Міністерство освіти і науки України
Харківський національний університет імені В. Н. Каразіна
Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки

«Затверджую»
Зав. кафедри теоретичної та
прикладної системотехніки
_____ д.т.н., проф. С. І. Шматков
«___» _____ 2024 р

Пояснювальна записка

до кваліфікаційної роботи
бакалавра

на тему: «**WEB-ДОДАТОК AGILE УПРАВЛІННЯ ІТ-ПРОЄКТАМИ**»

Захищено на засіданні
Атестаційної комісії № 42
протокол № 4101-5/909 від
03.05.2024 р.
Оцінка _____ / _____
Голова Атестаційної комісії
_____ **СКОБ Ю. О.**

Виконав:
студент 4 курсу, групи КІ–41
Галузь знань: 12 – Інформаційні технології
Спеціальність: 123 – Комп'ютерна інженерія.
ЩОМА Дмитро Русланович

Керівник: к.т.н., доцент ЗВО кафедри теоретичної та прикладної системотехніки
БУЛАВІН Дмитро Олексійович

Рецензент: д.т.н., доцент ЗВО; професор ЗВО кафедри ТПІ
РУККАС Кирило Маркович

Харків – 2024

АНОТАЦІЯ

Пояснювальна записка до кваліфікаційної роботи бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і чотирьох додатків. Загальний обсяг роботи складає сімдесят чотири сторінки, із яких п'ятдесят сторінок основної частини з двадцять вісім рисунками, сімнадцять таблиць, вісім найменувань списку використаних джерел та чотирьма додатками.

Ця робота присвячена розробці веб-додатку для Agile управління IT-проєктами. Додаток реалізовано з використанням Python, Django та PostgreSQL і включає наступні функції: авторизація користувачів, управління профілем, створення та управління проєктами, робота з задачами, а також адаптація інтерфейсу за допомогою Bootstrap. Додаток забезпечує гнучкість і зручність у використанні для команд, які працюють за методологією Agile.

Ключові слова: Python, PostgreSQL, Django, DjangoORM, Bootsrtap, HTML, CSS, JavaSrcipt, реалізація проєкту.

ABSTRACT

The explanatory note to the bachelor's thesis consists of an introduction, three chapters, conclusions, a list of references and four appendices. The total volume of the work is seventy-four pages, including fifty pages of the main part with twenty-eight figures, seventeen tables, eight references, and four appendices.

This work is devoted to the development of a web application for Agile IT project management. The application is implemented using Python, Django, and PostgreSQL and includes the following functions: user authorization, profile management, project creation and management, task management, and interface adaptation using Bootstrap. The application provides flexibility and ease of use for teams working according to the Agile methodology.

Keywords: Python, PostgreSQL, Django, DjangoORM, Bootstrap, HTML, CSS, JavaSript, project implementation.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних;

HTML – HyperText Markup Language;

CSS – Cascading Style Sheets;

ВСТУП

У зв'язку з лавиноподібним поширенням електронних систем керування складними системами, також електронних систем діагностування (як людей, живих істот, так і об'єктів неживої природи) все більш актуальною постає проблема раціонального складу вимірювальної компоненти подібних систем та раціональності алгоритму їх роботи.

Актуальність роботи. У сучасному світі інформаційні технології розвиваються надзвичайно швидкими темпами, що вимагає від компаній гнучкості та оперативності у впровадженні змін. Методології управління проєктами, зокрема Agile, дозволяють досягати високої адаптивності та ефективності в умовах постійно змінного середовища [1]. Проте, для успішного застосування Agile підходів необхідні спеціалізовані інструменти, які забезпечують зручне та ефективне управління проєктами. Розробка веб-додатку для Agile управління IT-проєктами відповідає потребам сучасних компаній у підвищенні продуктивності та якості роботи команд, що робить цю тему надзвичайно актуальною.

Метою дослідження підвищення ефективності керування проєктами, завдяки розробленому веб-додатку для Agile управління IT-проєктами з використанням технологій Python, Django та PostgreSQL.

Об’єкт дослідження – це процеси, управління IT-проєктами, які реалізуються за методологією Agile.

Методи дослідження: Для досягнення поставленої мети в дослідженні використовувалися наступні методи:

- Аналіз літературних джерел з метою визначення основних вимог до систем управління проєктами за методологією Agile.
- Проєктування програмного забезпечення, включаючи моделювання бази даних та розробку архітектури веб-додатку.

- Розробка веб-додатку за допомогою технологій Python, Django та PostgreSQL.
- Тестування програмного забезпечення для забезпечення його коректного функціонування та відповідності вимогам.

Предмет дослідження – веб-додаток для управління IT-проєктами, який реалізує функціонал, необхідний для підтримки методології Agile.

Завдання дослідження

1. Виконати аналіз існуючих інструментів для Agile управління проєктами та визначити їх основні переваги та недоліки.
2. Розробити архітектуру веб-додатку для Agile управління IT-проєктами, враховуючи вимоги до функціональності та зручності використання.
3. Реалізувати функції авторизації, управління профілем користувача, створення та управління проєктами, а також управління задачами з використанням технологій Python, Django та PostgreSQL.
4. Забезпечити адаптивність інтерфейсу веб-додатку та можливість вибору темної теми за допомогою Bootstrap.
5. Провести тестування розробленого додатку для перевірки його коректного функціонування та відповідності поставленим вимогам.

РОЗДІЛ 1

АНАЛІЗ AGILE МЕТОДОЛОГІЇ УПРАВЛІННЯ ІТ-ПРОЄКТАМИ

1.1 Вступ до методології Agile

Методологія Agile [1] є популярною серед ІТ-компаній завдяки своїй гнучкості та швидкості реагування на зміни. Agile управління проєктами дозволяє командам адаптуватися до нових вимог і забезпечує безперервне покращення продукту протягом усього циклу розробки.

