Тестування

При розробці та тестуванні системи нами було застосоване модульне тестування — це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Для кожного методу наявних контроллерів був створений окремий метод Unit Test. Всього частково чи повністю тестами було покрито 68,5% методів. Скріни запуску тестування відображено на рисунку 21. Приклад коду тестів для методів контроллера ProjectController прикріплено в Додатку Б.

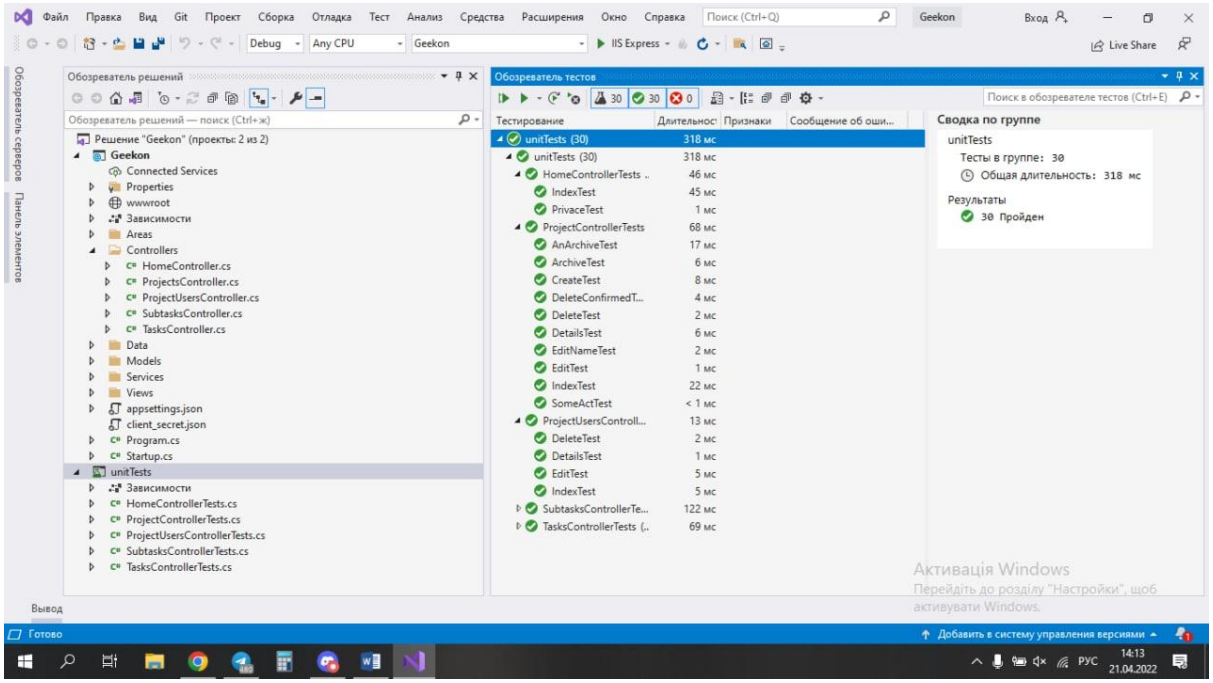


Рисунок 21. Результат запуску тестів

АНАЛІЗ ПЕРЕВАГ ТА НЕДОЛІКІВ СИСТЕМИ "GEEKON"

Проаналізувавши переваги і недоліки систем управління проектами, що представленні на ринку, нами була створена система, яка мала на меті об'єднати найкращі риси різних систем у собі.

На початку роботи з такими системами потрібно пройти реєстрацію. В деяких з систем наявних на ринку від користувача вимагали великої кількості інформації про себе, що не дуже зручно, і це може стати перепоною для використання. Тому реєстрація в нашій системі відбувається за рахунок аккаунта в Google чи Facebook, або ж за допомогою пошти, не вимагаючи інших даних.

Наступний пункт на який ми звернули увагу, – простота у використанні, адже це дуже важливо для приваблення нових користувачів на перших етапах. Ми вирішили зробити наш продукт нативним у використанні для користувача, аби не було потреби у навчанні персоналу для роботи в системі управління проектами "GeekOn".

Наступним важливим пунктом у розробці проекту був цікавий дизайн, адже більшість систем на ринку мають мінімалістичний дизайн і не сприяють бажанню користувача залишатися в системі на довше, або ж поділитись ресурсом з колегами.

Іншим, на наш погляд, важливим моментом у використанні подібних за стосунків є наявність зручної та зрозумілої статистики щодо виконання завдань. Саме тому, була реалізована статистика, яка демонструє нам результати виконання завдань проекту з цікавими рішеннями в області виведення даних.

Проєкт знаходиться в стані продовження розробки. Тому деякі з запланованих функцій не реалізовані (як, наприклад, сортування та фільтрація завдань, інтеграція з іншими системами та сервісами) або реалізовані не в тій мірі, в якій були заплановані. Однією з таких функцій є відстеження часу, який виділений на завдання та відстеження в календарі всіх завдань.

Наступною функцією, що не реалізована на даний момент, є можливість додавання двох або більше виконавців на одне завдання. Наразі таке завдання можна розділити між кількома користувачами тільки шляхом створення декількох різних завдань менших за обсягом. Вирішено реалізувати дану функцію на більш пізніх етапах.

У таблиці 8 тезисно описані основні переваги та недопрацювання системи "GeekOn".

Таблиця 8. Переваги та недоліки системи "GeekOn"

Переваги	**Недоліки**
Нативність у використанні	Відсутнє зручне відстеження часу виділеного на виконання завдання
Зручна реєстрація	Одне завдання для одного користувача і ніяк інакше
Не потребує попереднього навчання персоналу для використання	Не має можливості роботи з зображеннями та файлами
Привітна і компетентна підтримка	
Наявна статистика виконання завдань проекту	
Цікавий та зручний дизайн, що не відволікає від роботи	

ВИСНОВКИ

В результаті виконання даного навчального проекту ми, перш за все, здобули чи поглибили навички та досвід роботи в команді. Здобули чимало нових знань у сфері менеджменту та управління IT-проектами. Розробили власну систему для управління проектами "GeekOn".

"GeekOn" – веб-платформа керування роботою, створена, щоб допомогти командам організовувати, відстежувати й керувати проектами та процесами. Підвищення продуктивності команди неможливе без продуктивного інструмента. "GeekOn" –- це універсальний і безкоштовний таск-менеджер. Сайт однозначно допоможе не загубитися у стрімкому потоці нових завдань, збирати в одному місці всі необхідні для них файли та контролювати прогрес. Окремо варто відмітити вдало підібрану стилістику та дизайн сайту, а також достатньо зрозумілий для початківців інтерфейс.

Хоча платформа створювалася як таск-менеджер для команд, її зручно використовувати і як персональний планер. Всі необхідні функції для цього є.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке управління проектами: як здійснюється і що в себе включає?[Електронний ресурс] – Режим доступу до ресурсу: https://blog.agrokebety.com/shcho-take-upravlinnya-proektamy
2. ClickUp [Електронний ресурс] – Режим доступу до ресурсу: https://clickup.com/
3. monday.com [Електронний ресурс] – Режим доступу до ресурсу: https://monday.com/lang/ru/
4. Asana [Електронний ресурс] – Режим доступу до ресурсу: https://asana.com
5. Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/uk
6. Jira [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/software/jira
7. W3schools[Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/css.
8. Boxicons [Електронний ресурс] – Режим доступу до ресурсу https://boxicons.com/

9. Itzik Ben-Gan "Microsoft SQL Server 2012 T-SQL Fundamentals"
10. Swiper. Офіційна документація [Електронний ресурс] – Режим доступу до ресурсу: https://www.swiper.com.cn/api/start/new.html

Додаток А

Посилання на репозиторій проекту на GitHub:
https://github.com/KaplenkoOleksandr/GeekOn

Додаток Б

Код Unit тестів для методів контроллера ProjectController:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xunit;
using Geekon.Controllers;
using Geekon.Models;
using Microsoft.AspNetCore.Mvc;
namespace unitTests
{
    public class ProjectControllerTests
    {
        [Fact]
        public void SomeActTest()
        {
            ProjectsController PC = new ProjectsController();
            Assert.IsType<PartialViewResult>(PC.SomeAct(null));
            return;
        }
        [Fact]
        public void IndexTest()
        {
            ProjectsController PC = new ProjectsController();
            Assert.IsType<Task<IActionResult>>(PC.Index(null));
            Assert.IsType<Task<IActionResult>>(PC.Index(-1));
            Assert.IsType<Task<IActionResult>>(PC.Index(1));
            return;
        }
        [Fact]
        public void DetailsTest()
        {
            ProjectsController PC = new ProjectsController();
            Assert.IsType<Task<IActionResult>>(PC.Details(null));
            Assert.IsType<Task<IActionResult>>(PC.Details(-1));
            Assert.IsType<Task<IActionResult>>(PC.Details(1));
            return;
        }
        [Fact]
        public void CreateTest()
        {
            ProjectsController PC = new ProjectsController();
            Projects P = new Projects();
            Assert.IsType<Task<IActionResult>>(PC.Create(P));
            return;
        }
        [Fact]
        public void EditTest()
        {
            ProjectsController PC = new ProjectsController();
```