1.2 Основні характеристики Agile

Agile методологія відрізняється такими характеристиками:

- **Ітеративний підхід**: Робота над проєктом розбивається на невеликі, керовані етапи, що дозволяє командам швидко адаптуватися до змін.
- **Інкрементальна розробка**: Продукт створюється поступово, з постійним додаванням нових функцій.
- **Постійний зворотний зв'язок**: Регулярне отримання зворотного зв’язку від клієнтів та користувачів дозволяє вчасно вносити необхідні корективи.
- **Командна робота**: Важливість співпраці між усіма членами команди та зацікавленими сторонами для досягнення спільної мети.
- **Гнучкість**: Здатність швидко адаптуватися до змін вимог або умов ринку.

1.3 Порівняння Agile з іншими методологіями

Для кращого розуміння переваг Agile, важливо порівняти її з іншими методологіями управління проєктами, такими як Waterfall та Kanban.

- **Waterfall [2]**:

    - **Підхід**: Послідовний підхід, де кожен етап розробки завершується перед початком наступного.
    - **Гнучкість**: Низька, важко вносити зміни після завершення етапів.
    - **Зворотний зв'язок**: Обмежений, отримується переважно на етапі тестування.
    - **Ризики**: Високі, оскільки можливі помилки виявляються на пізніх етапах проєкту.
- **Kanban [3]**:
    - **Підхід**: Методологія візуального управління роботою, орієнтована на постійний потік завдань.
    - **Гнучкість**: Висока, дозволяє швидко реагувати на зміни.
    - **Зворотний зв'язок**: Постійний, забезпечує оперативне вирішення проблем.
    - **Ризики**: Знижені, завдяки постійному моніторингу прогресу та швидкому реагуванню на зміни.
- **Agile**:
    - **Підхід**: Ітеративний та інкрементальний підхід з акцентом на гнучкість та постійне вдосконалення.
    - **Гнучкість**: Дуже висока, що дозволяє командам швидко адаптуватися до змін.
    - **Зворотний зв'язок**: Постійний, на кожній ітерації отримується зворотний зв’язок від клієнтів та користувачів.
    - **Ризики**: Знижені, завдяки частим ітераціям та регулярним рев’ю.

1.4 Переваги використання Agile в IT-проєктах

Agile підхід має численні переваги для IT-проєктів:

- **Покращена якість продукту**: Регулярне тестування та зворотний зв'язок дозволяють виявляти та виправляти помилки на ранніх етапах.

- **Підвищена продуктивність команди**: Завдяки чітким цілям на кожен спринт та постійній мотивації команди.
- **Вища задоволеність клієнтів**: Постійне залучення клієнтів до процесу розробки забезпечує створення продукту, який відповідає їхнім потребам.
- **Зменшення ризиків**: Часті релізи та постійний зворотний зв’язок дозволяють швидко виявляти та вирішувати проблеми.
- **Краща адаптивність**: Можливість швидко реагувати на зміни ринкових умов або вимог клієнтів.

1.5 Використання веб-додатків при AGILE управлінні IT-проєктами

Веб-додатки, призначені для управління IT-проєктами на основі методології Agile, забезпечують інструменти та функції для ефективної організації роботи команди. Такі додатки включають в себе можливості для планування спринтів, управління задачами, відстеження прогресу, комунікації та співпраці між членами команди. Вони також надають можливості для інтеграції з іншими інструментами, що використовуються в розробці програмного забезпечення, такими як системи контролю версій та платформи для безперервної інтеграції та доставки.

1.6 Переваги та недоліки веб-додатку при AGILE управлінні IT-проєктами

1.6.1 Переваги

1. **Функціональність для Agile управління проєктами**:
    - **Розширені можливості авторизації**: Реєстрація, логін, зміна паролю забезпечують безпечний доступ до додатку.
    - **Управління профілями**: Користувачі можуть переглядати та змінювати інформацію свого профілю, включаючи аватар.

- **Управління проєктами**: Створення, видалення та зміна проєктів, додавання користувачів до проєктів, пошук проєктів за назвою, чат проєкту та пагінація.
- **Управління задачами**: Створення, видалення та зміна задач, коментарі до задач, список задач з проєкту, пошук задач за назвою та пагінація.

2. **Покращена взаємодія та співпраця**:
    - **Чат проєкту**: Можливість комунікації всередині проєкту сприяє покращенню співпраці та швидкому обміну інформацією.
    - **Коментарі до задач**: Зручний спосіб залишати коментарі до задач забезпечує ефективну комунікацію між учасниками проєкту.
3. **Зручність використання**:
    - **Інтуїтивний інтерфейс**: Додаток має зрозумілий та зручний інтерфейс, що полегшує користування ним.
    - **Пагінація**: Зручне переглядання списків проєктів та задач завдяки пагінації, що підвищує продуктивність роботи з великим обсягом даних.

1.6.2 Недоліки

1. **Обмежена масштабованість у порівнянні з великими корпоративними рішеннями**:
    - Великі корпоративні рішення, такі як Jira або Microsoft Project, можуть пропонувати більш розширені можливості масштабування та інтеграції з іншими корпоративними інструментами. Веб-додаток може мати обмеження у масштабованості в порівнянні з цими рішеннями.
2. **Відсутність деяких розширених функцій**:
    - Веб-додаток може не мати деяких розширених функцій, таких як розширена аналітика, звіти, підтримка різних методологій

управління проєктами (наприклад, Kanban, Waterfall), які є в великих комерційних рішеннях.

3. **Потреба у додаткових ресурсах для налаштування та підтримки**:
    - Веб-додаток може вимагати додаткових ресурсів для налаштування, підтримки та масштабування в порівнянні з хмарними рішеннями, які надаються великими провайдерами з інтегрованою підтримкою та оновленнями.
4. **Обмежені можливості інтеграції з деякими сторонніми сервісами**:
    - Веб-додаток може мати обмежені можливості інтеграції з деякими специфічними сторонніми сервісами або платформами, що можуть бути доступні в комерційних продуктах, таких як Atlassian Suite або Microsoft Azure DevOps.

1.6.1 Порівняння з іншими рішеннями

Порівняння з Jira [4]

Jira: Платформа для управління проєктами та задачами, яка підтримує різні методології (Agile, Kanban, Waterfall). Містить розширені можливості для звітів, аналітики та інтеграції з іншими сервісами.

- **Переваги Jira**: Розширені функції, інтеграція з іншими продуктами Atlassian, велика спільнота користувачів та підтримка.
- **Недоліки Jira**: Висока вартість для великих команд, складність налаштування та використання для новачків.

Порівняння з Trello [5]

Trello: Інструмент для управління задачами на основі дошок та карток. Легкий у використанні, підходить для невеликих команд та особистого використання.

- **Переваги Trello**: Простота у використанні, візуальний інтерфейс, безкоштовна версія для невеликих команд.

- **Недоліки Trello**: Обмежені можливості для великих проєктів та команд, відсутність розширених функцій для аналітики та звітів.

Порівняння з Microsoft Project [6]

Microsoft Project: Потужний інструмент для управління проєктами, який підтримує різні методології та надає розширені можливості для планування, моніторингу та звітності.

- **Переваги Microsoft Project**: Розширені можливості для управління проєктами, інтеграція з іншими продуктами Microsoft, підтримка великих корпоративних проєктів.
- **Недоліки Microsoft Project**: Висока вартість, складність у використанні та налаштуванні, потреба у спеціальному навчанні для ефективного використання.

1.7 Постановка задачі дослідження

Задачею даного дослідження є розробка та впровадження веб-додатку для Agile управління IT-проєктами, який дозволяє ефективно керувати проєктами, задачами та користувачами, забезпечуючи при цьому високий рівень безпеки та зручності користування.

Висновки за розділом 1

У цьому розділі було розглянуто основні аспекти методології Agile, функціональні можливості розробленого веб-додатку та використані технології. Наступні розділи детально аналізуватимуть кожен з компонентів додатку, методи їх реалізації та впровадження, а також оцінюватимуть ефективність та практичність використання даного рішення.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ AGILE УПРАВЛІННЯ IT-ПРОЄКТАМИ

2.1 Дослідження Предметної Області

2.1.1 Основні функціональні можливості веб-додатку

Для забезпечення ефективного управління IT-проєктами був розроблений веб-додаток, який включає такі функціональні можливості:

Аутентифікація користувачів є ключовим елементом безпеки та контролю доступу до функцій додатку. Вона включає такі підфункції:

- **Реєстрація:** Користувачі можуть створювати нові облікові записи, що дозволяє їм отримати доступ до всіх функцій додатку.
- **Авторизація:** Після реєстрації користувачі можуть увійти до системи, використовуючи свої облікові дані.
- **Зміна паролю:** Користувачі можуть змінювати свої паролі для забезпечення безпеки своїх облікових записів.

Функції профілю дозволяють користувачам переглядати та оновлювати свою особисту інформацію:

- **Перегляд профілю:** Користувачі можуть переглядати інформацію свого профілю, включаючи особисті дані та аватар.
- **Зміна інформації профілю:** Користувачі можуть оновлювати свої особисті дані для підтримки актуальності інформації.
- **Зміна аватару профілю:** Можливість завантажувати та змінювати аватар профілю для персоналізації облікового запису.

Управління проєктами включає створення, редагування та видалення проєктів, а також додавання користувачів до проєктів:

- **Створення, видалення, зміна проєкту:** Користувачі можуть створювати нові проєкти, змінювати їх параметри або видаляти, коли вони більше не потрібні.

- **Додавання користувачів до проєкту:** Можливість додавати інших користувачів до проєкту для спільної роботи.
- **Пошук проєкту за назвою:** Інструмент для швидкого пошуку потрібного проєкту за його назвою.
- **Чат проєкту:** Інтегрований чат для комунікації між учасниками проєкту.
- **Пагінація:** Функціональність для зручного перегляду довгих списків проєктів.

Управління задачами дозволяє створювати, редагувати та видаляти задачі, а також встановлювати дедлайни та додавати коментарі:

- **Створення, видалення, зміна задач:** Користувачі можуть керувати задачами, створюючи нові, змінюючи існуючі або видаляючи їх.
- **Встановлення дедлайну:** Можливість встановлювати кінцеві терміни для виконання задач.
- **Коментарі до задач:** Інструмент для додавання коментарів до задач, що сприяє кращій комунікації та координації.
- **Список задач з проєкту:** Перегляд усіх задач, пов'язаних з певним проєктом.
- **Пошук за назвою:** Інструмент для швидкого пошуку задач за їх назвою.
- **Пагінація:** Функціональність для зручного перегляду довгих списків задач.

2.1.2 Випадки Використання

На рис. 2.1. зображено діаграму варіантів використання, яка представляють собою сценарії взаємодії користувачів із системою для виконання певних завдань.