```
        int? id = null;
        Assert.IsType<Task<IActionResult>>(PC.Edit(id));
        Assert.IsType<Task<IActionResult>>(PC.Edit(-1));
        Assert.IsType<Task<IActionResult>>(PC.Edit(0));
        return;
    }
    [Fact]
    public void EditNameTest()
    {
        ProjectsController PC = new ProjectsController();
        Assert.IsType<Task<IActionResult>>(PC.EditName(0, "newName"));
        return;
    }
    [Fact]
    public void ArchiveTest()
    {
        ProjectsController PC = new ProjectsController();
        Assert.IsType<Task<IActionResult>>(PC.Archive(0));
        Assert.IsType<Task<IActionResult>>(PC.Archive(-1));
        return;
    }
    [Fact]
    public void AnArchiveTest()
    {
        ProjectsController PC = new ProjectsController();
        Assert.IsType<Task<IActionResult>>(PC.AnArchive(0));
        Assert.IsType<Task<IActionResult>>(PC.AnArchive(-1));
        return;
    }
    [Fact]
    public void DeleteTest()
    {
        ProjectsController PC = new ProjectsController();
        Assert.IsType<Task<IActionResult>>(PC.Delete(null));
        Assert.IsType<Task<IActionResult>>(PC.Delete(0));
        Assert.IsType<Task<IActionResult>>(PC.Delete(-1));
      return;
    }
    [Fact]
    public void DeleteConfirmedTest()
    {
        ProjectsController PC = new ProjectsController();
        Assert.IsType<Task<IActionResult>>(PC.DeleteConfirmed(0));
        Assert.IsType<Task<IActionResult>>(PC.DeleteConfirmed(-1));
        return;
    }
}
                                        }
```

ПРОЄКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ

Єлизавета Валтеріс, Сергій Микитчин, Ілля Дубінін, Юрій Бєліков, Юлія Соломаха, Юлія Недавня

ПОСТАНОВКА ЗАДАЧІ

Команді розробників було поставлено задачу: розробити систему управління ІТ-проєктами (надалі Система). Під час онлайн-зустрічей із викладачами (надалі Замовник) та розробниками було уточнено тематику, вимоги та очікуваний для користувача функціонал Системи. Програмний продукт розроблено як повноцінний засіб управління проєктами для менеджерів та розробників і комунікації між ними.

Можливості системи описано за допомогою UML (Додатки 1, 2).

Актуальність роботи. На даний момент пандемія популяризувала віддалену роботу в багатьох проєктних командах. З'являється тренд створення розподілених команд інженерів, члени якої можуть бути не лише розділені фізичною відстанню, а і годинними поясами. Таким чином, зручна, швидка, інтуїтивно зрозуміла система взаємодії між шаром розробників та менеджменту значно сприяла би покращенню роботи таких команд.

Користувач / цільова група. Цільовою аудиторією є користувачі Системи. Цільова аудиторія представлена такими групами користувачів, які задіяні у життєвому циклі ІТ-проєкту:

- менеджери, які керують проєктами, створюють задачі та синхронізуються з розробниками з приводу їх статусу та проблем, що виникають під час виконання;
- члени команди розробників, які виконують поставлені задачі, створюючи певний програмний продукт для кінцевого клієнта.

Короткий опис програмного продукту. Система надає можливість для командної взаємодії при розробці ІТ-проєктів. Зареєстровані користувачі Системи можуть мати одну з наступних двох ролей: Manager, Employee. В залежності від ролі, обраної під час реєстрації, користувачу надаються відповідні права.

Manager може створювати/переглядати/редагувати/видаляти проєкти, де він є керівником, приєднуватись до проєктів та створювати нові завдання для своїх співробітників (Employee). Employee може переглядати та приєднуватись до проєктів, переглядати деталі завдань. Обидві категорії користувачів можуть приєднуватись до проєктів шляхом використання унікальних посилань, які вони можуть отримати від своїх колег або користувача Manager, який є керівником потрібного проєкту.

Незареєстрований користувач системи не має доступу до перегляду та будь якої взаємодії з проєктами та завданнями.

ОРГАНІЗАЦІЯ РОБОТИ РОБОТИ В КОМАНДІ

Учасники команди та розподіл ролей. Команда складається з 6 інженерів. Студенти 4-го курсу: Єлизавета Валтеріс, Сергій Микитчин, Ілля Дубінін, Юрій Бєліков. Студенти 2-го курсу: Юлія Соломаха, Юлія Недавня. Розподіл ролей відбувався з урахуванням власних побажань. Ознайомитись із зонами відповідальності кожного члена команди можна за допомогою таблиці 1.

Варто відмітити, що наведений вище розподіл є більше формальним, оскільки в реальному процесі виконання завдань та створення продукту ролі розробників дуже часто змінювались. Кожен мав змогу спробувати себе на різній посаді. Зрозуміло, що це не зовсім відповідає процесу розробки на реальних проєктах, проте таким чином студенти отримали нові цікаві знання та досвід.

Таблиця 1. Зони відповідальності інженерів команди

Учасник	Роль учасника
Єлизавета Валтеріс	Модератор, керівник проєкту
Сергій Микитчин	Бекенд розробник
Ілля Дубінін	Тестувальник, фахівець з документації
Юрій Бєліков	Фахівець з документації, аналітик
Юлія Соломаха	Фронтенд розробник
Юлія Недавня	Фронтенд розробник

Під час виконання задач часто траплялось, що один з учасників команди допомагав іншим у вирішенні питань, що виникли у процесі розробки.

АНАЛІЗ ІСНУЮЧИХ НА РИНКУ РІШЕНЬ

Командою було проаналізовано наступні конкурентні рішення:

- **Google Tasks.** Сервіс для менеджменту задач з мінімальною функціональністю. Орієнтований на особисте користування. Взаємодія з Gmail. У сервісі можна призначати час і дату для завдань, робити списки. Неможливо додати детальний опис завдання, приєднати посилання чи файл. Інтеграція з сторонніми сервісами відсутня.
- **Microsoft To-Do.** Орієнтований на особисте управління справами. У ньому можна створювати списки справ і завдань, групувати їх, розбивати на підзадачі. Є інтеграція з Microsoft Outlook.
- **Wrike.** Орієнтований на управління справами та проєктами в командах, можливе використання і в особистих цілях. У безкоштовній версії можуть працювати не більше 5 користувачів. Можлива інтеграція з хмарними сервісами. У платній версії доступна більшість кількість сервісів, інтеграція більшої кількості користувачів, розділення користувачів по рівню доступу.
- **Focuster.** Можливе створення завдань та їх списків. Підтримується групування завдань. Інтеграція з календарями і Trello.
- **Pyrus.** Надає можливість менеджменту завдань і проєктів командою, може бути використаний і для особистих цілей. Безкоштовна версія підтримує необмежену кількість користувачів, але не більше 100 завдань. Сервіс дозволяє комунікацію користувачів. У платній версії доступні необмежена кількість завдань та більша кількість ГБ у хмарі. Є інтеграція з OneDrive, DropBox, Google диск.
- **Basecamp.** Призначений для управління завданнями і організації командної роботи. Є лише платна версія (99$). Надає можливість створювати списки справ, розклад, комунікувати зі співробітниками. Відсутня інтеграція з іншими сервісами.
- **Worktek.** Орієнтований на підвищення особистої і командної ефективності. Можлива робота з інформаційними потоками, завданнями. Доступна тільки платна версія.
- **Trello.** Канбан-дошка зі списками карток, яку можна використовувати для управління завданнями. В картках можна комунікувати з іншими користувачами. У платній версії наявні додаткові засоби для командної роботи та більша кількість сервісів для інтеграції.[1]

Висновки стосовно кожного продукту можна підсумувати у вигляді таблиці 2.

Таблиця 2. Аналіз конкурентів на ринку

Продукт	Переваги	Недоліки
Google Tasks	безкоштовний; доступна інтеграція з Gmail	орієнтація на одного користувача; неможливо додати опис завдання
Microsoft To-Do	безкоштовний; можливість групувати завдання; інтеграція з Outlook	орієнтація на одного користувача
Wrike	орієнтований на командну роботу; інтеграція з хмарними сервісами	безкоштовна версія дозволяє працювати максимум 5 користувачам; розподілення користувачів по рівню доступу можливе тільки у платній версії
Focuster	групування завдань; інтеграція з іншими сервісами	орієнтація на одного користувача
Pyrus	орієнтований на командну роботу; необмежена кількість користувачів у безкоштовній програмі; комунікація між співробітниками; інтеграція з хмарними сервісами	кількість завдань у безкоштовній версії обмежена 100
Basecamp	орієнтований на командну роботу; можна створювати списки завдань; комунікація між співробітниками	є тільки платна версія
Worktek	орієнтований на підвищення ефективності	є тільки платна версія
Trello	підходить і для командної і для особистої роботи; комунікація між користувачами в картках	більше підходить для Scrum, Kanban; не орієнтований для роботи за стандартними підходами; безкоштовна версія містить обмеження на інтеграцію з іншими технологіями.

МЕТОДОЛОГІЯ ТА СИСТЕМА УПРАВЛІННЯ ПРОЄКТОМ

Для контролю та управління процесом розробки Системи було обрано методологію Kanban. В якості системи перегляду задач, їх виконавців, дедлайнів та статусів було обрано Trello. Було визначено наступні можливі стани виконання задачі (рис. 1):

- Backlog – задача поставлена;
- To-Do – задача поставлена, визначено відповідального за її виконання інженера(-ів);
- Doing – інженер(-и) почали виконання задачі;
- Code Review – відкрито Pull Request в основну гілку розробки, рішення очікує проходження оцінки якості з боку інших учасників команди;
- Done – робота над задачею завершена.

На початку роботи проводились зустрічі раз у тиждень разом із Замовником для обговорення статусів задач, проблем та ідей їх вирішення. За необхідністю комунікації між всією командою або окремими інженерами здійснюються в будь-який момент часу в разі виникнення проблем, що потребують термінового вирішення так і за наявності нагальних питань від будь-кого з членів команди. Комунікація команди відбувалася за допомогою Zoom, Google Meet, Google Docs. Для постійного зв'язку був створений чат в Telegram.

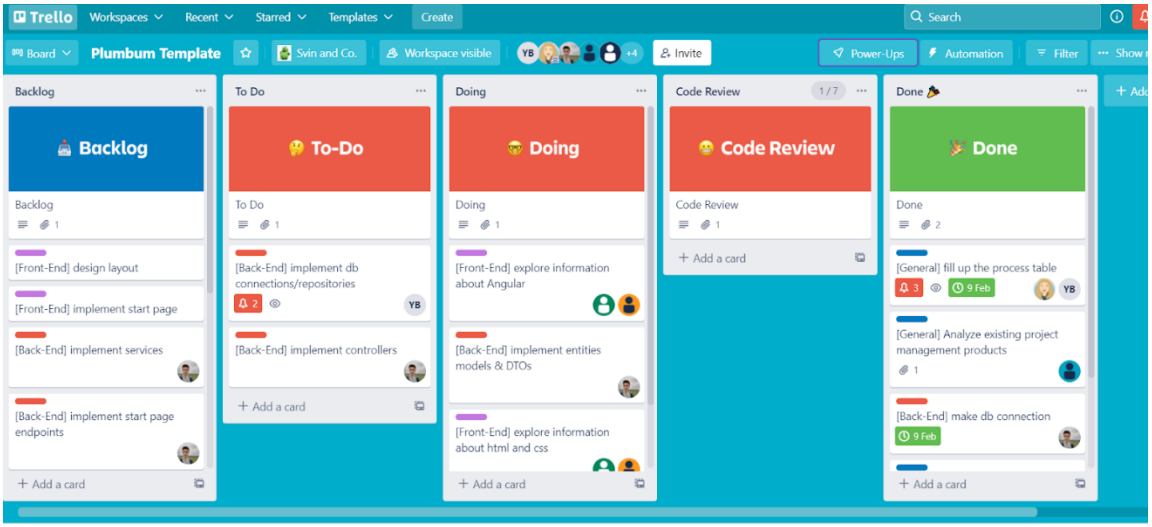


Рисунок 1. Фрагмент дошки проєкту в Trello

Також за допомогою Telegram проводилися Code Review. Один з членів команди надсилав виконану частину роботи, а інший перевіряв її і надсилав

Code Review .txt файлом. Це дуже допомагало покращити якість продукту, а також сприяло кращому розумінню матеріалу і обміну досвідом та знаннями. Зокрема, подальша оптимізація є невід'ємною частиною розвитку [8] нашої Системи.

Приклади Code Review можна побачити на рисунках 2 і 3.

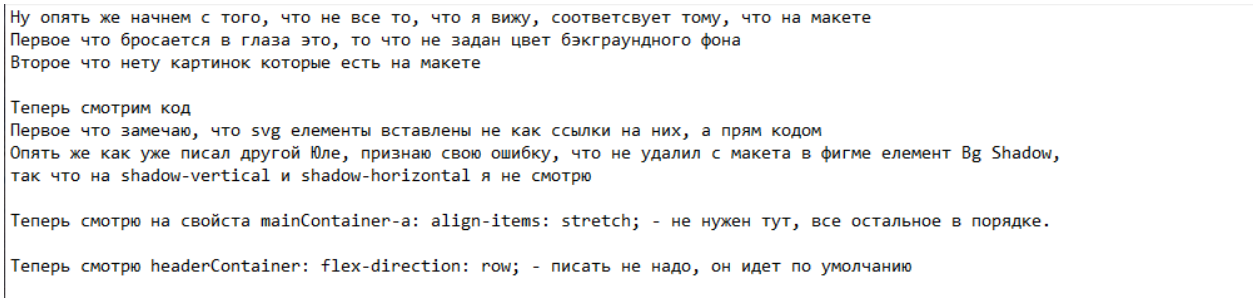


Рисунок 2. Фрагмент Code Review коду Юлії Соломахи Сергієм Микитчиним

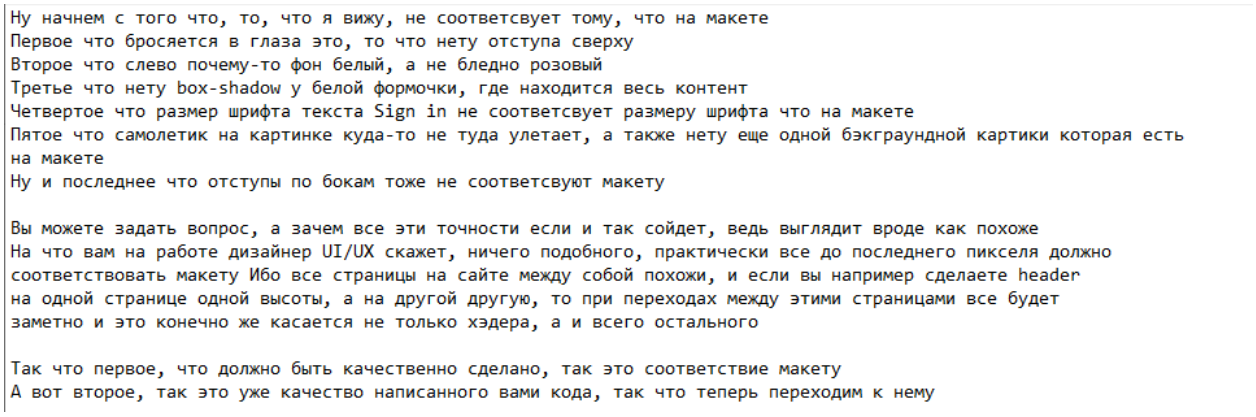


Рисунок 3. Фрагмент Code Review коду Юлії Недавньої Сергієм Микитчиним

ТЕХНІЧНЕ ЗАВДАННЯ

Абревіатури, скорочення. 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 (табл.3).

Таблиця 3. Скорочення використані в проєктній документації

№ з/п	Термін	Скорочення, абревіатури	Коментар
1	Зареєстрований /незареєстрований користувач	Користувач	-
2	Програмний продукт	Система	-
3	Система керування базами даних	СКБД	-
4	База даних	БД	-
5	CRUD-операції	CRUD	4 базові функції управління даними: "створення, зчитування, зміна і видалення"

Мета і актуальність проєкту. Система управління IT-проєктами "Plumbum Kolab" (далі – програмний продукт) розробляється як комплексне рішення для організації керування проєктами менеджерською ланкою та ефективної комунікації із командою

розробників. Актуальність проєкту в першу чергу полягає у необхідності створення умов швидкої комунікації і ефективної роботи проєктних команд у розподіленому віддаленому форматі роботи.

Інструментарій розробки. Бекенд Системи розробляється на мові програмування JavaScript. Використано технології Express, Node.js. Як СКБД обрано PostgreSQL. Розробка проводиться у IDE VS Code.

Для деплойменту використано платформу Heroku.

Фронтенд частина проєкту розроблена з використанням HTML/CSS, JavaScript, React.

Для розробки тестування фронтенд частини використовувалася бібліотека Cypress.

Мета створення Системи. Система централізованого адміністрування IT проєктами та доступу до деталізованої інформації про них, як для представників проєктного менеджменту так і розробників. Продукт, який забезпечує можливості письмової комунікації між менеджерами і розробниками в рамках певного проєкту стосовно окремо взятих задач.

Цілі і переваги користувача. Система повинна надавати наступні можливості для користувача:

- можливість доступу до проєктної інформації, розміщеної в Системі;
- можливість створювати нові проєкти та задачі в рамках конкретного обраного проєкту (для менеджера);
- можливість поширити доступ до проєкту та інформації, що стосується його деталей за допомогою унікальних посилань
- можливість видалити проєкт або конкретне завдання в його рамках
- можливість лишати коментарі стосовно конкретного завдання в рамках проєкту
- можливість редагувати інформацію про деталі проєкту або завдань
- можливість редагувати власний профіль у Системі

Структура та опис Системи. Use-case діаграми системи наведено в Додатку 1 та Додатку 2.

Мова Системи. Основна мова Системи − англійська

Ролі користувачів Системи. У Системі користувач може мати одну із наступних ролей:

У Системі користувач може мати одну із наступних ролей:

- Manager
- Employee

Роль Manager має можливість:

- зареєструватись в Системі;
- увійти у свій обліковий запис в Системі;
- вийти із облікового запису в Системі;
- переглянути веб-сторінку із проєктами якими він керує;
- відредагувати інформацію про певний проєкт;
- видалити проєкт;
- доєднатися до проєкту за допомогою посилання;
- створити проєкт;
- заповнити коротку форму із основними відомостями про проєкт;
- проглянути деталі конкретного проєкту;
- створити завдання;
- переглянути детальну інформацію про завдання;
- надіслати повідомлення в рамках відкритого завдання;
- скопіювати посилання на проєкт для подальшого розповсюдження;

- переглянути власний профіль;
- відредагувати власний профіль у Системі.

Роль Employee має можливість:

- зареєструватися в Системі; у
- війти у власний обліковий запис в Системі;
- вийти із облікового запису;
- переглянути веб-сторінку із проєктами, в яких він приймає участь;
- приєднатися до проєкту за допомогою унікального посилання;
- скопіювати унікальне посилання на конкретний проєкт;
- проглянути веб-сторінку із деталями обраного проєкту;
- проглянути деталі завдання в рамках проєкту;
- надіслати повідомлення, що стосується обраного завдання;
- переглянути свій профіль користувача;
- відредагувати свій профіль користувача.

Зареєстрований користувач має доступ до функціоналу який відповідає обраною при реєстрації роллю. Неавторизовані користувачі не мають можливості перегляди списки проєктів, проєктних завдань, лишати коментарі до них, тощо.

Вимоги до стилістичного оформлення Системи. Візуально система має співпадати з макетом, який створено за допомогою Figma та погоджений із Замовниками. Макет можна знайти за посиланням.

Дизайн повинен бути адаптивним: розміри та взаємне розташування елементів сторінки динамічно змінюються в залежності від розміру вікна браузера або розміру смартфону користувача Системи.

Системні вимоги. Система має коректно працювати та відображати дані в наступних браузерах:

- Mozilla Firefox (версія 98 і новіше)
- Google Chrome (версія 99 і новіше)
- Safari (версія 14.1 і новіше)

ІНСТРУМЕНТАРІЙ РОЗРОБКИ

Для розробки бек-енду було обрано мову програмування JavaScript. Це мова, що є дуже популярною у наш час. В більшості випадків застосовується для створення веб-застосунків. Для неї розроблено багато фреймворків та розширень, деякі з них використано і в нашому проєкті [4].

В процесі розробки було використано такі технології: Node.js, Express. Node.js – програмна платформа, що надає можливості мови загального призначення JavaScript. Вона додає можливість JavaScript взаємодіяти з пристроями вводу-виведення через власний API, розроблений на C++. Також можна підключати різноманітні зовнішні бібліотеки, написані різними мовами, звертаючись до них з JavaScript-коду. Дана платформа здебільшого використовується на сервері, як веб-сервер. Ця функціональність буде використана і в нашому випадку. Express – фреймворк веб-застосунків, що використовується на платформі Node.js, реалізований як вільне і відкрите програмне забезпечення. Спроєктований для створення веб-застосунків і API. Можна сказати, що Express є стандартним каркасом для Node.js. Часто використовується в парі з React, де останній є фронт-енд фреймворком. У обраних технологій створено детальну документацію, яка полегшує процес розробки їх користувачам.[5]

Як СКБД обрано PostgreSQL. Це – вільна об’єктно-реляційна СКБД. Наявні її реалізації для багатьох платформ. PostgreSQL базується на SQL та підтримує більшість її стандартів. До її сильних сторін належать:

- високоефективні і надійні механізми транзакцій і реплікації;
- розширювана система вбудованих мов програмування;
- наслідування;
- можливість індексування геометричних об’єктів та ГІС;
- підтримка слабоіндексованих даних в форматі JSON;
- розширюваність [2].

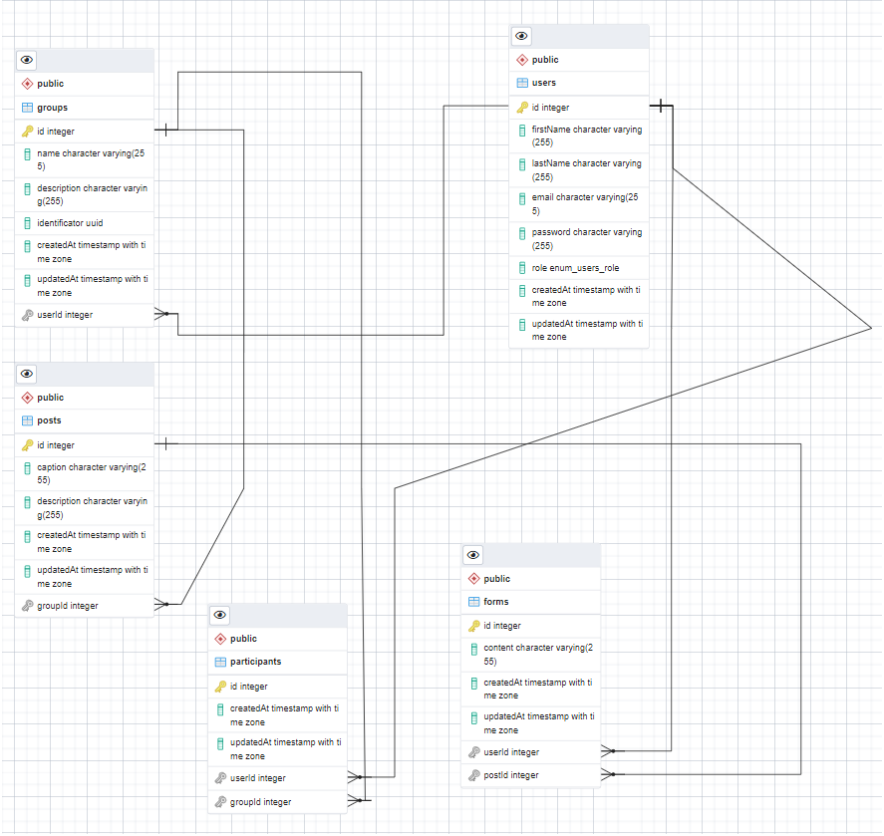


Рисунок 4. Діаграма спроєктованої БД

Розробка проводилась у IDE VS Code. Вона створена Microsoft та є зручним засобом для створення додатків. Є багато можливостей кастомізації середовища для зручності розробників.

Сервер був завантажений на платформу Heroku.

Фронт-енд частина застосунку розроблена з використанням таких технологій: HTML/CSS, JavaScript, React.

HTML – мова розмітки, яка разом з CSS відповідає за позиціонування та стилі елементів на сторінці.

React – безкоштовна фронт-енд бібліотека заснована на JavaScript. Застосовується для побудови інтерфейсів користувача. Підтримується Facebook та спільнотою індивідуальних розробників. Мета бібліотеки – надати високу швидкість, легкість та масштабованість до процесу розробки.

Зазначимо, що також було розроблено початкову версію застосунку на іншому інструментарії. Початково було вибрано технології C#, .NET Framework для бек-енду, MS SQL Server – СКБД і фреймворк Angular для фронт-енду. Проте команда розробників вирішила перейти на вищеописані засоби розробки і створила систему, що відповідає поставленим вимогам.

Також були розроблені тести для фронденд частини з використанням бібліотеки Cypress.

Cypress – це JavaScript-фреймворк для наскрізного тестування, що полегшує налаштування, написання, запуск та налагодження тестів у браузері.

ІНСТРУКЦІЯ КОРИСТУВАЧА

Користувач може перейти за посиланням https://kolab-frontend.herokuapp.com і почати користуватись розробленим продуктом.

На домашній сторінці можна побачити привітання та кнопки для логіну та реєстрації (Sign In і Sign Up відповідно) (рис. 5).

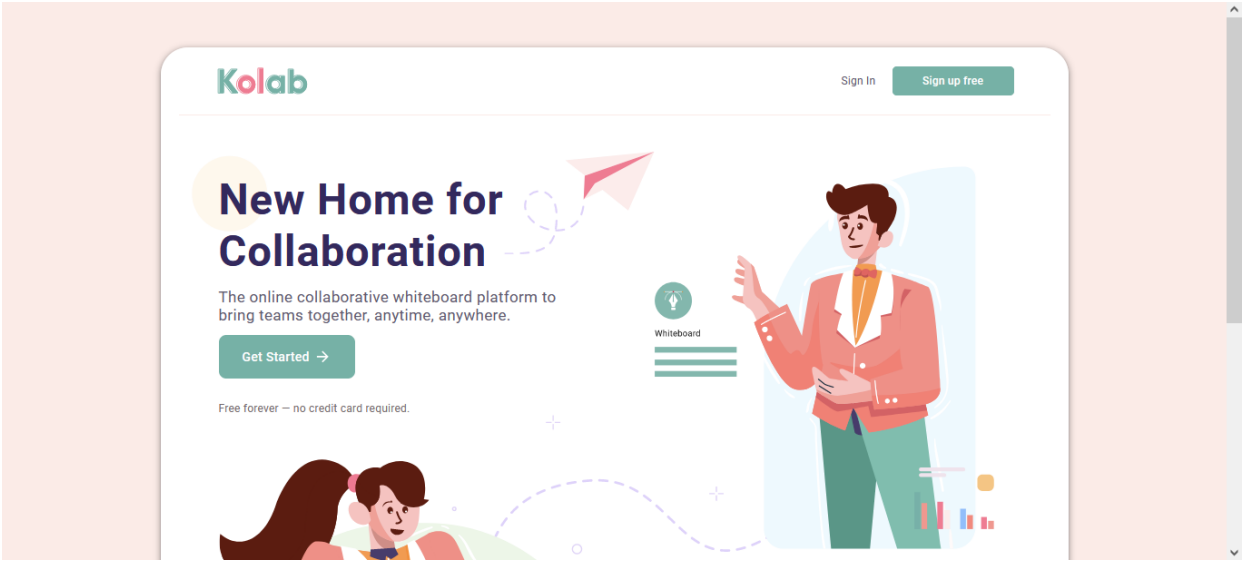


Рисунок 5. Домашня сторінка

На сторінці реєстрації користувачу потрібно ввести своє ім’я і прізвище, email, пароль з підтвердженням та обрати роль.(рис. 6). Після реєстрації користувач має залогінитись для входу в систему.

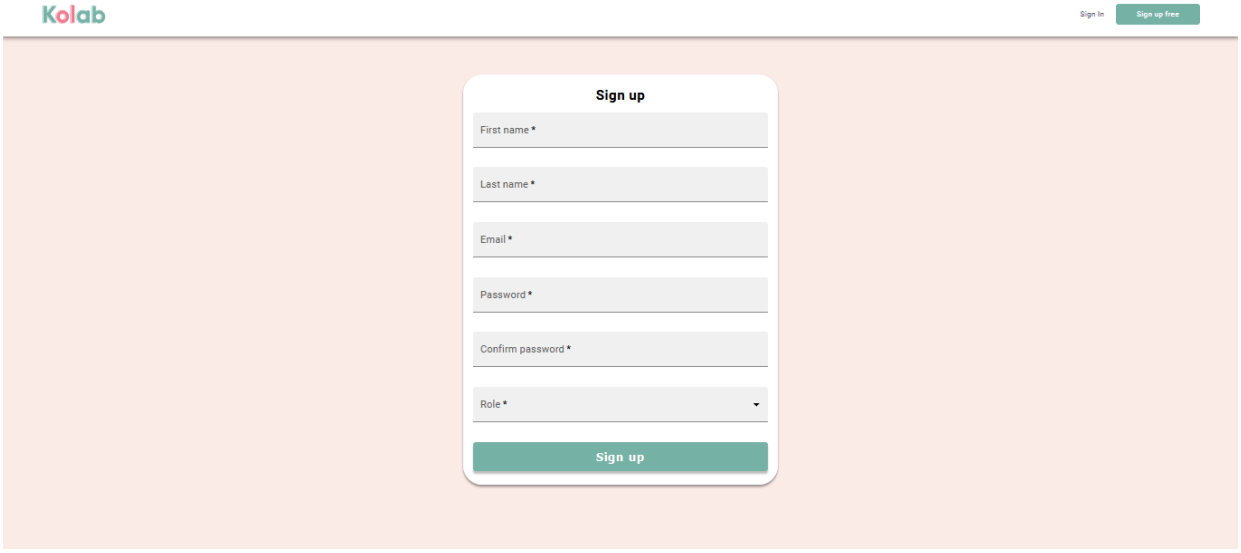


Рисунок 6. Сторінка реєстрації

На сторінці логіну потрібно ввести свою пошту та пароль (рис. 7).

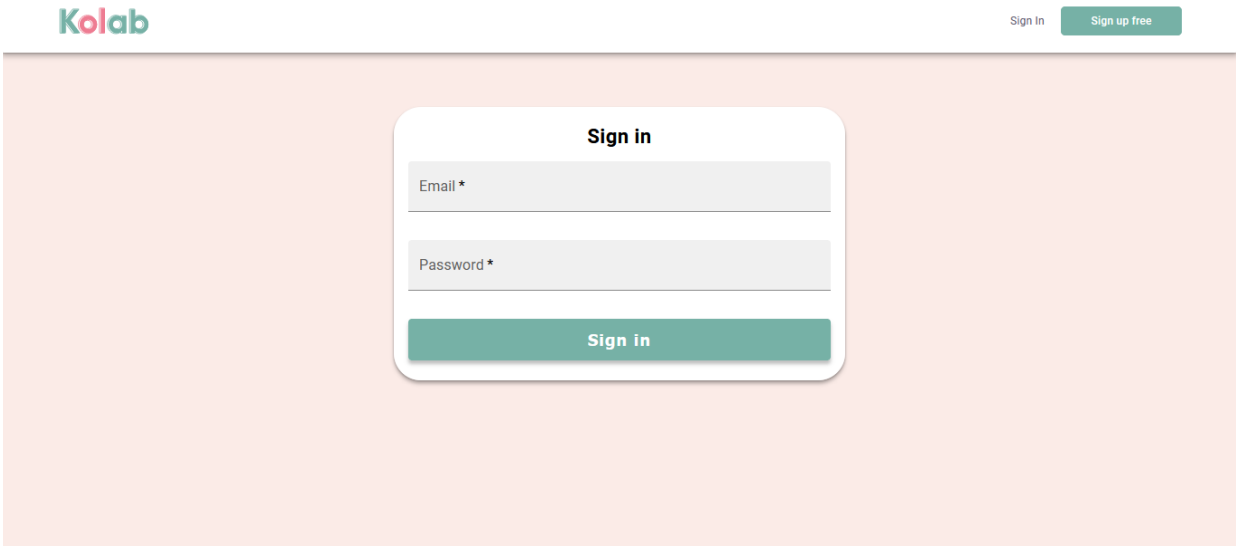


Рисунок 7. Сторінка логіну

Після входження у акаунт, користувачу стає доступний список його проєктів (див. рис. 8).

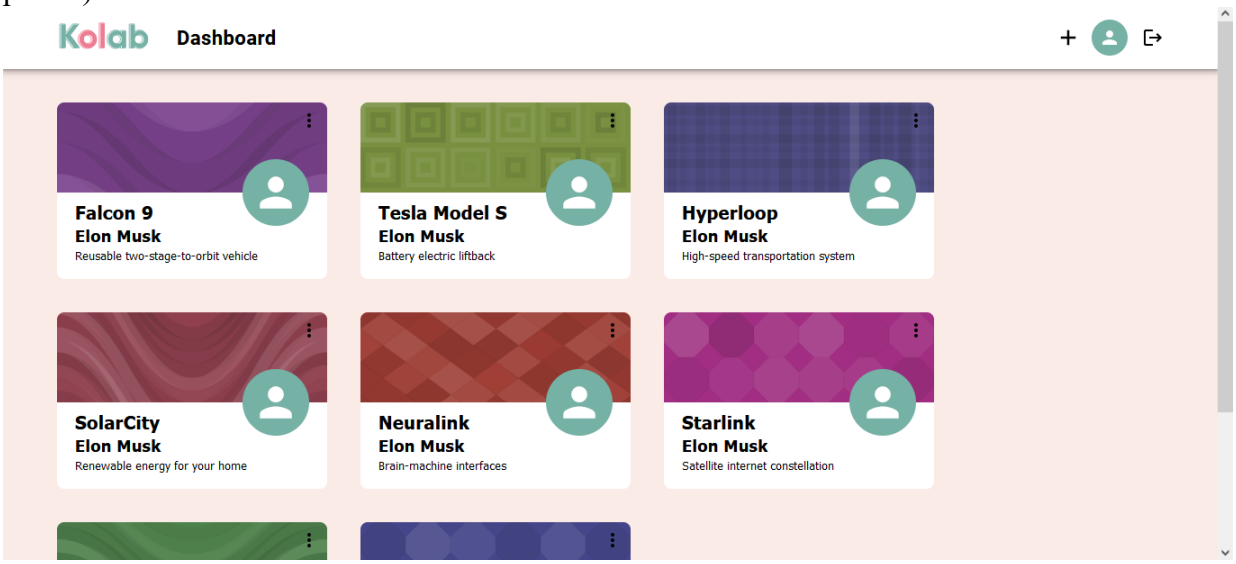


Рисунок 8. Сторінка з проєктами

Якщо натиснемо на потрібний нам проєкт, перейдемо на його сторінку. Якщо ми перебуваємо у ролі Manager, то маємо можливість створювати нові завдання та додавати їх короткий опис. Користувачі, роль яких Employee можуть обмінюватись повідомленнями у створеному завданні. При цьому можливості створювати завдання їм не надано (рис. 9, 10).

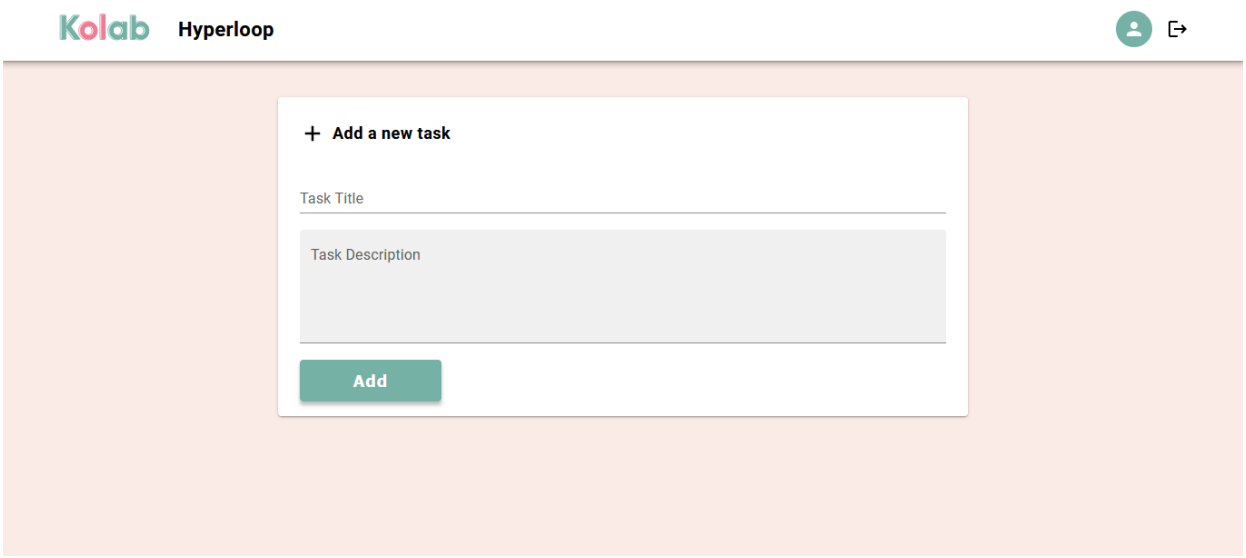


Рисунок 9. Сторінка проєкту(форма для додавання завдання)

Після створення завдання користувачі можуть обмінюватись в ньому повідомленнями.

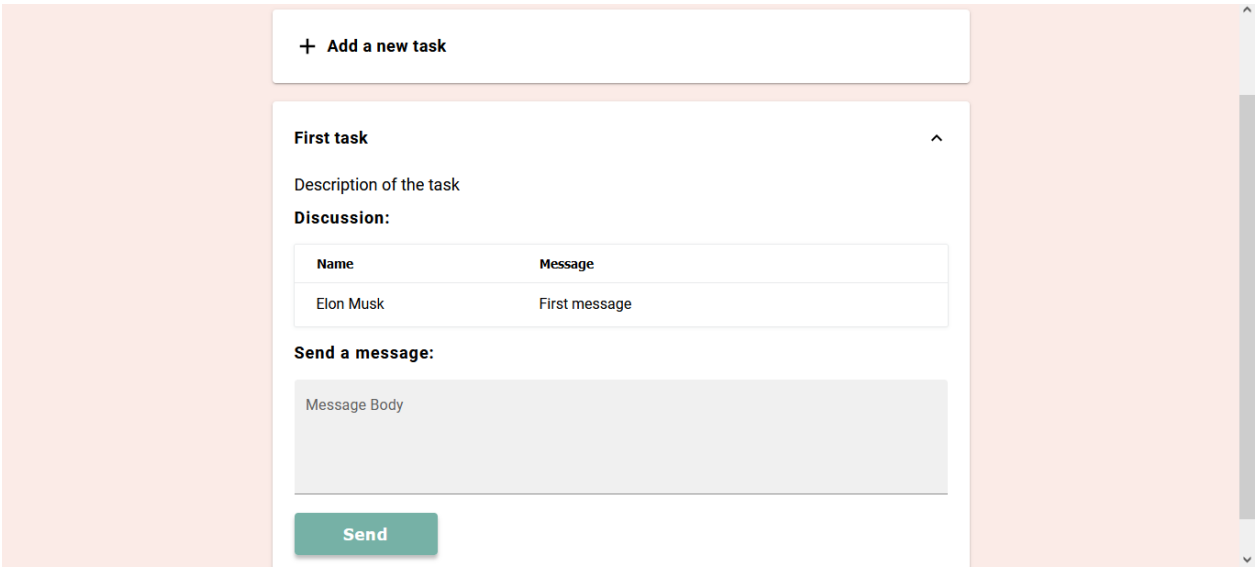


Рисунок 10. Сторінка проєкту(форма завдання)

Перейдемо на сторінку профілю (рис. 11). Для цього потрібно клікнути на іконку користувача справа вгорі екрану. На цій сторінці можна редагувати дані. Можливо додавати фото до профілю.

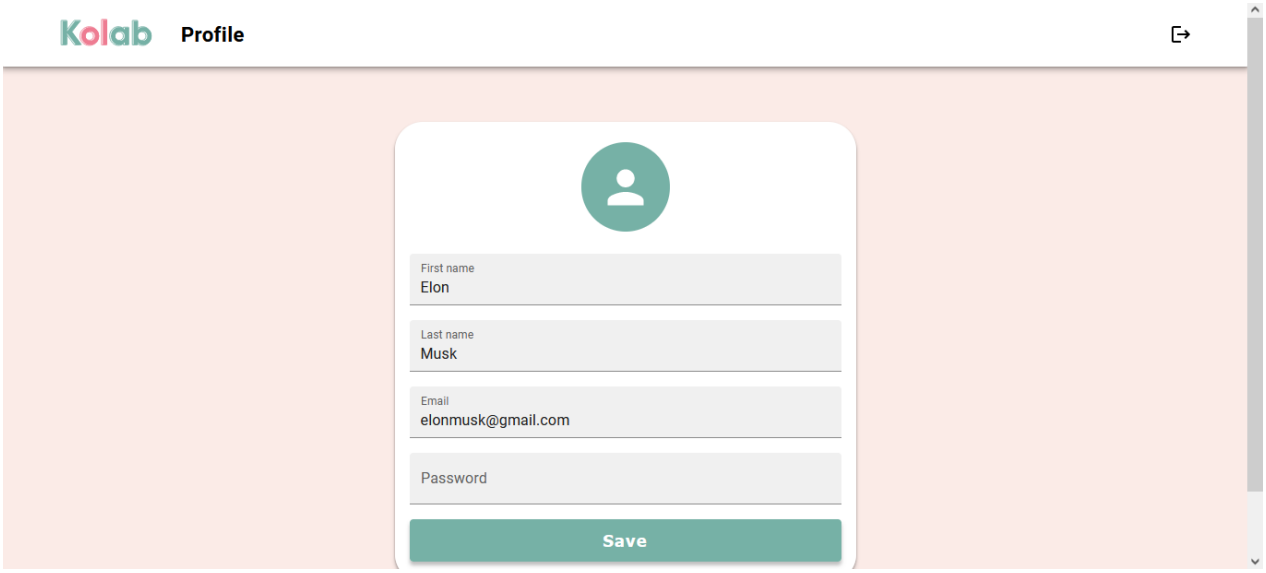


Рисунок 11. Сторінка профілю

Щоб вийти з акаунту потрібно натиснути на значок виходу справа вгорі сторінки.

Для того, щоб додати новий проєкт потрібно на сторінці зі списком проєктів натиснути на кнопку "+" і обрати "Create a project" (див. рис. 12).

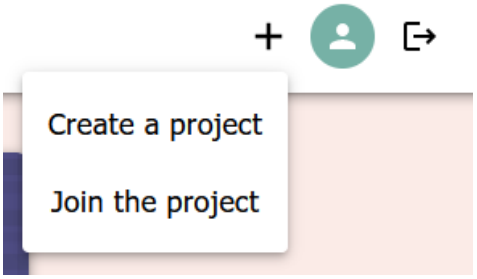


Рисунок 12. Іконки для створення проєкту та для виходу з акаунту

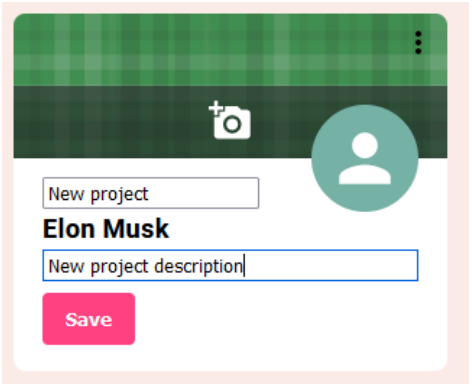


Рисунок 13. Форма для створення проєкту

Також були розроблені адаптивні версії на телефон і планшет, які ви можете побачити на рис. 14-15.

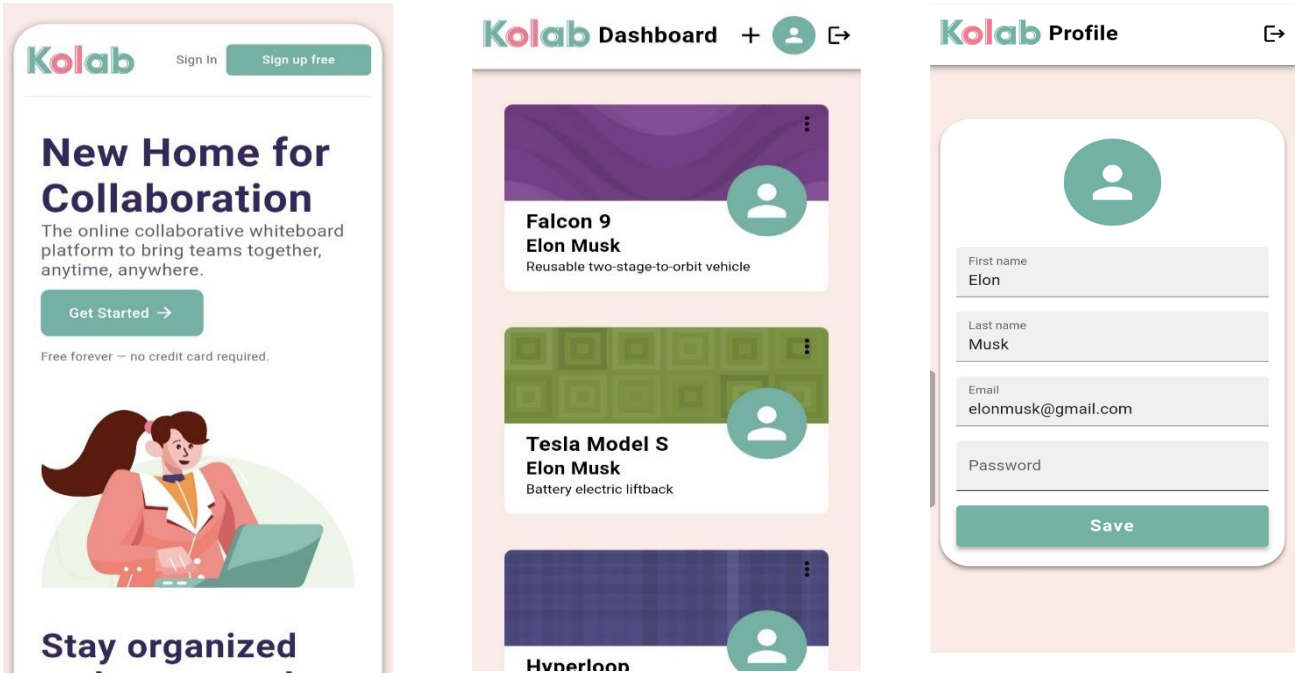


Рисунок 14. Адаптивна версія на телефон і планшет домашньої сторінки, сторінки із проєктами та сторінки профілю

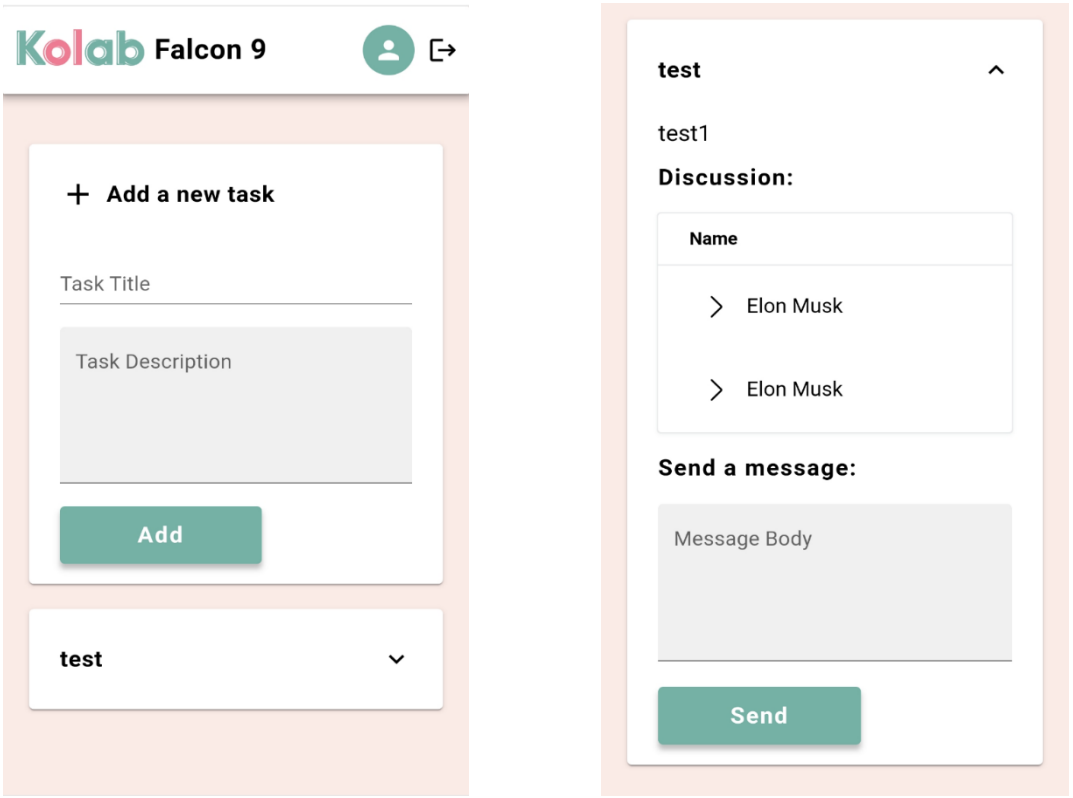


Рисунок 15. Адаптивна версія на телефон і планшет сторінки проєкту (форми додавання для завдання і завдання)

ВИСНОВКИ

В результаті роботи було розроблено проєкт системи управління IT-проєктами Plumbum Kolab. Студенти отримали цінний досвід роботи в команді з використанням реальних методологій розробки, на зразок Kanban. 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: JavaScript, Express, Node.js.Фронт-енд: HTML/CSS, JavaScript, React, Figma. Деплоймент: Heroku.

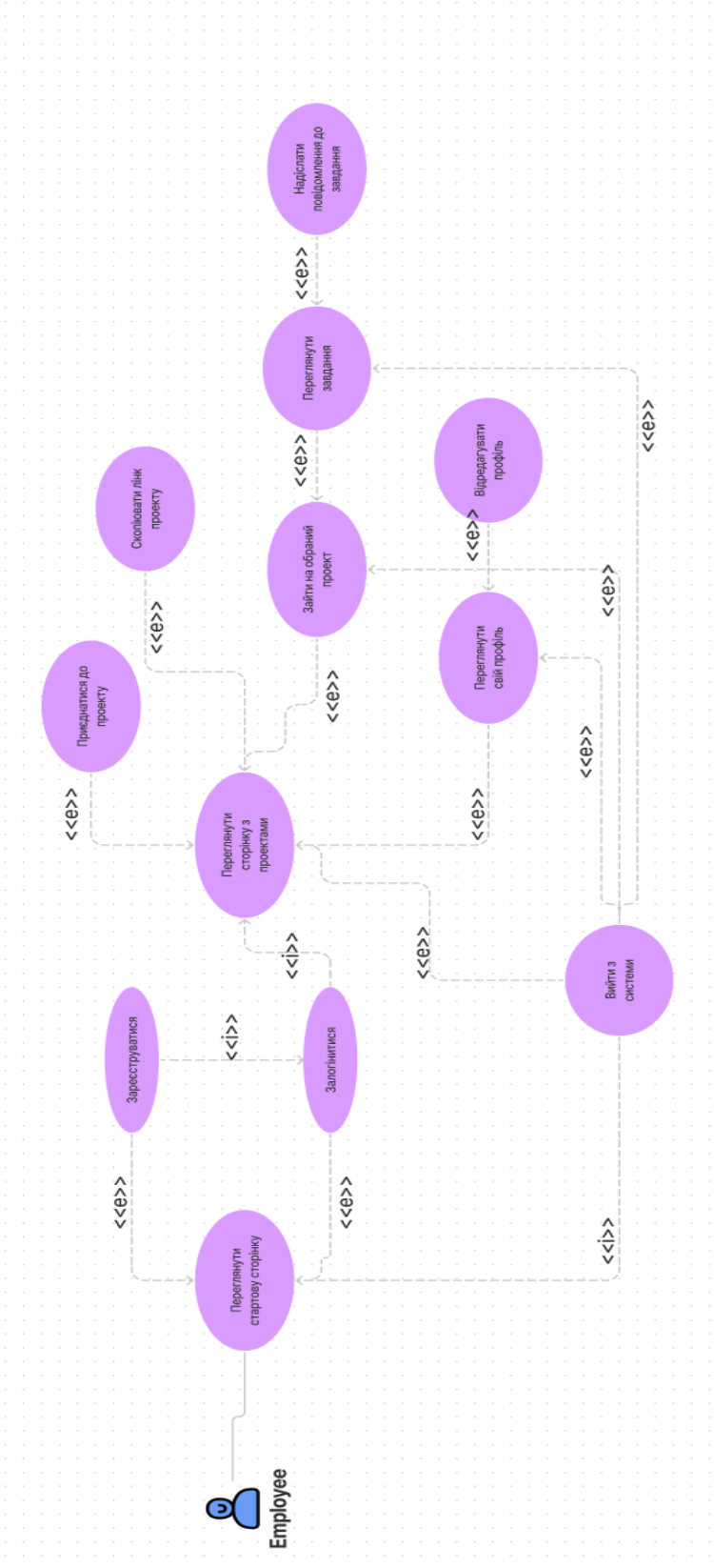
В кінці роботи було отримано повноцінний продукт, який відповідає наведеним вимогам, що свідчить про успішне виконання поставленої задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

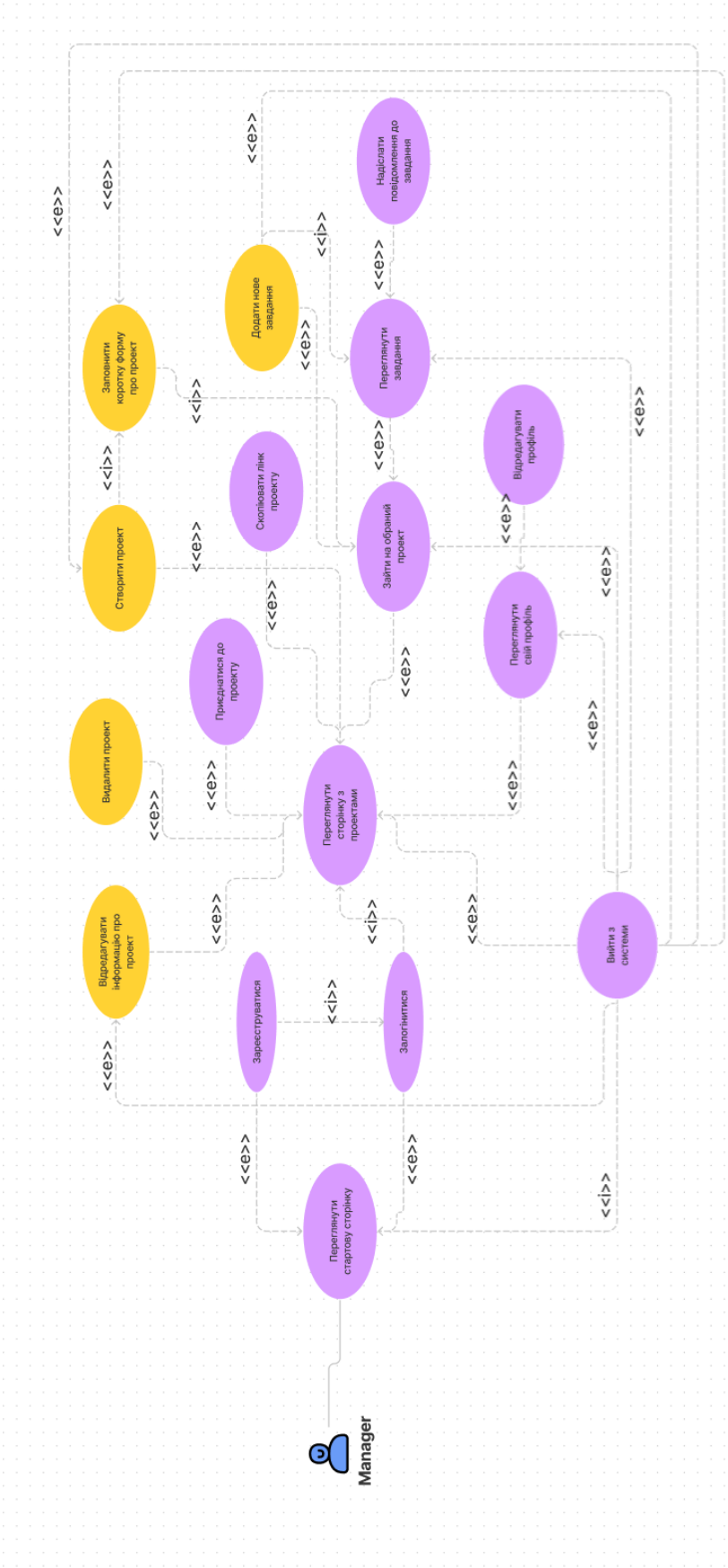
1. Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/
2. PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/
3. W3Schools [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com/
4. JavaScript | MDN [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/javascript
5. Express.js Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/expressjs-tutorial
6. Heroku [Електронний ресурс] – Режим доступу до ресурсу: https://www.heroku.com/
7. Tielens M.T. React in Action Mark / Tielens M.T. - Manning Publications, 2018. - 300 s.
8. Фаулер М. Рефакторинг кода на JavaScript: улучшение проєкта и существующего кода / Фаулер. М. - Вид-во "Диалектика" 2022. - 22 с.

ДОДАТКИ

Додаток 1. Use-case діаграма для користувача з роллю Employee



Додаток 2. Use-case діаграма для користувача з роллю Manager



РОЗРОБКА СИСТЕМИ ПІДТРИМКИ КЕРУВАННЯ ІТ-ПРОЄКТАМИ

Ольга Бережняк, Євгеній Васильєв, Максим Васильчук,
Олександр Галавай, Адальберт Макарович

ПОСТАНОВКА ЗАДАЧІ

Світ невпинно рухається вперед, а разом з ним і розвиваються технології. В наш час важко уявити серйозну ІТ-компанію, яка б досі проводила лише очні зустрічі для комунікації стосовно того чи іншого питання. А обставини нездоланної сили взагалі можуть "покласти" ІТ-сектор, якщо не використовувати засоби дистанційного керування проєктами.

Також варто зауважити, що ІТ-сектор займає третє місце в структурі економіки України після металургії та сільського господарства. Що, безумовно, свідчить про стратегічну важливість стабільности даної галузі, не зважаючи на зовнішні фактори.

Актуальність роботи. У зв'язку із пандемією COVID-19, а також з початком гарячої фази російсько-української війни для забезпечення безпеки працівників абсолютна більшість компаній функціонує онлайн. Тому керування проєктами в компаніях потребує певних зручних систем, в яких можна організувати процес їх управління. Тому така тема проєкту є надзвичайно актуальною у сучасних реаліях.

Мета та завдання роботи. Метою роботи є створення прототипу системи підтримки керування ІТ-проєктами, яка дасть можливість забезпечити учасникам команди з розробки певного програмного продукту координуватися і таким чином керувати процесом реалізації кінцевого завдання команди. Для реалізації мети роботи поставлено наступні завдання:

- проаналізувати подібні системи керування іт-проєктами на ринку;
- сформулювати основний функціонал майбутньої системи;
- розробити діаграми прецедентів та класів;
- розробити технічне завдання до програмного продукту;
- розробити бекенд та дизайн системи.

Об'єкт, методи та засоби розробки. Об'єктом розробки системи є сам процес керування ІТ-проєктами. Предметом роботи є прототип системи для забезпечення можливости підтримки керування ІТ-проєктами.

Перед розробкою програмної системи був проведений аналіз можливостей готових систем керування ІТ-проєктами. Розробка основного функціоналу була зроблена мовою програмування Java у середовищі Intelij IDEA Community edition IDE від компанії JetBrains.

ОРГАНІЗАЦІЯ РОБОТИ В КОМАНДІ

Учасники команди та розподіл ролей. До складу команди "ЛаМпочки ТТП" увійшло 5 людей, студентів 4 курсу: Ольга Бережняк, Максим Васильчук, Євгеній Васильєв, Олександр Галавай, Адальберт Макарович. Розподіл ролей (див. табл. 1) відбувався з урахуванням побажань студентів та був збережений впродовж усієї роботи над проєктом.

Таблиця 1. Розподіл ролей в команді

Учасник	**Роль учасника**
Ольга Бережняк	Фронтенд, UI/UX design
Євгеній Васильєв	Бекенд, тестування
Максим Васильчук	Фронтенд, аналітик
Олександр Галавай	Фахівець з документації, аналітик
Адальберт Макарович	Модератор, керівник проєкту

Методологія та система управління проєктами. Для керування проєктом було обрано методологію Scrum. Зустрічі проводилися регулярно (2-3 рази на тиждень) впродовж перших трьох тижнів роботи над проєктом. Внаслідок розпочатої гарячої фази російсько-української війни з 24 лютого роботу над проєктом було заморожено. 5 квітня замовниками було прийнято рішення про відновлення розробки у форматі прототипу, тобто з найбільш базовим функціоналом. Також проводився запис прогресу проєкту (Додаток А) за допомогою електронної таблиці (https://docs.google.com/spreadsheets/d/1qEcjcFNovF6uRyIZ3V5pZeCwaAfZIKKDyGsfzlygay4/edit?usp=sharing).

На першій зустрічі у Google Meets команда приняла рішення проводити регулярні зустрічі 2-3 рази на тиждень замість Daily Meetings, які, на думку учасників команди, у даному проєкті були б малоінформативними та безсенсовими, так як впродовж одного дня учасниками команди не вдасться виконати отримані завдання. Тому для оперативної комунікації було створено чат у Telegram, в якому на рівні команди приймалися певні мікрорішення. Зустрічі проводилися з використанням Google Meets та Zoom. Результати виконання завдань розміщувалися учасниками команди у спеціально створеному замовниками Google Class. Для зберігання коду було створено проєкт LaMpochky-TTP у сервісі GitHub, який складається з двох частин: бекендної та фронтендної.

Також для більшої структурованості та пріоритезації завдань, які необхідно реалізувати в проєкті, та їх розстановки учасникам команди використовувалася система Trello (рис. 1)

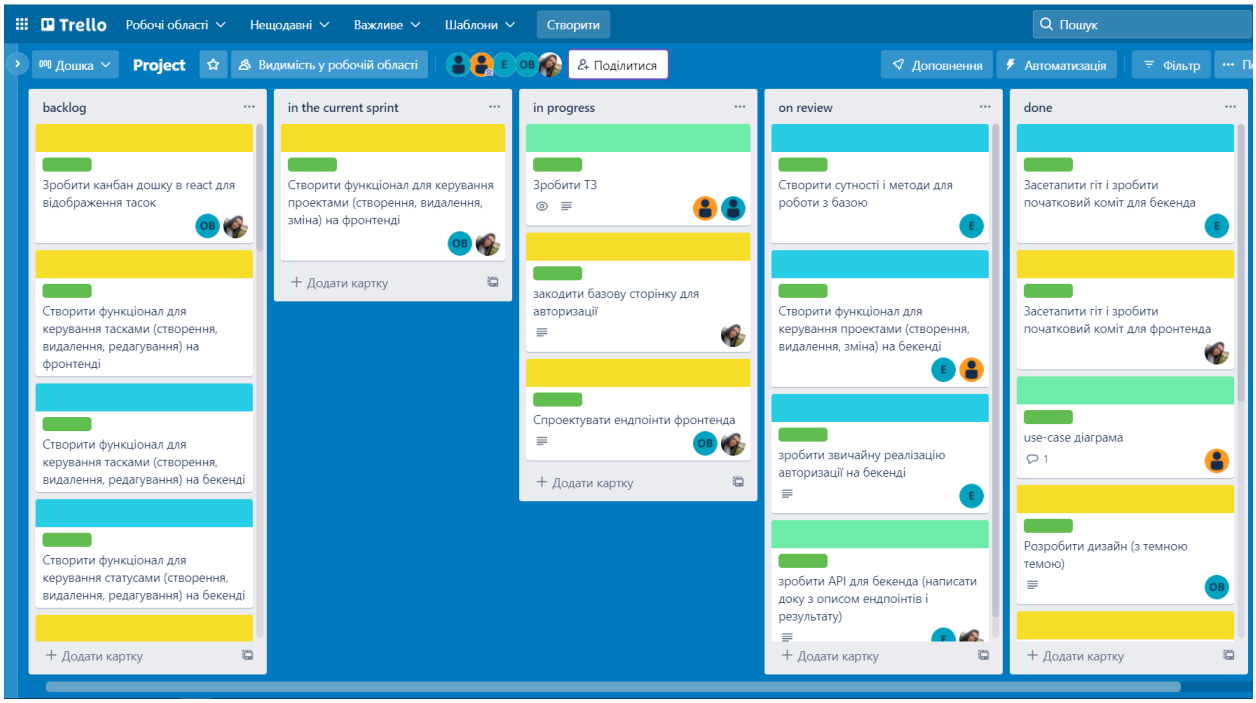


Рисунок 1. Дошка Trello станом на 23 лютого

Примітка: стан та кількість завдань на дошці у Trello відповідає статусу розробки системи станом на вечір 23 лютого.

ОГЛЯД ІСНУЮЧИХ НА РИНКУ ПРОДУКТІВ

Командою було розглянуто декілька варіантів систем керування, серед яких було виокремено Jira, Trello, Worksection. Найвагомішими критеріями вибору саме цих систем

для нас стали доступність, простота входу в систему, простий дизайн та мінімальний набір функцій.

Jira має достатньо великий функціонал, тому в цій системі важко налаштовується під гнучкі, часто змінні, вимоги до нетривалого проєкту. Також більшість розширень для Jira є платними, хоча функціонал з ними значно розширюється.

Trello має широкий функціонал як для безкоштовної системи, навіть не дивлячись на те, що за додаткову плату можна розширити фунціонал.

Worksection має теж широкий функціонал, наприклад, відстеження часу роботи над задачами та проєктами. В ній достатньо гнучке управління проєктами.

Найбільш прийнятним вибором, на думку команди, здалася системи Trello і вітчизняна Worksection. В Системі буде можливість створення своїх проєктів, додання людей до них і визначення для користувачів своїх ролей. Тому продукт буде базуватися на певному симбіозі цих двох з впровадженням певного більш цікавого функціоналу.

ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Інструментарій (технології проєктування). *Creately* [1] надає попередньо визначені шаблони та елементи діаграм для включення в проєкти. Він забезпечує функцію перетягування, за допомогою якої можна включати як попередньо визначені, так і виготовлені на замовлення фігури для створення потрібної діаграми, в той час як одне робоче середовище можна використовувати для спільної роботи кільком особам. За допомогою Creately було спроєктовано діаграму прецедентів для Системи. Це середовище було обрано за свою зручність створення таких діаграм та широкі можливості, наприклад, редагування діаграм декількома учасниками команди.

Google Sheets [2] розроблені з урахуванням потреб організацій, для яких важлива гнучкість робочих процесів. Технології з урахуванням штучного інтелекту допомагають отримувати точні дані, необхідних прийняття вірних бізнес-рішень. Хмарна інфраструктура дозволяє працювати спільно з колегами на будь-якому пристрої та будь-де. Команда використовувала цю систему для запису прогресу розробки проєкту за результатами регулярних зустрічей. Такі таблиці дозволяють зручно і зрозуміло описувати мету зустрічі та її результат.

Trello [3] – це одна з популярних систем управління проєктами в режимі онлайн, яка користується особливим попитом серед невеликих компаній. Тому вона дозволяє ефективно організувати роботу за методологією канбан-дошки.

Інструментарій (технології розробки). *Java* [4] – це об’єктно-орієнтована мова програмування. За допомогою неї програмісти пишуть інструкції, використовуючи команди англійською мовою, замість того, щоб писати у числових кодах. Вона відома як високорівнева мова, тому що її код легко читається та пишеться програмістами. Як і будь-яка мова, Java має перелік правил, які визначають написання інструкцій Ці правила відомі як "синтаксис". Після написання програми інструкції високого рівня переводяться в числові коди, які комп'ютери можуть зрозуміти і виконати. Дана мова була обрана як найбільш популярна мова програмування, а також така, якою володіють розробники бекенду.

Spring Framework [7] – це платформа Java, яка забезпечує всебічну підтримку інфраструктури для розробки додатків Java. Spring [8] обробляє інфраструктуру, щоб можна зосередитися на додатку. Spring дозволяє створювати програми з "звичайних старих Java-об'єктів" (POJO) і застосовувати корпоративні послуги неінвазивно до POJO. Ця можливість застосовується до моделі програмування Java SE і до повної і часткової Java EE. Приклади того, як ми, як розробники додатків, можемо використовувати перевагу платформи Spring: зробити метод Java виконаним у транзакції бази даних без необхідності мати справу з API транзакцій; зробити локальний метод Java віддаленою

процедурою без роботи з віддаленими API. Spring Framework складається з функцій, організованих у близько 20 модулів. Ці модулі згруповані в контейнер Core, доступ до даних / інтеграцію, Web, AOP (програмне забезпечення, орієнтоване на аспект), приладобудування і тест. Бекенд розробники наполягали на використанні саме цієї платформи, як найбільш використовуваною.

PostgreSQL [9]— це потужна об'єктно-реляційна система баз даних з відкритим вихідним кодом з більш як 30-річною активною розробкою, завдяки якій вона заслужила міцну репутацію завдяки надійності, стійкості функцій і продуктивності. В офіційній документації можна знайти велику кількість інформації, яка описує, як встановити та використовувати PostgreSQL.

MaterialUI [10] пропонує широкий вибір високоякісних компонентів, які дозволили нам швидше поставляти функції. MaterialUI використовують понад сотня інженерів з різних організацій. Більше того, добре продумана система налаштування MUI дозволила компанії виділитися на ринку. На заміну звичному Bootstrap розробники фронтенду вирішили використати саме цей пакет бібліотек, які на їх погляд є кращими за дизайнерським рішенням.

React [11] спрощує створення інтерактивних інтерфейсів. Лише потрібно описати, як різні частини інтерфейсу виглядають у кожному стані додатку і React ефективно оновить та відрендерить лише потрібні компоненти, коли дані зміняться. Декларативні інтерфейси роблять код більш передбачуваним і його набагато легше налагоджувати. Дана технологія була обрана завдяки своїй популярности для використання на фронтенді.

Інструментарій (технології тестування). *Модульне тестування (Unit testing)* – тестування кожної атомарної функціональності додатку окремо, в штучно створеному середовищі. Саме потреба у створенні штучної робочого середовища для певного модуля, вимагає від тестувальника знань в автоматизації тестування програмного забезпечення, деяких навичок програмування. Дане середовище для деякого юніта створюється за допомогою драйверів і заглушок. Такий метод тестування було обрано завдяки можливлости відокремленого тестування модулів та доступности використання тестувальником.

Hamcrest — це фреймворк для написання тестів, що дозволяє декларативно визначати правила "відповідності". Існує ряд ситуацій, коли потрібно проводити тестування, наприклад, перевірка інтерфейсу користувача або фільтрація даних, але саме в області написання гнучких тестів найчастіше використовуються відповідники. Під час написання тестів іноді буває важко знайти правильний баланс між надмірним визначенням тесту і недостатнім конкретизацією. Наявність інструменту, який дозволяє вам точно вибрати аспект, який тестується, і описати значення, які він повинен мати, з контрольованим рівнем точності, дуже допомагає в написанні тестів. Такі тести зазнають невдачі, коли поведінка досліджуваного аспекту відхиляється від очікуваної поведінки, але продовжують проходити, коли в поведінку вносяться незначні зміни. Тому саме цьому команда обрала даний фреймворк для тестування проєкту.

ПРИЗНАЧЕННЯ СИСТЕМИ

Призначенням прототипу Системи підтримки керування ІТ-проєктами "СПіКерПро-ЛаМпочки" є автоматизація процесу створення проєктів та їх подальшої підтримки. Така система дає можливости користувачам брати участь у керування прогресом розробки проєкту, а саме створювати нові завдання, змінювати їх статус, додавати нових учасників команди.

Робота передбачає:

- аналіз схожих програмних продуктів на ринку;

- визначення функціональних можливостей для майбутньої системи;
- проєктування діаграм згідно поставлених вимог;
- програмна реалізація системи.

ЦІЛІ СТВОРЕННЯ СИСТЕМИ

"СПіКерПро-ЛаМпочки" була створена з метою:

- можливості керування створеними іт-проєктами;
- комунікації учасників проєкту з приводу його організації;
- представлення проєктів третім особам (замовникам);

В тому числі прототип системи призначений для більш оперативної взаємодії між учасниками проєкту.

ВИМОГИ ДО СИСТЕМИ

Система повинна забезпечити можливості користувачів створювати проєкти та підтримувати їх часниками команди. Повний функціонал поданий в use-case діаграмі (Додаток Б).

Система повинна забезпечувати реалізацію основних наступних завдань:

- можливість створювати нові проєкти;
- можливість створення списків завдань;
- можливість додавати у списки нових завдань;
- можливість змінювати статус завдання і коментувати його;

Цільовою авдиторією є користувачі "СПіКерПро-ЛаМпочки", користувачі, які представлені в системі трьома можливими ролями: адмін (він же керівник проєкту), розробники та спостерігачі. Тому в Системі передбачається виділення трьох функціональних підсистем: адміністративна, призначена для керівника(-ів) проєкту, підсистема розробників проєкту та підсистема переглядачів проєкту.

Вимоги до системи в цілому. Система повинна мати архітектуру, побудовану на сучасних технологіях зберігання, обробки, аналізу даних та доступу до них; забезпечувати одночасну роботу декількох користувачів. Рішення щодо побудови Системи повинні базуватися на:

- застосуванні сучасних інформаційних технологій;
- реалізації концепції створення єдиного інформаційного простору;
- застосуванні правила централізованого накопичення, зберігання та обробки інформації;
- підтримці актуальности, повноти, несуперечности, цілісности та доступности інформації;
- забезпеченні надійного захисту інформації від порушення її цілісності, витоку та блокування згідно з вимогами нормативно-правових документів в галузі захисту інформації;
- забезпеченні надійности, резервування компонентів технічного забезпечення Системи;
- забезпеченні централізованого управління, безперервного контролю функціонування та централізованого налаштування Системи і модулів її компонентів;
- використанні сучасних засобів програмної інженерії при розробці програмного прикладного забезпечення.

Система представляє собою комплекс інформаційних, програмних, технічних, організаційно-методичних та інших необхідних засобів, що забезпечують збір, обробку, зберігання та передачу даних. Архітектура Системи повинна передбачати максимальну незалежність програмно технічних модулів від Виконавця. Інформаційна архітектура

Системи повинна відповідати сучасним вимогам щодо побудови інтерфейсів користувача.

Вимоги до ергономіки. Рішення щодо ергономіки повинно забезпечувати:

- зрозумілу логічну побудову сторінок та переходів відповідно до інформаційної архітектури;
- вбудовані механізми валідації значень, що визначаються для окремих полів, комбінацій полів, контроль значень полів.

Вимоги до надійности. Надійність функціонування Системи повинна забезпечуватися:

- використанням сучасних технологій розробки (модернізації) прикладного програмного забезпечення та забезпеченням якісного його тестування;
- резервуванням основних компонентів;
- регламентом організації резервного копіювання та архівного зберігання інформації;
- оперативністю заміни програмно-технічних засобів, що вийшли з ладу;
- сумісністю технічних засобів та програмного забезпечення.

Повинно бути реалізованим гаряче резервування, у відповідності до якого дублюючі компоненти знаходяться у режимі "гарячого" резерву. У разі відсутності відклику основного компонента здійснюється перенаправлення трафіку на резервну систему. Вимоги щодо надійності Системи можуть бути уточнені Виконавцем.

Вимоги до підсистеми керівника проєкту.

Функції та задачі, які стосуються керівників проєкту наведено нижче.

Таблиця 2. Функціонал ролі "керівник проєкту"

Функція	Задача
Редагування проєкту	Змінити назву проєкту
	Редагування статусу користувачів**
	Запросити користувача до проєкту
Редагування статусів на дошці проєкту**	Створити новий статус
	Змінити статус
	Видалити статус
Додатковий функціонал для скраму**	Додати завдання в теперішній спринт
	Видалити завдання з теперішнього спринта
	Налаштувати час та тип зустрічі команди
	Закінчити спринт і почати новий

** - даний функціонал є додатковим і може бути нереалізованим внаслідок дії нездоланної сили (форс-мажор).

Вимоги до підсистеми розробника проєкту.

Функції та задачі, які стосуються розробників проєкту наведено нижче.

Таблиця 3. Функціонал ролі "розробник проєкту"

Функція	Задача
Змінити завдання	Створити нове завдання
	Змінити статус
	Змінити опис
	Змінити дедлайн
	Змінити назву
	Змінити час початку

	Змінити пріоритет
	Змінити відповідального
	Додати тег
	Експортувати завдання з Jira**

** - даний функціонал є додатковим і може бути нереалізованим внаслідок дії нездоланної сили (форс-мажор).

Вимоги до підсистеми переглядача проєкту.
Функції та задачі, які стосуються переглядача проєкту наведено нижче.

Таблиця 4. Функціонал ролі "переглядач проєкту"

Функція	Задача
Вибрати проєкт	Переглянути чат проєкту
	Переглянути форми з результатами зустрічей**
	Підписатися на сповіщення в телеграмі**
	Додати зустріч в календар**
Перегляд дошки із завданнями	Змінити назву
	Фільтрувати завдання, що відображаються**
	Передивитися лог змін**
	Прокоментувати обране завдання
	Подивитися статистику по завданням, що відображаються на дошці**

** - даний функціонал є додатковим і може бути нереалізованим внаслідок дії нездоланної сили (форс-мажор).

Вимоги до додаткової підсистеми авторизації/реєстрації користувача.
Функції та задачі, які стосуються авторизації/реєстрації користувача наведено нижче.

Таблиця 5. Вимоги до реєстрації/авторизації

Функція	Задача
Авторизація користувача	Звичайна авторизація
	Авторизація через GitHub**
	Авторизація через Google Auth**
Реєстрація користувача	Звичайна реєстрація
	Реєстрація через GitHub**
	Реєстрація через Google Auth**

** - даний функціонал є додатковим і може бути нереалізованим внаслідок дії нездоланної сили (форс-мажор).

Системні вимоги. Сайт повинен відображатися в інтернет-браузерах Microsoft Edge 99.0.0.0 і вище, Mozilla Firefox 1.7 і вище, Opera 7.54 і вище, Chromium 99.0.0.0 і вище, Google Chrome 99.0.0.0 і вище.

Операційна система: Windows 10 і вища.

Джерелом даних для Системи повинна бути інформаційна система PostgreSQL.

ОПИС ОРГАНІЗАЦІЇ ІНФОРМАЦІЙНОЇ БАЗИ

Логічна структура бази даних. При проєктуванні Системи було розроблено наступну структуру класів, які подані в таблиці 6.

Таблиця 6. Перелік таблиць, спроєктованих для Системи

Номер	Таблиця	Опис
1	Users	Таблиця для збереження інформації про всіх зареєстрованих користувачів у системі
2	Project	Таблиця для збереження інформації про всі наявні проєкти в системі
3	UserProject	Таблиця для зв'язку проєктів з користувачами
4	List	Таблиця для представлення списків, які наявні у проєкті
5	Task	Таблиця для опису завдання, яке представлено у конкретному списку
6	Tag	Таблиця для означення тегу (статусу) для завдань
7	Message	Таблиця для повідомлень під завданням (чат завдання)
8	Action	Таблиця для представлення події – перенесення завдання з одного списку в інший або створення нового

Опис таблиць. У таблицях 7-14 представлені основні таблиці інформаційної бази з атрибутами та їх коротким описом (Додаток В).

Таблиця 7. User

id	integer	Ключ користувача
username	varchar(20)	Ім'я користувача
email	varchar(30)	Електронна пошта
password	varchar(100)	Пароль до аккаунту

Таблиця 8. Project

id	integer	Ключ проєкту
name	varchar(20)	Назва проєкту

Таблиця 9. UserProject

id	integer	Ключ даного проєкту даному користувачу
user_id	integer	Ключ користувача
project_id	integer	Ключ проєкту
role	integer	Роль користувача у даному проєкті
confirmed	boolean	Завершеність проєкту

Таблиця 10. List

id	integer	Ключ даного проєкту даному користувачу
name	varchar(20)	Назва списку в проєкті
project_id	integer	Ключ проєкту

Таблиця 11. Task

id	integer	Ключ даного завдання
name	varchar(20)	Назва завдання
description	varchar(10000)	Опис завдання
priority	integer	Пріоритет завдання
date_to_start	date	Дата початку виконання завдання

date_to_finish	date	Дата завершення виконання завдання
assighed_user_id	integer	Ключ відповідального за виконання користувач
creator_id	integer	Ключ користувача, який створив завдання
list_id	integer	Ключ списку, якому належить завдання

Таблиця 12. Tag

id	integer	Ключ тега для завдання
name	varchar(40)	Назва тега
project_id	integer	Ключ проєкту

Таблиця 13. Message

id	integer	Ключ повідомлення
name	varchar(20)	Назва повідомлення
text	varchar(1000)	Детальне повідомлення
task_id	integer	Ключ завдання, у якому дане повідомлення
user_id	integer	Ключ користувача, який написав дане повідомлення

Таблиця 14. Action

id	integer	Ключ до історії змін списку
date	dateTime	Час внесеної зміни
task_id	integer	Ключ завдання, який було змінено
user_id	integer	Ключ користувача, який здійснив зміну
list_id	integer	Ключ списку, якому належить завдання

ЕТАП ТЕСТУВАННЯ

Тестування системи. Тестування програмного забезпечення – це процес виконання програмної системи або її компоненти за заданих умов з аналізом або записом результатів і оцінкою результатів та оцінкою деяких властивостей тестованого об’єкта.

Цей етап роботи допомагає команді чітко зрозуміти, в якому стані розробки знаходиться проєкт, що варто змінити, а від чого потрібно взагалі позбутися. В наш час існує цілий пласт кваліфікованих працівників, які відповідають за це.

Технології тестування постійно розвиваються. Навіть існують технології, які в свою чергу дозволяють записувати дії користувача в браузері та конвертувати їх в кроки автоматичного тесту. Це дозволить значно скоротити процес розробки автоматичних тестів, хоча об’єм роботи над створенням набору тестів для покриття принаймні базового функціоналу, вимагатиме значних ресурсів та часу на розробку.

Таким чином тестування – це передусім відповідність між отриманою поведінкою програми та тої, яка зазначена у специфікації, на обраному певним чином наборі тестів спеціалістом з тестування.

Ручне тестування – це найпростіший вид тестування, при якому програмний продукт перевіряється вручну спеціалістами з ручного тестування на відповідність вимогам програмного забезпечення та стандартам якості продукту.

Команда обрала ручне тестування так як це найбільш просте і зрозуміле тестування. Воно проводилося за стратегією Bottom-Up. Така стратегія була обрана, тому що вона підходить для поступового опанування засобів та технологій тестування навіть недосвідченими тестувальниками, тобто фрагменти системи тестуються у порядку зростання складности, а тому і у порядку зростання складности написання до них самих тестів.

Модульне тестування. Для системи проводилося модульне (unit) тестування за допомогою мови програмування Java. Для цього було використано окремий фреймфорк Hamcrest. Його було обрано завдяки тому, що він підтримує створення налаштованих правил-узгоджувачів, що дозволяє декларативно визначити правила збігу, а також тому що він достатньо гнучкий і зрозумілий розробнику. Всі тести було виокремлено в окремий проєкт, а для тестування методів було розроблено окремий клас. Наведемо декілька прикладів тестів та результатів їх виконання.

Таблиця 15. Тест "login_success"

Показник	Значення
Опис тесту	Перевірка на успішність входу зареєстрованого користувача
Вхідна умова	email = "test_email@gmail.com" password = "test_pwd"
Очікуваний результат	За умови зареєстрованости користувача авторизація пройде успішно
Фактичний результат	Тест пройдено – помилок не виявлено.

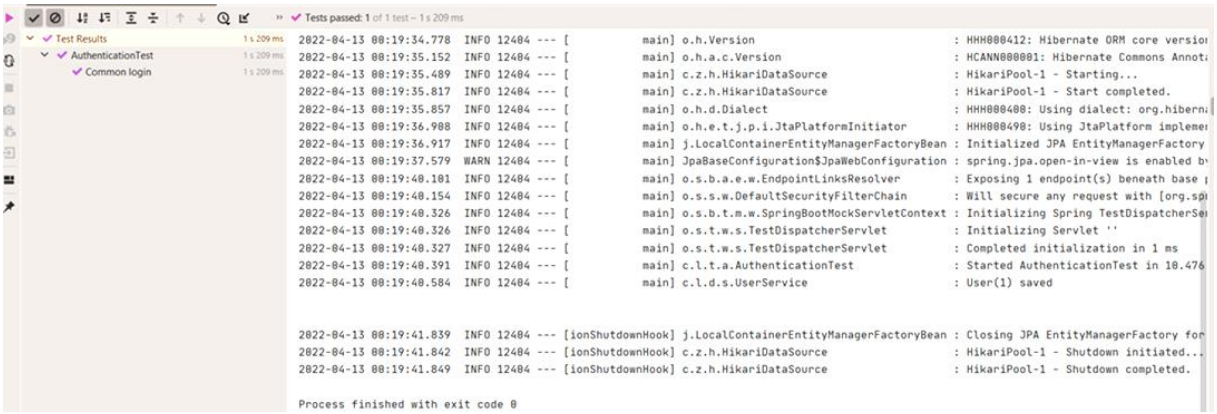


Рисунок 2. Перевірка на успішність авторизації

Таблиця 16. Тест "getProject_success"

Показник	Значення
Опис тесту	Отримання проєкту за ключем від певного користувача
Вхідна умова	id = 1 userEmail = "email_1@gmail.com"
Очікуваний результат	Отримання проєкту за ключем
Фактичний результат	Тест пройдено – помилок не виявлено. Отримано проєкт у json-представленні

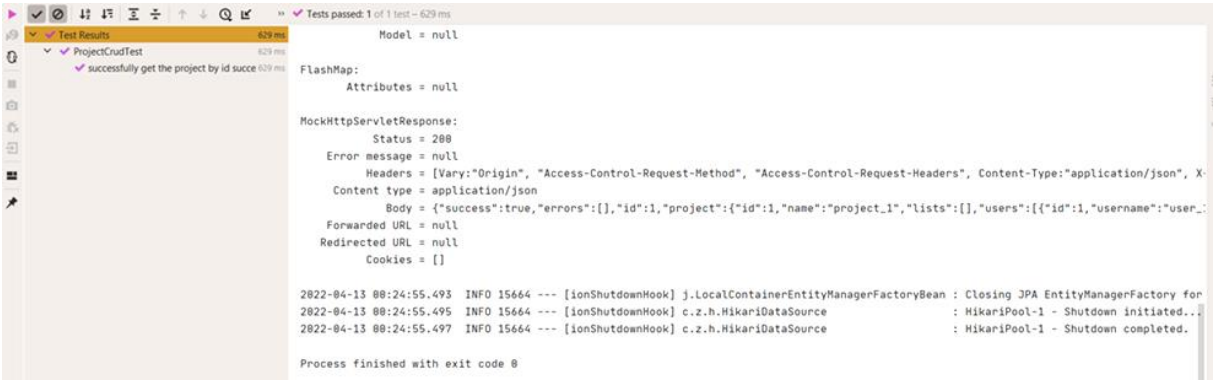


Рисунок 3. Отримання проєкту за його ключем

Таблиця 17. Тест "getTask_fail_taskDoesNotExist"

Показник	Значення
Опис тесту	Перевірка на наявність задачі у проєкті розробника
Вхідна умова	taskId = 10 userEmail = "nor@gmail.com"
Очікуваний результат	Значення параметру taskId невалідне
Фактичний результат	Тест пройдено – помилок не виявлено. Отримано повідомлення "TASK_NOT_FOUND"

Рисунок 4. Перевірка на наявність задачі у проєкті розробника

Далі наведено скріни деяких кодів тестів та їх загальне виконання, адже самих тестів у проєкті понад 150 штук (рис. 5-7).

```java
@Test
@Order(4)
@DisplayName("successfully refuting inviting to the project")
public void refuteInvitingToProject_success() throws Exception {
    final Integer projectId = 1;
    final String userEmail = "email_3@gmail.com";
    User user = getUserByEmail(userEmail);
    Project project = getProjectById(projectId);
    ObjectNode request = mapper.createObjectNode()
            .put( fieldName: "confirm",  v: false);

    UserProject userRelation = userProjectService.findByUserAndProject(user, project).orElse( other: null);
    Assumptions.assumeTrue( assumption: userRelation != null && !userRelation.getConfirmed(),
             message: "no unconfirmed relation found in the db");

    mvc.perform(put( urlTemplate: "/data/project/" + projectId + "/invite")
                .contentType(MediaType.APPLICATION_JSON)
                .content(request.toString())
                .with( user(new UserSecurity(user)) ))
            .andExpect(status().isOk())
            .andExpect(jsonPath( expression: "success", is( value: true)))
            .andExpect(jsonPath( expression: "errors", emptyIterable()))
            .andExpect(jsonPath( expression: "projectId", is(projectId)));

    userRelation = userProjectService.findByUserAndProject(user, project).orElse( other: null);

    Assertions.assertNull(userRelation);
}
```

Рисунок 5. Тест на успішність відмови під запиту на приєднання до проєкту

```java
@ParameterizedTest
@MethodSource
@DisplayName("Multiple user errors")
public void multipleErrorsUserValidation_fail(User user, List<String> errors){
    Assertions.assertFalse(userValidation.validate(user));
    Assertions.assertEquals(errors.size(), userValidation.getErrors().size());
    Assertions.assertTrue(userValidation.getErrors().containsAll(errors));
}

public static Stream<Arguments> multipleErrorsRequestValidation_fail(){
    return multipleErrorsUserValidation_fail()
            .map(arguments -> Arguments.of(
                    convertToRequests((User)arguments.get()[0]),
                    arguments.get()[1]
            ));
}

@ParameterizedTest
@MethodSource
@DisplayName("Multiple errors in request")
public void multipleErrorsRequestValidation_fail(RegisterRequestDto request, List<String> errors){
    Assertions.assertFalse(userValidation.validate(request));
    Assertions.assertEquals(errors.size(), userValidation.getErrors().size());
    Assertions.assertTrue(userValidation.getErrors().containsAll(errors));
}
```

Рисунок 6. Тести на появу декількох помилок одночасно при запиті

```java
@Test
@Tag("login")
@DisplayName("Login without email")
public void loginNoEmail_fail() throws Exception {
    ObjectNode jsonRequestBody = mapper.createObjectNode();
    jsonRequestBody.put( fieldName: "password",  v: "test_pwd");