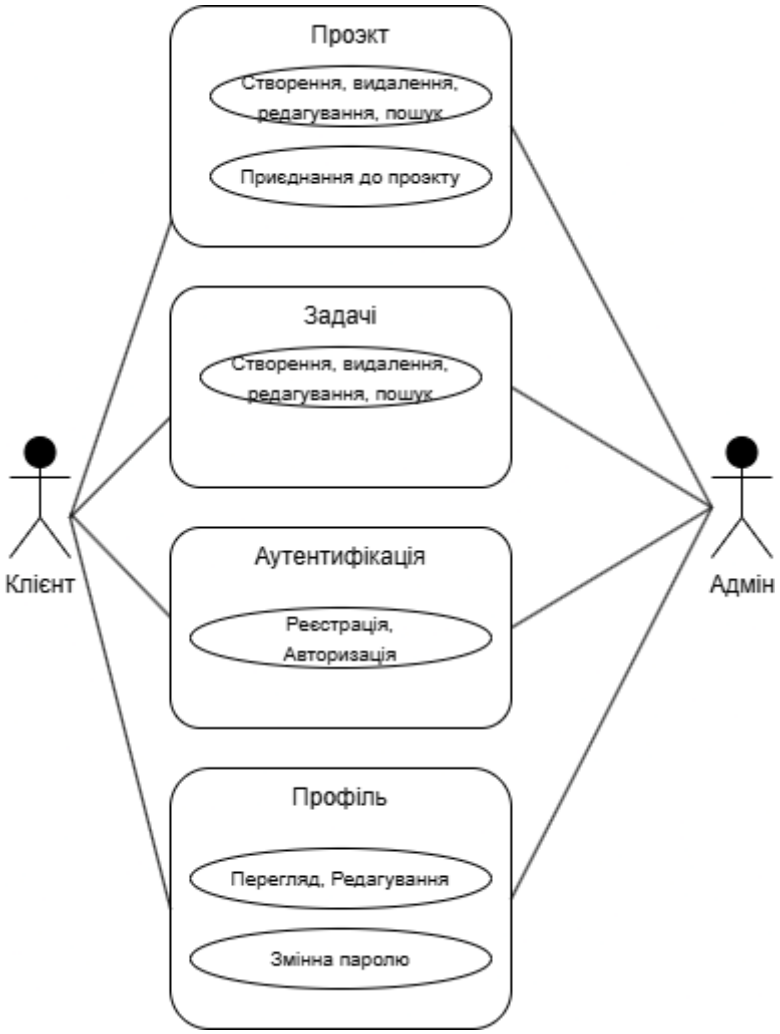

Рис. 2.1. Діаграма варіантів використання

2.1.3 Докладний Опис Випадків Використання

Серед випадків використання включено такі функції, як авторизація, перегляд профілю, взаємодія з проєктами, взаємодія з задачами. У таблицях 2.1 – 2.18 наведено ключові варіанти використання діаграми.

Аєтентифікація описує процеси реєстрації та авторизації.

Таблиця 2.1

Варіант використання «Реєстрація»

Назва	Реєстрація
Актори	Новий користувач
Передумови	Користувач має доступ до інтернету та сторінки реєстрації

Основний Потік	-Користувач відкриває сторінку реєстрації -Вводить необхідні дані (ім'я, електронна пошта, пароль, роль) -Підтверджує введення натисканням кнопки "Зареєструватися" -Система перевіряє введені дані та створює обліковий запис -Користувач потрапляє на сторінку входу
Результат	Обліковий запис створено

Таблиця 2.2

Варіант використання «Авторизація»

Назва	Авторизація
Актори	Зареєстрований користувач
Передумови	Користувач зареєстрований у системі
Основний Потік	-Користувач відкриває сторінку входу -Вводить електронну пошту та пароль -Натискає кнопку "Увійти" -Система перевіряє введені дані та авторизує користувача -Користувач перенаправляється на головну сторінку системи
Результат	Користувач успішно увійшов до системи

Профіль описує процеси перегляду, зміни інформації профілю та зміни паролю.

Таблиця 2.3

Варіант використання «Перегляд інформації профілю»

Назва	Перегляд інформації профілю
Актори	Зареєстрований користувач
Передумови	Користувач увійшов у систему
Основний Потік	-Користувач відкриває свій профіль -Система відображає інформацію профілю
Результат	Відображена актуальна інформація профілю

Таблиця 2.4

Варіант використання «Зміна інформації профілю»

Назва	Зміна інформації профілю
Актори	Зареєстрований користувач
Передумови	Користувач увійшов у систему
Основний Потік	-Користувач відкриває налаштування профілю -Вводить нову інформацію -Підтверджує зміни натисканням кнопки "Зберегти" -Система оновлює інформацію профілю. -Користувач переходить до особистого профілю
Результат	Інформація профілю успішно змінена

Таблиця 2.5

Варіант використання «Змінна паролю»

Назва	Зміна паролю користувача
Актори	Зареєстрований користувач
Передумови	Користувач увійшов у систему
Основний Потік	-Користувач вибирає опцію "Змінити пароль". -Вводить поточний пароль та новий пароль -Підтверджує зміни натисканням кнопки "Зберегти" -Система перевіряє поточний пароль і оновлює пароль користувача -Користувач отримує підтвердження про успішну зміну паролю.
Результат	Пароль успішно змінено

Проєкт описує процеси створення, видалення, зміни проєкту, додавання користувачів, пошуку проєкту, чату проєкту.

Таблиця 2.6

Варіант використання «Створення нового проєкту»

Назва	Створення нового проєкту
Актори	Зареєстрований користувач
Передумови	Користувач увійшов у систему
Основний Потік	-Користувач входить в систему -Вибирає опцію "Створити новий проєкт" -Вводить дані про проєкт (назва, опис) -Зберігає інформацію. -Система створює новий проєкт та зберігає його в БД
Результат	Новий проєкт успішно створено

Таблиця 2.7

Варіант використання «Видалення проєкту»

Назва	Видалення проєкту
Актори	Зареєстрований користувач
Передумови	Зареєстрований користувач має права доступу до проєкту
Основний Потік	-Користувач відкриває список проєктів -Вибирає проєкт для видалення -Підтверджує видалення -Система видаляє проєкт з системи -Користувач перенаправляється до списку проєктів
Результат	Проєкт успішно видалено

Таблиця 2.8

Варіант використання «Редагування проєкту»

Назва	Редагування проєкту
Актори	Зареєстрований користувач
Передумови	Зареєстрований користувач має права доступу до проєкту
Основний Потік	-Користувач відкриває список проєктів -Вибирає проєкт для редагування Вносить зміни (назва, опис) -Зберігає зміни. -Система оновлює дані проєкту в базі даних -Користувач перенаправляється до списку проєктів
Результат	Дані проєкту успішно оновлені