    mockMvc.perform(post( urlTemplate: "/auth/login")
            .contentType(MediaType.APPLICATION_JSON)
            .content(jsonRequestBody.toString()))
            .andExpect(status().isOk())
            .andExpect(jsonPath( expression: "$.email", nullValue()))
            .andExpect(jsonPath( expression: "$.username", nullValue()))
            .andExpect(jsonPath( expression: "$.token", nullValue()))
            .andExpect(jsonPath( expression: "$.success", is( value: false)))
            .andExpect(jsonPath( expression: "$.errors", not(emptyIterable())));
}
```

Рисунок 7. Тест на можливість входу без email

Загальні результати виконання всіх тестів у проєкті представлені на рис. 8-11.

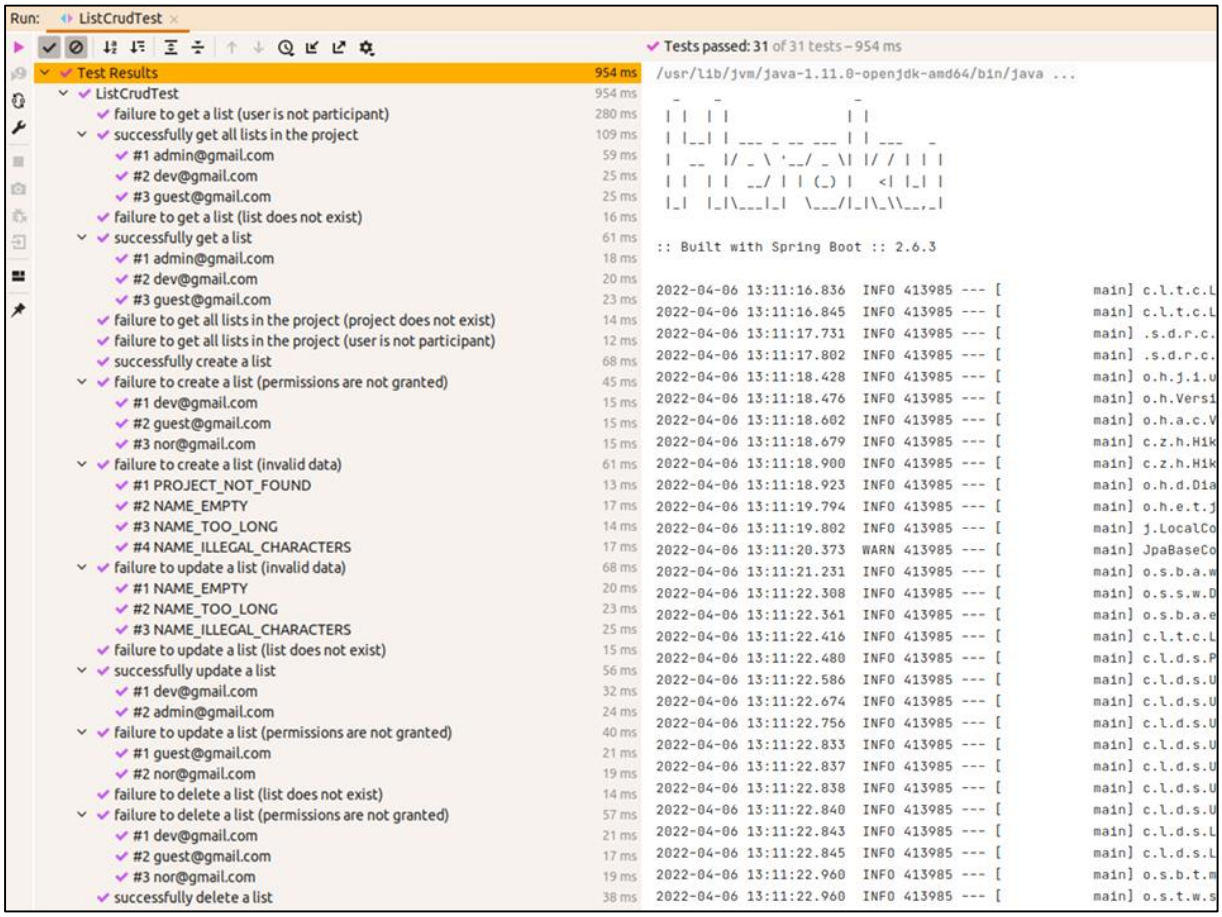


Рисунок 8. Тести класу ListCrudTest

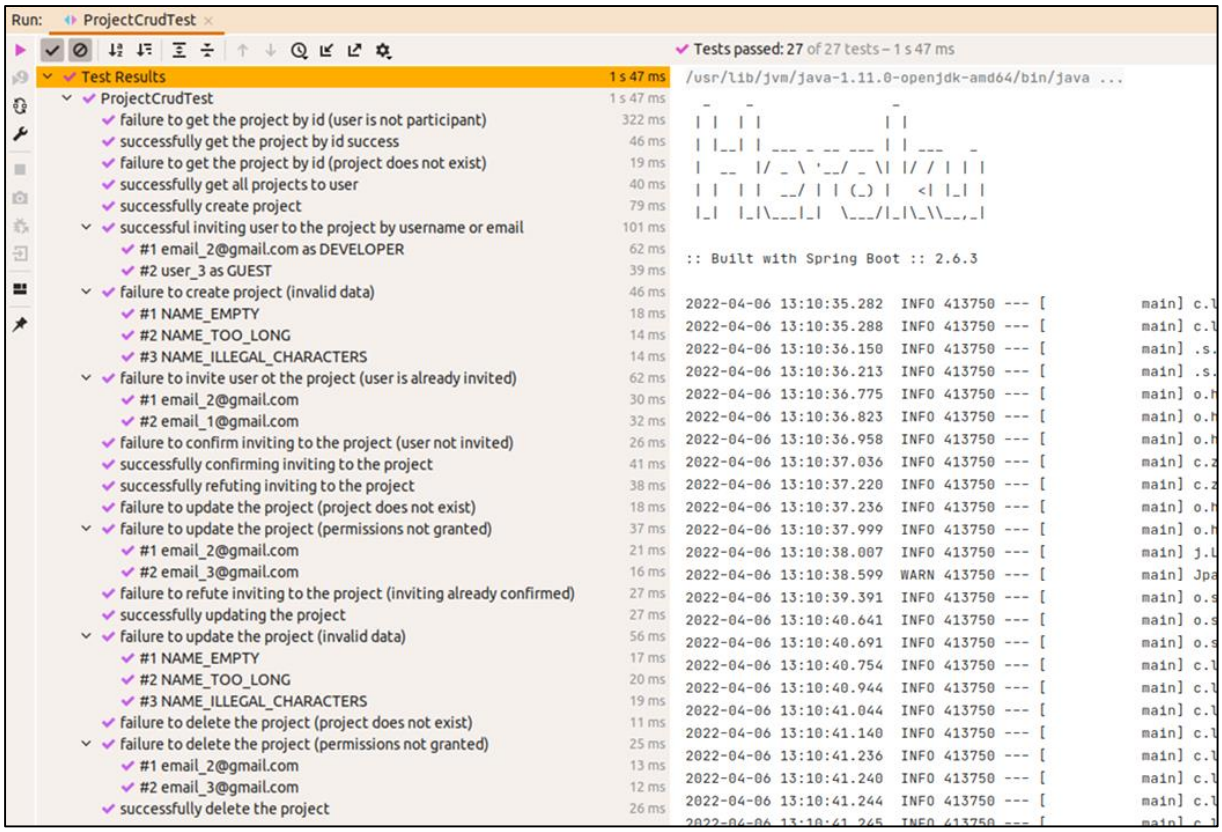


Рисунок 9. Тести класу ProjectCrudTest

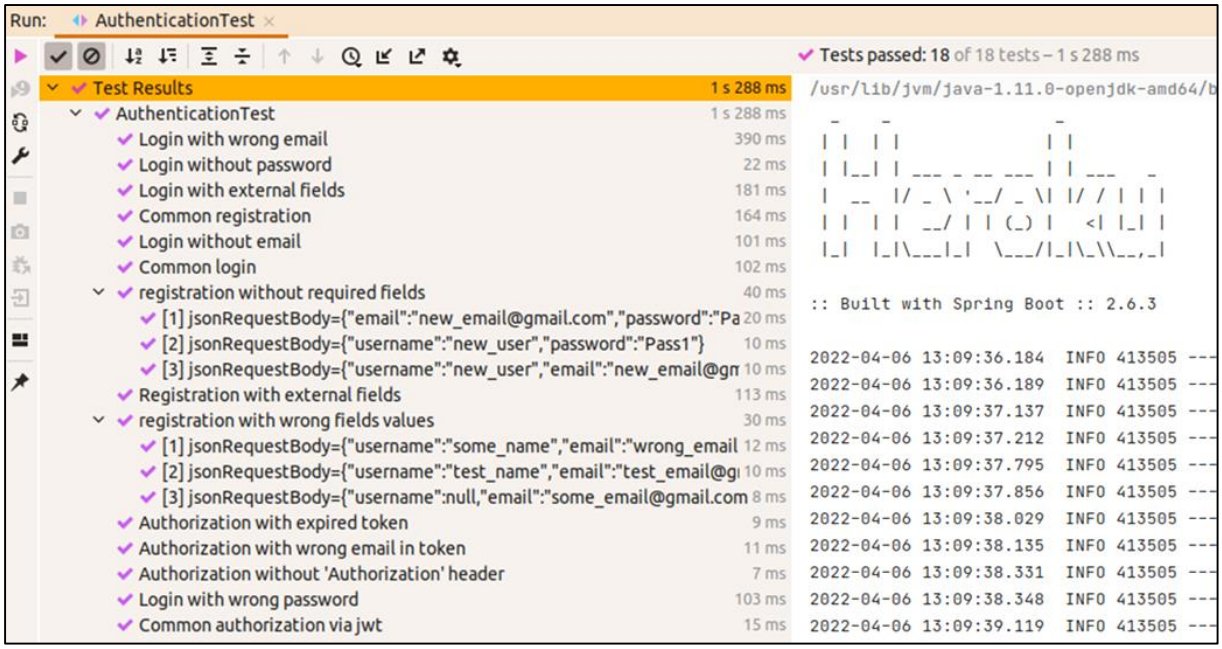


Рисунок 10. Тести класу AuthenticationTest

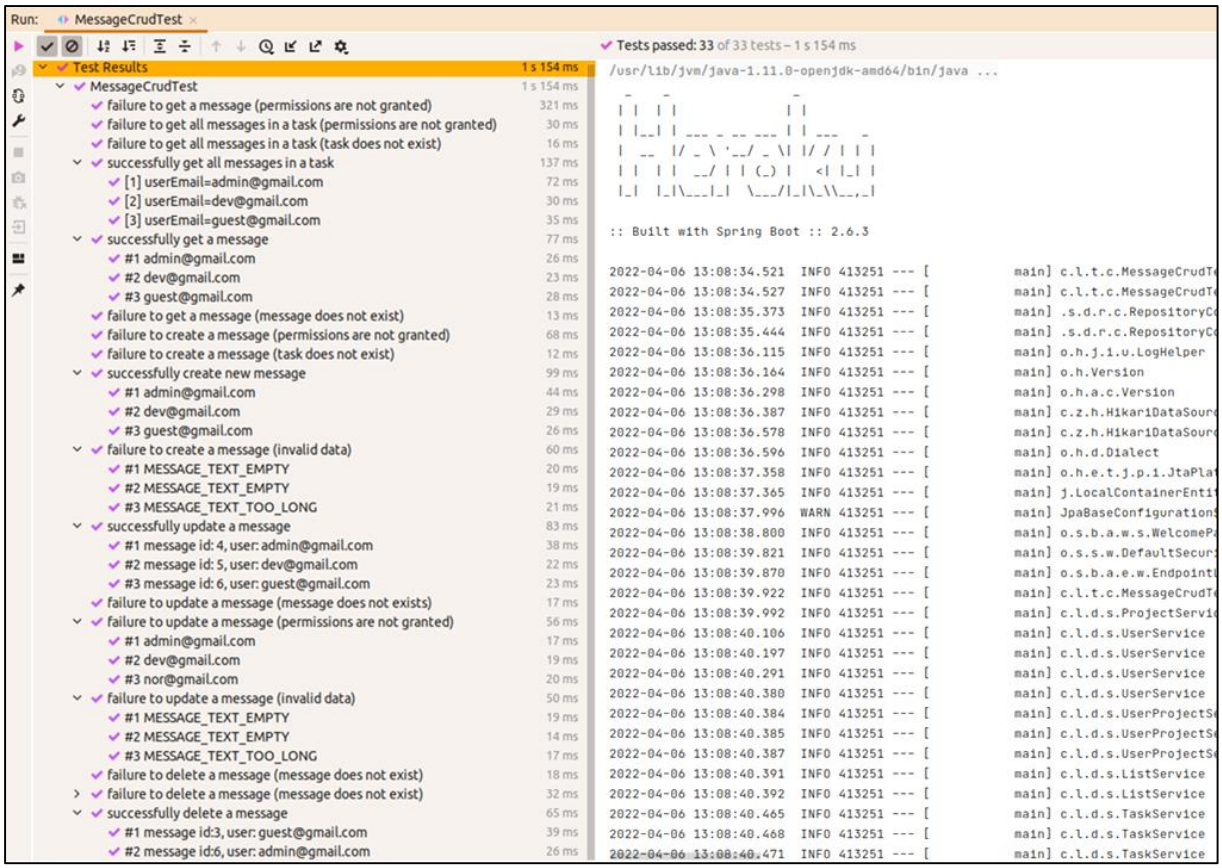


Рисунок 11. Тести класу MessageCrudTest

ІНСТРУКЦІЯ КОРИСТУВАЧА

При відкритті, кожен користувач потрапляє на головну сторінку (рис. 12а, 12б). Для користувача доступно дві теми зовнішнього вигляду (темна та світла), між якими він може перемикатись в ході роботи з системою.

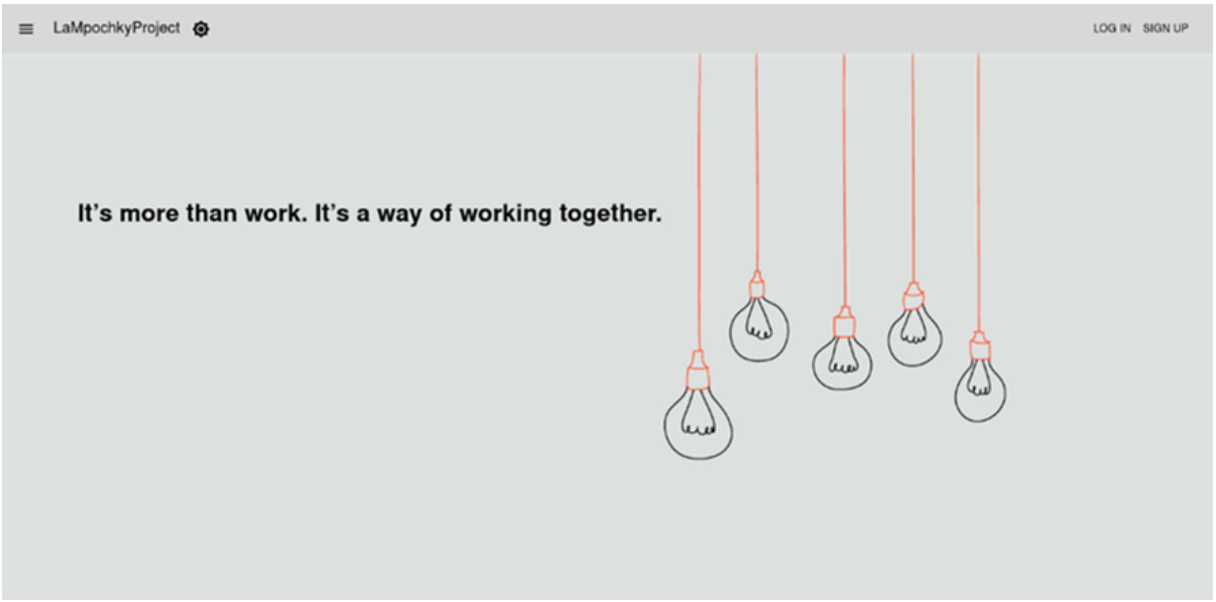


Рисунок 12а. Головна сторінка сайту (темна версія)

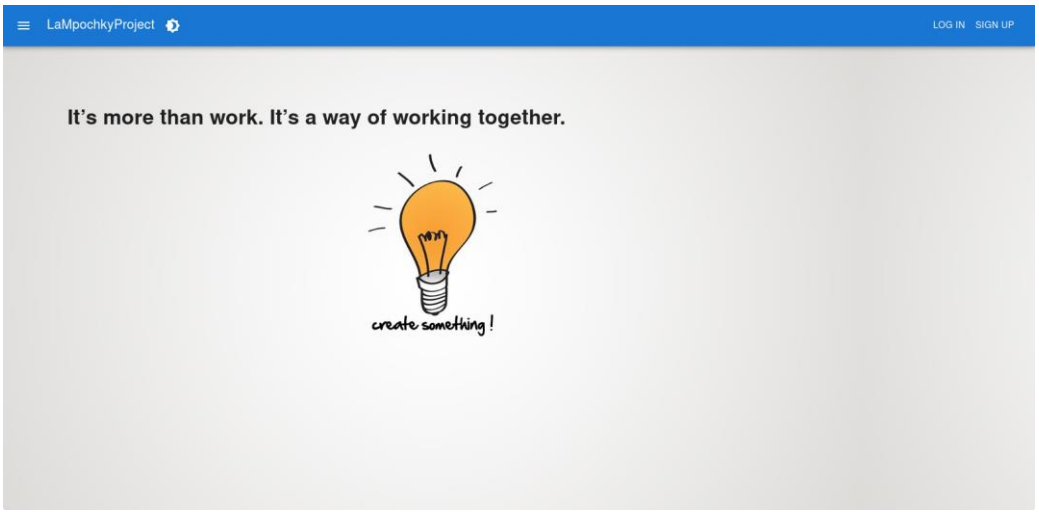


Рисунок 12б. Головна сторінка сайту (світла версія)

Неавторизований користувач. Для неавторизованого користувача функціонал практично недоступний. Для нього доступно лише 3 опції:

1. Перегляд надзвичайно красиву головну сторінку.
2. Зареєструватись (якщо користувач вперше у системі).
3. Залогінитись (якщо користувач реєструвався раніше).

При реєстрації необхідно заповнити такі поля: ім'я, email, пароль, його підтвердження (рис. 13-14). А при вході в акаунт, який вже існує — достатньо ввести лише значення в поля email та пароль (рис. 3).

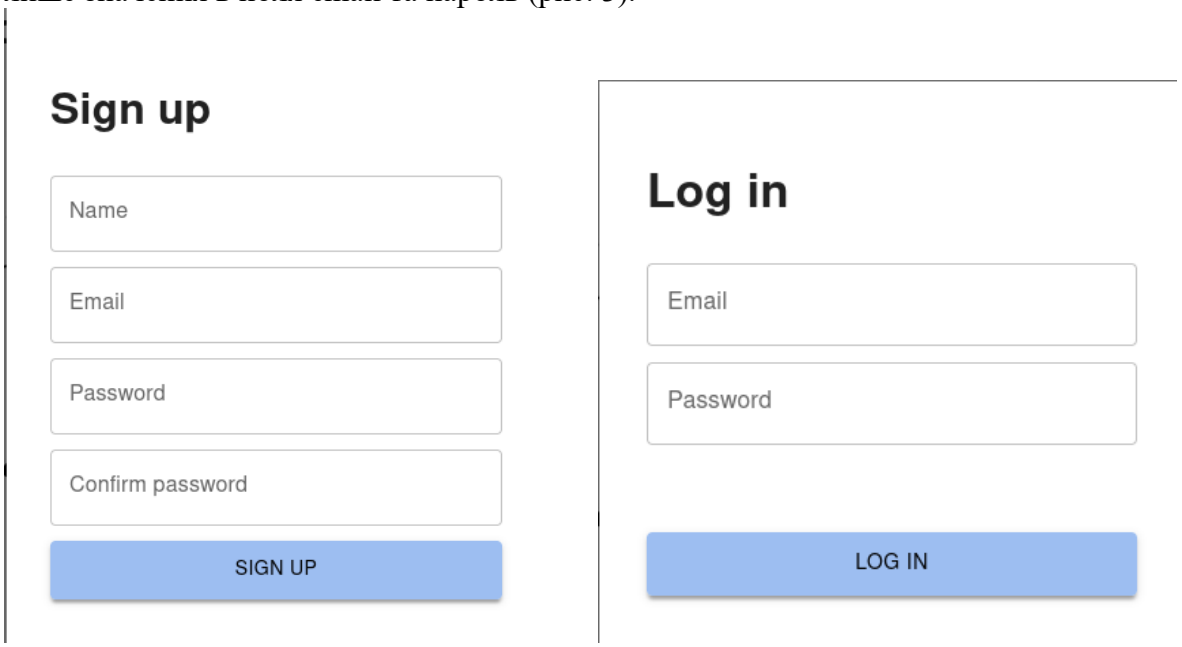


Рисунок 13-14. Реєстрація та авторизація користувача

Авторизовані користувачі. Уже авторизовані користувачі мають однакові привілеї в самій системі. Кожен з користувачів має доступ до бургер-меню, де він може бачити проєкти (та роль у кожному них), у яких він бере участь, або створити новий проєкт (рис 15.).

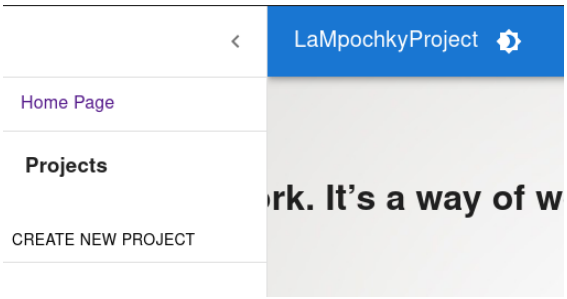


Рисунок 15. Меню навігації між проєктами

Також там присутній перехід на головну сторінку і, зокрема, там будуть з’являтись проєкти, до яких користувача запрошують приєднатись.

Якщо користувач хоче створити проєкт, то він просто тисне кнопку "CREATE NEW PROJECT" і перед ним з’являється модальне вікно, куди він має ввести назву цього проєкту, після чого тисне кнопку "CREATE" і проєкт з’являється у списку (рис. 16).

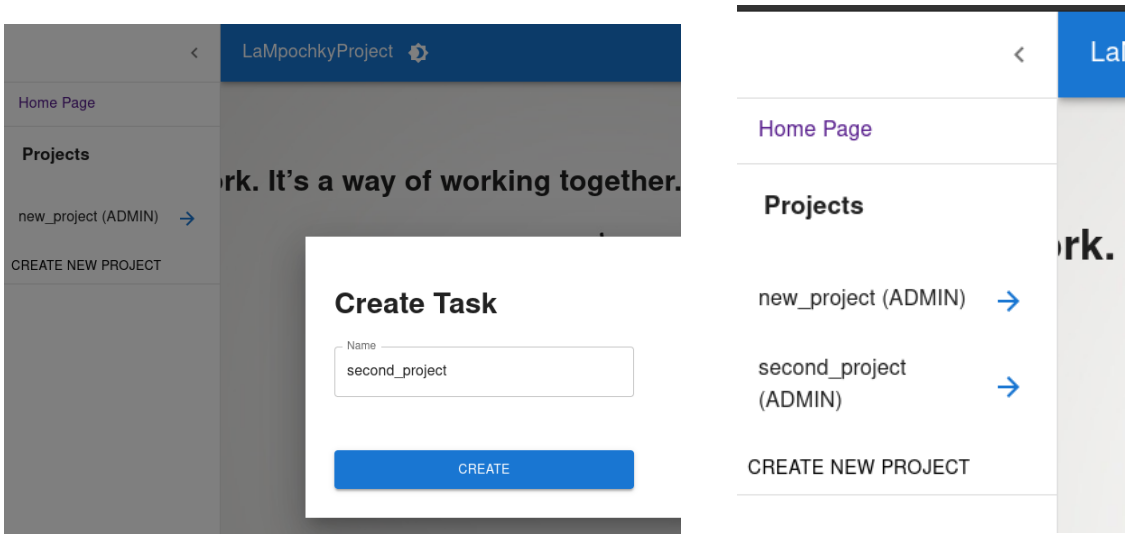


Рисунок 16. Створення нового проєкту

Для переходу на потрібний проєкт користувач має натиснути стрілочку біля назви проєкту, після чого відкриється робочий простір (рис 17.), у якому й відбувається основна робота.

Функціонал ролей у проєкті. У проєкті користувач може мати 3 різні ролі:

1. ADMIN — адміністратор проєкту, може користуватись всіма функціями управління проєктом, що доступні у системі.
2. DEVELOPER — розробник, має трішки менші права, ніж адміністратор. Може користуватись практично всіма функціями системи.
3. GUEST — гість, фактично може лише переглядати робочий простір проєкту.

Перша відмінність між адміністратором та розробником полягає в тому, що перший може створювати, редагувати та видаляти списки завдань, а другий — лише редагувати. Стосовно операцій із самими завданнями, то тут вони мають однакові права обоє можуть і створювати, і редагувати, і видаляти їх.

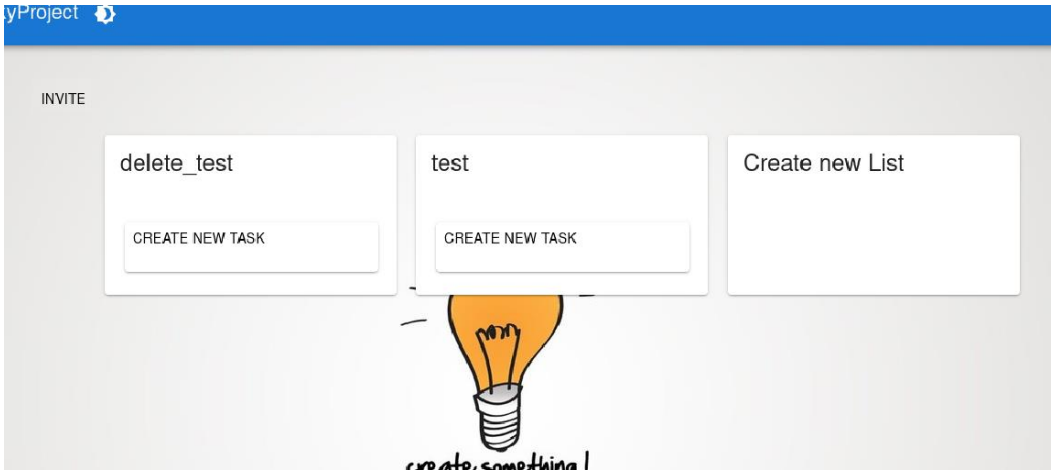


Рисунок 17. Вигляд робочого простору

Розробники та адміністратори можуть створювати (зокрема оновлювати та видаляти) завдання у кожному із списків завдань. Для цього необхідно внизу потрібного списку натиснути кнопку "CREATE NEW TASK", після чого відкриється звичне нам модальне вікно із полями, які нам необхідно заповнити:

1. Name — назва самого завдання.
2. Date to Start — дата, коли варто почати роботу над цих завданням.
3. Date to Finish — дата, коли необхідно завершити роботу на завданням.
4. Priority — пріоритет завдання (числове значення у межах від 1 до 10, чим більше значення, тим більший пріоритет виконання завдання).
5. Description — більш детальний опис самого завдання.
6. Assigned User — користувач, який повинен працювати над цим завданням (поле може залишатися порожнім).
7. Tag — тег завдання.

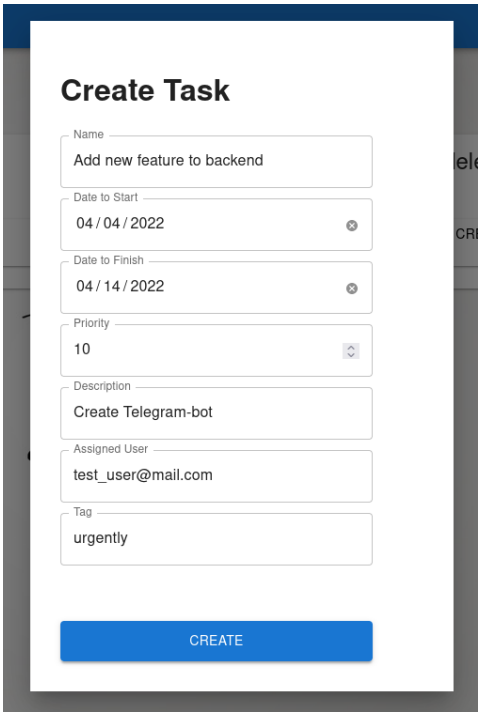


Рисунок 18. Створення нового завдання

Лише адміністратор проєкту має право запрошувати нового користувача до проєкту. Для цього йому потрібно натиснути кнопку "INVITE", яка є лише у нього на робочому просторі проєкту. Після чого відкриється модальне відкно запрошення нового користувача, де буде потрібно ввести email або ім’я користувача в систем та вибрати його роль у проєкті.

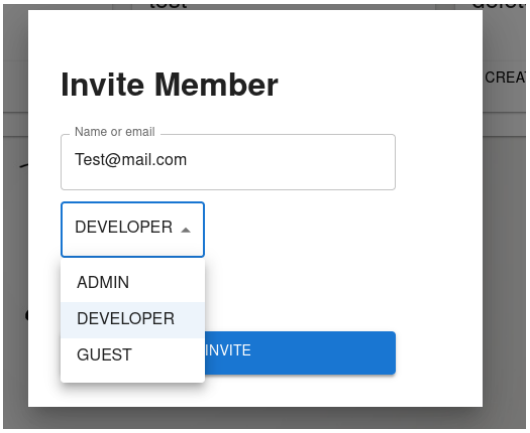


Рисунок 19. Запрошення нового користувача до проєкту

Коли адміністратор надішле запрошення користувачу, то у того в меню навігації з’явиться новий проєкт, для якого у нього будуть доступні дві опції:

1. Погодитись на участь в проєкті, після чого користувача буде переадресовано на робочий простір того проєкту
2. Відмовитись від участі, після чого цей проєкт зникне із меню навігації між проєктами

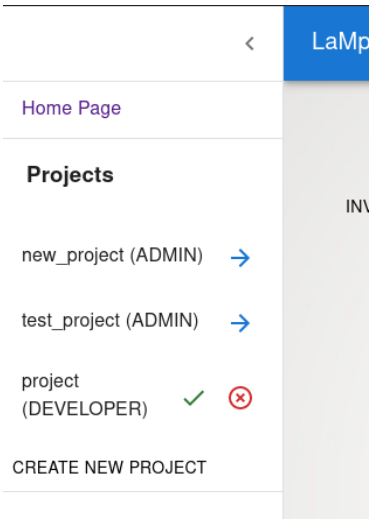


Рисунок 20. Меню навігації, коли користувач отримує запрошення до проєкту

ВИСНОВКИ

Командою було розроблено прототип системи підтримки керування IT-проєктами, яка надає можливості забезпечити учасникам команди з розробки певного програмного продукту координуватися і таким чином керувати процесом реалізації кінцевого завдання команди. В ході виконання роботи команда провела аналіз схожих систем на ринку та визначили головні риси майбутньої системи. Також було спроєктовано діаграми прецедентів та класів за затвердженими вимогами. Внаслідок цього було розроблено прототип даної системи, так як замороження проєкту, спровокованого початком гарячої фази збройної агресії проти України, сильно сплинуло на можливости даної системи.

Кожен з учасників команди за час розробки системи навчився багато новому та поглибив свої знання з певних галузей. У висновках ми б хотіли зазначити, як розробка командного проєкту вплинула на кожного з учасників.

Для Ольги це був значний досвід роботи в команді над великим проєктом:

- використання Scrum на практиці;
- закріплення навичок в обраних технологіях;
- важливість зустрічей і формулювання ТЗ.

Євгеній виділив для себе наступні корисні навички:

- знайомство зі Scrum;
- робота в екстремальних умовах;
- вивчення нових технологій.

Максим виділив для себе наступні переваги:

- поглиблення навичок роботи в команді;
- важливість проєктування на початкових стадіях роботи;
- новий досвід роботи з технологіями;
- необхідність оптимізації часу роботи над проєктом.

Олександр зазначив наступні переваги:

- покращення своїх комунікативних навичок;
- досвід роботи в команді;
- досвід правильного введення документації, тестування та підсумовування зустрічей;
- робота в екстремальних умовах.

Адальберт набув наступних навичок для себе:

- проведення Scrum-зустрічей;
- користування Trello;
- побудова use-case діаграм в Creately;
- керування командою, розподіл задач учасникам;
- покращення своїх Soft skills.

Таким чином кожен з учасників команди після завершення роботи над проєктом отримав для себе новий досвід та удосконалив свої знання з різних галузей.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Creately [Електронний ресурс]. – Режим доступу: https://creately.com/
2. Google Sheets [Електронний ресурс]. – Режим доступу: https://www.google.com/intl/uk_ua/sheets/about/
3. Trello [Електронний ресурс]. – Режим доступу: https://trello.com/
4. Java [Електронний ресурс]. – Режим доступу: https://www.java.com/
5. GitHub [Електронний ресурс]. – Режим доступу: https://github.com/
6. Heroku [Електронний ресурс]. – Режим доступу: https://www.heroku.com/
7. Spring Framework [Електронний ресурс]. – Режим доступу: https://spring.io/projects/spring-framework
8. Spring [Електронний ресурс]. – Режим доступу: https://spring.io/
9. PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/
10. MaterialUI [Електронний ресурс]. – Режим доступу: https://mui.com/
11. React [Електронний ресурс]. – Режим доступу: https://uk.reactjs.org/
12. Hamcrest [Електронний ресурс]. – Режим доступу: http://hamcrest.org/

ДОДАТКИ

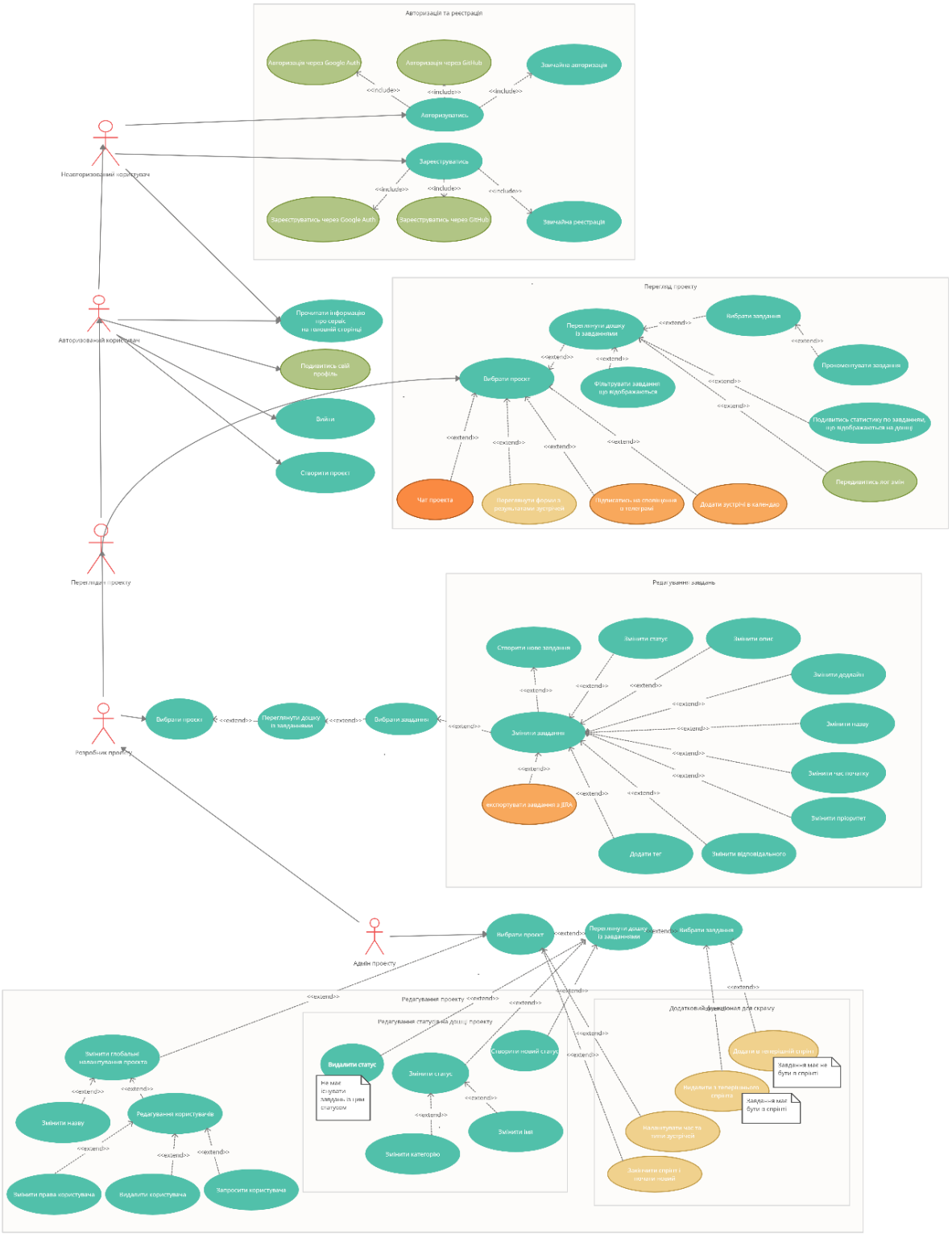
ДОДАТОК А – ХРОНОЛОГІЯ РОЗРОБКИ ПРОЄКТУ

№	Дата	Мета зустрічі	Бережняк Ольга	Васильєв Євгеній	Васильчук Максим	Галавай Олександр	Макарович Адальберт	Результат зустрічі
1.	02.02.2022	Знайомство. Визначення назви команди. Попереднє обговорення завдання	Ознайомитися з предметною областю та продуктами, які представлені в ній.	Ознайомитися з предметною областю та продуктами, які представлені в ній.	Ознайомитися з предметною областю та продуктами, які представлені в ній.	Ознайомитися з предметною областю та продуктами, які представлені в ній.	Ознайомитися з предметною областю та продуктами, які представлені в ній.	Відбулося знайомство учасників команди. Затверджено назву команди. Призначення зустрічі на 4.02. Вивчити предметну область, знайти приклади. Попередньо розподілено ролі в команді
2.	04.02.2022	Узгодження ролей. Визначення основних функціональних можливостей майбутньої системи	Заповнення поля очікування від проєкту у файлі "Процес розробки проєкту". Ознайомитися, які ще є продукти в даній предметній області, та обдумати, які фічі можуть бути використані.	Заповнення поля очікування від проєкту у файлі "Процес розробки проєкту". Ознайомитися, які ще є продукти в даній предметній області, та обдумати, які фічі можуть бути використані.	Заповнення поля очікування від проєкту у файлі "Процес розробки проєкту". Ознайомитися, які ще є продукти в даній предметній області, та обдумати, які фічі можуть бути використані.	Заповнення поля очікування від проєкту у файлі "Процес розробки проєкту". Ознайомитися, які ще є продукти в даній предметній області, та обдумати, які фічі можуть бути використані.	Заповнення поля очікування від проєкту у файлі "Процес розробки проєкту". Ознайомитися, які ще є продукти в даній предметній області, та обдумати, які фічі можуть бути використані.	Результат зустрічі було записано у відповідний файл, в якому були визначені основні ідеї та можливості майбутньої системи. Затверджено ролі в команді
3.	07.02.2022	Затвердження основних можливостей майбутньої системи. Обговорення можливих інструментарії в для проєктування і розробки завдання.	Ознайомлення з дизайнами і зовнішніми структурами уже існуючих проєктів, щоб мати основу для проєктування власного дизайну	Заповнення поля очікування від проєкту у файлі "Процес розробки проєкту". Обдумати структуру бази даних для майбутньої системи.	Ознайомлення з дизайнами і зовнішніми структурами уже існуючих проєктів, щоб мати основу для проєктування власного дизайну	Формулювання постановки завдання, технологій проєктування і розробки	Обдумати структуру бази даних для майбутньої системи.	Було майже заповнено необхідні дані у файлі "Процес розробки проєкту". Затверджено проміжний перелік всіх можливостей майбутньої системи.
4.	09.02.2022	**Зустріч із замовниками.** Аналіз І тижня: проміжні результати і підсумки	Знайти, які графічні "фічі" можуть бути застосовані для оформлення проєкту.	Засетапити гіт і зробити початковий коміт для бекенда	Засетапити гіт і зробити початковий гіт для фронтенда. Визначити особливості авторизації за допомогою Google OAuth	Розпочати розробку технічного завдання згідно із стандартами.	Спроєктувати та намалювати схему бази даних для проєкту. Розробити первісну use-case діаграму.	Замовниками було затверджено попередню постановку завдання. Встановлено терміни двох демо та завершення

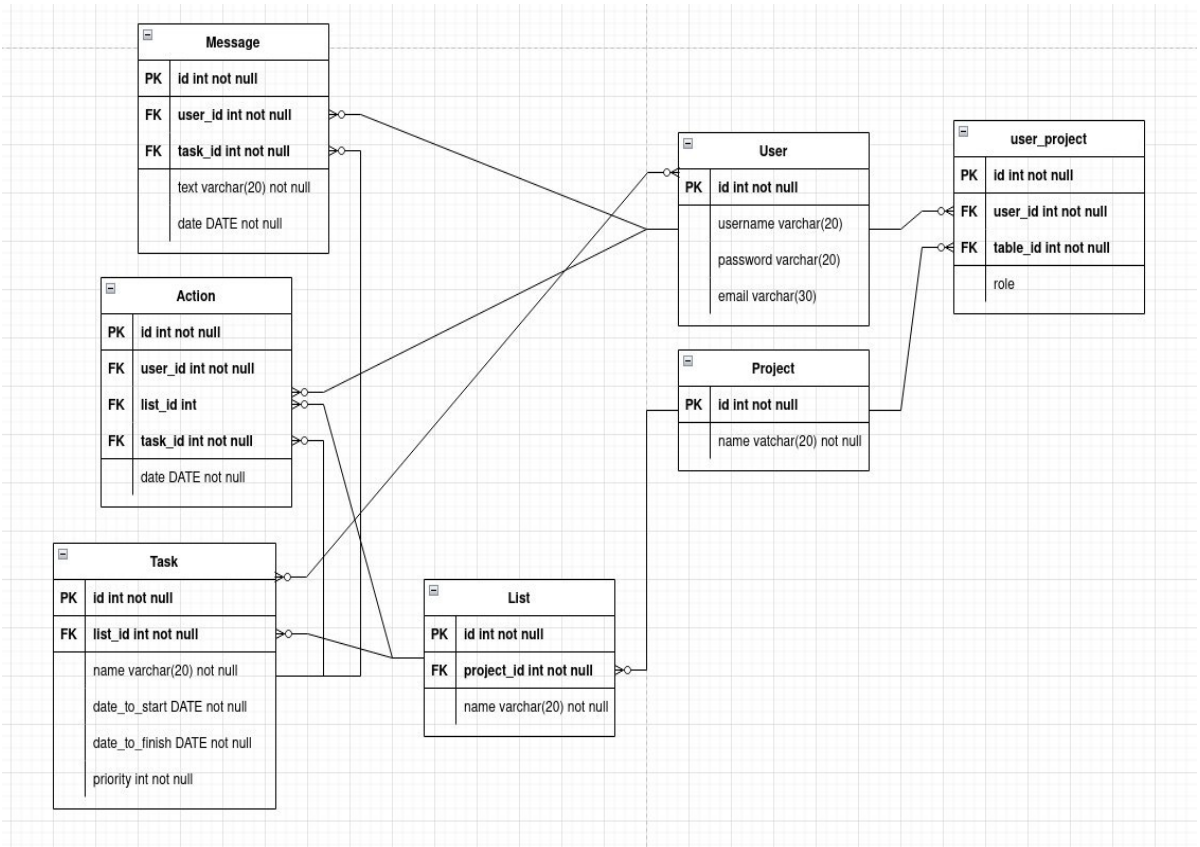
					та розпочати його реалізацію (для фронта).			робіт. Попередньо обрана "фішка" проєкту. Вирішено провести ширше обговорення наступного дня.
5.	10.02.2022	Визначити пріоритетні таски на наступний тиждень. Отримання учасниками проєкту завдань на тиждень.		Визначити особливості авторизації за допомогою Google OAuth та розпочати його реалізацію (для бека).				Формування backlog-файлу у системі керування проєктами Trello. Підключення всіх учасників до Trello і Github. Обміркувати, які ще таски можна додати у спринт.
6.	14.02.2022	Сформулювати спринт на наступний тиждень	Намалювати головну сторінку та подумати про лого проєкту. Розробити початковий дизайн форми для авторизації.	Намалювати базову схему бази даних.	Закодити базову сторінку для авторизації.	Зробити юридичні моменти технічного завдання.	Розробити use-case діаграму. Засетапити базу даних на PostgreSQL.	Визначено основні задачі на наступний тиждень, які були записані у спринт. Розподілено задачі учасникам.
7.	16.02.2022	**Зустріч із замовниками**. Аналіз II тижня.	Намалювати шаблони інших сторінок проєкту	Внести зміни в діаграми прецедентів та класів	Внести зміни в базову сторінку авторизації	Внести зміни в технічне завдання згідно з порадами замовників	Внести зміни в діаграми прецидентів та класів	Представлено перші версії use-case діаграми та діаграми класів.
8.	19.02.2022	Проведення аудиту поточних завдань у спринті, зміна їх пріоритетності.	Внести зміни до шаблону всіх сторінок проєкту					Внесено зміни до use-case діаграми. Були представлені шаблони сторінок для фронтенду.
9.	21.02.2022	*Дорадча зустріч*: обговорення графічних елементів для фронту	Спроєктувати ендпоінти для фронтенту.	Розпочати розробку API для бекенда. Сформулювати документ з описами ендпоінтів та результатами їх виведення.	Розпочати розробку API для бекенда. Спроєктувати ендпоінти для фронтенту.	Узагальнивши обговорення на зустрічі, змінити вимоги до технічного завдання	Створення основного функціоналу (CRUD) для керування проєктами на бекенді.	Обрано варіанти логотипів для майбутнього проєкту
10.	22.02.2022	Сформулювати попередній перелік завдань на спринт нового тижня.						Пріоритезація тасків у backlog-файл. Винесеення на

								голосування шаблону лого для проєкту. Затвердження шаблону лого. Затверджено способи фільтрації та сортування завдань у проєкті.
11.	23.02.2022	**Зустріч із замовниками.** Аналіз III тижня.	Розробка логотипу продукту	Розробити функціонал бекенду згідно з use-case діаграмою та діаграмою класів	Створення шаблону для фронтенду	Сформовати остаточне технічне завдання	Координація учасників команди, їх консультування	Представлено результати роботи: діаграми прецедентів та класів, шаблони сторінок на фронтенд, базове технічне завдання.
Указами Президента України "Про введення воєнного стану в Україні" №64/2022 від 24.02.2022 та "Про продовження строку дії воєнного стану в Україні" №133/2022 від 14.03.2022 та наказом ректора Київського національного університету імені Тараса Шевченка від 25 лютого 2022 року № 118-32 "Про організацію освітнього процесу у період з 28 лютого по 13 березня 2022 року" розробка проєкту була заморожена на період від 24 лютого 2022 року до 3 квітня 2022 року								
12.	05.04.2022	Обговорення поточного стану справ. Пошук шляхів для завершення розробки проєкту	Проводити дорадчі консультації для оформлення фронтенду	Проводити дорадчі консультації з приводу структури бекенду	Розробка фронтенду за основі готового бекенду з додатковими консультаціями	Почати злиття технічного завдання до звіту про проєкт	Проводити координацію учасників групи, консультувати з питань оформлення звіту до проєкту	Розроблено прототип бекенду проєкту з спрощеними вимогами.
13.	06.04.2022	**Зустріч із замовниками.** Аналіз поточних справ				Зробити звіт за проробленою роботою		

ДОДАТОК Б – USE-CASE ДІАГРАМА



ДОДАТОК В – ДІАГРАМА КЛАСІВ



Примітка: діаграма класів може частково не відповідати реальній структурі бази даних внаслідок дій обставин нездоланної сили.

СТВОРЕННЯ СИСТЕМИ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ "ПандоRA"

Олександра Євтушенко, Павло Циронін, Нікіта Ткач,
Дмитро Прохорчук, Антон Юрченко

ВСТУП

Актуальність обраної теми. Розробка якісного програмного продукту – складний та довготривалий процес, що поєднує багато людей. Тому, керівники компаній, проджект менеджери, продукт овнери постійно шукають шляхи покращення та оптимізації процесу створення продукту. Надзвичайно корисним інструментом у цьому випадку виступають системи управління ІТ-проєктами, які є потужним засобом організації діяльності команд. Тому, розробка такого додатку є дуже актуальною.

Мета і завдання роботи. Метою роботи є розробка веб-застосунку, який є зручним прикладним інструментом для управління ІТ-проєктами у невеликих командах. Ключовою особливістю додатку є зрозумілий та швидкий інтерфейс. Для досягнення цієї мети було поставлено наступні завдання:

- вивчення існуючих на ринку застосунків для управління ІТ-проєктами
- огляд та вибір існуючих технологій для проєктування та реалізації веб-додатку
- розробка технічного завдання до програмного продукту
- розробка інтерфейсу та архітектури додатку

ОГЛЯД НАЯВНИХ НА РИНКУ СИСТЕМ

Звісно, на ринку представлена велика кількість додатків для управління маленькими та великими проєктами з різних сфер життя. Але розглянемо деякі найбільш відомі з них.

Система Jira

Jira - це інструмент управління проєктами, що підходить для різних випадків, від керування вимогами і сценаріями тестування до agile-розробки програмного забезпечення.

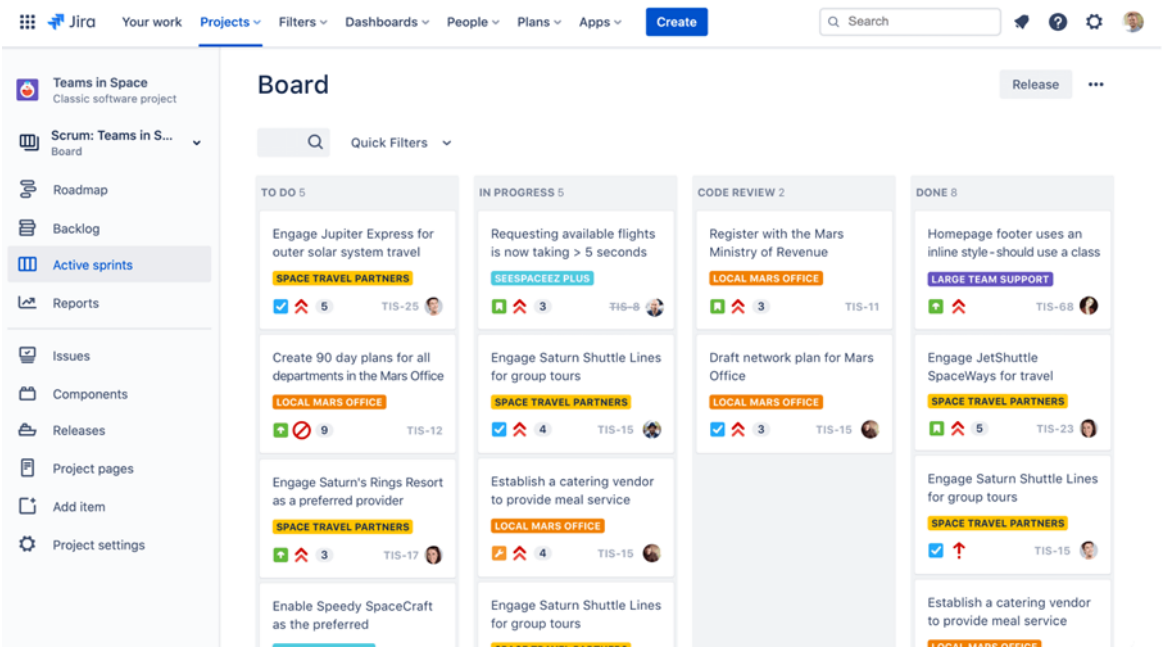


Рисунок 1. Інтерфейс Jira

Переваги: надає величезний набір функціоналу та можливостей для налаштування, надає потужний API, доволяє зручно планувати спринти, наявний Agile-беклог, є можливість розрахунку швидкості та обсягу роботи команди за спринт. Прогрес роботи з кожного проекту можна виміряти та візуалізувати.

Недоліки: система складна у використанні, особливо для не технічних спеціалістів, інтерфейс часто змінюється, складно налаштовувати власні робочі процеси у кожному окремому проекті, замало можливостей для автоматизації.

Система Trello

Trello – це одна з найпопулярніших систем управління проектами в режимі онлайн, яка має особливий попит серед невеликих компаній і стартапів. Вона дозволяє ефективно організовувати роботу за японською методологією канбан-дошок.

Рисунок 2. Інтерфейс Trello

Переваги: основні операції із завданнями вимагають не більше двох кліків, інформативні дошки, гнучке налаштування карток. Ідеально підходить для найпростіших лінійних завдань. Легко ділитися картками, запрошувати сторонніх людей на дошки, додавати швидкі нотатки та показувати їх колегам.

Недоліки: незручний для великої кількості проєктів, довго шукати потрібну дошку, мало можливостей для класифікації карток. Для великих проєктів стандартних звітів недостатньо.

Система Miro

Miro — це платформа для спільної віддаленої роботи за допомогою онлайн-дошки. Дошка підходить для складання проектів, креативу, дизайн-концепцій, брейнстормінгу та освітніх цілей. На дошку можна додавати завантажені файли та документи, малювати, робити нотатки та вставляти стікери. Для створення дошки можна використовувати готові шаблони або створити її з нуля.

Переваги: можливість додавати файли та документи на картки, клеїти стікери, робити нотатки, малювати, відзначати важливе маркером, складати mind-карти та to-do листи. Будь-яку дошку можна експортувати у вигляді картинки або PDF, можна розмістити на своєму сайті або у стрічці соцмереж. Можна спільно редагувати зображення, відео, PDF, Google-документи. Є текстові, голосові та відеочати, а також перегляд дошки у реальному часі, коли учасники бачать курсори та дії один одного. Є готові шаблони для управління проектами та різноманітних бізнес-процесів.

Недоліки: обмеження на розмір дошки для малювання та нотаток, її не можна зробити більше або поєднати з іншою дошкою. Немає пароля для дощок, який міг би захистити його від редагування або випадкового видалення анонімним користувачем. Складно відстежити історію змін на дошці: хто, коли і що зробив, хто якийсь елемент додав.

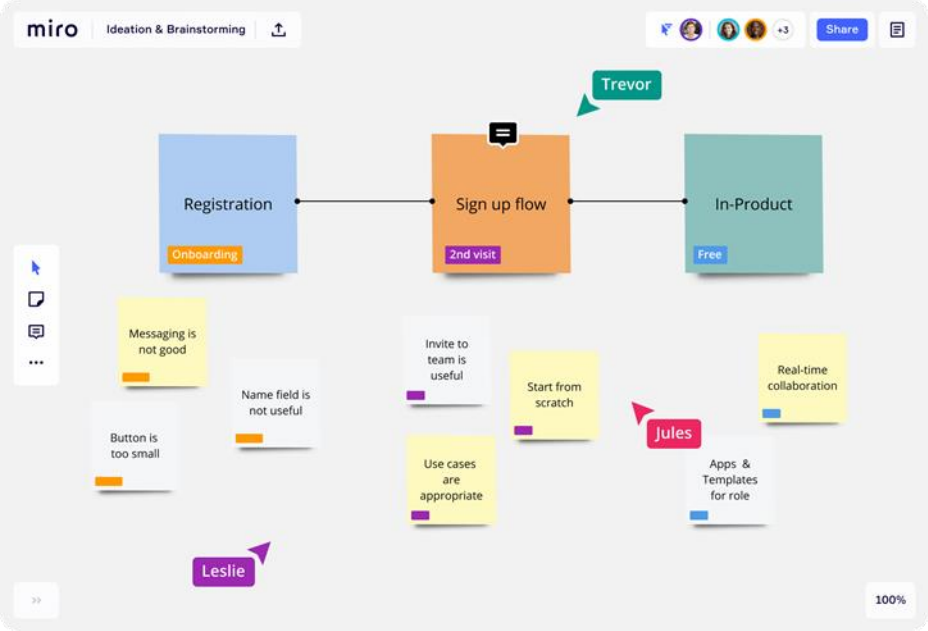


Рисунок 3. Інтерфейс Miro

Система Google Tasks

Google Tasks - інтегрований з Gmail додаток для керування задачами. Програма дозволяє створювати завдання та підзадачі, а також створювати нагадування для окремих дат та додавати до них нотатки. Завдання можна систематизувати за датою або пріоритетом, змінюючи їх місцями у списку.

Переваги: дуже простий інтерфейс, дозволяє легко скопіювати дані по завданням та ділитися ними, інтеграція з гугл-аккаунтом (поштою, зустрічами).

Недоліки: нестача функціоналу. У цій системі управління завданнями немає можливостей для організації хоч скільки серйозних робочих процесів, не можна призначити завдання на когось, вказати його пріоритет, додати до нього позначки.

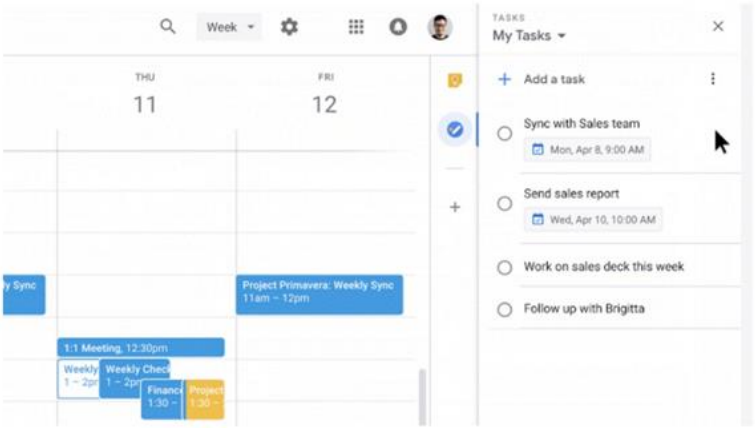


Рисунок 4. Інтерфейс Google Tasks

Система YouTrack

YouTrack – це інструмент управління проектами, який легко адаптується до процесів. Можна планувати проекти та відстежуйте завдання, використовувати Agile-дошки, організовувати спринти та релізи, вводити базу знань, використовувати звіти та панелі моніторингу, створювати робочі процеси

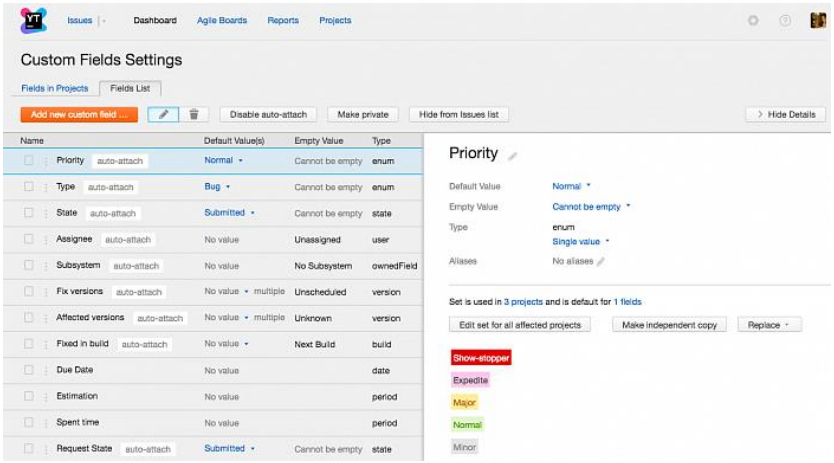


Рисунок 5. Інтерфейс YouTask

Переваги: швидка генерація звітів, просте використання клавіатурних "гарячих клавіш" та командного синтаксису, усіма робочими процесами легко керувати за допомогою клавіатури. Дуже гнучка система: можна налаштувати атрибути задач під себе (пріоритети помилок, типи помилок, дані про спрінти та інше). Є крута можливість пакетно змінювати поля завдань, призначати виконавців, пов'язувати між собою завдання тощо. Зручний інтелектуальний пошук за допомогою пошукових запитів з підказками та підсвічуванням помилок.

Недоліки: складний інтерфейс, менше можливостей для інтеграції порівняно з більш відомими системами.

Система Monday

Monday - платформа для спільної роботи та управління проектами. Найчастіше використовується у відділах продажу та маркетингу для контролю виконання завдань за допомогою Workflow, оптимізації вирви продажів та як CRM. Простий та зрозумілий у використанні інструмент.

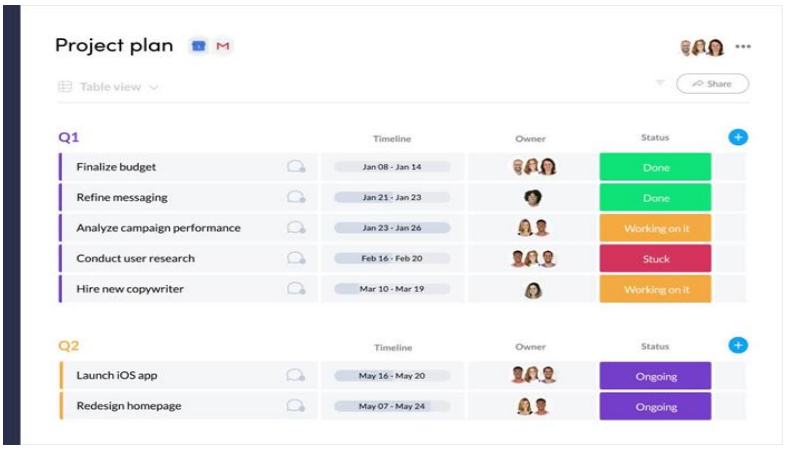


Рисунок 6. Інтерфейс Monday

Переваги: гнучкість, великий простір для налаштувань системи під себе, легко відстежувати ключові показники ефективності на рівні співробітника, відділу чи бізнесу загалом, лаконічні та інтуїтивно зрозумілі інтерфейси, пропонує багато шаблонів робочих дощок на різні завдання, наприклад, організацію цілого відділу продажів. Кожне завдання можна деталізувати: визначити рейтинг, оцінки, статуси, лінки. Можна створити окремі дошки для роботи з партнерами та клієнтами.

Недоліки: не зберігаються завершені завдання, зникають підзавдання після їх архівації, не підходить для великих проєктів, замало можливостей для автоматизації рутинних процесів.

ВИМОГИ ДО СИСТЕМИ ТА ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Технічні вимоги.

Мета і актуальність проєкту

Веб-система "ПандоRA" розробляється як легке в користуванні рішення для організації роботи IT-проєктів для невеликих команд.

Користувач/цільова група

- проектні менеджери (Project manager)
- власники продукту (Product owner)
- власники компаній (CEO)
- розробники
- тестувальники
- дизайнери
- інші працівники компанії, що безпосередньо беруть участь у розробці продукту

Системні вимоги

На пристрою користувача повинен бути встановлений один з браузерів

- Google Chrome
- Opera
- Microsoft Edge
- Mozila Firefox
- Safari

та бути ввімкнена функція підтримки JavaScript.

Вимоги до стилістичного оформлення додатку

Стилістичне оформлення додатку має відповідати дизайн-макету, погодженого з командою розробників та керівником проєкту. Використання колірних схем, графічних елементів, шрифтів погодженого дизайн-макету в процесі створення додатку є обов’язковим.

Вимоги до верстки та програмування

Система повинна бути оптимізованою і зверстаною у відповідності зі стандартами W3C для роботи в різних стабільних версіях браузерів (без помилок). Всі стилі потрібно максимально прибрати в зовнішні файли CSS, включаючи зображення, що відносяться до дизайну. Всі JS скрипти повинні бути оптимізовані на максимальну продуктивність системи і заховані в зовнішні файли JS. Верстка сторінок Системи повинна бути виконана з урахуванням максимальної продуктивності Системи; код чистий без помилок і без зайвих тегів. Сторінки Системи не повинні містити флеш-контент.

Вимоги до вікон додатку

Система повинна забезпечувати коректне відображення даних в наступних браузерах актуальних версій:

- Google Chrome;
- Microsoft Edge
- Opera

- Mozilla Firefox
- Safari

Структура та опис системи

Мова додатку

Основна мова додатку - українська. Задля збереження коректності термінології, деякі загальноприйняті назви, пов’язані з IT-сферою, подані англійською.

Ролі в системі

Функціонал різних користувачів системи подано на Use-case діаграмі (Додаток А).

У ролі Адміністратора системи може виступати будь-який представник IT-компанії, який має повноваження керувати процесом розробки продукту. У системі передбачено наявність лише одного Адміністратора.

У ролі Користувача можуть виступати будь-які співробітники компанії, які беруть безпосередню участь у розробці програмного продукту. Кількість Користувачів не обмежена.

Користувач має доступ до головної сторінки робочого простору. Адміністратор має доступ до головної сторінки робочого простору та сторінки адміністратора. Додати нового користувача до робочого простору може тільки адміністратор, використовуючи електронну адресу працівника.

Кожен Користувач чи Адміністратор може бути учасником лише одного проекту (робочого простору).

Реєстрація

Користувач може зареєструватися в Системі. Сторінка реєстрації містить:

- текст заголовок: "Реєстрація";
- текст і поля для реєстрації:
- прізвище*;
- ім’я*;
- пароль*;
- електронна пошта*.
- кнопка "Реєстрація";
- кнопка переходу до авторизації "Авторизація".

* поля обов'язкові для заповнення

Авторизація

Користувач може авторизуватися в Системі. Сторінка авторизації містить:

- текст заголовок: "Login";
- кнопка переход до реєстрації "Register";
- текст і поля для авторизації:
- пошта*;
- пароль*.

* поля обов'язкові для заповнення

Відновлення паролю

Користувач може відновити пароль до свого облікового запису в Системі. Сторінка відновлення пароля містить:

- текст заголовок: "Забув пароль";
- текст: "Введіть адресу Вашої пошти";
- текст і поля для заповнення:
- пошта*;
- кнопка "Відновити".

Після натискання на кнопку "Відновити" відкривається сторінка "Відновлення пароля" з вмістом: "Ми відправили вам лист з посиланням для зміни пароля. Будь ласка, перевірте свою електронну пошту і натисніть на посилання, щоб продовжити."

Користувачеві приходить лист з наступним вмістом: "Вітаємо, %username%! Ви отримали цей лист, тому що Ви запросили відновлення пароля на … . Щоб скинути пароль, перейдіть за наступним посиланням або скопіюйте і вставте його в веб-браузер: %посилання% Якщо ви не хочете змінювати свій пароль, видаліть цей лист."

Перейшовши за цим посиланням, користувач потрапляє на сторінку "Відновлення пароля", яка містить:

- текст "Введіть новий пароль нижче";
- текст і поля для заповнення:
- новий пароль;
- кнопка "Підтвердити".

Після натискання кнопки "Підтвердити" завантажується головна сторінка, користувач автоматично авторизований.

Огляд системи для Користувача

Модуль системи, сторінка "Головна сторінка"

У верхньому меню відображається аватар та ім’я користувача Системи.

На сторінці зліва відображається список наявних завдань у проекті, згрупованих за категоріями. Справа зображена діаграма Ганта для відстеження виконання проекту відповідним учасником.

При натисканні на задачу, що належить користувачу, відкривається форма зміни статусу задачі.

Огляд системи для Адміністратора

Модуль системи, сторінка "Головна сторінка"

У верхньому меню відображається пошта та ім’я адміністратора Системи.

На сторінці зліва відображається список наявних завдань у проекті, згрупованих за категоріями. Справа зображена діаграма Ганта для відстеження виконання проекту відповідним учасником.

При натисканні на задачу на діаграмі Ганта, відкривається форма редагування задачі (зміна дедлайну, опису, учасників).

При натисканні на кнопку "Додати учасника" у верхній панелі, можна додавати нових та змінювати ролі учасників проекту.

Тестування. Для тестування буде використано xUnit.net та Moq.

Розподіл ролей. Склад команди та розподіл ролей наведені в Таблиці 1.

Таблиця 1. Розподіл ролей у команді

Євтушенко Олександра	керівник проекту, аналітик, фахівець з документації, модератор
Ткач Нікіта	бекенд-розробник, фахівець з документації, модератор
Циронін Павло	бекенд-розробник, модератор
Прохорчук Дмитро	фронтенд-розробник, тестувальник
Юрченко Антон	фронтенд-розробник, дизайнер

Методологія та система управління проєктами. Для організації роботи команди було вирішено обрати методологію Scrum. Кожен спрінт тривав один тиждень, якому передував Daily Meting під час якого обговорювалось, що було зроблено під час минулого спрінта, що планується зробити на наступному спрінті та які проблеми

виникали під час роботи, та чи вдалось їх вирішити. Облік результатів Daily Meeting виконувався в гугл-таблиці, стан якої на певному етапі проекту зображено на рисунку.

Daily Meetings

	02.02.2022	03.02.2022	09.02.2022	15.02			22.02		
Учасник	Знайомство, генерація ідей з приводу майбутнього проєкту	Уточнення та деталізація функціоналу майбутнього проєкту	Фінальне узгодження функціоналу проєкту, розробка макету дизайну	Зроблено	Планується	Проблеми	Зроблено	Планується	Проблеми
Євтушенко Олександра				Написання ТЗ	Оформлення специфікації	немає	Створення календарного плану проєкту,	Виправлення документації	немає
Ткач Нікіта				Написання ТЗ	Підключення гугл-авторизації	немає	Підключення гугл-авторизації	Верстка сторінки авторизації та	немає
Циронін Павло				Схема БД, Репозиторій GitHub	Створення БД, створення MVC архітектури	немає	Створення БД у MS SQL, створення сутностей та	CRUD учасників та CRUD функціоналу	немає
Прохорчук Дмитро				Початок верстки головної сторінки, вивчення JS	Верстка сторінки та Діаграми Ганта	немає	Верстка головної сторінки: діаграма Ганта,	Верстка головної сторінки	немає
Юрченко Антон				Створення мокапів майбутнього сайту	Вивчення JavaScript, верстка сторінки	немає	хворіє	-	-

Рисунок 7. Звіти Daily Meetings

Для спілкування всередині команди було створено Telegram-чат, за потреби організовувались додаткові дзвінки у Google Meet.

Також, щотижня, кожен учасник команди оцінював свій настрій (рівень задоволеності роботи над проєктом) від шкалі від 1 до 5. Такий підхід дозволяє завжди підтримувати виконавців у стані високої продуктивності, запобігати вигорянню та конфліктам. Як видно на малюнку, команда зберігала високі показники протягом усього проєкту.

Рисунок 8. Стан команди під час розробки проєкту

Необхідною частиною Scrum-методології є канбан-дошка. Її було вирішено створити у додатку Trello, тому що, на нашу думку, він є найбільш зручним для використання у невеликих проєктах, має зрозумілий інтерфейс та можливості для налаштування під потреби команди. Дошку було поділено на наступні блоки:

- All Tasks – список задач на весь проєкт;
- To do – список задач на спрінт з присвоєними виконавцями;
- In progress – задачі, що знаходяться на етапі виконання;
- Testing – задачі, що знаходяться на етапі тестування;
- Done – виконані задачі;
- Backlog – задачі, які з певних причин були відкладені та перенесені на наступні спрінти.

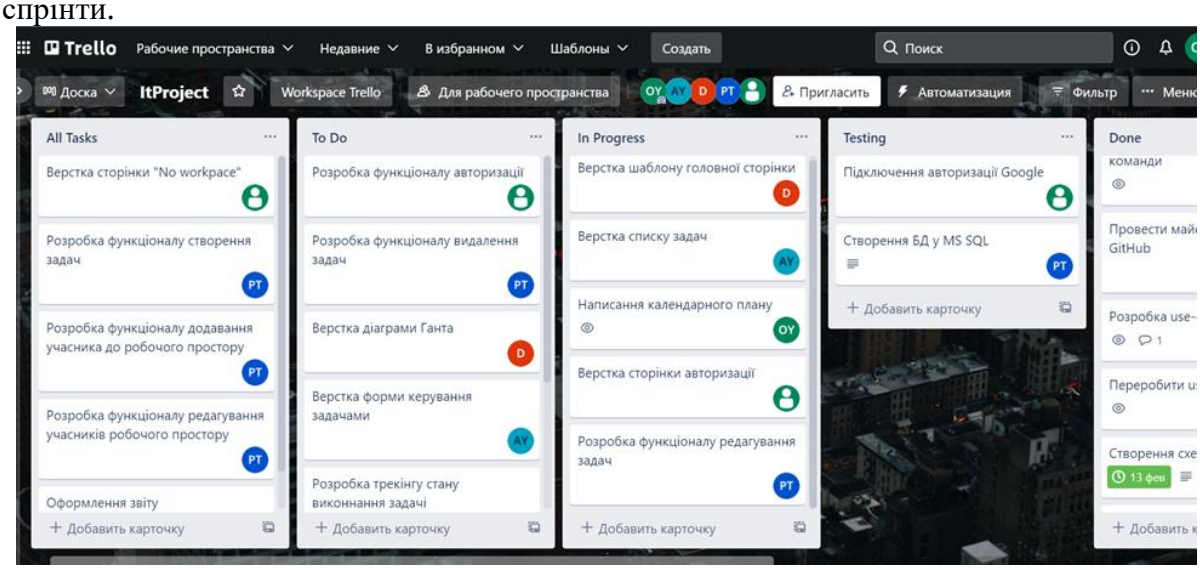


Рисунок 9. Trello-дошка проєкту

На початку проєкту було узгоджено наступний календарний план.

Таблиця 2. Календарний план виконання проєкту

№	Етап	Термін	Результат
1	Узгодження концепції продукту, створення технічного завдання, розробка дизайну	02.02.2022 - 15.02.2022	Технічне завдання, мокапи дизайну
2	Розробка додатку	16.02.2022 - 15.04.2022	Повна версія додатку до тестування
3	Тестування додатку	16.04.2022 - 20.04.2022	Завершений додаток, документація до додатку

Інструментарій. Під час розробки додатку було використано наступні технології.

Для організаці 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
- Balsamiq Mockups – інструмент для створення макету майбутнього додатку, дуже зручний у використанні, не потрібно витрачати час на написання коду, достатньо перемістити основні структурні елементи на шаблон, також було зручно одразу створити різні адаптивний макет для різних пристроїв

• Visual paradigm – онлайн-інструмент для створення діаграм з безліччю необхідних шаблонів

• Git – інструмент спільної роботи над кодом

• Microsoft Visual Studio – IDE для розробки клієнтської та серверної частини

• Microsoft SQL Management Studio – система управління базою даних

Технології розробки: .Net, Microsoft SQLServer, JavaScript (фреймворк Svelte), Html, CSS.

ОПИС ОРГАНІЗАЦІЇ ІНФОРМАЦІЙНОЇ БАЗИ

У якості основної СУБД було використано Microsoft SQL Server. Було створено наступні таблиці:

• Tasks – таблиця для зберігання інформації про завдання

• Epics – таблиця для зберігання груп завдань

• Workspaces – таблиця для зберігання робочих просторів

• AspNetUsers – таблиця для зберігання логіну, хеш паролю і загальної інформації про користувача

• AspNetUserRoles – таблиця для зберігання інформації про ролі користувача

• AspNetUserClaims – таблиця для зберігання інформації про значення Claims користувача для авторизації

• AspNetUserTokens – таблиця для зберігання токенів користувача, отриманих зі сторонніх сервісів

• AspNetUserLogins – таблиця для зберігання інформації про логіни користувача у сторонніх сервісах

• AspNetRoles – таблиця для зберігання інформації про ролі користувачів, що існують в застосунку

• AspNetRoleClaims – таблиця для зберігання інформації про існуючі види Claims застосунку

У Додатку Б зображена схема бази даних застосунку.

РЕАЛІЗАЦІЯ СИСТЕМИ

Розробка клієнтської частини. За допомогою Balsamic Mockups було створено макет додатку згідно створеному технічному завданню.

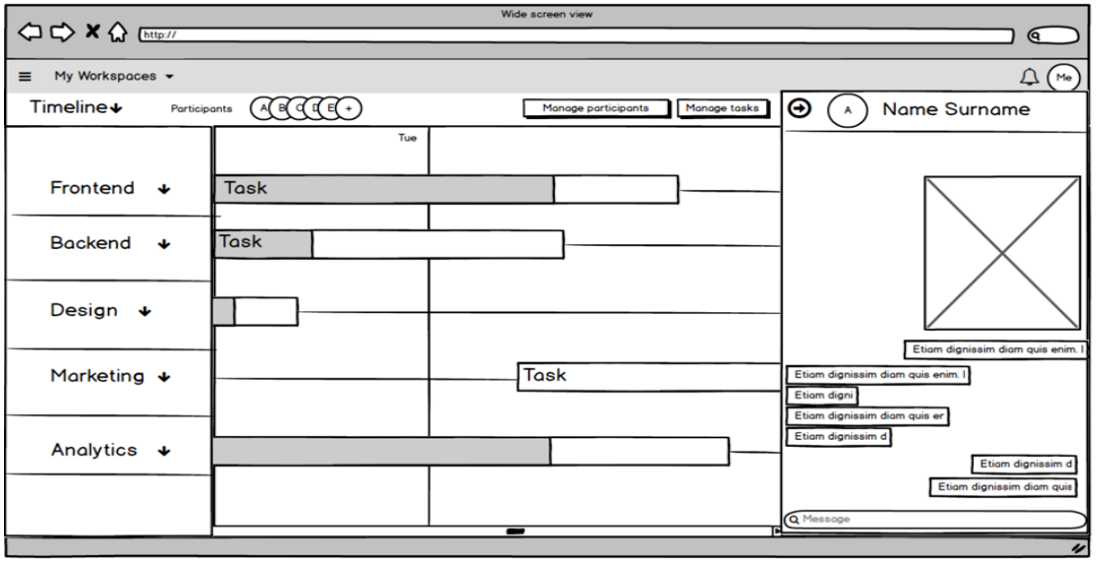


Рисунок 10. Макет головної сторінки

На головній сторінці розміщується список задач, згрупованих за призначенням, діаграма Ганта, кнопки керування задачами та вспливаюче вікно чату.

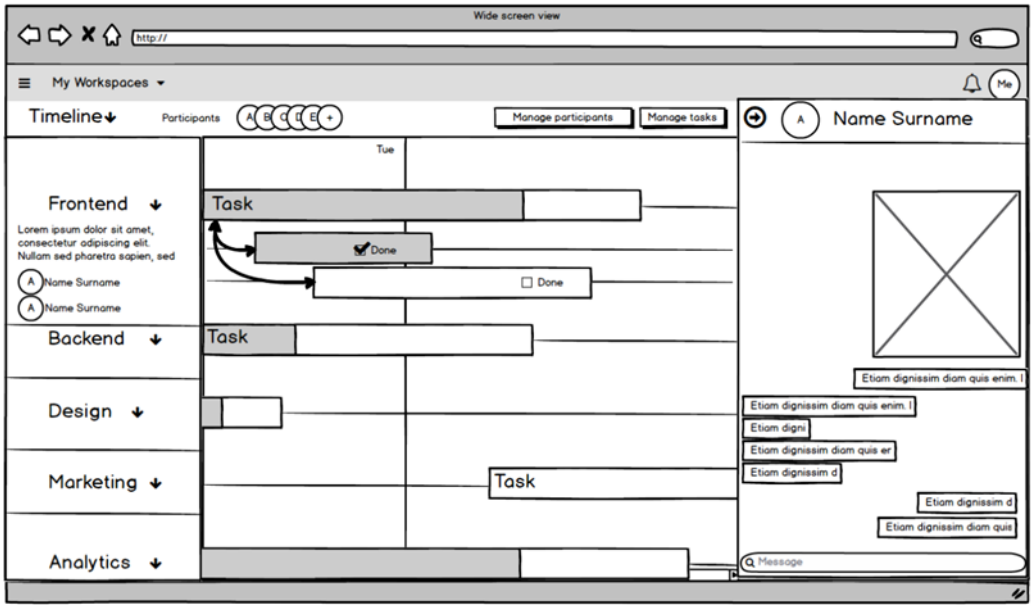


Рисунок 11. Робота з задачами

На діаграмі Ганта, кожна відображена задача є клікабельною, є можливість відстежувати її прогрес.

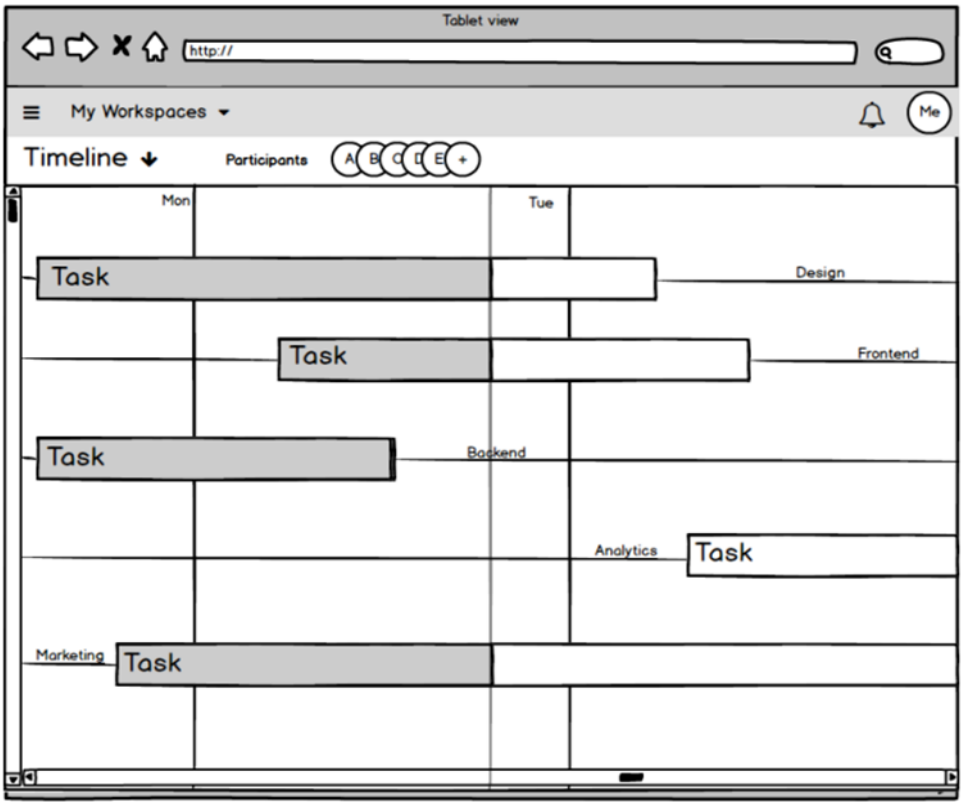


Рисунок 12. Макет головної сторінки (версія для планшетів)

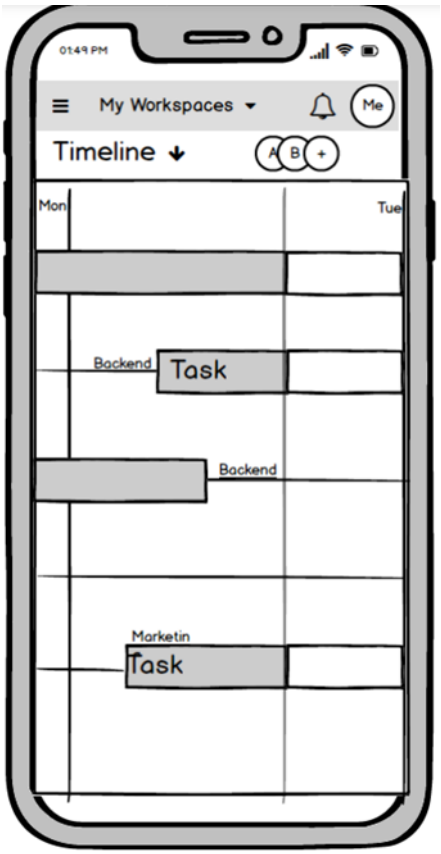


Рисунок 13. Макет головної сторінки (версія для мобільних пристроїв)

Також було створено версії для пристроїв з меншим екраном – планшетів та смартфонів.

Розробка серверної частини.

Для розробки бекенду була використана технологія .NET Core, з використанням паттерну MVC. Концепція патерну MVC передбачає поділ програми на три компоненти:

Модель (англ. model): описує дані, що використовуються в додатку, а також логіку, яка пов'язана безпосередньо з даними, наприклад, логіку валідації даних. Як правило, об'єкти моделей зберігаються у базі даних. У MVC моделі представлені двома основними типами: моделі уявлень, що використовуються уявленнями для відображення та передачі даних, та моделі домену, які описують логіку управління даними. Модель може містити дані, зберігати логіку управління цими даними. У той самий час модель має містити логіку взаємодії з користувачем і має визначати механізм обробки запиту. Крім того, модель не повинна містити логіку відображення даних у поданні.

Представлення (англ. view): відповідають за візуальну частину або інтерфейс користувача, нерідко html-сторінка, через який користувач взаємодіє з додатком. Також уявлення може містити логіку, пов'язану з відображенням даних. У той же час, подання не повинно містити логіку обробки запиту користувача або управління даними.

Контролер (англ. controller): представляє центральний компонент MVC, який забезпечує зв'язок між користувачем та додатком, представленням та сховищем даних. Він містить логіку обробки запиту користувача. Контролер отримує дані, що вводяться користувачем, і обробляє їх. І в залежності від результатів обробки надсилає користувачеві певний висновок, наприклад, у вигляді уявлення, наповненого даними моделей.

Наведемо деякі приклади коду моделі та контролера:

```
public class User : IdentityUser
{

    public int WorkspaceId { get; set; }


    public string Name { get; set; }


    public string Surname { get; set; }

    Ссылок: 0
    public string Picture { get; set; }


    Ссылок: 0
    public virtual Workspace Workspace { get; set; }

    Ссылок: 0
    public virtual ICollection<Task> Tasks { get; set; }

}
```

Рисунок 14. Код моделі

```
public class AccountController : Controller
{
    private readonly UserManager<User> _userManager;
    private readonly SignInManager<User> _signInManager;
    private readonly ApplicationDbContext _context;

    public AccountController(UserManager<User> userManager, SignInManager<User> signInManager)
    {
        _userManager = userManager;
        _signInManager = signInManager;
        var optionsBuilder = new DbContextOptionsBuilder<ApplicationDbContext>();
        optionsBuilder.UseSqlServer("DefaultConnection");
        _context = new ApplicationDbContext(optionsBuilder.Options);
    }
    [HttpGet]
    Ссылок: 0
    public async Task<IActionResult> Login(string returnUrl = null)
    {
        LoginViewModel model = new LoginViewModel
        {
            ReturnUrl = returnUrl,
            ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList()
        };
        return View(model);
    }
```

Рисунок 15. Код контролера

```
public IActionResult ExternalLogin(string provider, string returnUrl)
{
    var redirectUrl = Url.Action("ExternalLoginCallback", "Account", new { ReturnUrl = returnUrl });
    var properties = _signInManager.ConfigureExternalAuthenticationProperties(provider, redirectUrl);
    return new ChallengeResult(provider, properties);
}

Ссылок: 0
public async Task<IActionResult> ExternalLoginCallback (string returnUrl=null, string remoteError = null)
{
    returnUrl = returnUrl ?? Url.Content("~/");

    LoginViewModel loginViewModel = new LoginViewModel
    {
        ReturnUrl = returnUrl,
        ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList()
    };
    if(remoteError != null)
    {
        ModelState.AddModelError(string.Empty, $"Error from external provider: {remoteError}");

        return View("Login", loginViewModel);
    }
    var info = await _signInManager.GetExternalLoginInfoAsync();

    if(info == null)
    {
        ModelState.AddModelError(string.Empty, "Error loading external login information.");
```

Рисунок 16. Код контролера

ТЕСТУВАННЯ

До запуску програми у публічний доступ, тобто, коли вона стане доступною користувачам, важливо переконатися, що ця програма вірно виконує свої функції. Для перевірки програми можна використовувати різні схеми та механізми тестування. Одним із таких механізмів є юніт-тести.

Юніт-тести дозволяють швидко та автоматично протестувати окремі компоненти програми незалежно від решти його частини. Юніт-тести несуть потенційні переваги при розробці, до яких слід віднести не тільки власне перевірку результату та тестування коду, але й інші, як, наприклад, написання слабопов'язаних компонентів відповідно до принципів SOLID. Адже щоб тестувати компоненти програми незалежно один від одного, нам треба, щоб вони були слабкими. А подібна побудова програми надалі може позитивно позначитися на його подальшій модифікації та підтримці.

Для тестування бекенду було використано бібліотеку Xunit, та засоби тестування ASP.NET Core. Приклади тестів:

```
43          [Fact]
44          public void ActionRecognizer_ActionIsAddTask_ReturnActionAddTask()
45          {
46              //Arrange
47              var recognizer = new ActionRecognizer();
48
49              var data = new ICRUDModel<GanttDataSource>()
50              {
51                  added = new List<GanttDataSource>() { task },
52                  changed = new List<GanttDataSource>() { epic },
53                  deleted = new List<GanttDataSource>()
54              };
55
56              //Act
57              UiAction action = recognizer.RecognizeAction(data);
58
59              //Assert
60              Assert.Equal(UiAction.AddTask, action);
61          }
```

Рисунок 17. Код тесту на правильність створення нової задачі

```
103     [Fact]
104     public void ActionRecognizer_ActionIsEditEpic_ReturnActionEditEpic()
105     {
106         //Arrange
107         var recognizer = new ActionRecognizer();
108
109         var data = new ICRUDModel<GanttDataSource>()
110         {
111             added = new List<GanttDataSource>(),
112             changed = new List<GanttDataSource>() { epic },
113             deleted = new List<GanttDataSource>()
114         };
115
116         //Act
117         UiAction action = recognizer.RecognizeAction(data);
118
119         //Assert
120         Assert.Equal(UiAction.EditEpic, action);
121     }
122
```

Рисунок 18. Код тесту на правильність створення нової групи задач

```
143     [Fact]
144     public void ActionRecognizer_ActionIsDeleteEpic_ReturnActionDeleteEpic()
145     {
146         //Arrange
147         var recognizer = new ActionRecognizer();
148
149         var data = new ICRUDModel<GanttDataSource>()
150         {
151             added = new List<GanttDataSource>(),
152             changed = new List<GanttDataSource>(),
153             deleted = new List<GanttDataSource>() { epic }
154         };
155
156         //Act
157         UiAction action = recognizer.RecognizeAction(data);
158
159         //Assert
160         Assert.Equal(UiAction.DeleteEpic, action);
161     }
162
```

Рисунок 19. Код тесту на правильність видалення групи задач

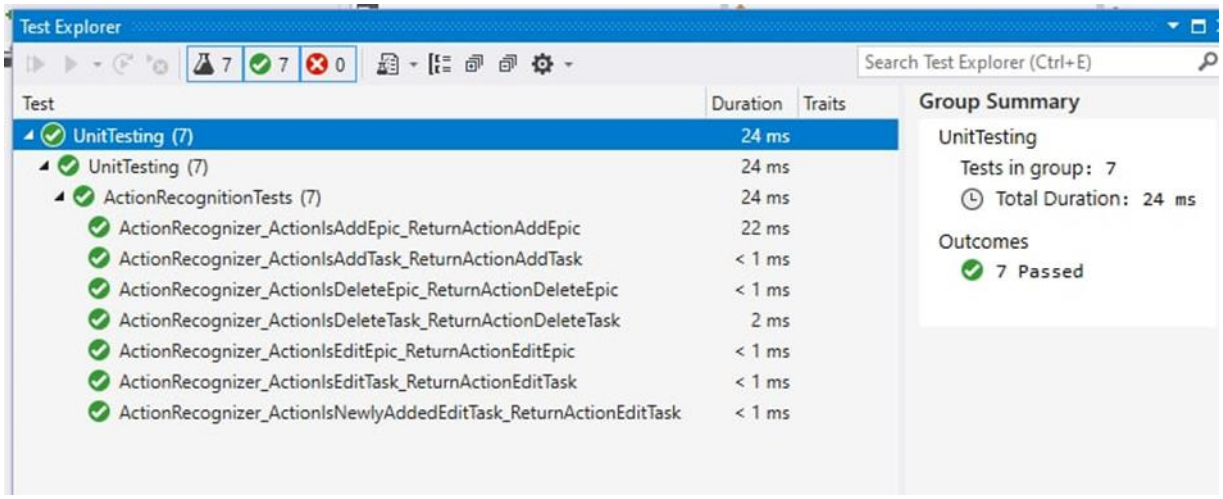


Рисунок 20 – Результат тестів

Тестування frontend частини проводилися мануально, тобто за оцінкою правильності тестувальниками.

ІНСТРУКЦІЯ КОРИСТУВАЧА

Після запуску додатку відкривається вікно авторизації користувача. У поточній версії додатку є можливість входу тільки через Гугл-акаунт. Для того, щоб це зробити, потрібно натиснути кнопку "Login" у верхньому правому куті екрану.

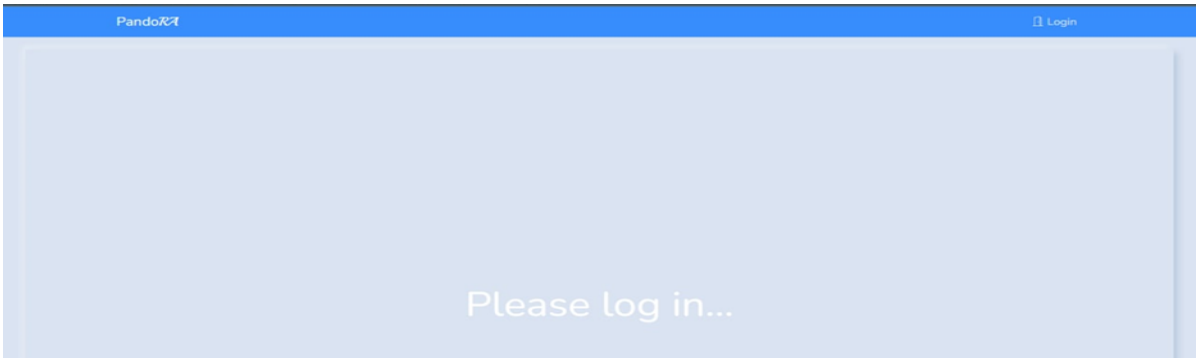


Рисунок 21. Стартова сторінка

Після цього відкриється наступна сторінка авторизації. Треба натиснути кнопку "Google" та вибрати свій Гугл-аккаунт для входу.

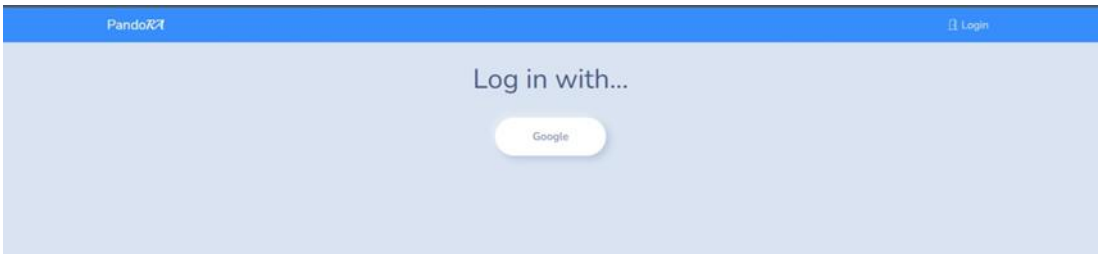


Рисунок 22. Авторизація Гугл

Після успішної авторизації, користувача буде направлено на сторінку робочого простору (якщо його було додано до нього). Для того, щоб створити нову задачу, треба натиснути кнопку "Add" в лівій верхній частині діаграми Ганта.

Після цього відкриється форма створення нового завдання.

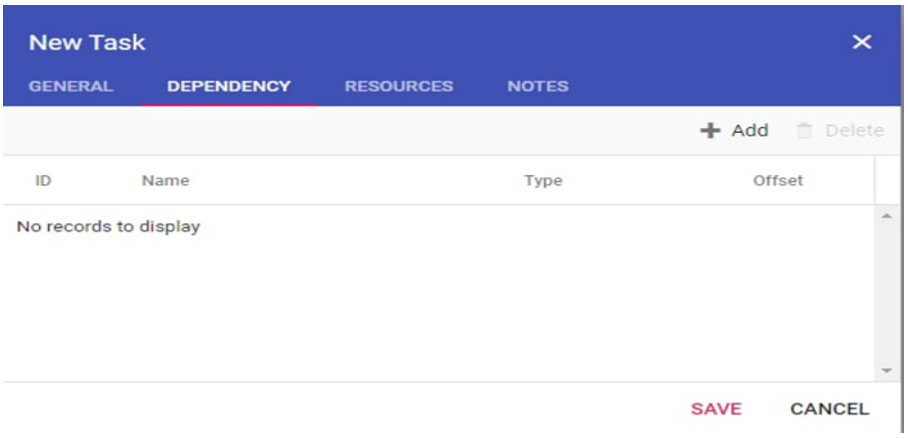


Рисунок 23. Нова форма створення задачі

Треба заповнити поля за зразком і натиснути кнопку "Save".

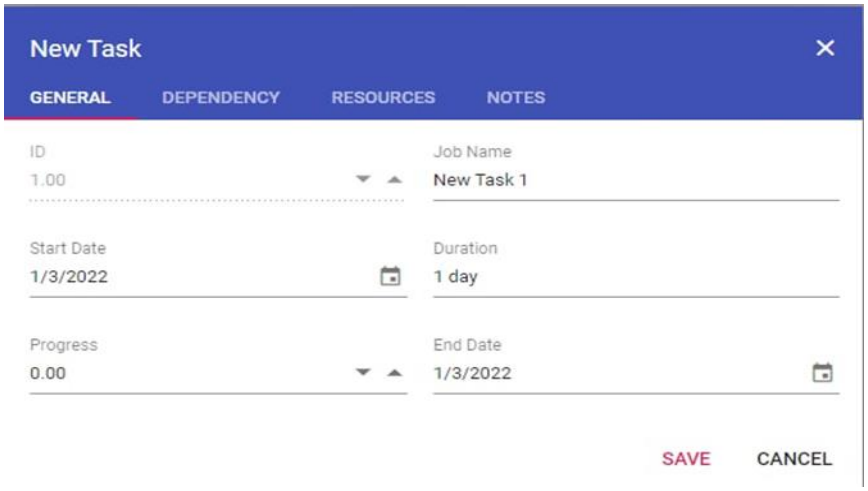


Рисунок 24. Приклад заповнення полів

Після додавання, задачі з'являться на діаграмі Ганта.

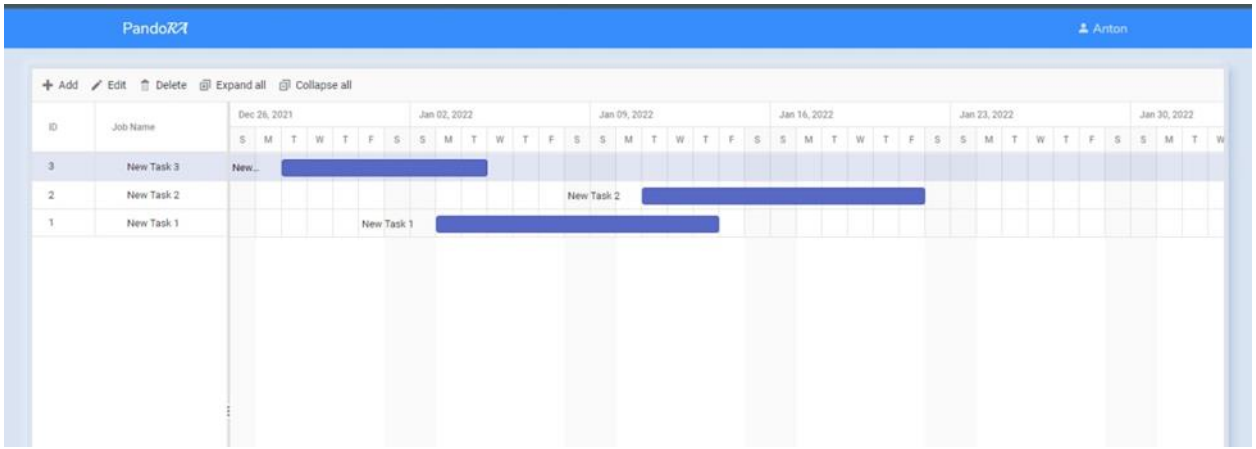


Рисунок 25. Головна сторінка проекту із доданими задачами

Для того, щоб переглянути інформацію про свій акаунт або розлогінитись, треба натиснути на ім'я користувача у верхньому правому куті екрану.

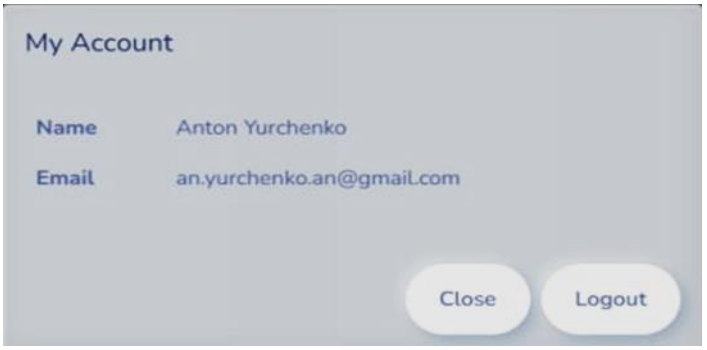


Рисунок 26. Інформація про акаунт

ВИСНОВКИ

У результаті виконаної роботи було вивчено та порівняно існуючі на ринку застосунки для управління IT-проєктами; проведено огляд та вибір технологій для проектування та реалізації веб-додатку; розроблено технічне завдання до програмного продукту; розроблено інтерфейс та архітектуру додатку. У результаті цього було створено веб-застосунок "ПандоRA", який є легким інструментом для планування та відстеження роботи невеликих команд.

Кожен з учасників проєкту здобув для себе нові навички та досвід.

Павло: покращив вміння роботи в команді та планування роботи, поглиблив знання в розробці та підтримці БД, підкріпив навички розробки MVC проектів

Нікіта: отримав досвід роботи в команді, покращив навички в розробці систем

Антон: отримав технічні навички роботи з asp.net mvc, entity framework, svelte.js, git, познайомився з методологією agile (scrum), отримав досвід у плануванні та виконанні групового проекту

Дмитро: навчився працювати в команді, дослідив нові технології, виробив стресостійкість

Олександра: отримала досвід організації та керування проєктом, розробки проєкту з нуля, покращила навички написання технічної документації

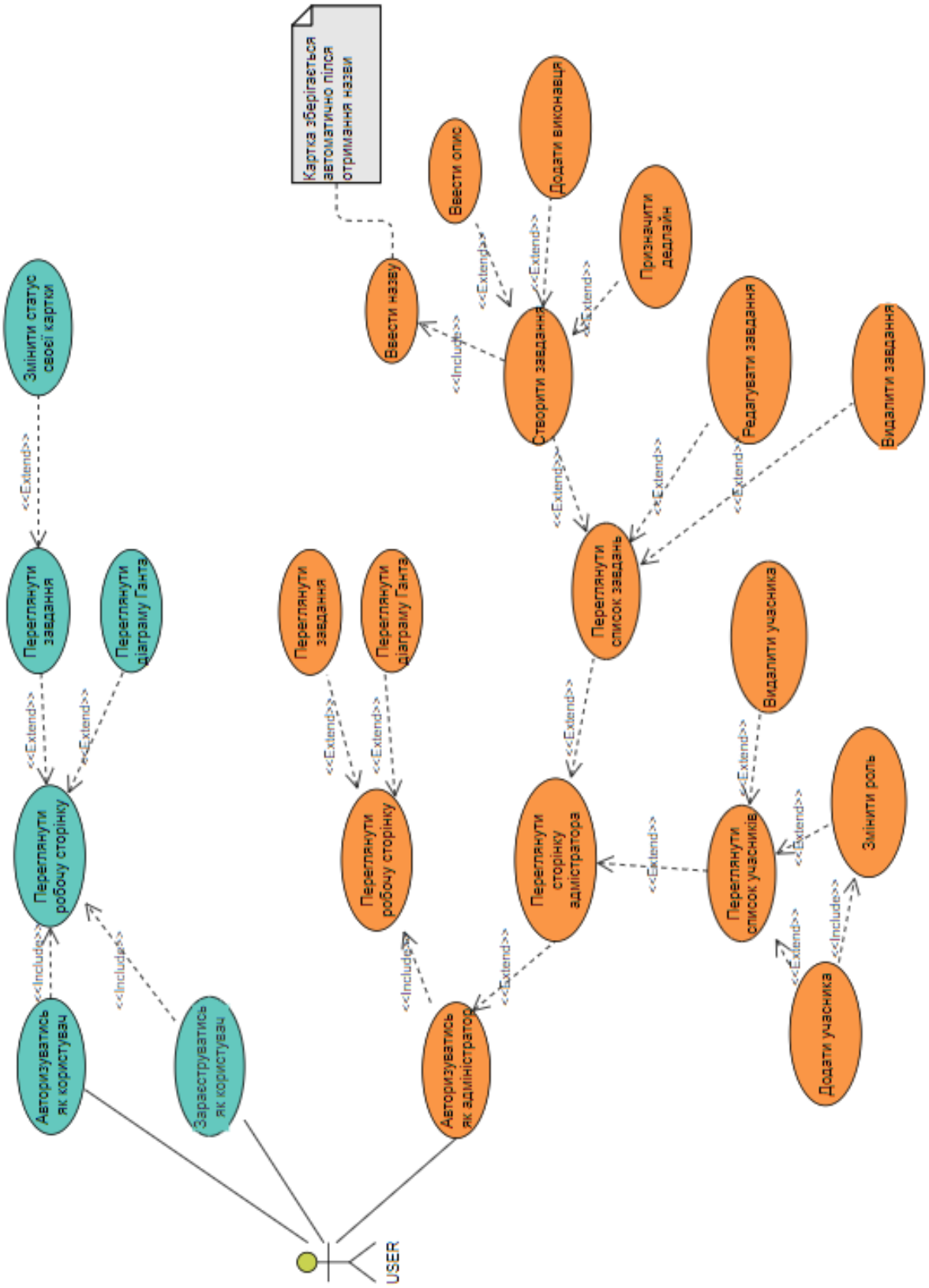
Подальший розвиток проєкту. Звісно, в умовах обмеженого часу, не вдалось реалізувати весь можливий потенціал додатку. Але та частина, яку вдалось втілити, вже є потужною базою для подальших розробок. Основні доопрацювання стосуються розширення функціоналу для охоплення всіх потреб IT-команд, в тому числі, доволі специфічних. Порівняння існуючих додатків допомогло виявити корисні та найчастіше застосовні функції тих чи інших додатків. Також, гарною є ідея провести опитування серед студентів, що працюють на реальних проєктах, та їх колег з приводу того, які функції вони використовують найчастіше та які, навпаки, хотіли би бачити.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

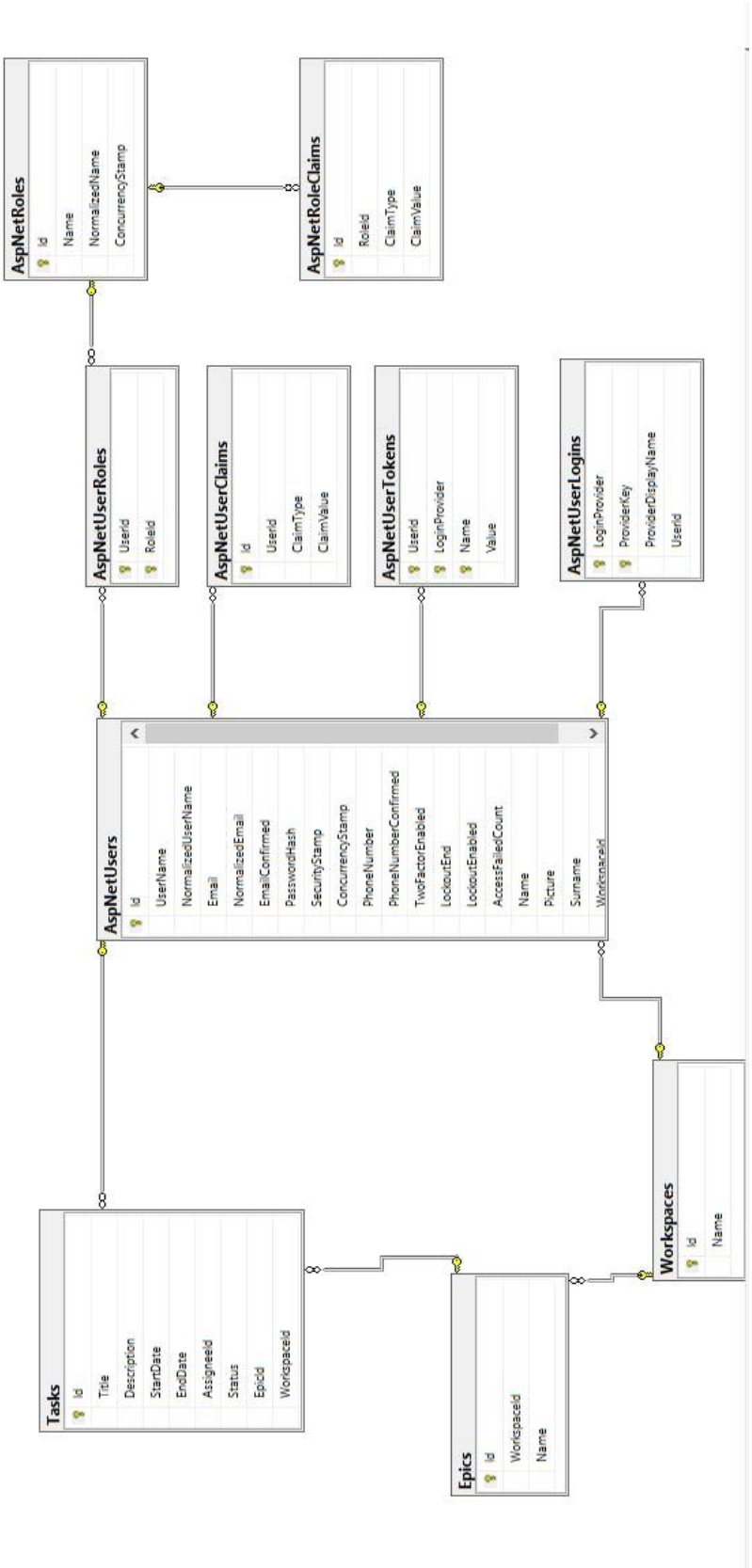
1. Jira software [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/ru/software/jira.
2. Trello Software [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/ru/software/trello.
3. Mirro [Електронний ресурс] – Режим доступу до ресурсу: https://miro.com/.
4. Monday [Електронний ресурс] – Режим доступу до ресурсу: https://monday.com.
5. YouTrack [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/youtrack/.
6. Scrum methodology [Електронний ресурс] – Режим доступу до ресурсу: https://www.digite.com/agile/scrum-methodology/.
7. ASP.NET Core 6.0 [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-6.0.
8. Testing ASP.NET Core services and web apps [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/architecture/microservices/multi-container-microservice-net-applications/test-aspnet-core-services-web-apps.

ДОДАТКИ

Додаток А



Додаток Б


AspNetRoles
Id
Name
NormalizedName
ConcurrencyStamp
AspNetRoleClaims
Id
RoleId
ClaimType
ClaimValue
AspNetUserRoles
UserId
RoleId
AspNetUserClaims
Id
UserId
ClaimType
ClaimValue
AspNetUserTokens
UserId
LoginProvider
Name
Value
AspNetUserLogins
LoginProvider
ProviderKey
ProviderDisplayName
UserId
AspNetUsers
Id
UserName
NormalizedUserName
Email
NormalizedEmail
EmailConfirmed
PasswordHash
SecurityStamp
ConcurrencyStamp
PhoneNumber
PhoneNumberConfirmed
TwoFactorEnabled
LockoutEnd
LockoutEnabled
AccessFailedCount
Name
Picture
Surname
WorkspaceId
Workspaces
Id
Name
Tasks
Id
Title
Description
StartDate
EndDate
AssigneeId
Status
EpicId
WorkspaceId
Epics
Id
WorkspaceId
Name

СТВОРЕННЯ СИСТЕМИ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ "Плінтус"

Всеволод Германюк, Всеволод Гелла, Олександр Кушніренко,
Віталій Ткаченко, Максим Савицький

ВСТУП

В наш час дуже актуальною є проблема організації робочого часу у проєкті. Для допомоги з цим існують системи управління проєктами, які полегшують комунікацію керівників та виконавців, дають їм повну видимість всіх процесів в ході виконання робіт. Ціллю розробки нової системи стало спрощення користування цільовими особами, отримання досвіду розробки та командної роботи. Основними концепціями є мінімалізм та швидкодія.

ОГЛЯД НАЯВНИХ НА РИНКУ СИСТЕМ

Розглянемо переваги на недоліки існуючих систем підтримки управління ІТ-проєктами та виділимо на основі цього вимоги до нашої системи.

Система "Trello"

Веб версія, розширення для Google Chrome, застосунки для Android та IOS.

Переваги:

- обмін файлами;
- простий у опануванні та інтуїтивний у користуванні;
- інтеграція з Gmail, Dropbox, Google Диск, чатами Slack, картами, календарем.

Недоліки:

- не підходить для великих проєктів (важко знайти необхідну дошку чи задачу);
- звіти за результатами роботи робити неможливо;
- відсутній трекер прогресу по часу;
- не можна класифікувати картки;
- відсутність діаграми Ганта;
- Лише Kanban.

Система "Asana"

Веб версія, застосунки для Android та IOS

Переваги:

- можливість розділити великий проєкт на сфери діяльності (створення команд), назначати спостерігачів за задачами;
- декілька варіантів візуалізації (не лише Kanban);
- висока деталізація проєктів;
- інтуїтивний інтерфейс;
- інтеграція з Dropbox, Google Drive, Slack, GitHub, Instagantt, Harvest, поштою;
- створення розширених звітів.

Недоліки:

- немає вбудованого представлення діаграми Ганта;
- немає ієрархії користувачів;
- важко розставляти пріоритети до проєктів;
- не підходить для ведення великої кількості проєктів одночасно.

Система "Jira"

Веб версія, застосунки для Android та IOS

Переваги:

- кастомізація, класифікація задач;
- найкраще заточена під корпоративні стандарти;

- зручна для багтрекінгу;
- Вибір модулей: Scrum, Kanban, Basic software development, Process management, Project management, Task management;
- зручне планування спринтів;
- підходить для команд будь-якого розміру;
- розрахунок та візуалізація швидкості та об’єму роботи, детальні звіти;
- інтеграція з дуже великим списком сервісів.

Недоліки:
- дуже важка в опануванні;
- потребує багато часу на налаштування;
- для деяких проєктів буває багато зайвого функціоналу.

Система "YouTrack"

Веб версія, застосунки для Android та IOS

Переваги:
- швидка генерація звітів та обробка проблем;
- зручний для багтрекінгу;
- широке використання клавіатурних скорочень та командного синтаксису;
- зручний пошук;
- можливість пакетно змінювати поля задач, виконавців;
- наявність вбудованої бази знань.

Недоліки:
- складний інтерфейс, що потребує досвіду користування з ним;
- набагато менше варіантів інтеграцій з зовнішніми сервісами;
- маленький об’єм даних, що може зберігатись.

Система "Notion"

Веб версія, застосунки для Android та IOS

Переваги:
- дуже велика кількість інструментів, може замінити одразу багато сервісів (створення таблиць, баз знань, календаря, дошок, таймлайну);
- велика кількість шаблонів;
- модульність (працює як конструктор - можна комбінувати багато елементів на одній сторінці, вкладати сторінки одна в іншу);
- простий у користуванні;
- легко ділитись файлами та коментувати.

Недоліки:
- заплутана навігація;
- немає можливості створювати події, що повторюються;
- незручно кастомізувати, мало можливостей для кастомізації;
- не підійде для великих проєктів;
- відсутня можливість робити звіти.

На основі проведеного аналізу існуючих систем підтримки управління IT-проєктами можна виділити наступні ключові пункти:

A. Базові можливості:
a. Реєстрація користувачів у сервісі;
b. Створення різних проєктів, додавання людей у проєкти;
c. Створення команд всередині проєкту та ієрархія користувачів;
d. Створення дошок для задач у рамках проєкту;
e. Створення карток-задач на дошках, що містять наступну інформацію:
    i. Створювач, виконавець(-ці), спостерігач(-чі) (за необхідністю) (з можливістю переназначати);

ii. Назва задачі, проєкт, до якого вона належить, очікуваний час на виконання, реальний час виконання;
iii. Опис задачі, з можливістю вставки списку, to-do, базовим редагуванням тексту, додаванням файлів у задачу;
iv. Виставлення часу початку та часу кінця (дедлайну) задачі;

f. Коментування задач, доступ на перегляд задачі за посиланням (без необхідності реєстрації);

g. Формування таймлайну з задач у рамках проєкту (діаграми Ганта);

h. Зберігання закінчених задач для подальшого створення звітності;

i. Наявніть інтелектуального пошуку (задачі, призначені на користувача, проєкти користувача, задачі у рамках проєкту, задачі з заданим тегом і т.д);

j. Легкий, інтуітивний інтерфейс, прозора навігація сторінками;

k. Інтеграція з сервісами Google;

B. Додаткові можливості:

a. Вбудована база знань та таблиці;

b. Кастомізація задач (кольорові мітки, символи);

c. Загальне зберігання даних у рамках дошки/проєкту;

d. Планування спринтів.

Огляд використаних технологій

ASP.NET MVC Framework — фреймворк для створення веб-застосунків, який реалізує шаблон Model-view-controller. Цей фреймворк доданий Microsoft в ASP.NET.

Blazor — безкоштовна веб-платформа з відкритим вихідним кодом, що дозволяє створювати веб-застосунки з використанням C# і HTML. Розробляється корпорацією Microsoft.

ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

Призначення системи

Призначенням системи підтримки управління проєктами є автоматизація процесу створення задач, встановлення часу на виконання, зміни їх статусу. Система має полегшити роботу невеликим командам за умови її використання всіма учасниками. Полегшення заключається у повній видимості менеджерами всіх задач проєкту, їх статусів, бажаних та фактичних строків їх виконання.

Робота передбачає:

- аналіз методів, методик і моделей, що застосовуються для розв’язання задач проєктування та реалізації веб сайтів;
- аналіз наявних програмних засобів у галузі управління проєктами;
- проєктування та програмну реалізацію системи підтримки управління проєктами.

Цілі створення системи

Система підтримки управління проєктами створюється з метою:

- забезпечення збору, обробки, аналізу інформації про технології для проектування та реалізації веб-сайтів;
- проаналізувати системи на ринку з урахуванням їх переваг та недоліків у майбутньому продукті;

Також система призначена для забезпечення зручного управління проєктами.

Управління проєктами – це дії щодо узгодження процесів, інструментів, учасників команди та навичок з метою випускати проєкти, що відповідають поставленій меті та вимогам.

Управління проєктами підвищує відсоток успіху. Воно дозволяє команді цілеспрямовано прикладати спільні зусилля для досягнення ясних цілей, допомагає

підвищити прозорість та наочність, оптимізувати комунікацію та окреслити сферу проєкту. Завдяки цьому команда має більше шансів на завершення проєкту.

Управління проєктом більше не використовується тільки в якихось виняткових випадках, а, навпаки, все частіше і швидше стає стандартним способом ведення бізнесу. Все більша частка робіт у звичайних компаніях виконується як проєкти. У майбутньому очікується збільшення важливості та ролі проєктів для стратегічного орієнтування розвитку організацій.

Вимоги ведення бізнесу, функціональні можливості інформаційних технологій створюють певну актуальність реалізації веб застосунків для конкретного бізнесу. Застосування веб сайтів у технології ведення бізнесу має вимірюваний економічний ефект, що й спричиняє рішення власників бізнесу замовити розробку сайту, а в багатьох випадках ведення бізнесу навіть неможливе без використання можливостей сайту. Звісно, для кожного випадку через аналітику приймається рішення щодо автоматизації тих чи інших технологічних процесів бізнесу, розраховується економічна основа розробки програмного забезпечення.

ВИМОГИ ДО СИСТЕМИ

Вимоги до системи в цілому

Система повинна мати архітектуру, побудовану на сучасних технологіях зберігання, обробки, аналізу даних та доступу до них; забезпечувати одночасну роботу декількох користувачів. Рішення щодо побудови системи повинні базуватися на:

●застосуванні сучасних інформаційних технологій;

●реалізації концепції створення єдиного інформаційного простору; застосуванні правила централізованого накопичення, зберігання та обробки інформації;

●підтримці актуальності, повноти, несуперечності, цілісності та доступності інформації;

●забезпеченні надійного захисту інформації від порушення її цілісності, витоку та блокування згідно з вимогами нормативно-правових документів захисту інформації;

●забезпеченні надійності, резервування компонентів технічного забезпечення Системи;

●забезпеченні централізованого управління, безперервного контролю функціонування та централізованого налаштування Системи і модулів її компонентів;

●використанні сучасних засобів програмної інженерії при розробці програмного прикладного забезпечення.

Система є комплексом інформаційних, програмних, технічних, організаційно-методичних та інших необхідних засобів, що забезпечують збір, обробку, зберігання та передачу даних. Архітектура Системи повинна передбачати максимальну незалежність програмно-технічних модулів від виконавця. Інформаційна архітектура Системи повинна відповідати сучасним вимогам щодо побудови інтерфейсів користувачів.

Вимоги до структури та функціонування системи

Система повинна мати наступний функціонал:

- Для адміністратора - реєстрація користувача як менеджера;
- Для менеджерів - можливість створення проєкту як деякої моделі роботи в команді деякого набору людей; Можливість редагування списку людей, залучених до команди проєкту, створення задач для команди проєкту, перегляд їх статусу та строків виконання, призначити відповідальних людей;
- Для учасників команди - можливість змінити статус задачі, що їм призначена, перегляд усієї інформації про задачу;

цільова аудиторія представлена такими групами користувачів:

- Менеджери проєктів, які хочуть отримати більше інформації про поточні статуси задач, хронологію подій у проєкті
- Працівники команди проєкту, які хочуть мати зручний інструмент звітування та механізм отримання задач

Адміністратор має права:

- Блокувати за потреби аккаунт користувача Системи;
- Створити "Організацію" - список користувачів, які працюють разом.

Користувач має права:

- Створювати новий проєкт(тоді його права розширюються пунктом "Власник проєкту" у межах створеного проєкту);
- Переглядати список проєктів, до яких він залучений, всю інформацію по кожному з цих проєктів, у тому числі задачі та всю інформацію, що їх стосується, додавати у них нові задачі;
- Якщо він є учасником задачі: змінювати статус задачі, додавати або змінювати інформацію стосовно цієї задачі.

Менеджер проєкту:

- Ті ж, що є в користувача
- Створити звіт щодо роботи проєкту;

Власник проєкту (якщо не вказано інше - користувач, що створив проєкт) має права:

- Ті ж, що є в менеджера;
- Додати до свого проєкту нового менеджера;
- Передати іншому менеджеру власність проєктом;

Незареєстрований або неавторизований користувач не може переглядати і редагувати жодну інформацію.

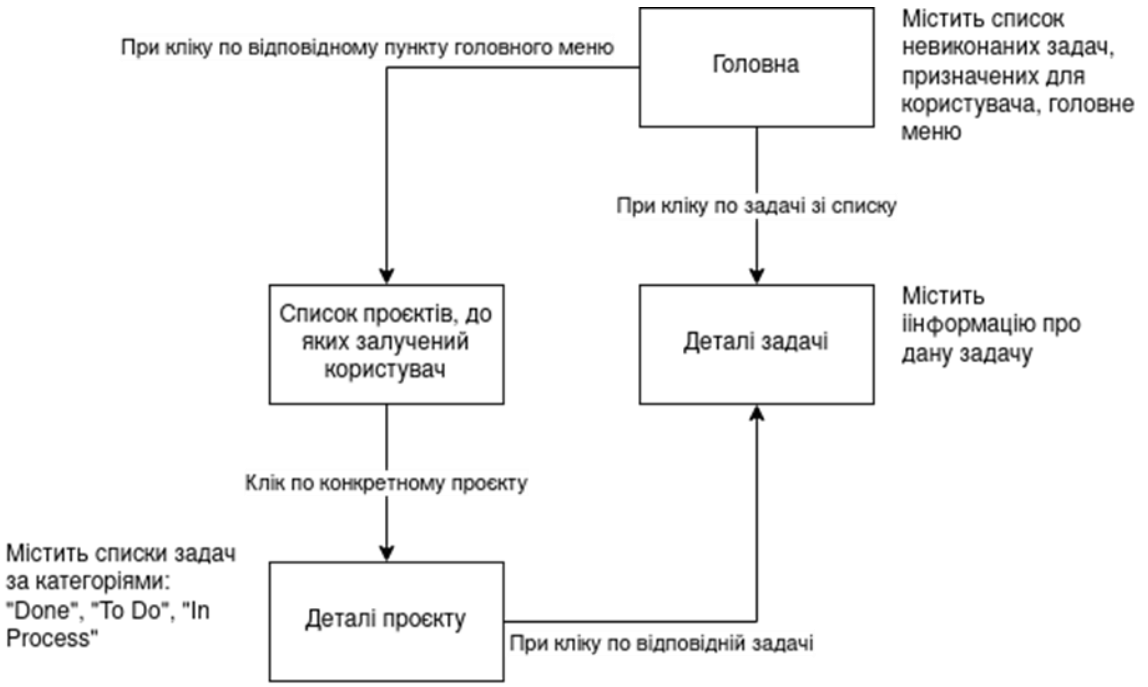


Рисунок 1. Блок-схема Системи

Вимоги до функцій, які виконуються системою

Підсистема менеджера проєкту

Таблиця 1. Перелік функцій, задач що підлягають автоматизації.

Функція	Задача
Реєстрація	Введення інформації про себе
	Підтвердження пошти
	Видалення інформації про себе
Авторизація	Введення пошти та пароля зареєстрованого акаунту
Робота з Системою	Зміна статусу задачі
	Створення коментаря до задачі
	Позначення часу виконання задачі
	Створення звіту щодо роботи проєкту
Створення проєкту	Встановлення назви
	Додавання учасників проєкту
	Встановлення менеджера проєкту
Створення завдання	Додавання опису
	Призначення виконавця
	Встановлення часу виконання
Редагування завдання	Зміна опису
	Зміна виконавця
	Зміна часу виконання

Підсистема адміністратора системи.

Таблиця 2. Перелік функцій, задач що підлягають автоматизації.

Функція	Задача
Реєстрація	Введення інформації про себе
	Підтвердження пошти
	Видалення інформації про себе
Авторизація	Введення пошти та пароля зареєстрованого акаунту
Робота з Системою	Зміна статусу задачі
	Створення коментаря до задачі
	Позначення часу виконання задачі
	Блокування користувача системи
	Створення звіту щодо роботи проєкту
Створення організації	Встановлення назви
	Додавання учасників організації
	Додавання опису
Створення проєкту	Встановлення назви
	Додавання учасників проєкту
	Встановлення менеджера проєкту
Створення завдання	Призначення виконавця
	Додавання опису
	Встановлення часу виконання
Редагування завдання	Зміна опису
	Зміна виконавця
	Зміна часу виконання
Редагування організації	Зміна опису
	Зміна учасників організації
Редагування проєкту	Зміна менеджера
	Додавання нового менеджера
	Створення звіту

Підсистема користувача системи.

Таблиця 3. Перелік функцій, задач що підлягають автоматизації.

Функція	Задача
Реєстрація	Введення інформації про себе
	Підтвердження пошти
	Видалення інформації про себе
Авторизація	Уведення пошти та пароля зареєстрованого акаунту
Робота з Системою	Зміна статусу задачі
	Створення коментаря до задачі
	Позначення часу виконання задачі
Створення проєкту	Встановлення назви
	Додавання учасників проєкту
	Встановлення менеджера проєкту
Створення завдання	Додавання опису
	Призначення виконавця
	Встановлення часу виконання
Редагування завдання	Зміна опису
	Зміна виконавця
	Зміна часу виконання

Системні вимоги

Система і МД та усі їх 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.

Дизайн повинен бути адаптивним: склад, розміри і взаємне розташування елементів всіх шаблонів динамічно змінюються в залежності від розміру вікна браузера, в якому відображається інформація. В процесі проєктування дизайну та структури елементів заохочується аналіз найкращих практик побудови веб-ресурсів.

Система повинна забезпечувати коректне відображення даних в наступних браузерах актуальних версій: Google Chrome, Safari, Opera, Internet Explorer, Mozilla Firefox.

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.

Система повинна бути наповнена контентом настільки, наскільки це необхідно для демонстрації її функціоналу.

Компонування сторінок Системи повинно забезпечувати автоматичне масштабування сторінок в залежності від ширини робочого поля браузера користувача.

Інформаційне забезпечення повинно відповідати таким вимогам та можливостям:

- забезпечення фізичної та логічної цілісності даних;
- мінімізація надмірності даних, що зберігаються;
- стандартизація представлення даних;
- достовірність та актуальність даних.

Джерелом даних для системи повинна бути інформаційна система (СУБД MS SQL).

ОПИС ОРГАНІЗАЦІЇ ІНФОРМАЦІЙНОЇ БАЗИ

Логічна структура бази даних

Під час розробки проєкту використовувалася база даних MSSQL. Нижче на рисунку в додатку А зображена діаграма бази даних.

Таблиця 5. Повний перелік таблиць.

Номер	Таблиця	Опис
1	User	Таблиця для збереження інформації про користувачів
2	UserTask	Таблиця для збереження наявних завдань у користувачів
3	Task	Таблиця для збереження інформації про завдання
4	Comment	Таблиця для збереження коментарів до завдання
5	Project	Таблиця для збереження інформації про проєкт
6	Organization	Таблиця для збереження інформації про організацію
7	Status	Таблиця для збереження статусів завдання
8	RoleProject	Таблиця для збереження наявних у проєкті ролей
9	RoleOrganization	Таблиця для збереження наявних у організації ролей
10	RootUserRoleProject	Таблиця для збереження інформації про проєкти користувачів
11	RootUserOrganRole	Таблиця для збереження інформації про організації користувачів

Опис таблиць

У наступних таблицях приведено опис даних у кожній з таблиць.

Таблиця 6. В таблиці User зберігається інформація про користувачів.

Атрибут	Тип	Опис
id	integer	Ідентифікатор
Login	nvarchar(50)	Логін користувача
Password	nvarchar(50)	Пароль аккаунту
Email	nvarchar(50)	Електронна пошта
Name	nvarchar(50)	Ім’я користувача

Таблиця 7. В таблиці UserTask зберігається інформація наявних завдань у користувачів.

Атрибут	Тип	Опис
TaskId	integer	Ідентифікатор завдання
UserId	integer	Ідентифікатор користувача

Таблиця 8. В таблиці Task зберігається інформація про завдання.

Атрибут	Тип	Опис
id	integer	Ідентифікатор
StatusId	integer	Статус завдання
ProjectId	integer	Ідентифікатор проєкту
Name	nvarchar(50)	Назва завдання
DateCreate	datetime	Дата створення завдання
DateDeadline	datetime	Кінцева дата виконання завдання
DateBegin	datetime	Дата початку виконання завдання
DateEnd	datetime	Дата завершення виконання завдання
Description	nvarchar(100)	Опис завдання

Таблиця 9. В таблиці Comment зберігаються коментарі до завдання.

Атрибут	Тип	Опис
id	integer	Ідентифікатор
TaskId	integer	Ідентифікатор завдання
UserId	integer	Ідентифікатор користувача
Text	nvarchar(150)	Текст коментаря
DateCreate	datetime	Дата додавання коментаря

Таблиця 10. В таблиці Project зберігається інформація про проєкт.

Атрибут	Тип	Опис
id	integer	Ідентифікатор
Name	nvarchar(50)	Назва проєкту
OrganizationId	integer	Ідентифікатор організації

Таблиця 11. В таблиці Organization таблиці зберігається інформація про організацію.

Атрибут	Тип	Опис
id	integer	Ідентифікатор
Name	nvarchar(50)	Назва організації

Таблиця 12. В таблиці Status зберігається інформація про статуси завдань.

Атрибут	Тип	Опис
id	integer	Ідентифікатор
Code	integer	Код статусу завдання
Name	nvarchar(50)	Назва статусу завдання
Description	nvarchar(100)	Опис статусу завдання

Таблиця 13. В таблиці RoleProject зберігається інформація про наявні у проєкті ролі.

Атрибут	Тип	Опис
id	integer	Ідентифікатор
Code	integer	Код ролі користувача
Name	nvarchar(50)	Назва ролі користувача
Description	nvarchar(100)	Опис ролі користувача

Таблиця 14. В таблиці RoleOgranization зберігається інформація наявних в організації ролей.

Атрибут	Тип	Опис
id	integer	Ідентифікатор
Code	integer	Код ролі користувача
Name	nvarchar(50)	Назва ролі користувача
Description	nvarchar(100)	Опис ролі користувача

Таблиця 15. В таблиці RootUserRoleProject зберігається інформація про проєкти користувача.

Атрибут	Тип	Опис
id	integer	Ідентифікатор
RoleId	integer	Ідентифікатор ролі
ProjectId	integer	Ідентифікатор проєкту
UserId	integer	Ідентифікатор користувача

Таблиця 16. В таблиці RootUserOrganRole зберігається інформація про організації користувача.

Атрибут	Тип	Опис
id	integer	Ідентифікатор
RoleId	integer	Ідентифікатор ролі
OrganizationId	integer	Ідентифікатор організації
UserId	integer	Ідентифікатор користувача

РЕАЛІЗАЦІЯ СИСТЕМИ

Система використовує фреймворк ASP.NET MVC для розробки backend.

Найбільш принциповими контролерами є контролер для об’єктів проєкту, задачі та користувача:

Отримання користувача за його електронною адресою:

```
// GET: api/Users/GetUserByEmail/email
[HttpGet("GetUserByEmail/{email}")]
public async Task<ActionResult<User>> GetUserByEmail(string email)
{
    var user = await _context.Users.Where(t => t.Email == email).FirstOrDefaultAsync();

    if (user == null)
    {
        return NotFound();
    }

    return user;
}
```

Отримання задач для заданих користувача та проєкту:

```
// GET: api/Tasks/GetTasksByProjectId/2/UserId/7
[HttpGet("GetTasksByProjectId/{projectId}/UserId/{userId}")]
public async Task<ActionResult<IEnumerable<Task>>>
GetTasksByProjectIdAndUserId(int projectId, int userId)
{
    var tasks = await _context.Tasks.Where(t => t.ProjectId == projectId).Where(t =>
t.Users.All(d => d.Id == userId)).ToListAsync();

    if (tasks == null)
    {
        return NotFound();
    }

    return tasks;
}
```

Отримання проєктів користувача та організації

```
// GET: api/Projects/GetProjectsByUserId/5/OrganizationId/6
[HttpGet("GetProjectsByUserId/{userId}/OrganizationId/{organizationId}")]
public async Task<ActionResult<IEnumerable<Project>>>
GetProjectsByUserIdAndOrganizationId(int userId, int organizationId)
{
    var projects = await _context.Projects.Where(t => t.RootUserRoleProjects.Any(d =>
d.UserId == userId)).ToListAsync();
    projects = projects.Where(t => t.OrganizationId == organizationId).ToList();
    if (projects == null)
    {
        return NotFound();
    }

    return projects;
}
```

ІНСТРУКЦІЯ КОРИСТУВАЧА

У системі є чотири режими роботи:

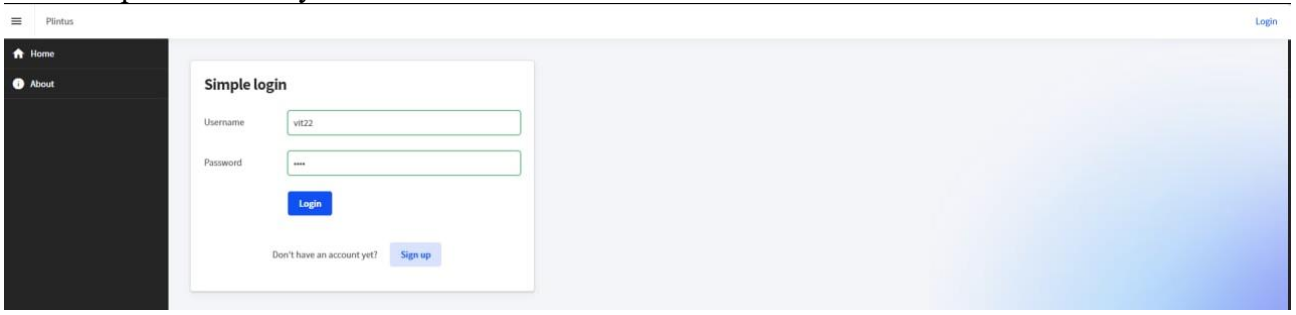
- Гостьовий
- Користувацький
- Менеджерський
- Адміністраторський

Гостьовий:

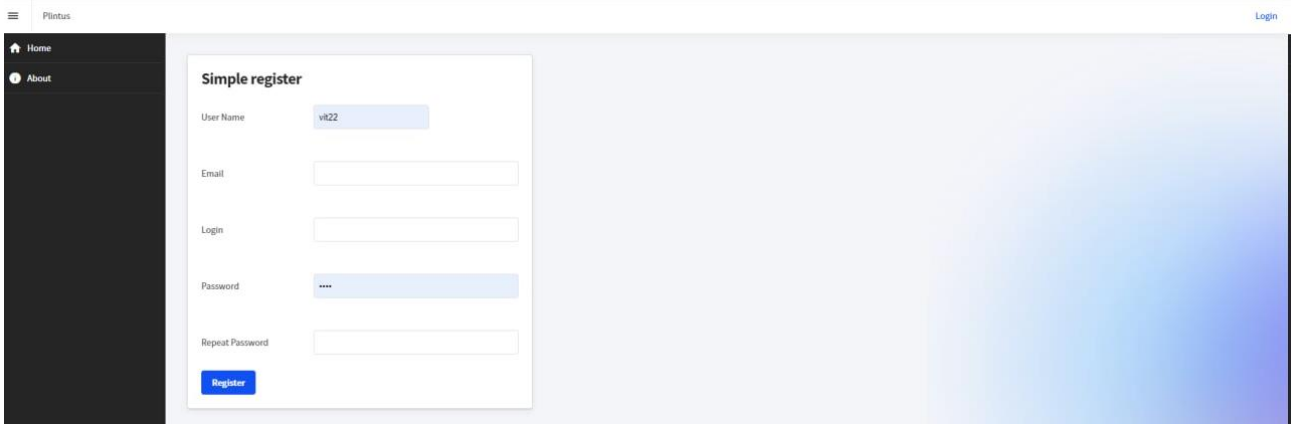
При вході на сайт без логіну система запрошує користувача залогінітись

Користувацький.

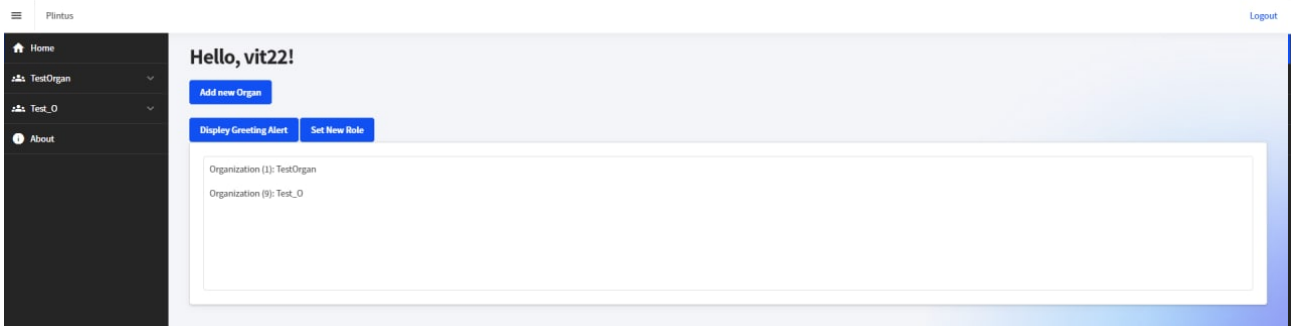
Форма для входу:



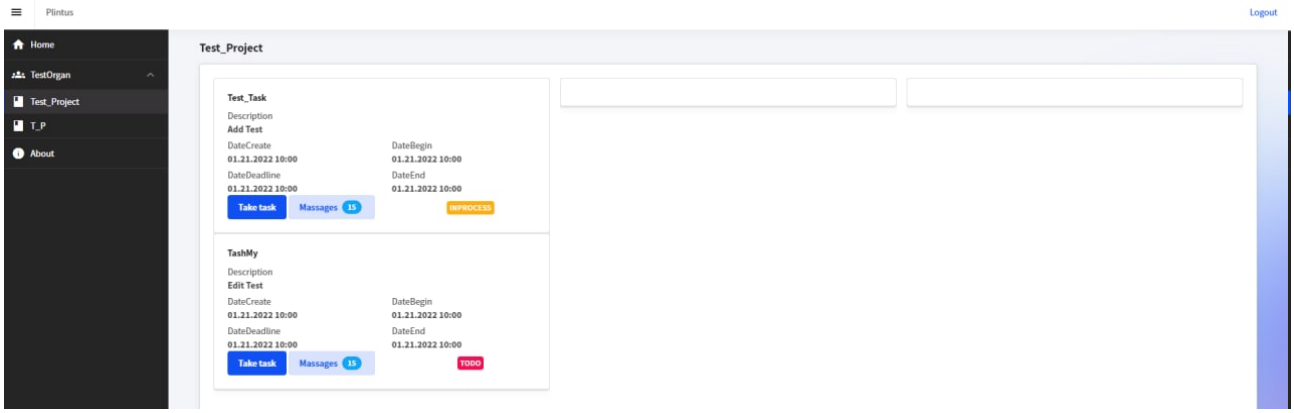
Форма для реєстрації:



Привітання залогіненого користувача, його меню:

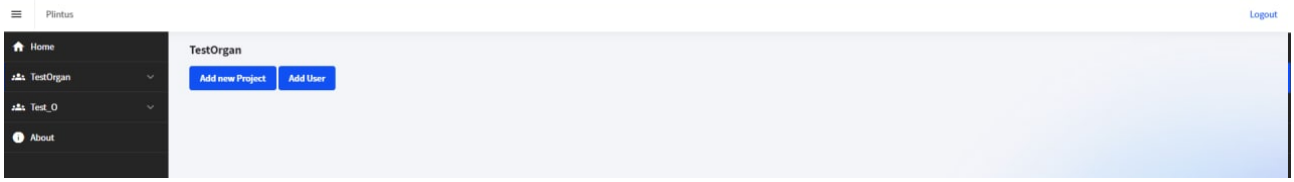


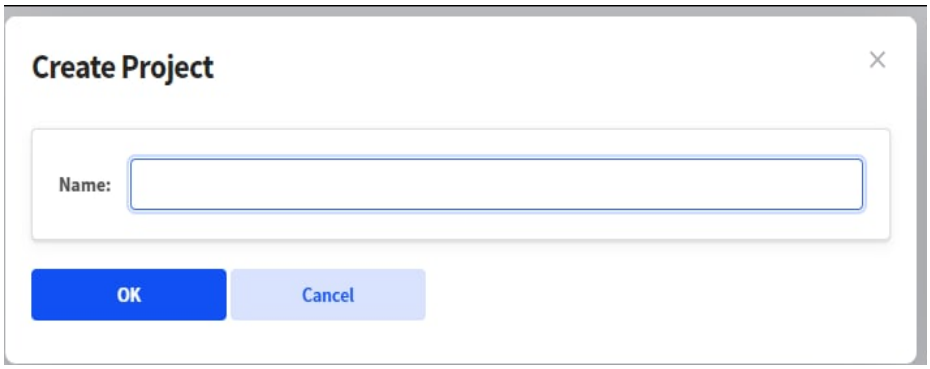
Вікно задач проєкту організації, до яких залучено користувача:

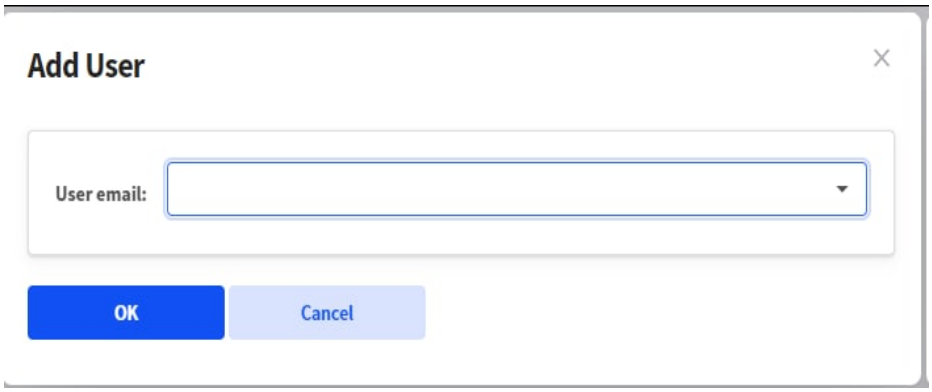


Менеджерський.

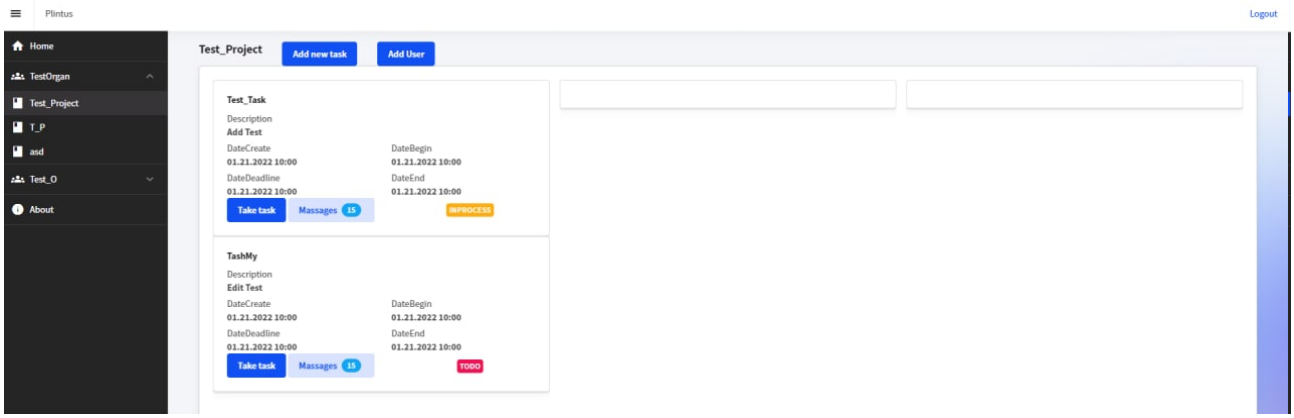
Додавання до організації нового проєкту або користувача:





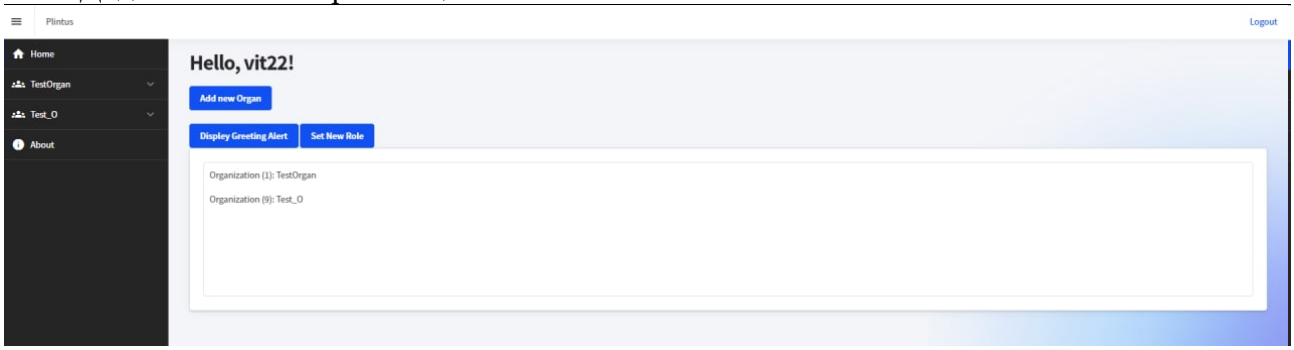


Управління проєктом:



Адміністраторський.

Додавання нової організації:



ВИСНОВКИ

Стосовно роботи в цілому:

- проведено аналіз методів, методик і моделей, що застосовуються для розв'язання задач проектування та реалізації веб сайтів;
- проведено аналіз наявних програмних засобів у галузі управління проєктами;
- спроектовано та програмно реалізовано систему підтримки управління проєктами.

Виконані вимоги:

- застосовано сучасні інформаційні технології;
- реалізовано концепцію створення єдиного інформаційного простору;
- застосовано правила централізованого накопичення, зберігання та обробки інформації;
- система підтримує актуальність, повноту, несуперечливість, цілісність та доступність інформації;
- забезпечено надійний захист інформації від порушення її цілісності, витоку та блокування згідно з вимогами нормативно-правових документів в галузі захисту інформації;
- забезпечена надійність, резервування компонентів технічного забезпечення Системи;
- забезпечене централізоване управління, безперервний контроль функціонування та централізованого налаштування Системи і модулів її компонентів;
- використано сучасні засоби програмної інженерії при розробці програмного прикладного забезпечення.

Вимоги до структури та функціонування системи:

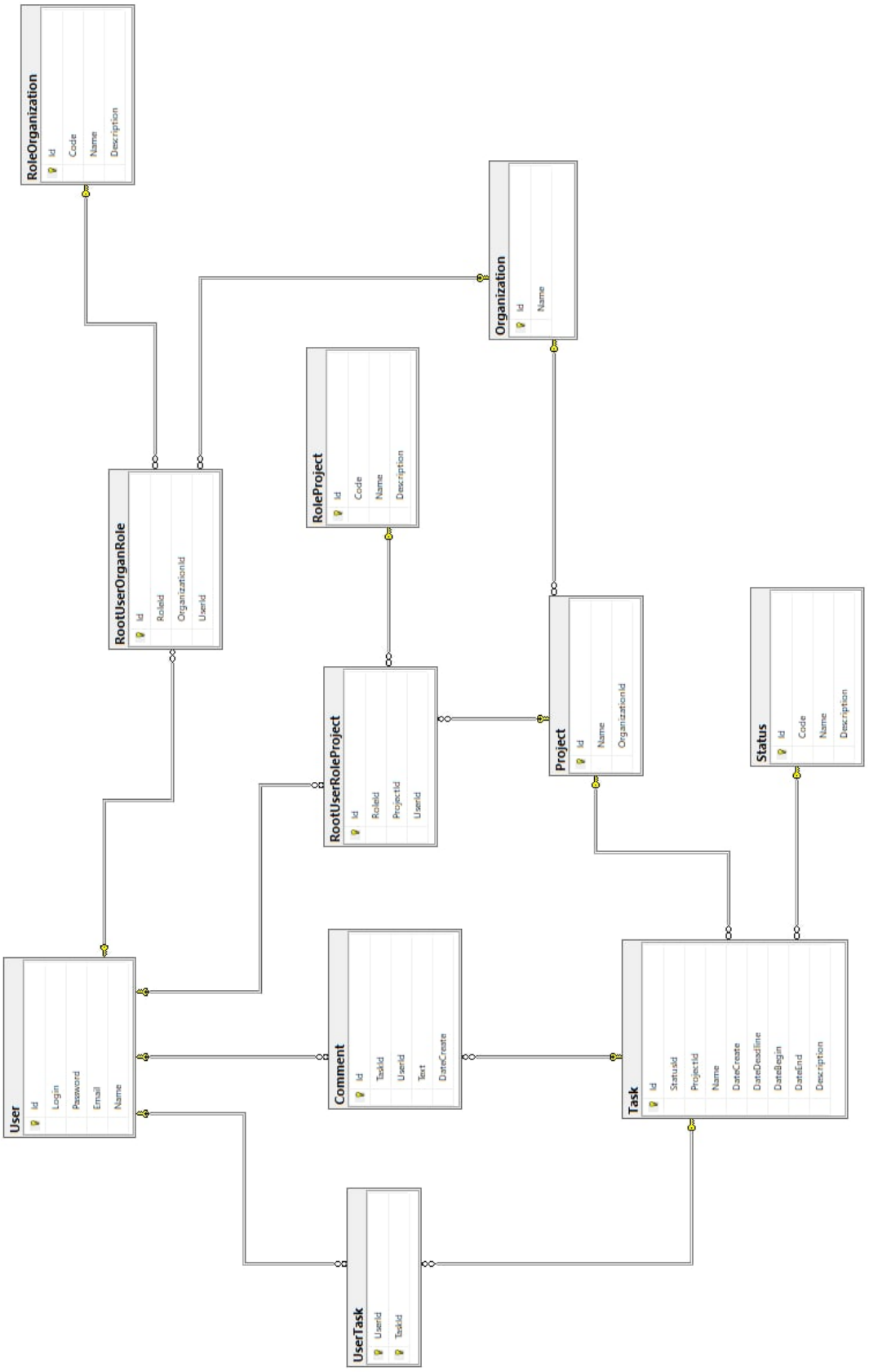
Система повинна має наступний функціонал:

- Для адміністратора - реєстрація користувача як менеджера;
- Для менеджерів - можливість створення проєкту як деякої моделі роботи в команді деякого набору людей; Можливість редагування списку людей, залучених до команди проєкту, створення задач для команди проєкту, перегляд їх статусу та строків виконання, призначення відповідальних людей;
- Для учасників команди - можливість змінити статус задачі, що їм призначена, перегляд усієї інформації про задачу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Омельчук Л.Л. Формальні методи специфікації програм. – К.: УкрІНТЕІ, 2010. - 78 с.
2. Омельчук Л.Л. Аксіоматичні системи специфікацій програм над номінативними даними: Дисертація к.ф.-м.н.: 01.05.01. - К., 2007. – 142 с.
3. J.M. Spivey. Understanding Z: A specification language and its formal semantics. – Cambridge University press. – 1988. – 131 p.
4. ASP.NET Core Blazor. Developer tools, technical documentation and coding examples | Microsoft Docs. URL: https://docs.microsoft.com/ru-ru/aspnet/core/blazor/?view=aspnetcore-6.0
5. ASP.NET MVC. Developer tools, technical documentation and coding examples | Microsoft Docs. URL: https://docs.microsoft.com/ru-ru/aspnet/mvc/

ДОДАТОК А. Схема бази даних



РОЗРОБКА СИСТЕМИ ПІДТРИМКИ УПРАВЛІННЯ ПРОЄКТАМИ

Ірина Потієнко, Артем Божок, Іван Єрьоменко, Ілля Риченко,
Вікторія Харченко, Дмитро Шимків

ВСТУП

Актуальність. У сучасному світі проєктне управління стало невід'ємною частиною успішної компанії. В умовах досить жорсткої конкуренції складно налагодити ефективну роботу підприємства, не плануючи терміни, витрати та не враховуючи ризики. На сучасному ринку успіху можуть досягти ті компанії, які враховують потреби ринку і випускають свою продукцію або надають свої послуги у конкретний момент часу, тобто, вчасно.

Відставання за термінами є одною із головних проблем майже у кожній сфері діяльності. Це веде до збільшення сукупних витрат і звичайно відзначається на прибутку підприємства. Використання технології проєктного управління дозволяє організаціям скоротити терміни реалізації проєктів, знижуючи сукупні витрати у середньому на 10-15%, тим самим підвищивши ефективність діяльності компанії. Методи проєктного управління роблять бізнес "прозорим", легко керованим і дозволяють реалізовувати проєкти в максимально стислий термін при обмежених ресурсах.

Цілі й завдання. Метою роботи є розробка веб застосунку для забезпечення комфортного управління проєктами з урахуванням недоліків та переваг наявних на ринку систем. Отримати досвід роботи у команді, освоїти інструменти для такої роботи, дізнатися детальніше про повний процес розробки програмного продукту. Покращити навички у застосуванні технологій верифікації та тестування.

Для досягнення поставлених цілей визначено такі завдання:

- дослідити наявні на ринку системи підтримки управління проєктами;
- дослідити застосування різних технологій для проєктування та реалізації веб сайту;
- розробити технічне завдання до програмного продукту;
- розробити інтерфейс та дизайн веб застосунку.

Методологія та система управління проєктами.

Для виконання поставлених задач команда обрала методологію управління проєктами Scrum. Scrum - це методика, яка допомагає командам вести спільну роботу. Методику Scrum найчастіше застосовують команди розробників додатків, але принципи та досвід її використання можна застосувати

до командної роботи будь-якого роду. Це одна з причин такої популярності методики.

Учасники команди Scrum проводять збори, використовують спеціальні інструменти і

приймають на себе особливі ролі, щоб організувати роботу і керувати нею. Зустрічі проводилися двічі на тиждень, обговорювалися наступні питання:

- Прогрес у виконанні поставлених задач;
- З якими проблемами зіштовхнулися, варіанти їх вирішення;
- План подальшої роботи.

Для спілкування було створено групу в месенджері Telegram.

Системою управління було обрано Trello [1]. Це одна з найпопулярніших систем управління проєктами в режимі онлайн, яка користується особливим попитом серед невеликих компаній і стартапів. Вона дозволяє ефективно організовувати роботу по методології kanban-дошок. На рисунку 1 представлена командна дошка з деякими завданнями.

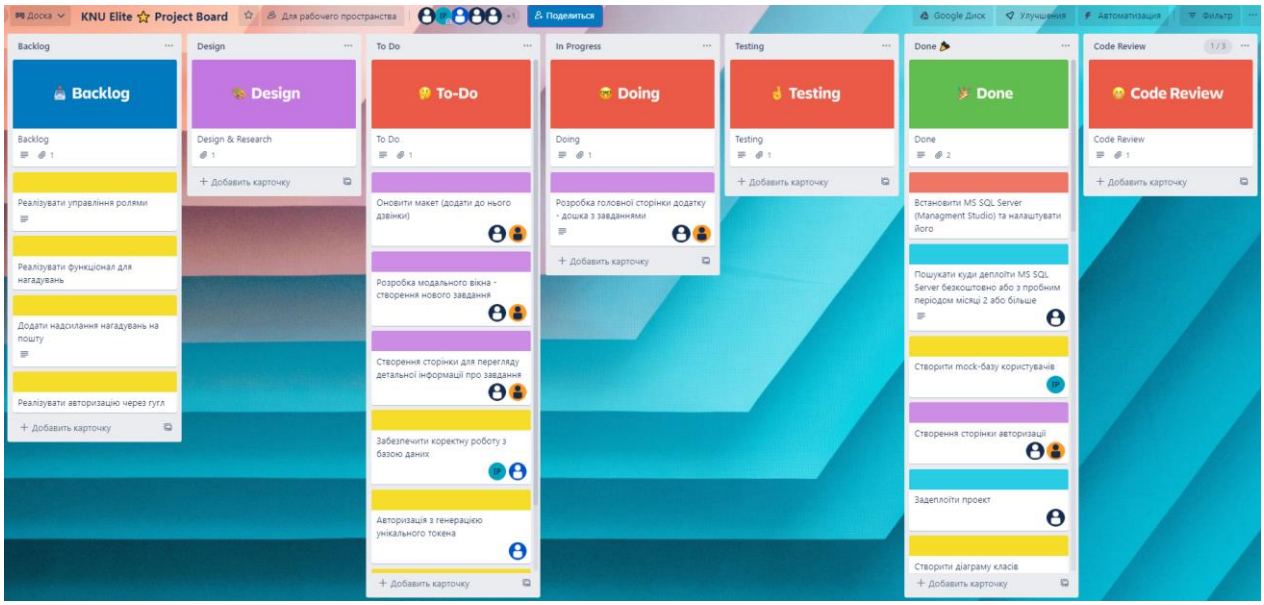


Рисунок 1. Дошка Trello

ОГЛЯД НАЯВНИХ НА РИНКУ СИСТЕМ

Було переглянуто найбільш популярні системи управління IT проєктами, проаналізовано їх переваги та недоліки, а також створено порівняльну таблицю для трьох програмних продуктів.

Нижче описана коротка інформація про них, а також деякий їх функціонал.

Jira.[2] Система відстеження помилок, призначена для організації спілкування з користувачами та управління проєктами.

Можна виділити деякі особливості та переваги:

- Достатньо добре заточена під корпоративні стандарти.
- Kanban-дошка – допомагає команді забезпечити прозорість роботи над проєктом, оптимізувати робочий процес, розподілити завдання з беклогу.
- Scrum-дошка - дозволяє управляти складним проєктом, об'єднати команди з різних напрямків розробки продукту для досягнень однієї мети.
- Картки для кожного багу можуть створюватися прямо з чату.
- Доступним є вибір потрібних модулів для відділу розробки: Scrum, Kanban, Basic software development, а також для бізнесу: Process management, Project management, Task management.
- Можна розрахувати швидкість та об’єм роботи команди за спрінт та подивитися скільки роботи було заплановано, а скільки виконано.

Недоліки:

- Інтерфейс здається досить складним для користувачів новачків;
- Через багатофункціональність буває складно знайти те, що потрібно;
- У користувачів, що мають досить багато прав адміністратора, довантажується багато додаткової інформації, у зв’язку з чим продукт працює повільніше;

Microsoft Project.[3] Система, що дозволяє організовувати сумісну роботу й планувати проєкти в складі Office 365. Легко інтегрується з іншими сервісами Microsoft. Допомагає розподілити ресурси на задачі, відслідковувати прогрес та аналізувати об’єми робіт.

Одною з основних функцій продукту є допомога в плануванні проєкту. З її допомогою можна визначити, які саме роботи приведуть до бажаного результату, що потрібно для досягнення поставленої мети, а також скільки часу на це знадобиться.

Також є функція контролю й відслідковування ходу проєкту, в рамках якої Microsoft Project дозволяє здійснювати порівняння виконаних розрахунків з результатами, яких вдалося досягти, таким чином контролюючи виконання проєкту за всіма параметрами.

YouGile.[4] Система управління проєктами та спілкування в команді.

Головною відмінністю є те, що кожна задача - це чат. Це змінює процес застосування та використання системи. Завдання ставляться як на Agile-дошки, так і в особистий планувальник "Мої завдання". Немає ситуацій, коли завдання нікуди занести. Для зв’язку відділів та формування моніторингу можна відфільтрувати завдання з інших дошок і проєктів. Структура компанії виглядає наступним чином: створюються відділи, вказуються посади співробітників та базові права в системі. Виявляється корисним, якщо в команді багато людей. Гнучка система побудови звітів та онлайн-моніторингу, лог подій – повна прозорість за людьми, проєктами, дошками.

Отже, проаналізувавши отриману інформацію, було прийнято рішення реалізувати власний програмний продукт з наступним функціоналом:

- Наявність авторизації користувачів
- Наявність ролей та відповідних до них можливостей
- Kanban-дошка
- Забезпечення зміни статусу задачі
- Відстежування часу, витраченого на задачу

Особливістю буде можливість планувати онлайн-зустрічі всередині системи.

ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Серверна частина.

Мова програмування C# [5] - об'єктно-орієнтована, орієнтована на компоненти мова програмування. C# надає мовні конструкції для безпосередньої підтримки такої концепції роботи. Завдяки цьому C# підходить для створення та застосування програмних компонентів. З моменту створення мова C# збагатилася функціями для підтримки нових робочих навантажень та сучасними рекомендаціями щодо розробки ПЗ.

ASP.NET Core [6] - це кросплатформний фреймворк з відкритим вихідним кодом для створення сучасних хмарних додатків, підключених до інтернету, таких як веб-програми мобільні серверні програми. Застосунки ASP.NET Core працюють на .NET Core, безкоштовній міжплатформній програмі з відкритим вихідним кодом. Фреймворк був розроблений так, щоб забезпечити оптимізовану структуру розробки для програм, які розгортаються в хмарі або запускаються локально. Він складається з модульних компонентів, тому зберігається гнучкість під час створення своїх рішень.

Entity Framework [7] - представляє спеціальну об'єктно-орієнтовану технологію на базі фреймворку .NET для роботи з даними. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework є більш високим рівнем абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. Якщо ми фізично оперуємо таблицями, індексами, первинними та зовнішніми ключами, то на концептуальному рівні, який пропонує Entity Framework, ми вже працюємо з об'єктами.

MySQL [8] - це реляційна система управління базами даних. Тобто дані у її базах зберігаються як логічно пов'язані між собою таблиці, доступ яких здійснюється з допомогою мови запитів SQL. Це вільно поширена система. Крім того, досить швидка,

надійна й проста у використанні СУБД, що цілком підходить для не надто глобальних проектів.

Heroku [9] - це контейнерна хмарна платформа. Розробники використовують її для розгортання, керування та масштабування сучасних програм. Дана платформа елегантна, гнучка та проста у використанні, пропонуючи розробникам найпростіший шлях до виведення своїх програм на ринок.

Клієнтська частина.

Мова програмування JavaScript [10] - це легковажна, інтерпретована або JIT-компільована, об'єктно-орієнтована мова з функціями першого класу. Найширше застосування знаходить як мова сценаріїв веб-сторінок, але також використовується і в інших програмних продуктах, наприклад, node.js або Apache CouchDB. JavaScript це прототипно-орієнтована, мультипарадигменна мова з динамічною типізацією, яка підтримує об'єктно-орієнтований, імперативний та декларативний (наприклад, функціональне програмування) стилі програмування.

CSS [11] - це мова ієрархічних правил (таблиць стилів), що використовується для представлення зовнішнього вигляду документа, написаного на HTML або XML. CSS описує, як елемент повинен відображатися на екрані, на папері, голосом або за допомогою інших засобів масової інформації.

React [12] - JavaScript-бібліотека з відкритим вихідним кодом для розробки інтерфейсів користувача.

Bootstrap React [13] - замінює Bootstrap JavaScript. Кожен компонент був створений з нуля як справжній компонент React, без непотрібних залежностей, таких як jQuery. Як одна з найстаріших бібліотек React, React-Bootstrap розвивався та виріс разом із React, що робить його чудовим вибором як основу інтерфейсу.

ПРИЗНАЧЕННЯ Й ЦІЛІ СТВОРЕННЯ СИСТЕМИ

Призначення системи

Призначенням системи підтримки управління проєктами є автоматизація процесу створення задач, встановлення часу на виконання, зміни їх статусу, організація спілкування між користувачами задля підвищення ефективності роботи користувачів.

Робота передбачає:

- аналіз методів, методик і моделей, що застосовуються для розв’язання задач проєктування та реалізації веб сайтів;
- аналіз наявних програмних засобів у галузі управління проєктами;
- проєктування та програмну реалізацію системи підтримки управління проєктами.

Цілі створення системи

Система підтримки управління проєктами створюється з метою:

- забезпечення збору, обробки, аналізу інформації про технології для проєктування та реалізації веб-сайтів;
- проаналізувати системи на ринку із урахуванням їх переваг та недоліків у майбутньому продукті;

Також система призначена для забезпечення зручного способу управління проєктами.

Вимоги до системи

Вимоги до системи в цілому

Система підтримки управління проєктами повинна мати архітектуру, побудовану на сучасних технологіях зберігання, обробки, аналізу даних та доступу до них; забезпечувати одночасну роботу декількох користувачів.

В Системі передбачається виділити наступні функціональні підсистеми: розробника, менеджера проєкту, тестувальника, керівника команди.

Запланований функціонал у залежності від ролі користувача відображений у Use Case діаграмах у додатках А-Г.

Вимоги до функцій, які виконуються системою

Підсистема розробника.

Таблиця 1. Перелік функцій, задач що підлягають автоматизації.

Функція	**Задача**
Реєстрація	Уведення інформації про себе
	Підтвердження пошти
	Видалення інформації про себе
Авторизація	Уведення пошти та пароля зареєстрованого акаунту
Робота з Kanban-дошкою	Зміна статусу задачі
	Створення коментаря до задачі
	Додавання файлу до задачі
	Позначення часу виконання задачі
	Отримання посилання на онлайн-зустріч

Підсистема менеджера проєкту.

Таблиця 2. Перелік функцій, задач що підлягають автоматизації.

Функція	**Задача**
Реєстрація	Уведення інформації про себе
	Підтвердження пошти
	Видалення інформації про себе
Авторизація	Уведення пошти та пароля зареєстрованого акаунту
Робота з Kanban-дошкою	Зміна статусу задачі
	Створення коментаря до задачі
	Додавання файлу до задачі
	Позначення часу виконання задачі
	Створення посилання на онлайн-зустріч
	Отримання посилання на онлайн зустріч
Створення проєкту	Встановлення назви
	Додавання учасників
Створення завдання	Додавання опису
	Призначення виконавця
	Встановлення часу виконання
	Встановлення пріоритету

Функція	Задача
Редагування завдання	Зміна опису
	Зміна виконавця
	Зміна часу виконання
	Зміна пріоритету

Підсистема тестувальника.

Таблиця 3. Перелік функцій, задач що підлягають автоматизації.

Функція	Задача
Реєстрація	Уведення інформації про себе
	Підтвердження пошти
	Видалення інформації про себе
Авторизація	Уведення пошти та пароля зареєстрованого акаунту
Робота з Kanban-дошкою	Зміна статусу задачі
	Створення коментаря до задачі
	Додавання файлу до задачі
	Позначення часу виконання задачі
	Отримання посилання на онлайн-зустріч
Створення баг завдання	Додавання опису
	Призначення виконавця
	Встановлення часу виконання
	Встановлення пріоритету

Підсистема керівника команди.

Таблиця 4. Перелік функцій, задач що підлягають автоматизації.

Функція	Задача
Реєстрація	Уведення інформації про себе
	Підтвердження пошти
	Видалення інформації про себе
Авторизація	Уведення пошти та пароля зареєстрованого акаунту
Робота з Kanban-дошкою	Зміна статусу задачі
	Створення коментаря до задачі
	Додавання файлу до задачі

Функція	Задача
	Позначення часу виконання задачі
	Створення посилання на онлайн-зустріч
	Отримання посилання на онлайн зустріч
Створення завдання	Додавання опису
	Призначення виконавця
	Встановлення часу виконання
	Встановлення пріоритету
Редагування завдання	Зміна опису
	Зміна виконавця
	Зміна часу виконання
	Зміна пріоритету

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 англійською мовою, в межах загального дизайну сайта.

Джерелом даних для Системи повинна бути інформаційна система (СУБД MySQL).

ОПИС ОРГАНІЗАЦІЇ ІНФОРМАЦІЙНОЇ БАЗИ

Логічна структура бази даних

Під час розробки проєкту використовувалася MySQL база даних. На рисунку нижче зображена діаграма.

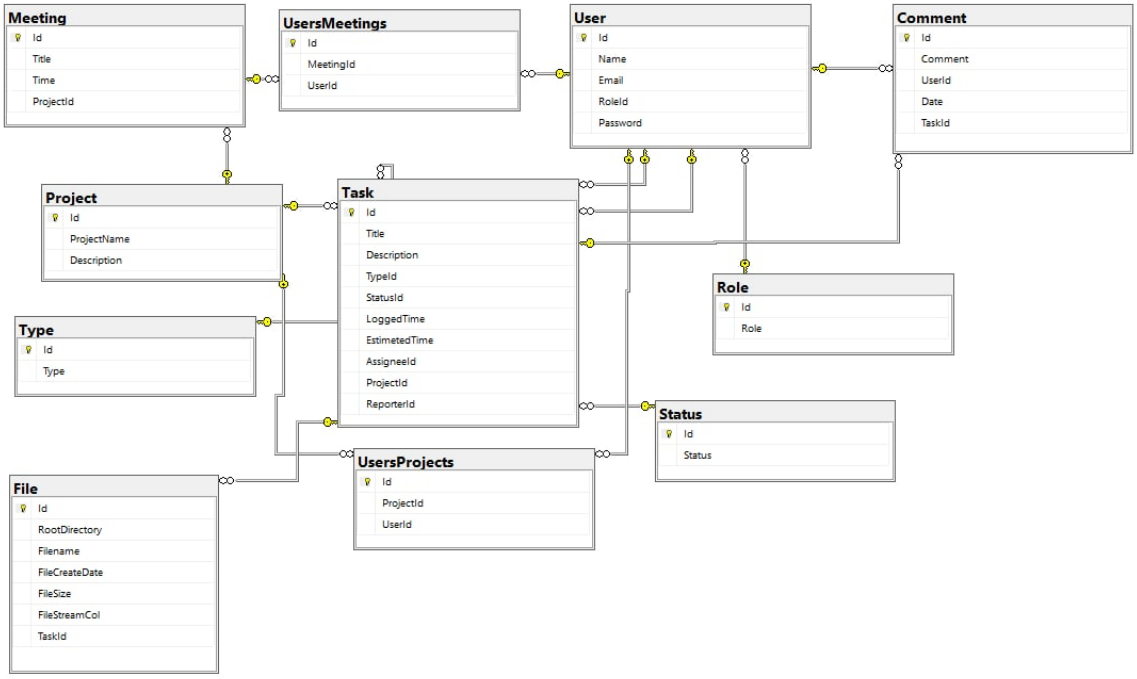


Рисунок 2. Діаграма бази даних

Таблиця 5. Повний перелік таблиць.

Номер	**Таблиця**	**Опис**
1	Meeting	Таблиця для збереження інформації про онлайн-конференції
2	UsersMeetings	Таблиця для збереження інформації про онлайн-конференції користувачів
3	User	Таблиця для збереження інформації про користувачів
4	Role	Таблиця для збереження наявних у проєкті ролей
5	Comment	Таблиця для збереження коментарів до завдання
6	Project	Таблиця для збереження інформації про проєкт
7	Task	Таблиця для збереження інформації про завдання
8	Type	Таблиця для збереження наявних типів завдань
9	Status	Таблиця для збереження статусів завдання
10	File	Таблиця для збереження інформації про файл, прикріплений до завдання
11	UsersProjects	Таблиця для збереження інформації про проєкти користувачів

Опис таблиць

Нижче приведений опис даних у кожній з таблиць.

У таблиці 6 містяться поля для збереження даних про онлайн-конференції.

Таблиця 6. Meeting

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
Title	varchar(255)	Назва конференції

Time	datetime	Час початку конференції
ProjectId	integer	Ідентифікатор проєкту

У таблиці 7 демонструються поля з інформацією про онлайн- конференції користувачів.

Таблиця 7. UsersMeetings

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
MeetingId	integer	Ідентифікатор конференції
UserId	integer	Ідентифікатор користувача

Таблиця 8 містить поля з інформацією про користувачів.

Таблиця 8. User

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
Name	varchar(255)	Ім’я користувача
Email	varchar(100)	Електронна пошта
RoleId	integer	Ідентифікатор ролі
Password	varchar(50)	Пароль аккаунту

У таблиці 9 наведено поля, що містить модель ролі - назва та її ідентифікатор.

Таблиця 9. Role

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
Role	varchar(50)	Назва ролі корисувача

Таблиця 10 використовується для збереження коментарів.

Таблиця 10. Comment

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
Comment	text	Текст коментаря
UserId	integer	Ідентифікатор користувача
Date	datetime	Дата додавання коментаря
TaskId	integer	Ідентифікатор завдання

Таблиця 11 описує деяку інформацію про проєкт.

Таблиця 11. Project

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
ProjectName	varchar(100)	Назва проєкту
Description	text	Опис проєкту

У таблиці 12 наведено поля моделі Task.

Таблиця 12. Task

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
Title	varchar(255)	Назва завдання
Description	text	Опис завдання
TypeId	integer	Ідентифікатор типу завдання
StatusId	integer	Статус завдання
LoggedTime	varchar(50)	Відмічений час виконання завдання
EstimatedTime	varchar(50)	Час на виконання завдання
AssigneeId	integer	Ідентифікатор користувача, що створив завдання
ProjectId	integer	Ідентифікатор проєкту
ReporterId	integer	Ідентифікатор виконавця завдання

У таблиці нижче показано поля моделі Type.

Таблиця 13. Type

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
Type	varchar(50)	Назва типу завдання

Таблиця 14 використовується для збереження статусів завдання

Таблиця 14. Status

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
Status	varchar(50)	Назва статусу завдання

Таблиця 15 зображає поля моделі File.

Таблиця 15. File

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
RootDirectory	varchar(255)	Шлях до файлу
FileName	varchar(50)	Назва файлу
FileCreateDate	datetime	Дата створення файлу
FileSize	bigint(20)	Розмір файлу
FileStreamCol	text	Потік файлу
FileTaskId	integer	Ідентифікатор завдання

Таблиця нижче використовується для зберігання проєктів користувача.

Таблиця 16. UsersProjects

Атрибут	**Тип**	**Опис**
id	integer	Ідентифікатор
ProjectId	integer	Ідентифікатор проєкту
UserId	integer	Ідентифікатор користувача

РЕАЛІЗАЦІЯ СИСТЕМИ

Серверна частина. Реалізована мовою C# за допомогою ASP.NET Core та Entity Framework.

Метод конфігурації зображено на рисунку 3.

```
    public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
    {
        if (env.IsDevelopment())
        {
            app.UseDeveloperExceptionPage();
        }
        else
        {
            app.UseExceptionHandler("/Error");
            // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
            app.UseHsts();
        }
        app.UseCors("AllowOrigin");

        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthorization();

        app.UseEndpoints(endpoints =>
        {
            endpoints.MapControllers();
        });
    }
}
```

Рисунок 3. Метод конфігурації

Розглянемо для прикладу сутність Task. На рисунку 4 - клас моделі.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace KNUElite_project_backend.Models
{
    public class Task
    {
        public Task()
        {
            Comments = new List<Comment>();
            Files = new List<File>();
        }
        public int Id { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
        public int TypeId { get; set; }
        public int StatusId { get; set; }
        public string LoggedTime { get; set; }
        public string EstimatedTime { get; set; }
        public int AssigneeId { get; set; }
        public int ReporterId { get; set; }
        public int ProjectId { get; set; }
        public virtual Type Type { get; set; }
        public virtual Status Status { get; set; }
        public virtual User Assignee { get; set; }
        public virtual User Reporter { get; set; }
        public virtual Project Project { get; set; }
        public List<Comment> Comments { get; set; }
        public List<File> Files { get; set; }

    }
}
```

Рисунок 4. Клас моделі Task

Клас контролера показано на рисунках 5-6.

```
using KNUElite_project_backend.Models;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace KNUElite_project_backend.Controller
{
    [Route("api/[controller]")]
    [ApiController]
    public class TaskController : ControllerBase
    {
        private ProjectContex _context;

        public TaskController(ProjectContex context)
        {
            _context = context;
        }

        [HttpGet]
        public IList<Models.Task> Get()
        {
            return (_context.Tasks.ToList());
        }
```

```
        [HttpGet("{id}")]
        public async Task<IActionResult> Get(int id)
        {
            var task = await _context.Tasks.FindAsync(id);

            if (task == null)
            {
                return NotFound();
            }

            return Ok(task);
        }

        [HttpPost]
        public async Task<IActionResult> Post(Models.Task task)
        {
            _context.Tasks.Add(task);
            await _context.SaveChangesAsync();

            return CreatedAtAction("Get", new { id = task.Id }, task);
        }

        [HttpDelete("{id}")]
        public async Task<IActionResult> Delete(int id)
        {
            var task = await _context.Tasks.FindAsync(id);
            if (task == null)
            {
                return NotFound();
            }

            _context.Tasks.Remove(task);
            await _context.SaveChangesAsync();

            return Ok();
        }
    }
}
```

Рисунки 5,6. Клас контролера Task

Інші сутності реалізовано подібним чином.