Таблиця 2.9

Варіант використання «Створення коду для запрошення до проєкту»

Назва	Створення коду для запрошення до проєкту
Актори	Зареєстрований користувач
Передумови	Зареєстрований користувач має права доступу до проєкту
Основний Потік	-Користувач відкриває запрошення до проєкту -Користувач отримує код
Результат	Код для запрошення проєкту успішно створенний

Таблиця 2.10

Варіант використання «Приєднання до проєкту»

Назва	Приєднання до проєкту
Актори	Зареєстрований користувач
Передумови	Користувач увійшов у систему
Основний Потік	-Користувач отримує код для приєднання до проєкту -Користувач відкриває приєданняння до проєкту -Користувач вводить код, котрий отримав -Надсилає код -Корисистувач приєднується до проєкту
Результат	Користувачі успішно додані до проєкту

Таблиця 2.11

Варіант використання «Пошук проєкта»

Назва	Пошук проєкта
Актори	Зареєстрований користувач
Передумови	Користувач увійшов у систему та має доступ до проєктів
Основний Потік	-Користувач вводить назву проєкту в поле пошуку -Натискає кнопку "Пошук" -Система виконує пошук проєктів за введеною назвою. -Відображає результати пошуку
Результат:	Відображені результати пошуку

Таблиця 2.12

Варіант використання «Чат проєкта»

Назва	Чат проєкта
Актори	Користувачі проєкту
Передумови	Користувачі додані до проєкту
Основний Потік	-Користувач відкриває проєкт. -Вибирає вкладку "Чат" -Вводить повідомлення -Натискає кнопку "Відправити" -Система відображає повідомлення в чаті проєкту
Результат	Повідомлення успішно відправлено та відображено в чаті

Задачі описують процеси створення, видалення, зміни задач, коментарів до задач, перегляду списку задач, пошуку задач за назвою.

Таблиця 2.13

Варіант використання «Створення нової задачі»

Назва	Створення нової задачі
Актори	Користувач проєкту
Передумови	Користувач доданий до проєкту
Основний Потік	-Користувач відкриває проєкт -Користувач відкриває задачі Вибирає опцію "Створити нову задачу" -Вводить дані про задачу (назва, опис, дедлайн, статус, складність) -Зберігає інформацію -Система створює нову задачу та зберігає її в базі даних
Результат	Нова задача успішно створена

Таблиця 2.14

Варіант використання «Видалення задачі»

Назва	Видалення задачі
Актори	Користувач проєкту
Передумови	Користувач доданий до проєкту та власник створенної задачі
Основний Потік	-Користувач відкриває проєкт -Користувач відкриває задачі -Обирає задачу для видалення -Натискає кнопку "Видалити" -Система видаляє задачу
Результат	Задачача успішно видалена

Таблиця 2.15

Варіант використання «Редагування задачі»

Назва	Редагування задачі
Актори	Користувач проєкту
Передумови	Користувач доданий до проєкту власник створенної задачі

Основний Потік	-Користувач відкриває проєкт -Користувач відкриває задачі -Вибирає задачу для редагування -Вносить зміни (назва, опис, дедлайн, статус, складність) -Зберігає зміни -Система оновлює дані задачі в базі даних
Результат	Дані задачі успішно оновлені

Таблиця 2.16

Варіант використання «Додавання коментарів до задачі»

Назва	Додавання коментарів до задачі
Актори	Користувач проєкту
Передумови	Користувач доданий до проєкту
Основний Потік	-Користувач відкриває задачу -Вводить коментар в спеціальне поле -Натискає кнопку "Додати коментар" -Система зберігає коментар в базі даних
Результат	Коментар успішно доданий до задачі

Таблиця 2.17

Варіант використання «Перегляд таблиці задач проєкту»

Назва	Перегляд таблиці задач проєкту
Актори	Користувач проєкту
Передумови	Користувач доданий до проєкту
Основний Потік	-Користувач відкриває проєкт -Вибирає вкладку "Задачі проєкту" -Система відображає список задач, пов'язаних з проєктом
Результат	Відображений список задач проєкту

Таблиця 2.18

Варіант використання «Пошук задач за назвою»

Назва	Пошук задач
Актори	Користувач проєкту
Передумови	Користувач доданий до проєкту

Основний Потік	-Користувач відкриває проєкт -Вибирає вкладку "Задачі проєкту" -Користувач вводить назву задачі в поле пошуку -Натискає кнопку "Пошук" -Система виконує пошук задач за введеною назвою -Відображає результати пошуку
Результат	Відображені результати пошуку задач

2.2 Проєктування системи

2.2.1 Логічна постановка задачі

Цей розділ описує логічну структуру проєкту та завдань, які потрібно вирішити для створення системи управління IT-проєктами на основі Agile.

Основні задачі проєкту:

1. **Розробка користувацького інтерфейсу**
   - Створення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів системи.
   - Вимоги до інтерфейсу: простота використання, швидкий доступ до основних функцій, адаптивність до різних пристроїв.
2. **Реалізація функціоналу авторизації та профілів**
   - Реєстрація нових користувачів, логін, зміна паролю.
   - Можливість перегляду та редагування інформації профілю, завантаження аватару.
3. **Управління проєктами та задачами**
   - Створення, видалення та редагування проєктів та задач.
   - Додавання користувачів до проєктів, можливість коментування задач.
4. **Реалізація пошуку та пагінації**
   - Пошук проєктів та задач за назвою.
   - Пагінація для зручного перегляду великих списків проєктів та задач.

2.2.2 Проєктування бази даних

Проєктування бази даних включає створення концептуальної та логічної моделей даних.

На концептуальному рівні моделювання визначаються основні сутності та їх зв'язки. Основними сутностями для нашої системи є:

- **Користувач (Worker)**

  Атрибути: ID, ім'я, прізвище, електронна пошта, пароль, аватар, позиція.

- **Проєкт (Project)**

  Атрибути: ID, назва, опис, творець, учасники, запрошувальний код.

- **Чат (ChatMessage)**

  Атрибути: ID, проєкт, поівідомлення, відправник, дата.

- **Задача (Task)**

  Атрибути: ID, назва, опис, дедлайн, пріорітет, статус, змінений, проєкт, творець.

- **Коментар (TaskComment)**

  Атрибути: ID, поівідомлення, відправник, задача, дата.

- **Тип Задачи (TaskType)**

  Атрибути: ID, назва.

- **Позиція (Position)**

  Атрибути: ID, назва

Проєктування логічної моделі даних деталізує концептуальну модель, додаючи ключі та інші обмеження. Логічна модель даних продемонстрована на рис. 2.2

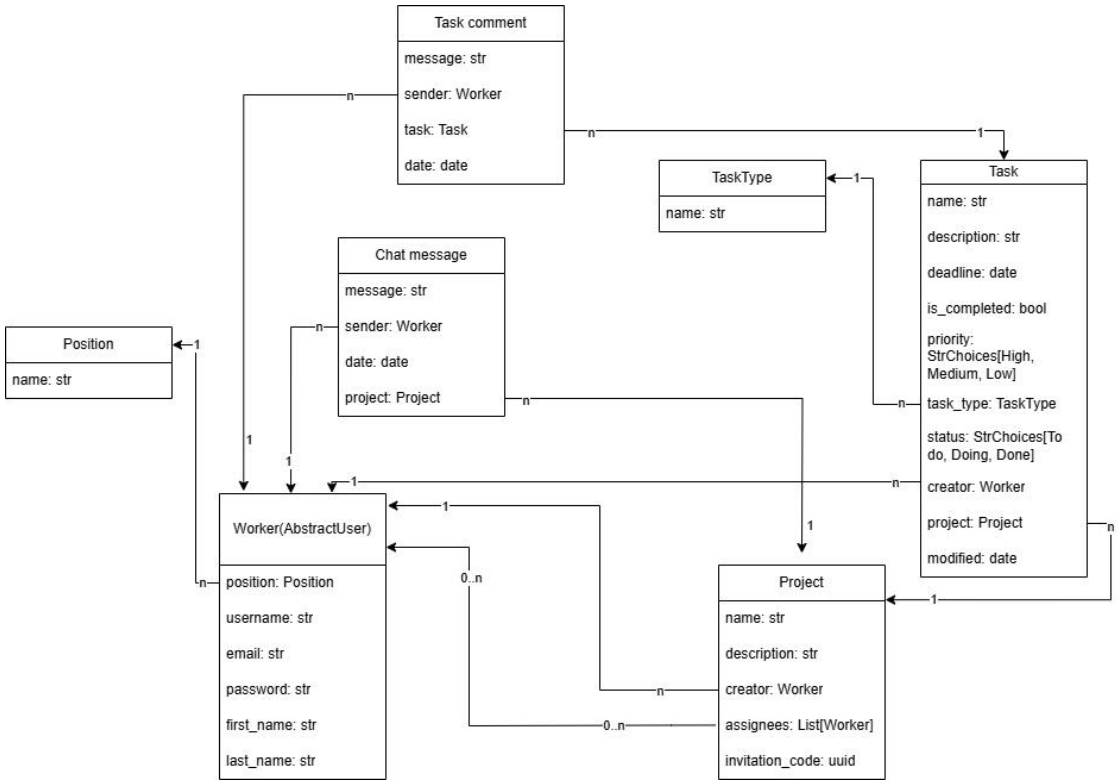

Рис. 2.2 – Діаграма логічної моделі даних

Взаємодія між класами:

- Worker створює Project і може бути доданий до інших проєктів.
- Worker створює Task і коментує завдання через Task comment.
- Worker надсилає повідомлення в чаті через Chat message.
- Project включає завдання (Task) і повідомлення в чаті (Chat message).
- Task включає тип завдання (TaskType), статус, пріоритет і коментарі (Task comment).

Ця модель дозволяє відстежувати завдання, коментарі до них, повідомлення в чаті для проєктів та управління працівниками в контексті їхніх ролей та позицій.

2.3 Архітектура системи

На рис. 2.3 зображено діаграму архітектури системи, яка використовує Модель-Вид-Шаблон (MVT) підхід. Ця діаграма відображає взаємодію компонентів: Клієнт, Відображення (view), Модель (model), Шаблон (template) та бази даних PostgreSQL. Діаграма ілюструє процес обробки запитів від клієнтів та генерації відповідей у вигляді HTML-сторінок.

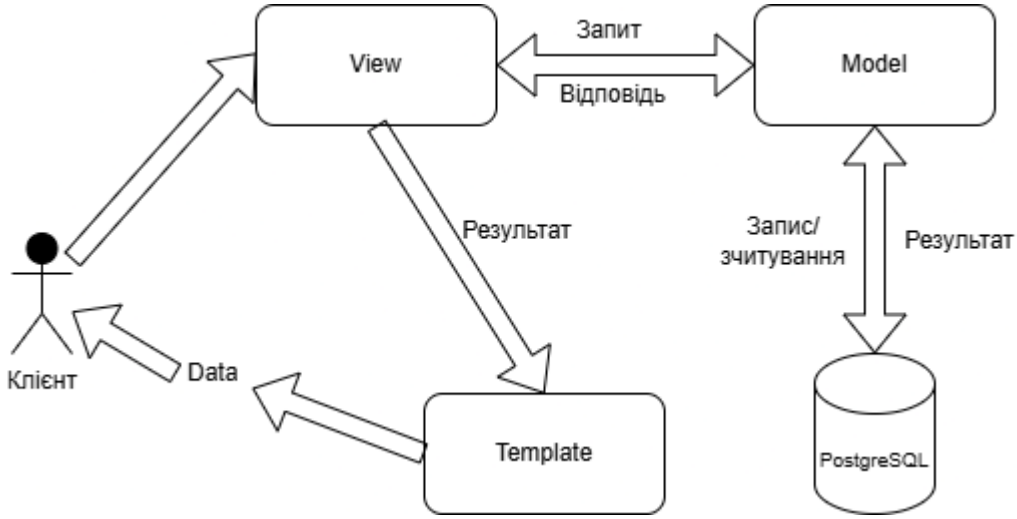

Рис. 2.2 – Діаграма архітектури системи

Клієнт: Це користувач або система, що взаємодіє з веб-додатком, відправляючи запити і отримуючи відповіді.