Клієнтська частина. Написана на мові JavaScript, CSS за допомогою бібліотеки React.

Реалізація сторінки Login представлена на рисунку 7.

```jsx
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import { Alert, Button, Form } from "react-bootstrap";
import "bootstrap/dist/css/bootstrap.min.css";
import "./styles.css";

function LogIn() {
  const navigate = useNavigate();
  const [logIn, setLogIn] = useState("");
  const [password, setPassword] = useState("");
  const [show, setShow] = useState("");

  const signIn = () => {
    if (logIn === "knu@gmail.com" && password === "11111") {
      navigate("/mainPage");
    } else {
      setShow(true);
    }
  };

  return (
    <>
      <h1 className="mainText">KNU ELITE</h1>
      <div className="logInBox">
        {show && (
          <Alert variant="danger" onClose={() => setShow(false)} dismissible>
            <Alert.Heading>Invalid Login or Password</Alert.Heading>
            <p>Please try again</p>
          </Alert>
        )}
        <div className="box">
          <h2 className="text">Sign In</h2>
          <div className="form">
            <Form.Group className="mb-3" controlId="exampleForm.ControlInput1">
              <Form.Label>Login</Form.Label>
              <Form.Control
                type="email"
                placeholder="Enter login"
                onChange={({ target }) => setLogIn(target.value)}
              />
            </Form.Group>
            <Form.Group className="mb-3" controlId="exampleForm.ControlInput1">
              <Form.Label>Password</Form.Label>
              <Form.Control
                type="email"
                placeholder="Enter password"
                onChange={({ target }) => setPassword(target.value)}
              />
            </Form.Group>
          </div>
          <div>
            <Button
              className="button"
              variant="warning"
              size="lg"
              onClick={signIn}
            >
              Sign In
            </Button>
          </div>
        </div>
      </div>
    </>
  );
}

export default LogIn;
```

Рисунок 7. Сторінка Login

На рисунку 8 показано як описані деякі додаткові стилі для сторінки Login.

```css
.logInBox {
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  background-color: #282c34;
  min-height: 100vh;
}

.mainText {
  color: #ffc107;
  font-family: Impact, Haettenschweiler, "Arial Narrow Bold", sans-serif;
  position: absolute;
  width: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
  top: 50px;
}

.text {
  text-align: center;
  font-size: x-large;
}

.box {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  background-color: white;
  border-radius: 10px;
  padding: 50px;
  height: 400px;
  width: 450px;
}

.button {
  width: 100%;
  border-radius: 50px;
}
```

Рисунок 8. Додаткові стилі для сторінки Login

Структура інших сторінок реалізована аналогічно.

ТЕСТУВАННЯ СИСТЕМИ

Таблиця нижче містить інформацію та результати тестування отримання завдання за id.

Таблиця 17. GetTaskByIdTest

Показик	**Значення**
Опис тесту	Отримати завдання за його id
Початкові умови	Id = 1
Очікуваний результат	Отримати завдання із заголовком Task1
Результат тесту	Тест пройдений- помилок не виявлено

Таблиця 18 показує опис тесту для отримання всіх існуючих завдань та його результати.

Таблиця 18. GetAllTasksTest

Показик	**Значення**

Опис тесту	Перевіряється коректність запиту на отримання списку всіх завдань
Початкові умови	-
Очікуваний результат	Отримати всі існуючі завдання, список "not null"
Результат тесту	Тест пройдений- помилок не виявлено

У таблиці 19 показаний результат перевірки коректності додавання нового завдання.

Таблиця 19. AddTaskTest

Показик	**Значення**
Опис тесту	Перевіряється коректність створення нового завдання
Початкові умови	Title = "TaskTest", AssigneeId = 1, ReporterId = 1, StatusId = 1, ProjectId = 1, TypeId = 1
Очікуваний результат	Модель валідна
Результат тесту	Тест пройдений- помилок не виявлено

Таблиця нижче містить інформацію та результат виконання тесту видалення завдання.

Таблиця 20. DeleteTaskTest

Показик	**Значення**
Опис тесту	Перевіряється коректність запиту на видалення завдання
Початкові умови	Id = 1
Очікуваний результат	Отримуємо статус "OK"
Результат тесту	Тест пройдений- помилок не виявлено

Також було протестовано контролер користувача. Наведемо деякі з них.

На таблиці нижче показано опис тесту перевірки отримання користувача за id.

Таблиця 21. GetUserByIdTest

Показик	**Значення**
Опис тесту	Перевіряється коректність запиту на отримання користувача за його id.
Початкові умови	Id = 1
Очікуваний результат	Отримуємо користувача з потрібним id
Результат тесту	Тест пройдений- помилок не виявлено

Таблиця 22 показує опис тесту для перевірки коректності видалення користувача.

Таблиця 22. DeleteUserTest

Показик	**Значення**
Опис тесту	Перевіряється коректність запиту на видалення користувача за його id.
Початкові умови	Id = 1
Очікуваний результат	Отримуємо статус "OK"
Результат тесту	Тест пройдений- помилок не виявлено

Скріншот виконання тестів наведено на рисунку 9.

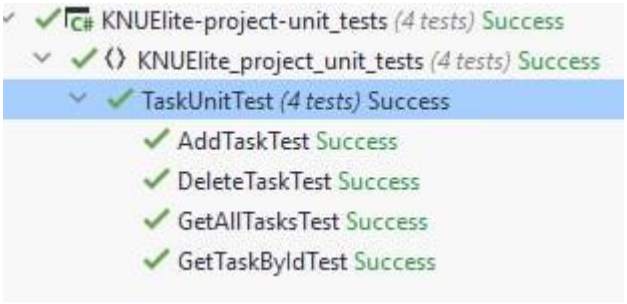


Рисунок 9. Результати тестів

7. Деплой

Для розгортання системи на віддаленій машині був використаний сервіс Heroku. Було створено 3 проєкти для таких частин:

- клієнтської;
- серверної;
- бази даних.

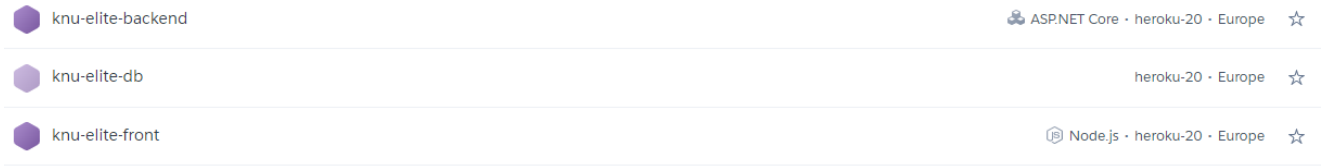


Рисунок 10. Проєкти, створені в Heroku

ІНСТРУКЦІЯ КОРИСТУВАЧА

Для початку роботи у системі потрібно зареєструватися або увійти. Вікно входу в аккаунт виглядає наступним чином:

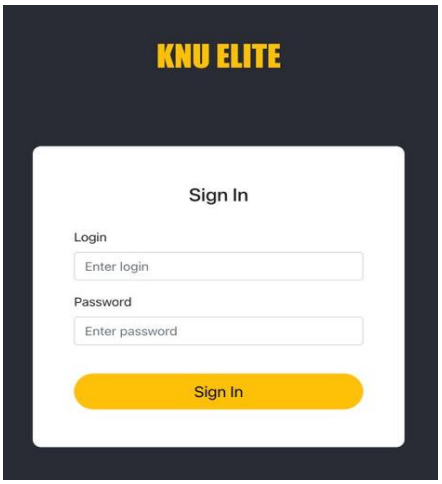


Рисунок 11. Вікно авторизації

Вводимо логін та пароль, натискаємо "Sign In". Введені дані перевіряються. При некоректному вводі отримаємо відповідне сповіщення:

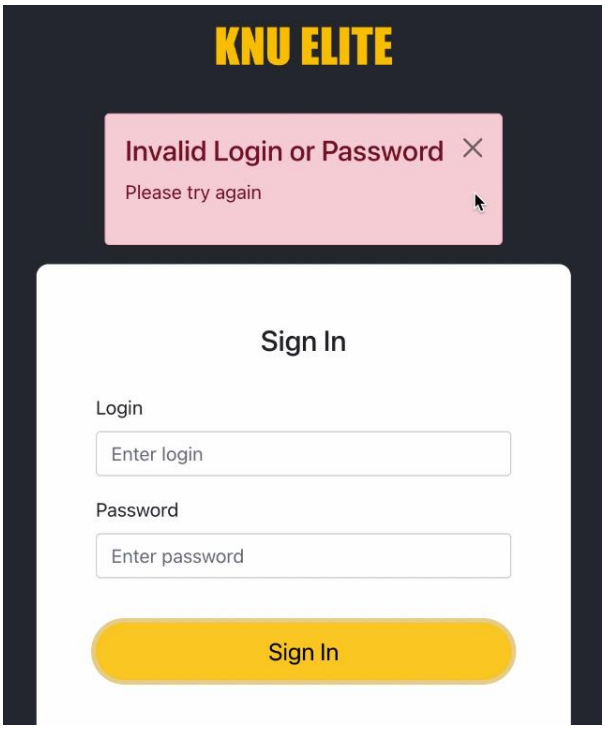


Рисунок 12. Неуспішний вхід

Після успішної авторизації потрапляємо на головну сторінку застосунку. Вона представляє собою Kanban-дошку із завданнями.

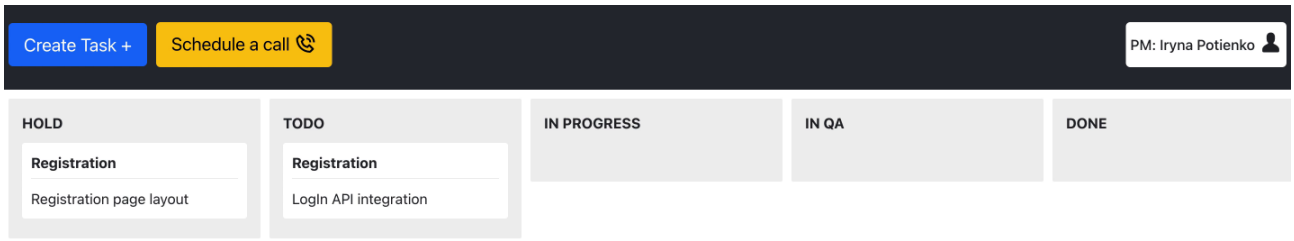


Рисунок 13. Kanban-дошка

Для того, щоб створити завдання потрібно натиснути кнопку "Create Task+" у лівому верхньому кутку.

Відкриється нове вікно, у якому потрібно ввести необхідні дані.

Для того, щоб запланувати онлайн-конференцію, слід натиснути кнопку "Schedule a call", після чого з’явиться завдання у окремій колонці CALLS із посиланням на зустріч для кожного учасника проєкту.

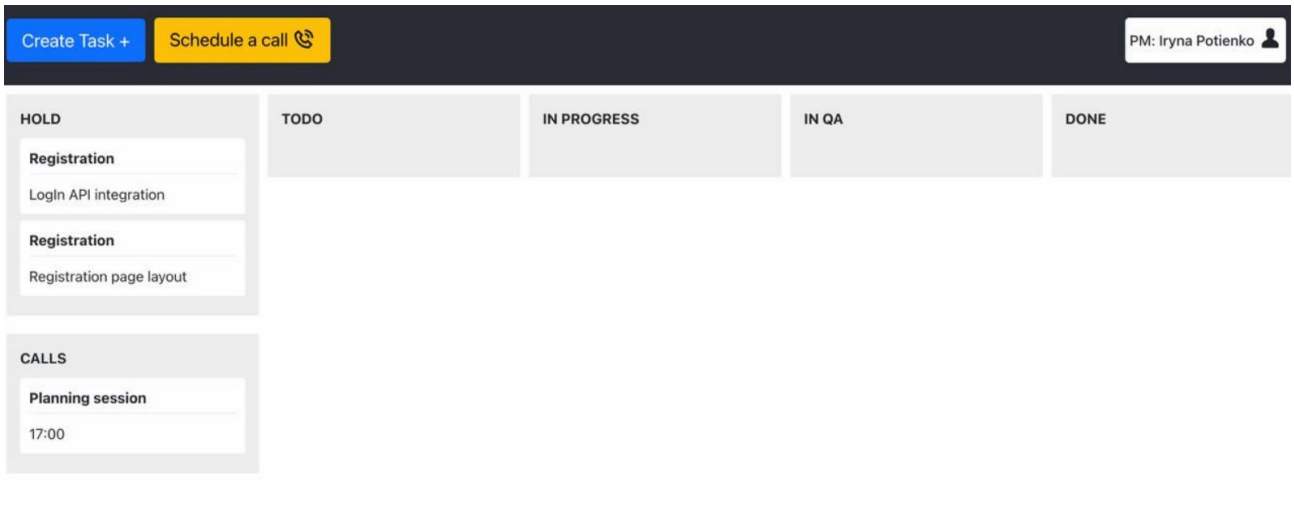


Рисунок 14. Дошка із запланованою конференцією

Завдання можна переміщати по дошці, змінюючи таким чином їх статус. Для цього натисніть на завдання та перетягніть його у відповідну колонку.

ВИСНОВКИ

У результаті роботи над груповим проєктом команда проаналізувала системи на ринку із урахуванням їх переваг та недоліків, розробила систему підтримки управління проєктами, підвищила рівень професійних та соціальних навичок. Кожен учасник отримав новий досвід у проєктуванні та реалізації веб-сайтів. Поглибнено знання в використанні таких технологій розробки:

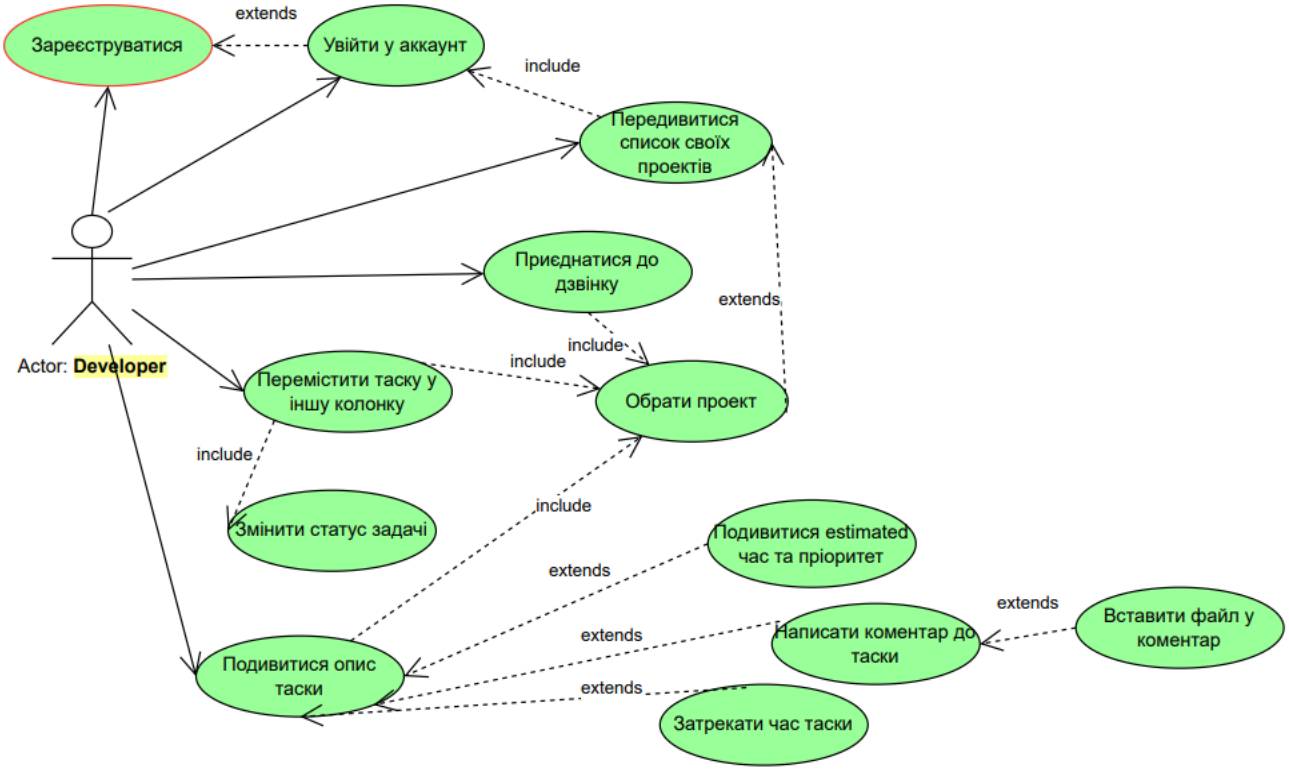
Сервіс: C#, ASP.NET Core, Entity Framework, MySQL; Деплоймент: Heroku; Фронтенд: JS, React, CSS, Bootstrap.

Підвищено рівень розуміння всіх етапів життєвого циклу програмного забезпечення, підходів до управління IT-проєктами, посилено практичну складову навчальних дисциплін, задіяних в проєкті. Було розроблені на використані Unit тести для тестування системи. Усі вони пройшли успішно. Створений продукт частково відповідає наданим технічним вимогам, реалізує основний функціонал.

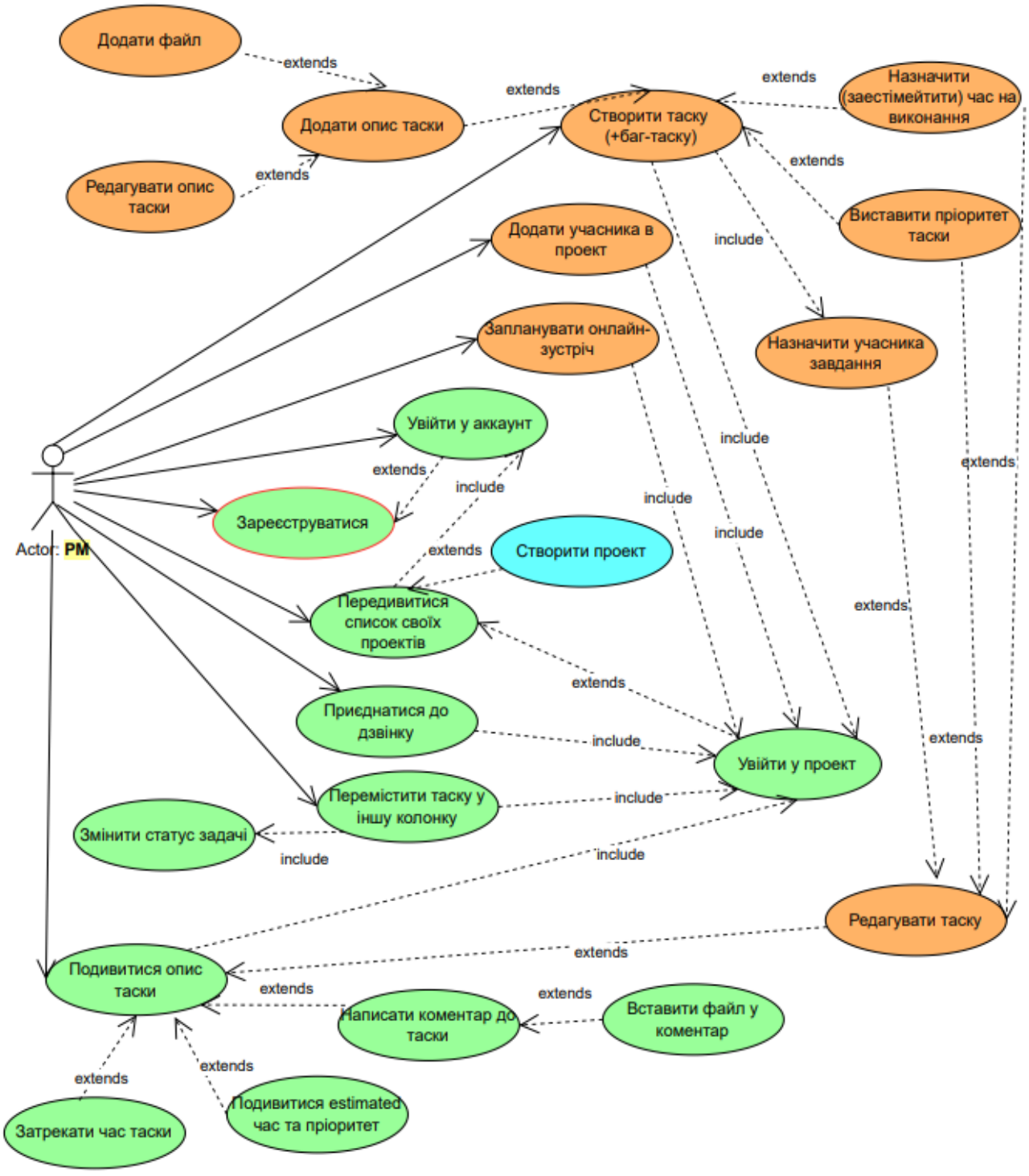
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/
2. Jira [Електронний ресурс]. – Режим доступу до ресурсу: https://www.atlassian.com/ru/software/jira
3. Microsoft Project [Електронний ресурс]. – Режим доступу до ресурсу: https://www.microsoft.com/en-us/microsoft-365/project/project-management-software
4. YouGile [Електронний ресурс]. – Режим доступу до ресурсу: https://en.yougile.com/
5. C# - Герберт Шилдт. C# 4.0: повне керівництво = C# 4.0 The Complete Reference. — М.: "Вильямс", 2010.
6. ASP.NET Core [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-6.0
7. Entity Framework [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/ef/
8. MySQL [Електронний ресурс] – Режим доступу до ресурсу: https://www.mysql.com/
9. Heroku [Електронний ресурс] – Режим доступу до ресурсу: https://www.heroku.com/
10. JavaScript - Kyle Simpson, You Don’t Know JS, 2015
11. CSS [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3.org/Style/CSS/Overview.en.html
12. React [Електронний ресурс] – Режим доступу до ресурсу: https://reactjs.org/
13. Bootstrap React [Електронний ресурс] – Режим доступу до ресурсу: https://react-bootstrap.github.io/

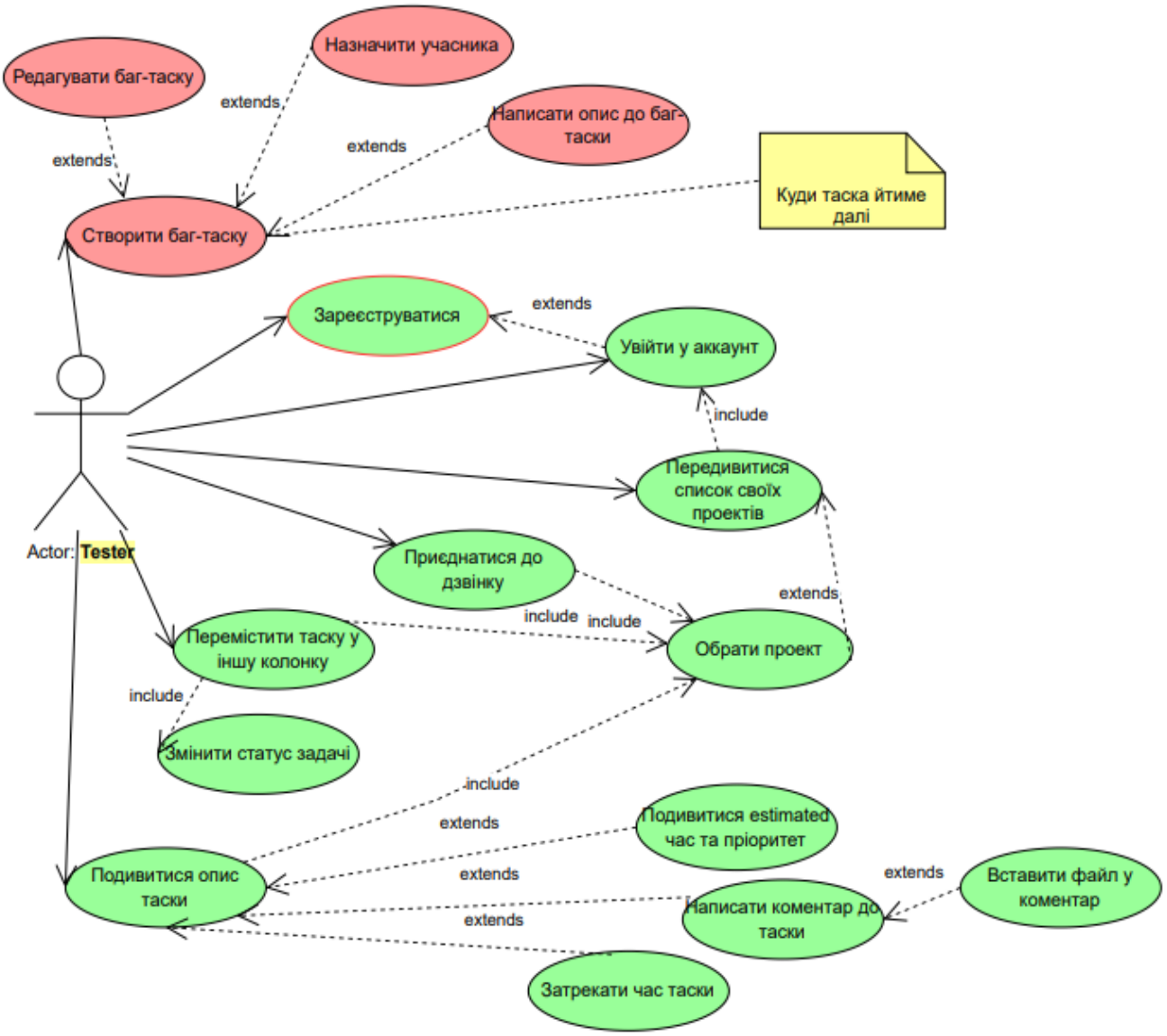
ДОДАТОК А. Use-case діаграма можливостей розробника



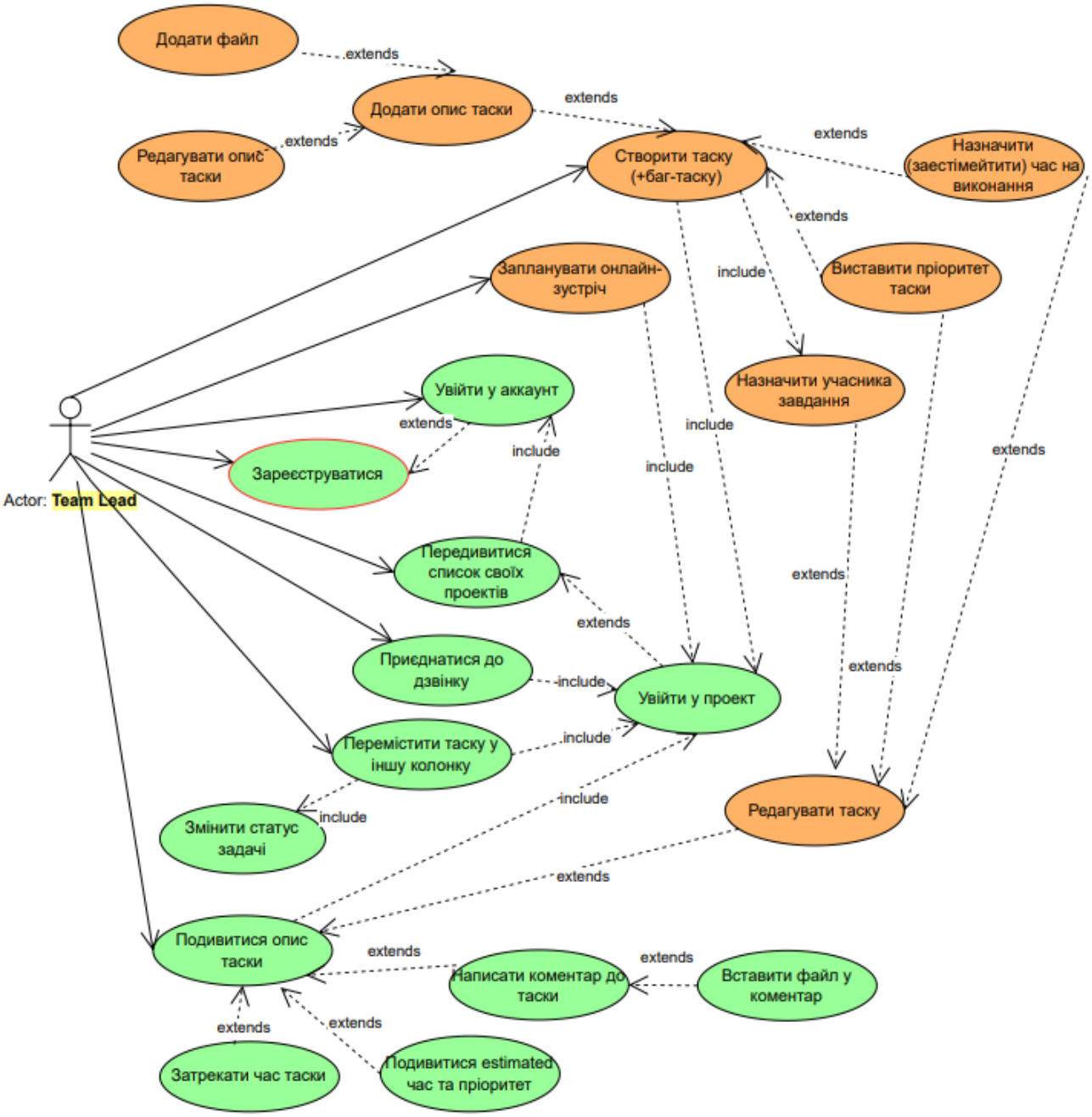
ДОДАТОК Б. Use-case діаграма можливостей менеджера проєкту



ДОДАТОК В. Use-case діаграма можливостей тестувальника



ДОДАТОК Г. Use-case діаграма можливостей керівника команди



РОЗРОБКА ВЕБСИСТЕМИ ПІДТРИМКИ УПРАВЛІННЯ ІТ-ПРОЕКТОМ

Максим Федоренко, Микита Дерябін, Денис Постільняк, Сергій Крук, Ігор Скоробогатько, Владислав Близнюк

ВСТУП

Актуальність. Управління проектом включає в себе багато різних аспектів і багато речей, які необхідно відстежувати і виконувати. Ви повинні визначити завдання проекту та їх виконавців, створити розклад, призначити ресурси, а також визначити й відстежувати проблеми та ризики. Система управління проектом – це засіб управління проектом шляхом планування, організації та управління різними його необхідними аспектами. Залежно від складності, система управління проектами може включати:

- Оціночна діяльність.
- Планування.
- Контроль витрат та управління бюджетом.
- Розподіл ресурсів.
- Управління якістю.
- Управління ризиками.
- Управління прийняттям рішень.

Оскільки з часом складність та масштаб проектів лише зростають, такі системи управління є наднеобхідними для належної організації роботи та підвищення успіху.

Цілі і завдання робити. Метою роботи є розробка веб-системи підтримки управління ІТ-проектами, яка надасть змогу вести онлайн облік виконання конкретних задач у межах проекту. Для досягнення цієї мети було поставлено такі завдання:

- дослідити існуючі системи для управління ІТ-проектами, зробити їх порівняння.
- дослідити застосування різних технологій для проєктування та реалізації вебсистеми.
- розробити технічне завдання до програмного застосунку.
- ознайомитися з фреймворком Angular та поглибити знання з веброзробки.
- розробити інтерфейс та функціонал вебсистеми.

Об’єкт, методи і засоби розробки. Об’єктом розробки вебсистеми є процес керування ІТ-проектом. Предметом роботи є вебзастосунок для забезпечення ведення обліку виконання задач у межах проекту. IntelliJ IDEA IDE було обрано було обрано в якості інструменту створення програмного засобу. Для unit-тестування було використано бібліотеку JUnit та фреймворк Mockito.

Методологія та система управління проєктами. Для управління проєктом було обрано Scrum. Зустрічі проводилися на початку реалізації проєкту кожного дня (іноді двічі на день). По мірі виконання завдань, кількість зустрічей була зменшена.

Розподіл ролей

Для виконання задач кожен учасник отримав роль:

Учасник	Роль
Денис Постільняк	Frontend-розробник
Максим Федоренко	Backend-розробник
Микита Дерябін	Backend-розробник
Сергій Крук	Фахівець з документації
Скоробогатько Ігор	Frontend-розробник
Владислав Близнюк	Frontend-розробник

ОГЛЯД ІСНУЮЧИХ НА РИНКУ СИСТЕМ

1. Trello [1]. Безкоштовна багатоплатформна система управління проектами. Переваги - продукт має інтуїтивно зрозумілий і знайомий інтерфейс керування завданнями в стилі Kanban-дошки. Проекти зображуються дошками, що містять списки. Списки містять картки, якими зображуються задачі. Серед мінусів – відсутня розширена функціональність, обмеженість використання у складних проектах.

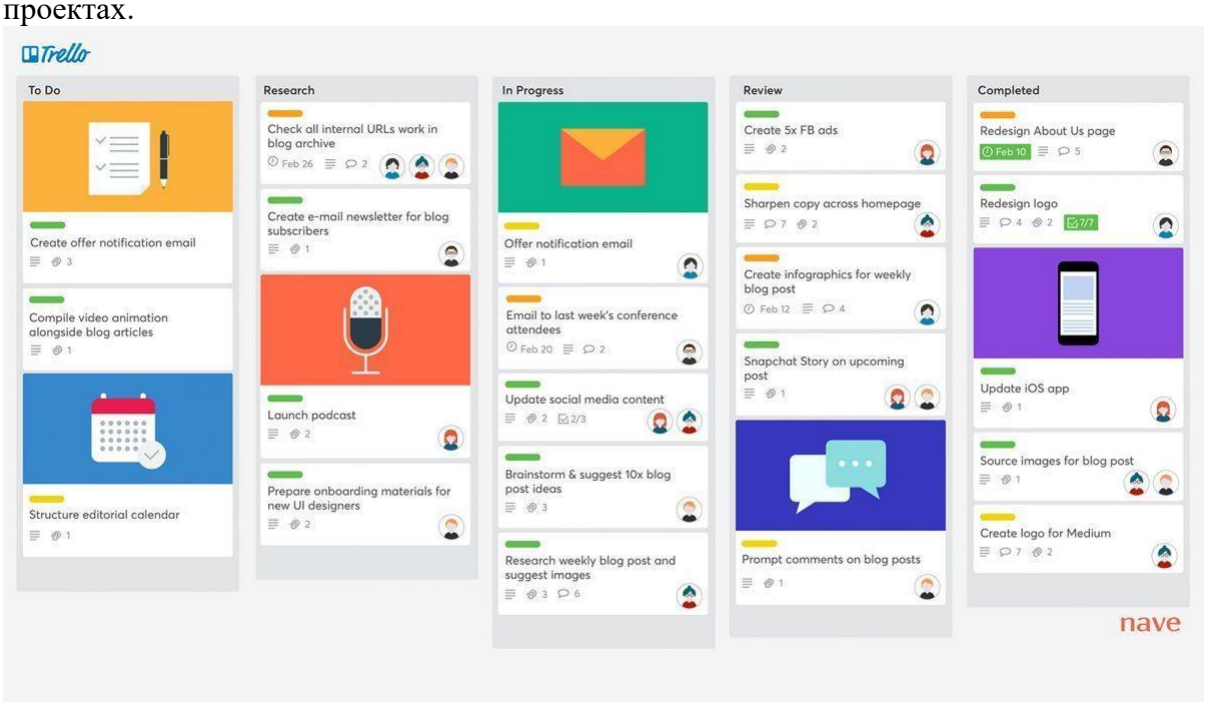


Рисунок 1. Інтерфейс "Trello"

2. Jira [2]. Система "Jira" — це програмне забезпечення, яке використовується для відстеження проблем та управління проектами. Інструмент, розроблений австралійською програмною компанією Atlassian, став широко використовуватися командами гнучких розробників для відстеження помилок, історій, епосів та інших завдань. Система Jira пропонує різні інформаційні борди та схеми для технік Scrum та Agile, дозволяє інтегруватися з такими програмами, як Confluence, Trello, Slack, GitHub, Bitbucket, Microsoft, Google тощо. Порівняно з

Trello, має більший функціонал, але менш інтуїтивний дизайн.

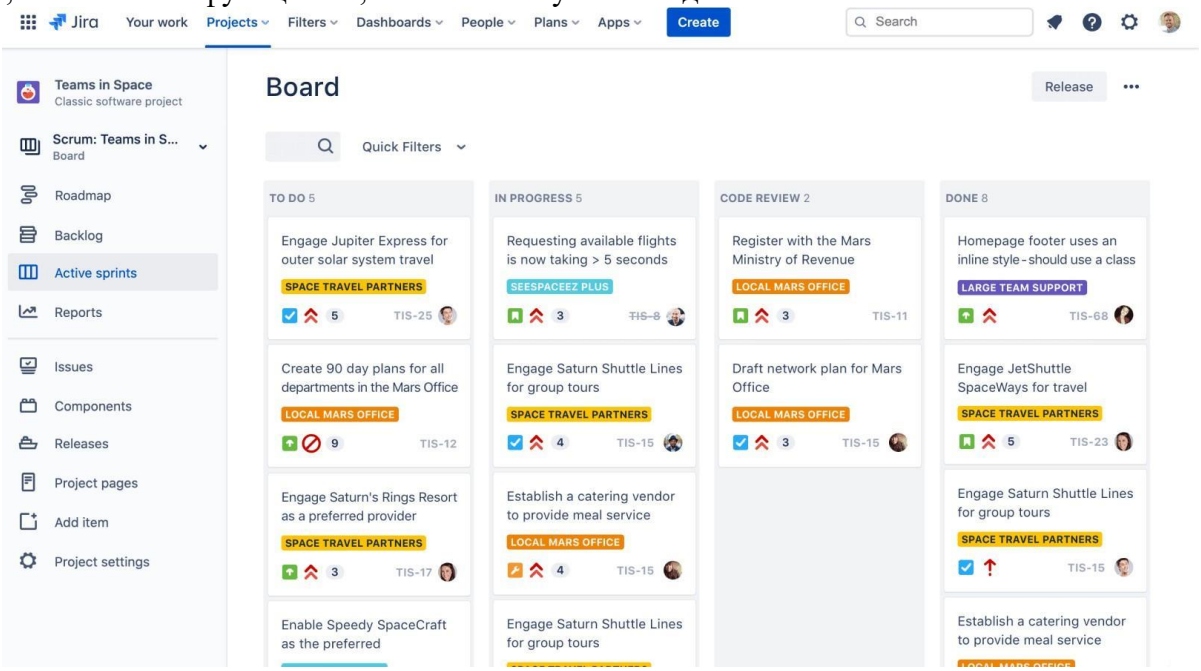


Рисунок 2. Інтерфейс Jira

Було прийняте рішення реалізувати свою систему у вигляді Kanban- дошки за аналогом Trello та розширити фунціонал, а саме: додати до

системи розподіл ролей та взаємодію через телеграм-бот задля перегляду поточних задач.

ПРИЗНАЧЕННЯ І ЦІЛІ СТВОРЕННЯ СИСТЕМИ

Призначення системи

Призначення вебсистеми "BlueprintHub" є автоматизація процесу управління IT-проектами, що дає змогу ефективно керувати процесами виконання задач.

Робота передбачає:

1) Аналіз методів, методик і моделей, що застосовуються для розв’язання задач створення комплексних вебсистем
2) Аналіз наявних програмних засобів в галузі управління проектами
3) Проєктування та програмну реалізацію системи "BlueprintHub"
4) Апробацію "BlueprintHub"

Цілі створення системи

"BlueprintHub" створюється з метою надання користувачам системи підтримки управління IT-проектами, що б покращити робочий процес для команд в межах певного проетку.

Вимоги до системи

Розробка системи повинна будуватись з використанням підходів ООП, з уніфікацією програмно-технічних засобів розробки прикладної функціональності з використанням сучасних веб-портальних, сервісно- орієнтованих технологій.

Вимоги до системи у цілому

Система "BlueprintHub" повинна мати архітектуру, побудовану на сучасних технологіях зберігання, обробки, даних та доступу до них; забезпечувати одночасну роботу декількох користувачів.

В системі передбачається виділити наступні функціональні підсистеми:

- адміністративна, призначена для керування користувачами, зміною їх ролей у проектах
- підсистема project manager'а, призначеня для керування конкретними проектами
- підсистема користувача, призначена для роботи з Kanban-дошкою для моніторингу поточних задач та їх виконання у кожному проекті

Вимоги до функцій, які виконуються системою

Підсистема адміністратора

Таблиця 1. Перелік функцій, задач, що підлягають автоматизації

Функція	Задача
Керування роботою користувачів системи	Надання конкретної ролі користувачам
	Видалення користувачів із системи

Підсистема PM'а

Таблиця 2. Перелік функцій, задач, що підлягають автоматизації

Функція	Задача
Управління проектами	Створення проекту
	Редагування проекту
	Видалення проекту
Управління задачами в межах проекту	Створення задач
	Редагування задач
	Видалення задач
	Додавання користувачів на задачу
	Надання задачі окремого статусу виконання

Підсистема користувача

Таблиця 3 Перелік функцій, задач, що підлягають автоматизації

Функція	Задача
Реєстрація	Введення інформації про себе

	Оновлення реєстрації
	Видалення інформації про себе
Робота в проекті	Під’єднання до проекту по коду
	Перегляд Kanban-дошки
	Написати коментар
	Надання задачі окремого статусу окрім "Done" і "Rejected"
Робота з Telegram-ботом	Під’єднання до системи "BlueprintHub"
	Перегляд поточних задач
	Отримання сповіщень про нові задачі

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.

ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Java [3] – об'єктно-орієнтована мова програмування, випущена 1995 року компанією "Sun Microsystems" як основний компонент платформи Java. З 2009 року мовою займається компанія "Oracle", яка того року придбала "Sun Microsystems". В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи.

Spring Framework [4] — це програмний каркас з відкритим кодом та контейнери з підтримкою інверсії управління для платформи Java. Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але є розширення для створення вебдодатків на платформі Java EE.

TypeScript [5] — мова програмування, представлена Microsoft восени 2012; позиціонується як засіб розробки вебзастосунків, що розширює можливості Java Script. TypeScript є зворотньо сумісним з JavaScript. Фактично, після компіляції програму на TypeScript можна виконувати в будь-якому сучасному браузері або використовувати спільно з серверною платформою Node.js.

Angular [6] — написаний на TypeScript front-end фреймворк з відкритим кодом, який розробляється під керівництвом Angular Team у компанії Google, а також спільнотою приватних розробників та корпорацій. Angular - це AngularJS, який був переосмислений та перероблений тією ж командою розробників. Angular JS став одним із найпопулярніших фреймворків JavaScript, оскільки він простий у використанні, забезпечує високе спрощення всього процесу розробки, а також підтримує останні стандарти JavaScript (ES6). Протягом багатьох років він став одним із

найдосконаліших фреймворків для створення модульних і адаптивних веб та мобільних додатків.

Trello - це одна з найпопулярніших систем управління проєктами в режимі онлайн, яка користується особливим попитом серед невеликих компаній і стартапів. Вона дозволяє ефективно організовувати роботу по японській методології канбан-дощок.

Дані технології були обрані через їх актуальність та добре організовану лаконічну документацію.

ОПИС ОРГАНІЗАЦІЇ ІНФОРМАЦІЙНОЇ БАЗИ

Для збереження даних вебсистеми було обрано СУБД PostgreSQL. Додатково для збереження фотографій профілю користувача було використано Firebase Storage[7].

Логічна структура бази даних

Перелік таблиць:

Номер	Таблиця	Опис
1	PasswordResetToken	Таблиця для збереження токену для відновлення паролю
2	ConfirmationToken	Таблиця для збереження токену для підтвердження реєстрації
3	Comment	Таблиця для збереження коментарів під завданнями
4	Project	Таблиця для збереження даних про конкретний проект
5	Task	Таблиця для збереження даних про конкретну задачу в межах проекту
6	User	Таблиця для збереження даних про користувача

Перелік користувацьких типів:

Номер	Тип	Значення	Опис
1	Role	PM, Admin, User	Роль користувача системи
2	Type	Bug, Front, Back	Тип задачі
3	Status	toDo, inProgress, Review, Done, Rejected	Статус виконання задачі
4	Category	Business, Software, Marketing	Категорія проекту

Опис таблиць

Таблиця PasswordResetToken

Атрибут	Тип	Опис
Id	Long	Ідентифікатор
Token	String	Токен
Created	Date	Дата створення

User	User	Користувач

Таблиця PasswordResetToken

Атрибут	Тип	Опис
Id	Long	Ідентифікатор
Token	String	Токен
Created	Date	Дата створення
Expired	Date	Дата закічення дії
User	User	Користувач

Таблиця Comment

Атрибут	Тип	Опис
Id	Long	Ідентифікатор
Text	String	Текст коментаря
User	User	Користувач, що написав коментар
Created	Date	Дата створення коментаря

Таблиця Project

Атрибут	Тип	Опис
Id	Long	Ідентифікатор
Name	String	Назва проекту
Owner	User	Власник
Category	Category	Категорія проекту
Created	Date	Дата створення
Team	List<User>	Перелік команди
Tasks	List<Task>	Перелік задач проекту

Таблиця User

Атрибут	Тип	Опис
Id	Long	Ідентифікатор

Username	String	Юзернейм
Email	String	Електронна пошта
firstName	String	Прізвище
lastName	String	Ім’я
birthDay	Date	Дата народження
Password	String	Пароль
Active	Boolean	Пітверджений акаунт
Locked	Boolean	Заблокований акаунт
Roles	List<Role>	Перелік ролей
Projects	List<Project>	Перелік проектів
Tasks	List<Task>	Перелік задач
imageName	String	Назва фотографії
imageUrl	String	URL фотографії
ownProjects	List<Project>	Перелік власних проектів
telegramID	Long	ID телеграм-юзера

Таблиця Task

Атрибут	Тип	Опис
Id	Long	Ідентифікатор
Title	String	Назва задачі
description	String	Опис задачі
Reporter	User	Доповідач
Status	Status	Статус задачі

Type	Type	Тип задачі
Members	List<User>	Перелік користувачів, які працюють над даною задачею
createdAt	Date	Дата створення задачі
updatedAt	Date	Дата оновлення задачі
comments	List<Comment>	Перелік коментарів

Для збереження фотографій у Firebase Storage було створено окрему директорію, де кожна фотографія матиме унікальне ім’я, що буде зберігатися у таблиці у полі "imageName".