View (Відображення): Компонент, який отримує запити від клієнта, обробляє їх і звертається до моделі для отримання або збереження даних. Після обробки запиту, View передає дані до шаблону для генерації HTML-відповіді.

Model (Модель): Компонент, який взаємодіє з базою даних. Він виконує запити до бази даних і повертає результати до View. Відповідає за бізнес-логіку додатку і доступ до даних.

Template (Шаблон): Компонент, який відповідає за рендеринг HTML-сторінок, використовуючи дані, отримані з View. Шаблон формує кінцевий вигляд сторінки, яку бачить клієнт.

Data: Template відправляє згенерований HTML-код з даними назад до клієнта. Клієнт отримує цю HTML-сторінку як відповідь і може переглянути дані в браузері.

PostgreSQL: Це база даних, в якій зберігаються всі дані додатку. Модель взаємодіє з PostgreSQL для зчитування і запису даних.

Процес взаємодії: Клієнт відправляє запит до View. View обробляє запит і робить запит до моделі. Модель взаємодіє з базою даних (PostgreSQL), щоб отримати необхідні дані або зберегти їх. Модель повертає результати до View. View передає результати до шаблону, який рендерить HTML. Клієнт отримує готову HTML-сторінку як відповідь.

Ця архітектура демонструє взаємодію компонентів у веб-додатку, де View обробляє запити від клієнтів, Model забезпечує доступ до даних у базі даних, а Template відповідає за відображення сторінок.

2.4 Аналіз критеріїв хостингу та його використання

Вибір хостингу для веб-додатка є важливим кроком у розробці та впровадженні системи управління IT-проєктами на основі Agile. Правильний вибір хостингу може забезпечити стабільну роботу додатка, високу продуктивність, безпеку даних та можливість масштабування. Ось основні критерії, за якими можна обирати хостинг:

1. Продуктивність
    1) Швидкість завантаження сторінок: Висока швидкість завантаження сторінок позитивно впливає на користувацький досвід.
    2) Ресурси: Достатня кількість виділених ресурсів (процесор, оперативна пам’ять, дисковий простір) для обробки запитів і забезпечення швидкої роботи додатка.
2. Масштабованість

1) Горизонтальне та вертикальне масштабування: Можливість легкого збільшення ресурсів при зростанні навантаження на додаток.
2) Автоматичне масштабування: Деякі хостингові провайдери пропонують автоматичне масштабування ресурсів у відповідь на зміну навантаження.

3. Сумісність і інтеграції
    1) Підтримка потрібних технологій: Хостинг повинен підтримувати технології, які використовуються у проєкті, Postgresql для бази даних, фреймворки для розробки бекенду - Django та DjangoORM і фронтенду - Bootstrap).

Висновки за розділом 2

У цьому розділі було детально розглянуто процес аналізу та проєктування системи для управління IT-проєктами на основі методології Agile.

На основі проведеного аналізу та проєктування було сформовано чітке розуміння архітектури системи та її основних компонентів, що дозволяє перейти до реалізації та тестування розробленого веб-додатка. Описані моделі даних, варіанти використання та вимоги до системи створюють основу для подальшого розвитку та вдосконалення проєкту, забезпечуючи його відповідність сучасним стандартам та вимогам користувачів.

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ ДЛЯ AGILE УПРАВЛІННЯ IT-ПРОЄКТАМИ

3.1 Засоби розробки

Мова програмування: Python

Python є вибраним мовою програмування для створення веб-додатку. Вона відома своєю простотою та читабельністю коду, що полегшує розробку та підтримку проєктів.

Фреймворк: Django

Django - це високорівневий веб-фреймворк на Python, який дозволяє швидко та ефективно розробляти веб-додатки. Він має вбудовану адміністративну панель, що спрощує роботу з базою даних та аутентифікацією користувачів.

База даних: PostgreSQL

PostgreSQL - це потужна об'єктно-реляційна система керування базами даних (СКБД), яка використовується для зберігання та організації даних у веб-додатку. Вона відома своєю надійністю, масштабованістю та дотриманням стандартів SQL.

Фронтенд: HTML, CSS, JavaScript, Bootstrap

Для створення користувацького інтерфейсу (фронтенду) використовуються стандартні технології веб-розробки: HTML для структури сторінок, CSS для стилізації та оформлення, JavaScript для динамічного змісту та взаємодії, а також фреймворк Bootstrap для швидкого та адаптивного розроблення.

Середовище розробки: PyCharm

PyCharm - це інтегроване середовище розробки (IDE) для Python, яке надає розширені можливості для роботи з кодом, такі як автоматичне доповнення коду, відладка, контроль версій та інші корисні інструменти.

Система контролю версій: Git, GitHub для зберігання коду

Git - це розподілена система керування версіями, яка використовується для відстеження змін у коді проєкту. GitHub - це хмарна платформа для спільної роботи з проєктами, де можна зберігати, спільно використовувати та вести розробку програмного забезпечення з використанням системи контролю версій Git.