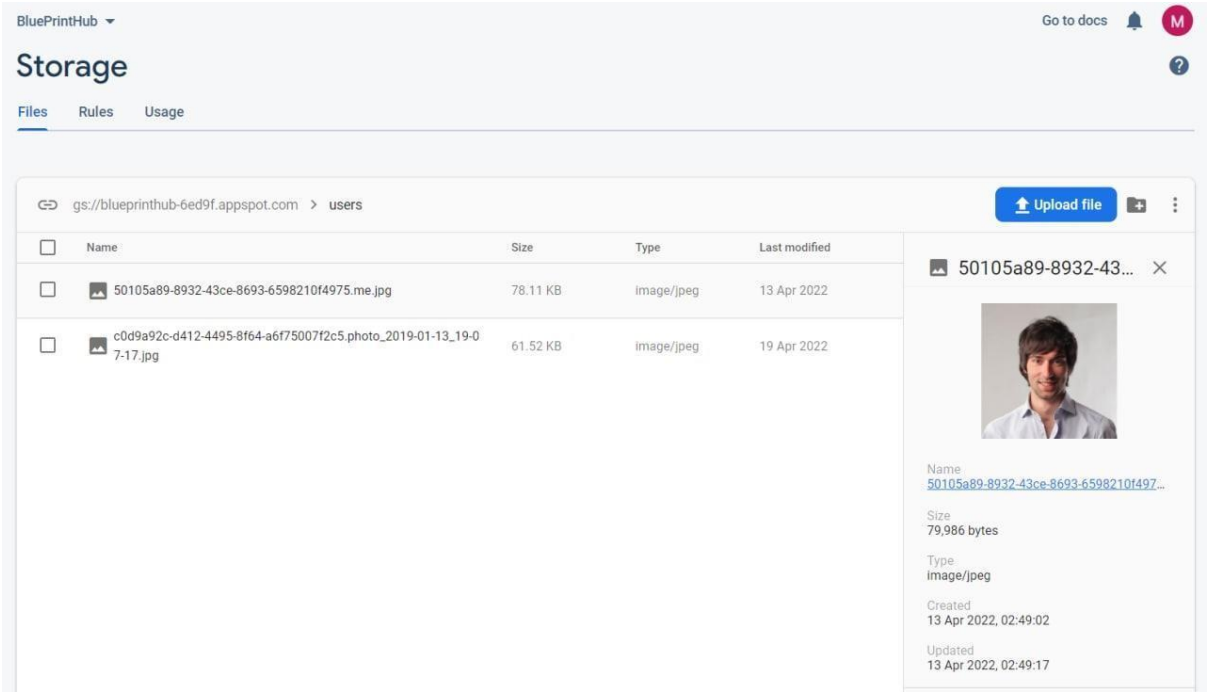


Рисунок 3. Збереження даних у Firebase Storage

РЕАЛІЗАЦІЯ СИСТЕМИ

Проект був розроблений з використання Java Spring Framework та Angular Framework. Spring Boot є модулем Spring Framework. Він використовується для створення автономних додатків з мінімальними затратами. Spring Boot дотримується багатошарової архітектури, в якій кожен шар взаємодіє з шаром безпосередньо під ним або над ним (ієрархічна структура). Перш ніж зрозуміти архітектуру Spring Boot, ми повинні знати різні шари та класи, присутні в ній. У Spring Boot є чотири рівні:

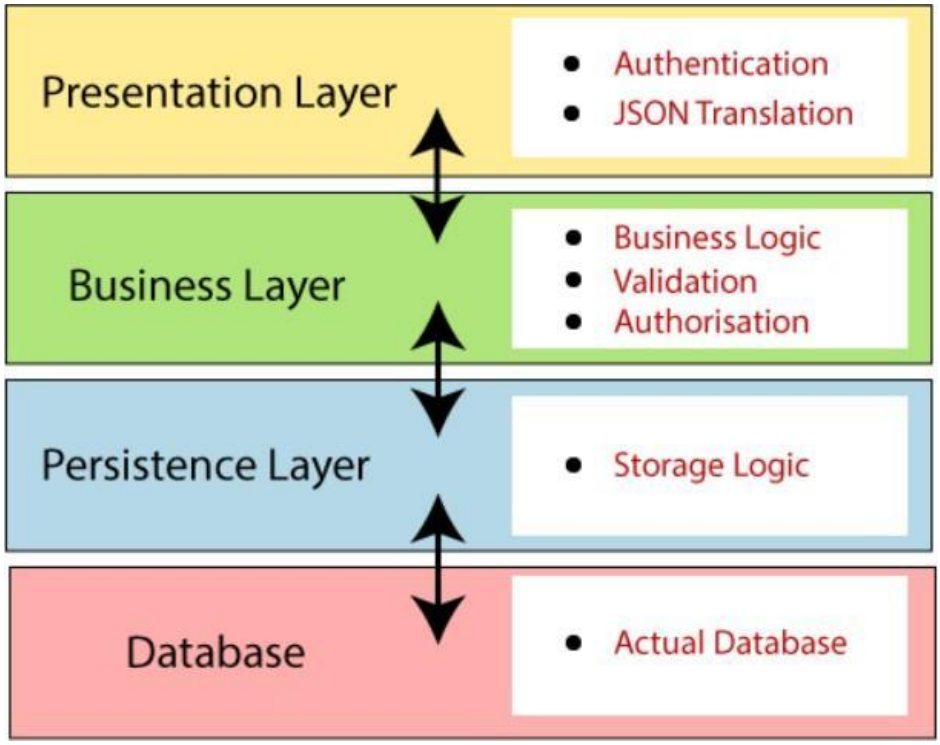


Рисунок 4. Рівні Spring Boot архітектури

Для збереження даних було використано СУБД PostgreSQL, для збереження зображень було використано Firebase API. Для інтеграції проекту з Telegram було використано Telegram API Bot.

URL схема REST API має наступний вигляд:

URL	**Тип HTTP-запиту**	**Опис**
/users	GET	Перегляд всіх користувачів системи
/used/{id}/change-role	POST	Зміна ролі користувача
/projects	GET	Перегляд всіх проектів системи
/login	POST	Логін у систему
/sign-out	POST	Вихід із систему
/register	POST	Реєстрація
/confirm/token	GET	Підтвердження профіля по токену
/reset-password/{token}	GET, POST	Зміна паролю через токен
/profile/projects	GET	Перелік проектів користувача
/profile/tasks	GET	Перелік задач користувача
/profile/edit	PUT	Редагування профілю користувача
/profile/change-password	PUT	Зміна паролю користувача
/project/{code}	GET	User

/connect	POST	Приєднання до проекту
/project/create	POST	Створення проекту
/project/{code}/edit	PUT	Редагування проекту
/project/{code}/delete	DELETE	Видалення проекту
/project/{code}/team	GET	Перегляд команди проекту
/project/{code}/tasks	GET	Перегляд задач проекту
/task/create	POST	Створення задачі
/task/{id}/edit	PUT	Редагування задачі
/task/{id}/change-status	PUT	Зміна статусу задачі
/task/{id}/delete	DELETE	Видалення задачі
/task/{id}/comments	GET	Перелік коментарів
/comment/add	POST	Додавання коментаря
/comment/{id}/delete	DELETE	Видалення коментаря

Для передачі даних між підсистемами програми було використано DTO(Data Transfer Object). Data Transfer Object використовуються для того, щоб не навантажувати функції великою кількістю аргументів, можна створити об'єкт, який міститиме значення в собі, та передати його.

Для прикладу DTO користувача при реєстрації:

```
@Data
public class RegisterDto {
    @Pattern(regexp = "^[a-zA-Z0-9]([._-](?![._-])|[a-zA-Z0-9])*[a-zA-Z0-9]$",
            message = "Username must contain only letters, digits and special characters(., -,  _)"
    )
    @Length(min = 3, max = 30, message = "Username be between 3 and 30 characters")
    private String username;

    @Pattern(regexp = "^[A-Z][a-z]*", message = "First name must contain only letters with first capital letter")
    @Length(min = 3, max = 30, message = "First name must be between 3 and 30 characters")
    private String firstName;

    @Pattern(regexp = "^[A-Z][a-z]*", message = "Last name must contain only letters with first capital letter")
    @Length(min = 3, max = 30, message = "Last name must be between 3 and 30 characters")
    private String lastName;

    @Email(message = "Email is not correct")
    @NotBlank(message = "Email cannot be empty")
    private String email;

    @DateTimeFormat(pattern="dd/MM/yyyy")
    private Date birthDay;

    @ValidPassword
    private String password;

    private String password2;
}
```

Рисунок 5. Контролер реєстрації нового користувача

Та відповідний контролер при реєстрації:

```
@PostMapping("/register")
public ResponseEntity<?> registration(@RequestBody @Valid RegisterDto registerDto, BindingResult bindingResult) {
    if (registerDto.getPassword() != null && !registerDto.getPassword().equals(registerDto.getPassword2()))
        return new ResponseEntity<>( body: "Passwords are different!", HttpStatus.BAD_REQUEST);

    if (bindingResult.hasErrors())
        return new ResponseEntity<>( body: "Error data!", HttpStatus.BAD_REQUEST);

    try {
        userService.signUpUser(registerDto);
        return new ResponseEntity<>( body: "User registered", HttpStatus.OK);
    }
    catch (Exception ex){
        return new ResponseEntity<>(ex.getMessage(), HttpStatus.BAD_REQUEST);
    }
}
```

Рисунок 6. Контролер реєстрації нового користувача

Структура фронтенду ґрунтується на системі модулів та компонент angular. Сторінка може бути розподілена на окремі компоненти, що

створило більше можливостей для внесення окремих змін. Для створення форм, дошки та іншого були залучені окремі зовнішні модулі. Можливість використання директив у html-файлах була використана для більш

гнучкого використання компонентів: наприклад, форми реєстрації та входу до системи, що використовують однакові частини коду,

реалізуються одним компонентом, де за допомогою директиви умови виводиться тільки те, що потрібно при конкретному використанні.

Телеграм-бот реалізований за принципом Webhook. Сервер Telegram миттєво буде перенаправляти повідомлення на обраний нами сервер.

Для початку треба було налаштувати бота в нашому проекті.

```
# Telegram Bot Config

telegrambot.userName=@blue_print_hub_bot
telegrambot.botToken=[redacted]
telegrambot.webHookPath=https://blue-print-hub.herokuapp.com/
```

Рисунок 7. Налаштування телеграм боту

- Поле webHookPath означає нашу адресу webhook, яку ми отримаємо на Heroku.
- Поле username означає назву нашого бота, яке ми отримаємо при реєстрації нашого бота в Telegram.
- Поле botToken це наш токен, який ми отримаємо при реєстрації нашого бота в Telegram.

Telegram сервер відправляє на зареєстровану адресу webhook

повідомлення у форматі JSON методом POST, наш контролер їх приймає та передає бібліотеці telegram у вигляді об'єкта Update.

Для того щоб бот працював коректно і міг обробляти повідомлення від різних користувачів і при цьому не конфліктував ми будемо для кожного юзера зберігати стан бота в кеші.

```java
public enum BotState {
    START,
    LOGIN,
    ENTER_USERNAME,
    ENTER_PASSWORD,
    MY_PROJECTS,
    OWN_PROJECTS,
    CONNECT,
    ENTER_CODE,
    CREATE_PROJECT,
    ENTER_PROJECT_NAME,
    ENTER_PROJECT_INFO,
    ENTER_PROJECT_CATEGORY,
    ENTER_TASK_TITLE,
    ENTER_TASK_INFO,
    ENTER_TASK_TYPE,
    ENTER_TASK_REPORTER,
    ENTER_TASK_TEAM
}
```

Рисунок 8. Стани телеграм боту

```java
@Service
@Setter
@Getter
public class BotStateCash {
    private final Map<Long, BotState> botStateMap = new HashMap<>();

    public void saveBotState(long userId, BotState botState) { botStateMap.put(userId, botState); }
}
```

Рисунок 9. Сервіс який зберігає стан бота для кожного користувача

У класі TelegramFacade реалізовується логіка обробки повідомлень. Повідомлення можуть бути двох типів – звичайні меседжі, тобто текст та CallbackQuery - цей об'єкт представляє запит зворотного зв'язку від інлайн-кнопки із заданим callback_data. Відповідно до кожного із цього типів у нас буде свій клас обробник. Далі приведемо приклад функцію яка обробляє повідомлення від користувача про його задачі та відповідно повертає йому у відповідь весь список завдань відповідного проекту:

```
public BotApiMethod<?> myTasksHandler(long chatId, long userId, String projectCode){
    SendMessage sendMessage = new SendMessage();
    sendMessage.setChatId(String.valueOf(chatId));

    User user = userService.findByTelegramId(userId);
    if(user == null) {
        sendMessage.setText(Emojis.MARK_FAILED + "Login to your BluePrintHub account to see your tasks");
        return sendMessage;
    }

    Project project = projectService.findByCode(projectCode);
    if(!project.getOwner().equals(user) && !project.getTeam().contains(user)){
        sendMessage.setText(Emojis.MARK_FAILED + "You have no rights");
        return sendMessage;
    }

    List<TaskDto> tasks = taskService.userTasksByProject(user, project);

    if(tasks.size() == 0){
        sendMessage.setText(Emojis.MARK_FAILED + " You have not any task");
        return sendMessage;
    }

    for(var task : tasks){
        String taskMessage = buildTask(task);
        telegramBot.sendMessage(userId, taskMessage,  markup: null);
    }

    botStateCash.saveBotState(userId, BotState.START);
    sendMessage.setText(Emojis.SUCCESS_MARK + " Tasks was successfully found");
    return sendMessage;
}
```

Рисунок 10. Приклад обробника списку завдань користувача

Структура фронтенду ґрунтується на системі модулів та компонент angular. Сторінка може бути розподілена на окремі компоненти, що

створило більше можливостей для внесення окремих змін. Для створення форм, дошки та іншого були залучені окремі зовнішні модулі. Можливість використання директив у html-файлах була використана для більш

гнучкого використання компонентів: наприклад, форми реєстрації та входу до системи, що використовують однакові частини коду,

реалізуються одним компонентом, де за допомогою директиви умови виводиться тільки те, що потрібно при конкретному використанні.

Телеграм-бот працює реалізований за принципом Webhook. Сервер

Telegram миттєво буде перенаправляти повідомлення на обраний нами сервер.

Frontend

Фронтенд частина розроблена за допомогою фреймворку Ангуляр.

Використана модульна архітектура, в якій за основу взято модулі. Кожен модуль відповідає своїй сторінці на UI частині користувача.

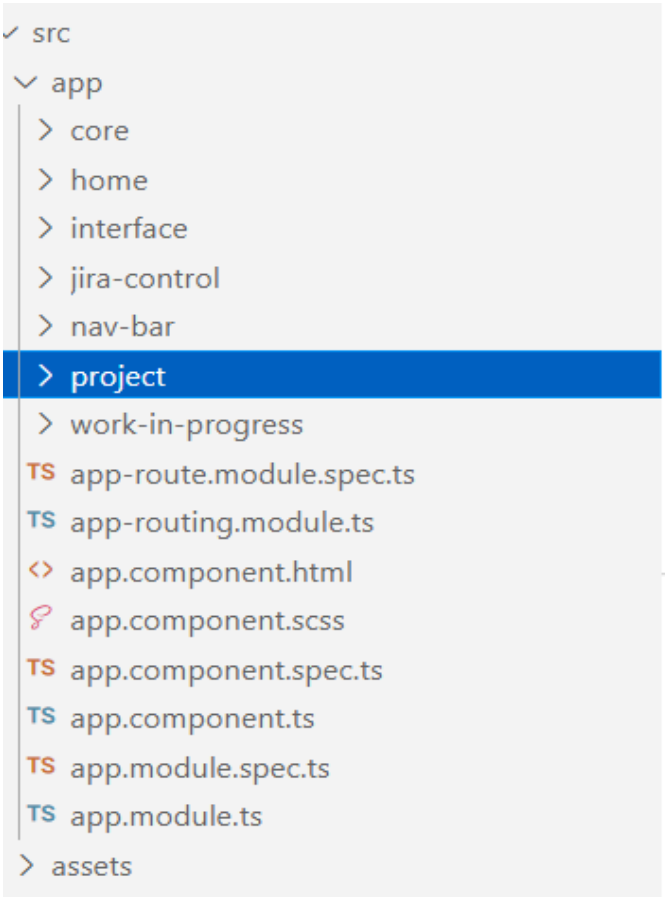


Рисунок 11. Структура модулів

Кожен модуль також поділяється на компоненти – мікро одиниці, які відповідають за відповідну одиницю в html сторінці. Ідея поділення на найменші компоненти для того щоб потім їх перевикористати.

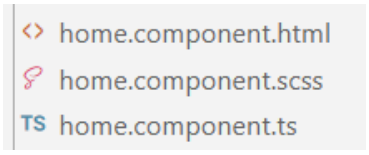


Рисунок 12. Структура компоненти

Для передачі данних були використані ДТО такі ж самі як і на бекенд частині для правильного мапінгу об’єктів.

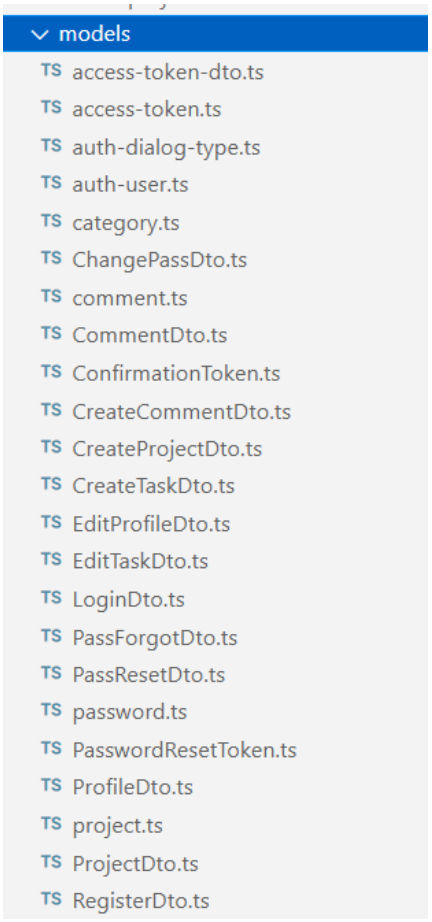


Рисунок 13. Моделі

Комунікація з бекендом відбувається через сервіси, в основі яких лежать http-запити.

```
@Injectable({ providedIn: 'root' })
export class UserService {
    public routePrefix = '/api/users';

    constructor(private httpService: HttpInternalService) { }

    public getUserFromToken() {
        return this.httpService.getFullRequest<User>(`${this.routePrefix}/fromToken`);
    }

    public getUserById(id: number) {
        return this.httpService.getFullRequest<User>(`${this.routePrefix}`, { id });
    }
```

Рисунок 14. Структура сервісу

ТЕСТУВАННЯ ВЕБСИСТЕМИ

Тестування бекенду відбувалося за допомогою unit-тестів. Для забезпечення коректності роботи основної логіки бекенд частини, тестуванню підлягали репозиторії та сервіси, які відповідають за реєстрацію, аутентифікацію, надсилання токену для підтвердження та зміни пароля, пошук (за ключовим полем, ім’ям та електронною поштою) користувача. Також були протестовані певні випадки виникнення виключень, наприклад, пошук неіснуючого користувача. Аналогічно було протестовано методи, які відповідають за створення,

редагування, видалення, пошуку проектів та задач, які може створювати користувач. Загалом тестувалися наступні класи та інтерфейси(рис.5):

- ConfirmationTokenRepository.
- PasswordResetTokenRepository.
- UserRepository.
- ConfirmationTokenRepository.
- MailSenderService.
- PasswordResetTokenService.
- UserService.
- ProjectService.
- TaskService.
- ProjectRepository.
- TaskRepository.

Для того, щоб відокремити збереження даних тестування, було використано окрему базу даних H2. Перевага даного рішення полягає в тому, що ми не маємо доступу до даних, які були завантажені під час використання застосунку. В основному, база даних H2 може бути налаштована для роботи як база даних пам'яті, що означає, що дані не будуть зберігатися на диску.

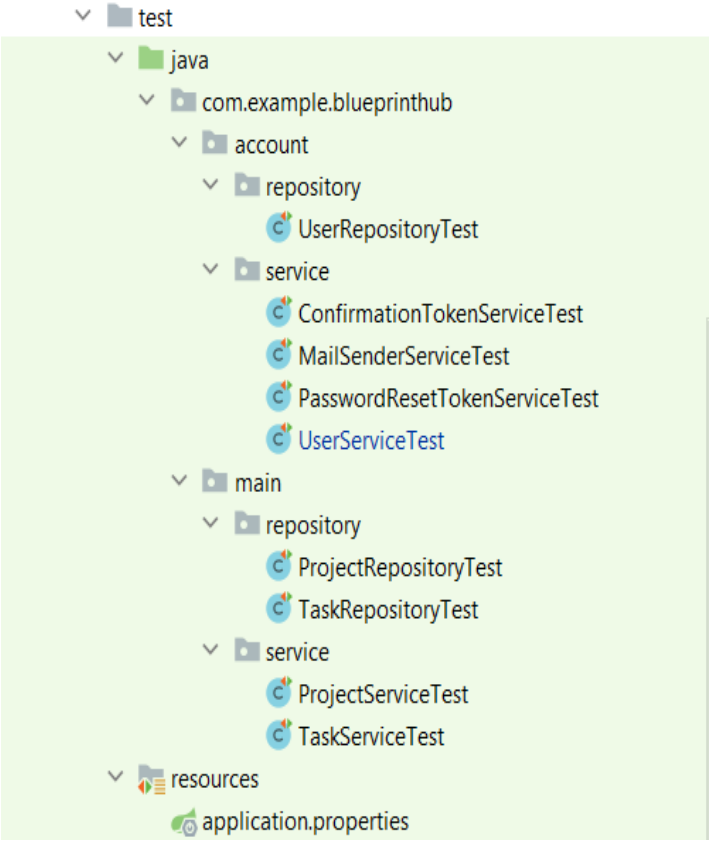


Рисунок 15. Структура класів тестування

Налаштування бази даних для тестування виглядає наступним чином:

```
# Database Config
spring.datasource.url=jdbc:h2://mem:db;DB_CLOSE_DELAY=-1
spring.datasource.username=qwer
spring.datasource.password=qwer
spring.datasource.driver-class-name=org.h2.Driver
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.show-sql=true
```

Рисунок 16. Налаштування бази даних H2

Приклади модульних тестів:

```
package com.example.blueprinthub.account.service;

import ...

@ExtendWith(MockitoExtension.class)
class UserServiceTest {

    @Mock
    private UserRepository userRepository;
    @Mock
    private PasswordEncoder passwordEncoder;
    @Mock
    private ConfirmationTokenService confirmationTokenService;
    @Mock
    ConfirmationToken confirmationToken;
    @Mock
    private PasswordResetTokenService resetTokenService;
    @Mock
    PasswordResetToken resetToken;
    @Mock
    private PasswordResetTokenRepository resetTokenRepository;
    @Mock
    private MailSenderService mailSenderService;

    @InjectMocks
    private UserService userService;

    private RegisterDto getRegisterDtoObj(){...}

    private User getUserObj(){...}

    @BeforeEach
    void setUp() {...}

    @AfterEach
    void tearDown(){...}

    @Test
    void canGetUserList() {...}

    @Test
    void canSignUpUser() throws Exception {...}

    @Test
    void userExistsWhileSignUp() throws Exception {...}

    @Test
    void canGetUserById() {...}

    @Test
    void canLoadUserByUsername() {...}
```

Рисунок 17. Основні поля та методи класу UserServiceTest

```
    @Test
    void failureToLoadUserByUserName(){...}

    @Test
    void canFindUserByEmail() {...}

    @Test
    void failureToFindUserByEmail(){...}


    @Test
    void checkConfirmUser() {...}

    @Test
    void checkForgotPassword() {...}

    @Test
    void checkResetPassword() {...}

    @Test
    void checkDeleteUser() {...}

    @Test
    void checkSendMessageConfirmType() {...}
}
```

Рисунок 18. Основні поля та методи класу UserServiceTest

Наступний метод демонструє перевірку коректної відповіді користувачу, який намагається зареєструватися з ім'ям, яке вже використовується кимось іншим.

```
@Test
void userExistsWhileSignUp() throws Exception {
    //Checking for failure to sign up if exists another user with the same username

    //given
    RegisterDto registerDto = getRegisterDtoObj();
    User user = getUserObj();

    //when
    when(userRepository.findByUsernameIgnoreCase(anyString())).thenReturn(Optional.of(user));


    //then
    assertThatThrownBy(() -> userService.signUpUser(registerDto)) AbstractThrowableAssert<capture of ?, capture of ? extends Throwable>
            .isInstanceOf(Exception.class) capture of ?
            .hasMessage("User with this username already exists");

}
```

Рисунок 19. Метод класу UserServiceTest

Покриття методів та рядків коду, які було протестовано для відповідних класів:

100% classes, 77% lines covered in 'all classes in scope'

Element	Class, %	Method, %	Line, %
com.example.blueprinthub.account.repository	100% (0/0)	100% (0/0)	100% (0/0)
com.example.blueprinthub.account.service	100% (4/4)	96% (24/25)	87% (85/97)
com.example.blueprinthub.main.service	100% (2/2)	100% (13/13)	77% (67/87)

Рисунок 20. Покриття методів та коду

Загалом було реалізовано 40 unit-тестів:

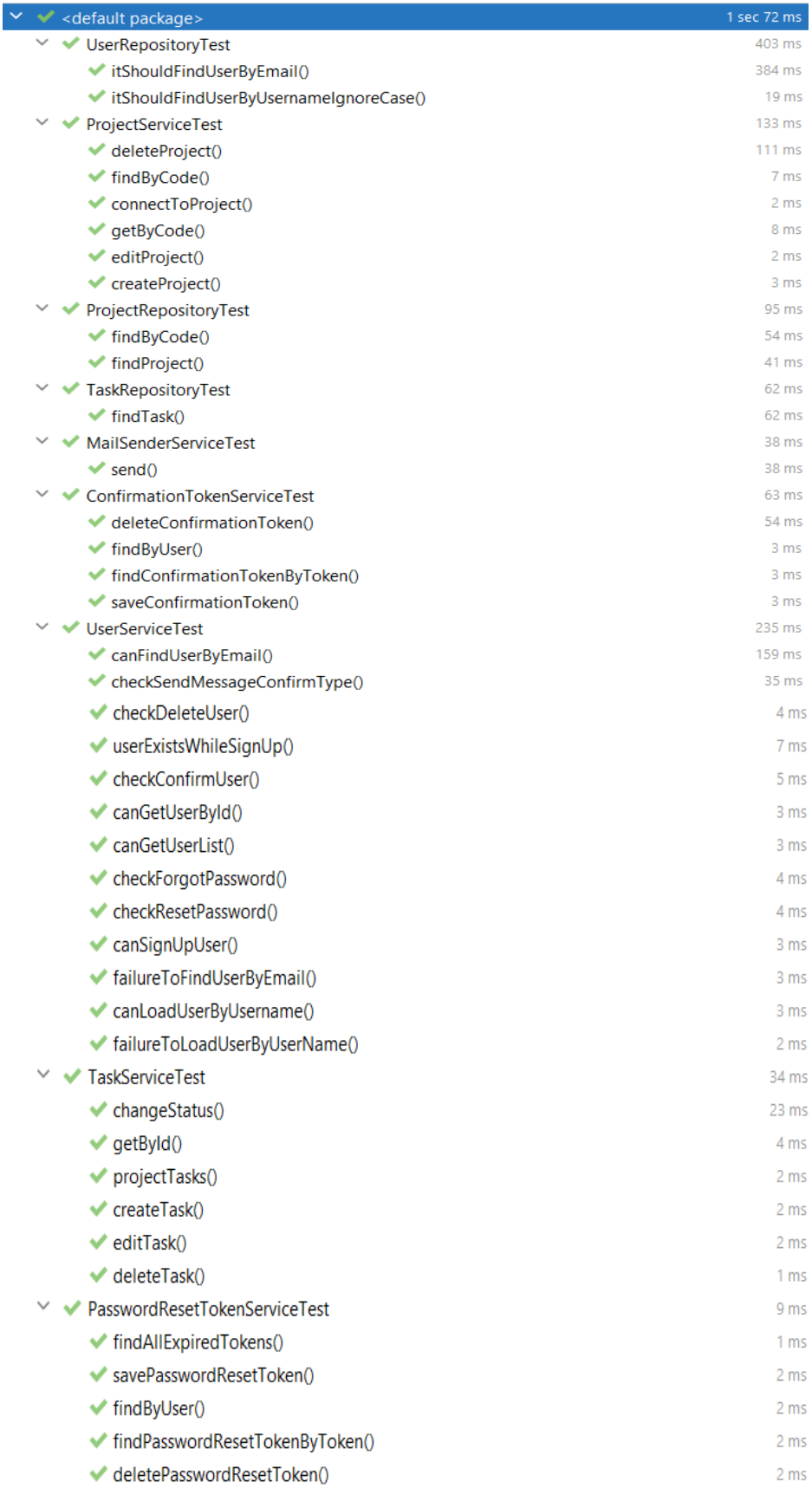


Рисунок 21. Результати тестування програми

ІНСТРУКЦІЯ КОРИСТУВАЧА

При переході на сайт користувачу потрібно авторизуватись або зареєструватись, якщо у нього не був створений акаунт.

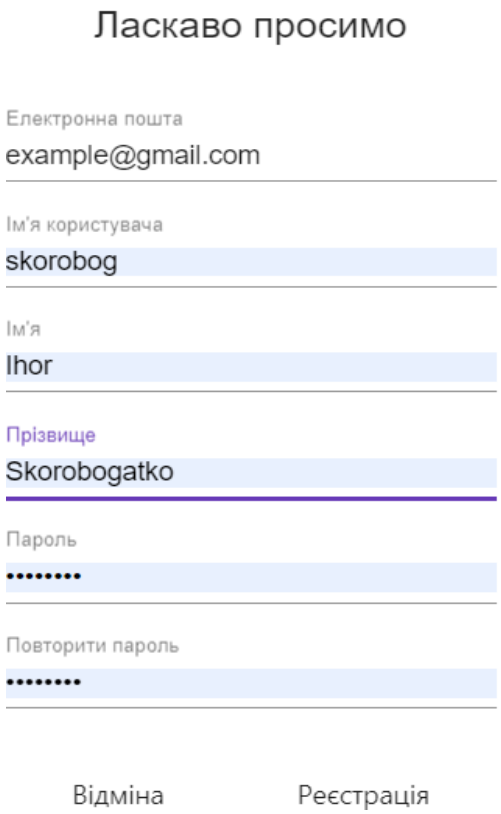


Рисунок 22. Форма реєстрації користувача

Далі за допомогою спеціального коду потрібно приєднатися до конкретного проекту. Після підключення до класу користувач має змогу переглядати список поточних задач та їх статус виконання, за необхідності змінювати їх статус на інший(окрім "Done", "Rejected"), писати коментарі. Користувач телеграм акаунт до проекту, що б отримувати сповіщення про нові задачі та переглядати список решти. Також є можливість переглянути всі свої проекти.

Рисунок 23. Перегляд списку задач для користувача у Telegram-боті

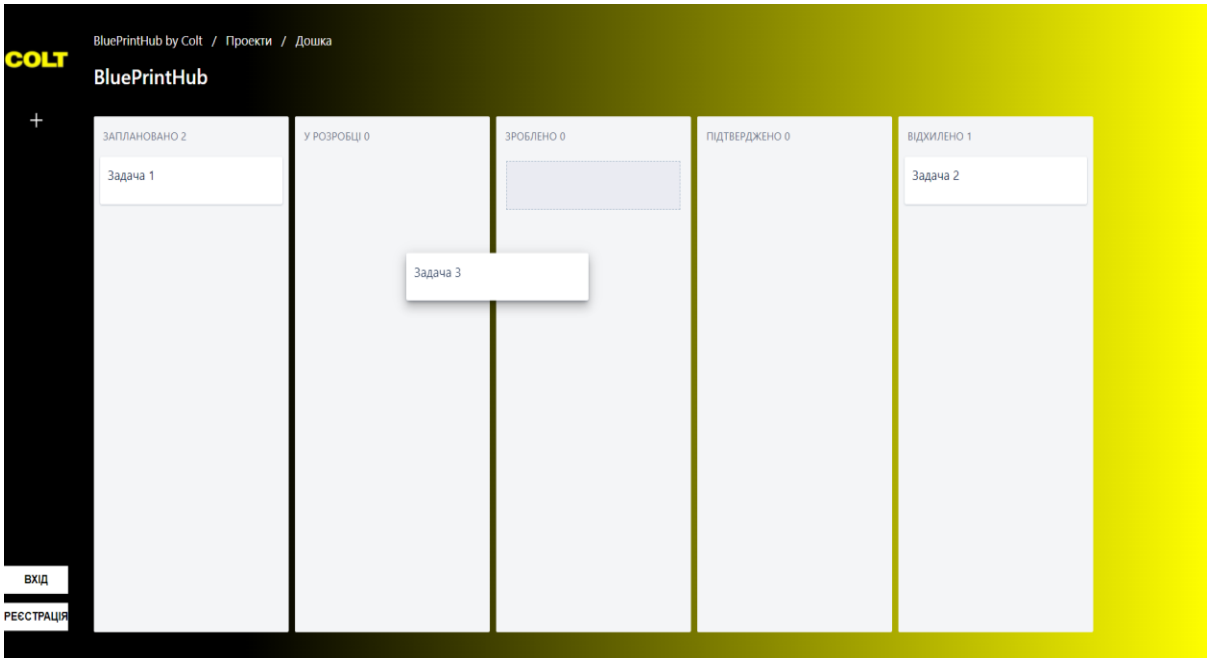


Рисунок 24. Kanban-дошка з наявними задачами

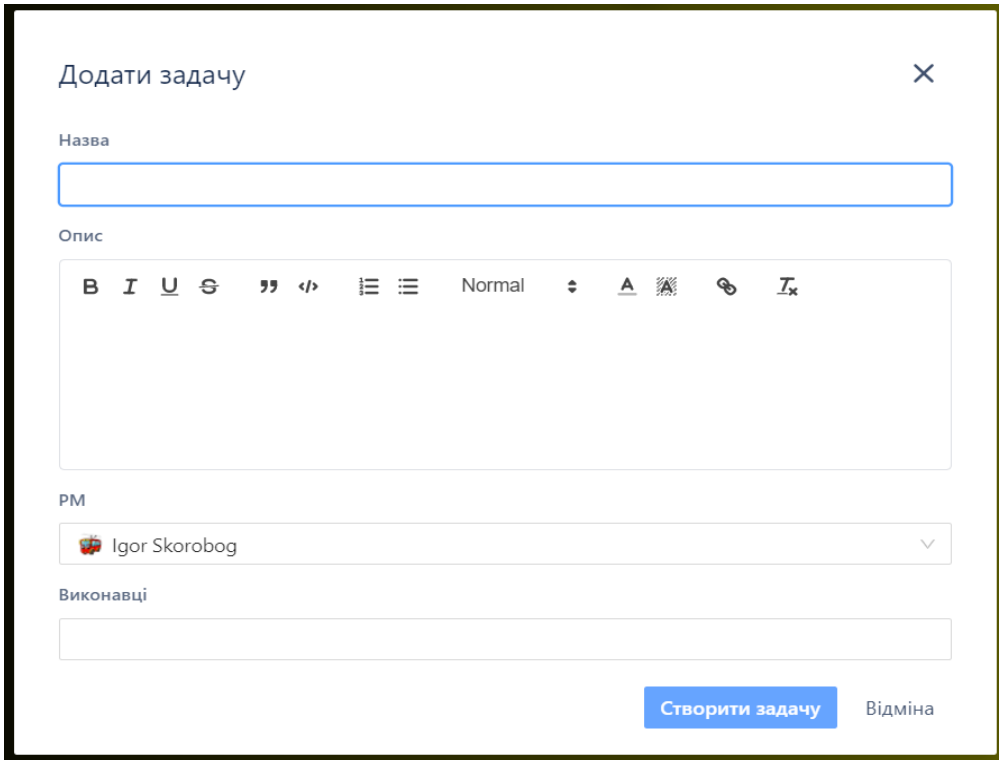


Рисунок 25. Форма додавання задачі

PM має змогу створювати проекти та задачі, редагувати та видаляти їх, міняти статус задач на будь-який.

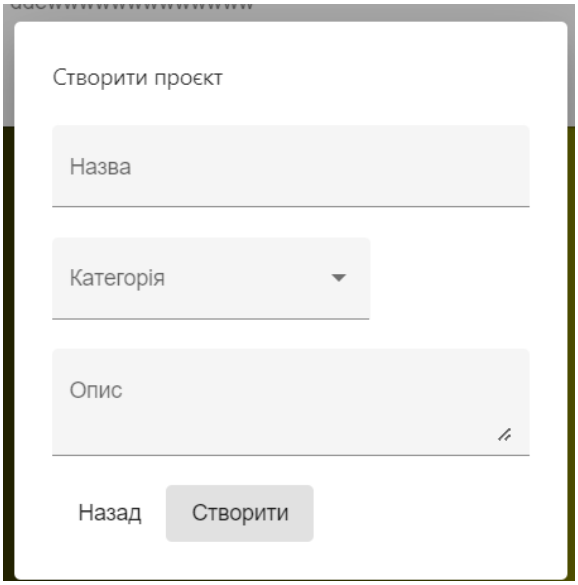


Рисунок 26. Форма створення проекту

PM, аналогічно до звичайного користувача, може приєднати акаунт до проекту, переглядати власні проекти. Крім цього він має створювати та видаляти нові проекти, створювати та видаляти нову задачу в межах проекту.

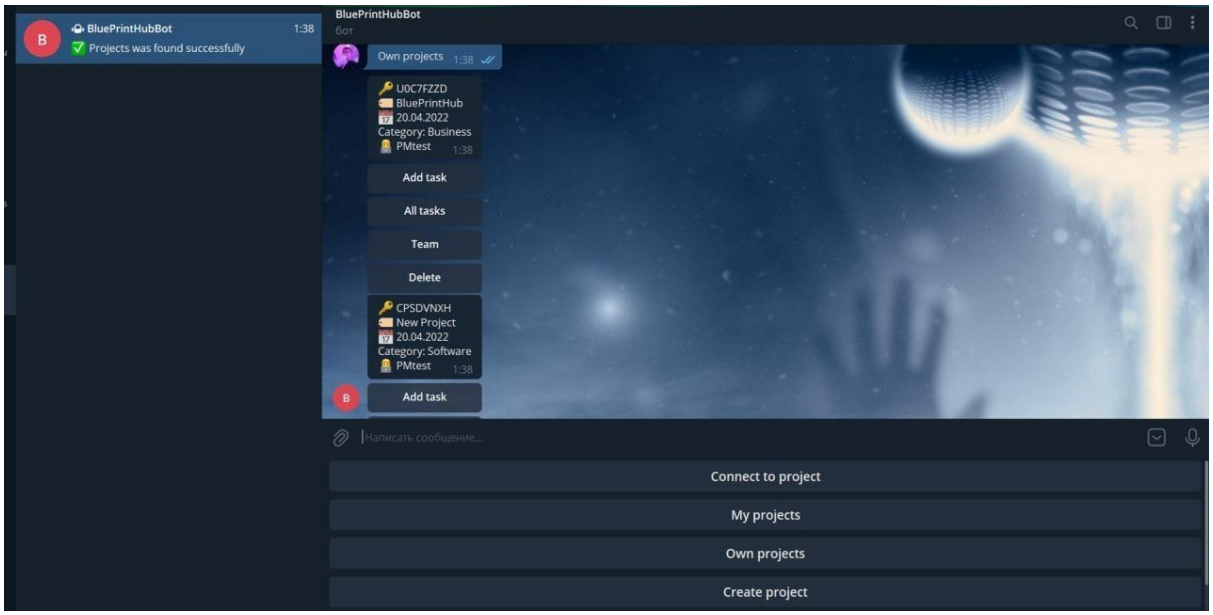


Рисунок 27. Інтерфейс telegram-боту для проджект-менеджера

Admin має змогу переглядати всіх користувачів, міняти їх ролі.

ВИСНОВКИ

Кожен з учасників команди у процесі роботи над розробкою вебсистеми навчився чомусь новому та покращив свої знання у певній сфері.

В результаті роботи над груповим проектом було окремо розроблено backend та frontend частину, що в майбутньому дозволить нам реалізувати повноцінний програмний продукт "BlueprintHub", що буде відповідати усім технічним вимогам. Більшість з учасників команди раніше не приймала участі у групових проектах, тому цей досвід був дуже важливим для кожного з нас. Студенти підвищили не лише soft-скіли, а й покращили свої знання у сфері інформаційних технологій та оформленні необхідної документації до проекту.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/uk
2. Jira [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/ru/software/jira
3. Java documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.oracle.com/en/java/
4. Spring Framework documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.spring.io/springframework/docs/current/reference/html/
5. Typescript documentation [Електрониий ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/docs/
6. Angular docs [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/docs
7. Firebase Storage [Електронний ресурс] – Режим доступу до ресурсу:

ДОДАТОК А Use Case діаграма вебсистеми

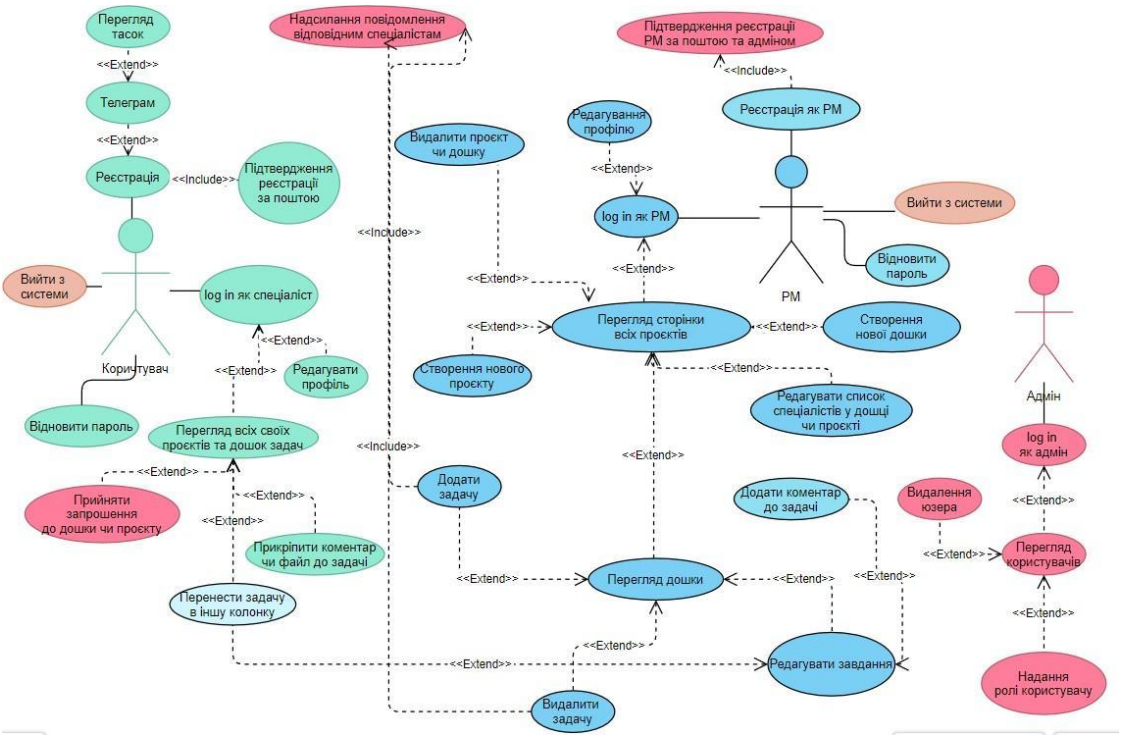


Рисунок А1. Use Case діаграма

ДОДАТОК Б Діаграма бази даних

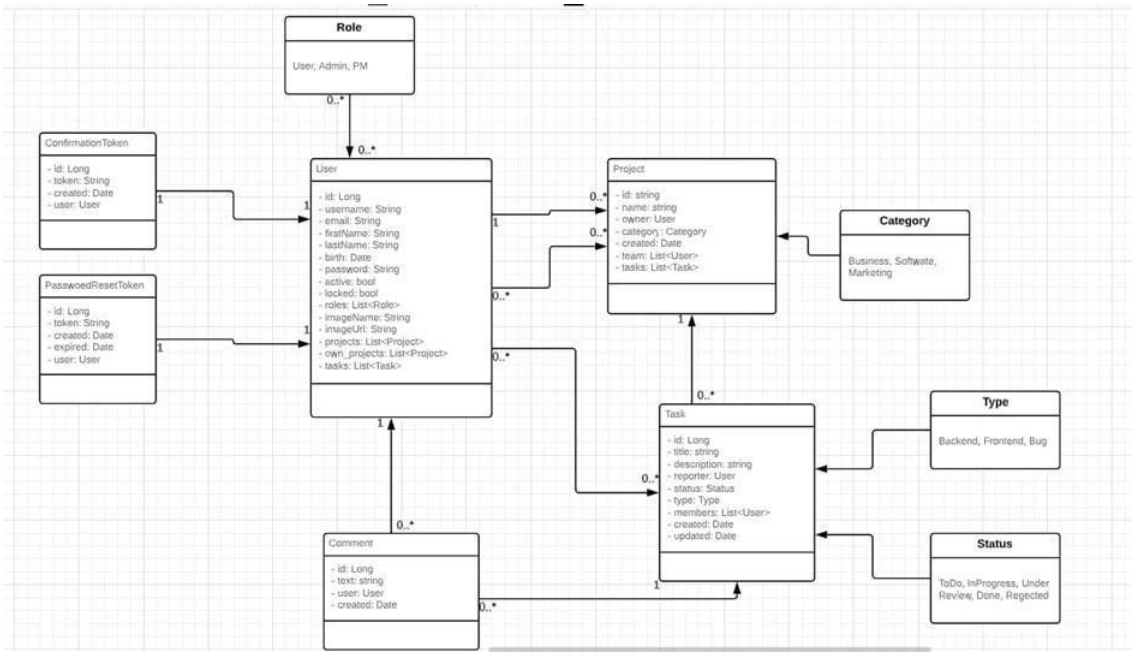


Рисунок Б1. Діаграма бази даних