3.2 Вимоги до технічного та програмного забезпечення

Серверне ПЗ:

- Python 3.10 або вище
- Django 3.0 або вище
- PostgreSQL 12 або вище

Операційні системи:

- Linux (Ubuntu 20.04), Windows Server або MacOS

Клієнтське ПЗ:

- Браузер з підтримкою JavaScript та CSS

Ці вимоги забезпечують сумісність та оптимальну продуктивність веб-додатку на різних рівнях: від серверного програмного забезпечення до операційних систем, на яких він запускається, та клієнтського програмного забезпечення, через яке користувачі взаємодіють з додатком через браузер.

3.3 Опис програмної реалізації

Структура проєкту складається з таких директорій:

media/
staticfiles/
task_manager/
tasks/
templates/
manage.py

Опис директорій та файлів:

1. **media/**
    - Директорія для зберігання медіафайлів, які завантажуються користувачами (наприклад, зображення, відео, документи).
    - Файли в цій директорії не є частиною статичного контенту і можуть змінюватися в процесі роботи додатку.
2. **staticfiles/**
    - Директорія для зберігання статичних файлів, таких як CSS, JavaScript, зображення та інші ресурси, які не змінюються динамічно.
    - Ці файли використовуються для оформлення та функціонування фронтенд частини веб-додатку.
3. **task_manager/**
    - Основна директорія проєкту Django, яка містить конфігураційні файли та налаштування додатку.
    - **__init__.py**: Файл, що вказує Python, що це директорія є пакетом.
    - **settings.py**: Конфігураційний файл Django, де налаштовуються бази даних, додатки, середовище виконання та інші глобальні налаштування.
    - **urls.py**: Файл, що містить маршрутизацію URL для проєкту.
    - **wsgi.py**: Файл, що використовується для розгортання проєкту на WSGI-сервері.
    - **asgi.py**: Файл для розгортання проєкту на ASGI-сервері (для підтримки асинхронних запитів).
4. **tasks/**
    - Додаток Django, що реалізує основний функціонал керування завданнями.
    - **__init__.py**: Файл, що вказує Python, що це директорія є пакетом.
    - **tests/**

        -Директорія для тестування додатку.

-Файли в tests/:

-test_commands.py: Файл, що містить тести для команд Django.

-test_forms.py: Файл, що містить тести для форм.

-test_models.py: Файл, що містить тести для моделей.

-test_views.py: Файл, що містить тести для представлень (views).

- **admin.py**: Файл для реєстрації моделей в адміністративній панелі Django.
- **apps.py**: Конфігураційний файл додатку.
- **models.py**: Файл, що містить моделі даних, які визначають структуру таблиць у базі даних.
- **views.py**: Файл, що містить функції та класи, які обробляють запити та повертають відповіді.
- **urls.py**: Файл, що містить маршрутизацію URL для додатку **tasks**.
- **forms.py**: Файл для визначення форм, які використовуються в додатку.
- **migrations/**: Директорія, що містить файли міграцій для бази даних.

5. **templates/**
    - Директорія, що містить шаблони HTML для рендерингу веб-сторінок.
    - Тут зберігаються всі HTML-файли, які використовуються в додатку для відображення інформації користувачам.
6. **manage.py**
    - Сценарій командного рядка, що дозволяє взаємодіяти з проєктом Django.
    - Використовується для виконання різних адміністративних завдань, таких як запуск сервера, застосування міграцій бази даних, створення суперкористувача та інше.

Ця структура забезпечує організоване зберігання коду та ресурсів, що полегшує розробку, підтримку та масштабування веб-додатку.

3.4 Проєктування інтерфейсу

Проєктування інтерфейсу користувача (UI) є важливим етапом у розробці веб-додатку. Мета цього етапу - створити зручний, інтуїтивно зрозумілий та естетично привабливий інтерфейс, який забезпечить ефективну взаємодію користувача з додатком.

Основні принципи проєктування інтерфейсу:

1. **Простота та зрозумілість**
    - Інтерфейс повинен бути максимально простим та зрозумілим для користувачів. Важливо уникати перевантаження сторінок елементами та забезпечити логічну структуру розташування компонентів.
2. **Консистентність**
    - Всі елементи інтерфейсу повинні бути стилістично і функціонально узгодженими між собою. Це забезпечує користувачам передбачуваність дій та знижує криву навчання.
3. **Адаптивний дизайн**
    - Інтерфейс має коректно відображатися на різних пристроях та екранах. Для цього використовується адаптивний дизайн, що забезпечує зручність користування як на десктопах, так і на мобільних пристроях.

Кожна сторінка має індивідуальний дизайн, але має спільні компоненти

Спільні компоненти інтерфейсу:

1. **Головна сторінка**
    - Містить оглядову інформацію про кількість проєктів та задач користувача.

2. **Навігаційне меню**
    - Розташовується на бічній панелі сайту.
    - Включає посилання на основні розділи додатку: список проєктів, завдання, завдання проєкту, чат, ауетентифікація, профіль
3. **Налаштування**
    - Розташовується на бічній панелі сайту та виглядає, як шестерня.
    - Відображає профіль користувача та можливість змінити пароль.
4. **Темна тема**
    - Можливість відобрадення додатку в темній темі.

Інструменти та технології для проєктування інтерфейсу:

1. **HTML -** Основна мова для створення структури веб-сторінок.
2. **CSS -** Використовується для оформлення та стилізації веб-сторінок.
3. **JavaScript -** Додає динамічність та інтерактивність до веб-сторінок.
4. **Bootstrap -** Фреймворк для швидкої розробки адаптивного та зручного у використанні інтерфейсу. Забезпечує готові стилі та компоненти, які можна легко використовувати та налаштовувати під потреби проєкту.

3.5 Взаємодія користувача із системою

3.5.1 Реєстрація та авторизація

На рис. 3.1 – 3.2 зображено сторінка Реєстрації та авторизації

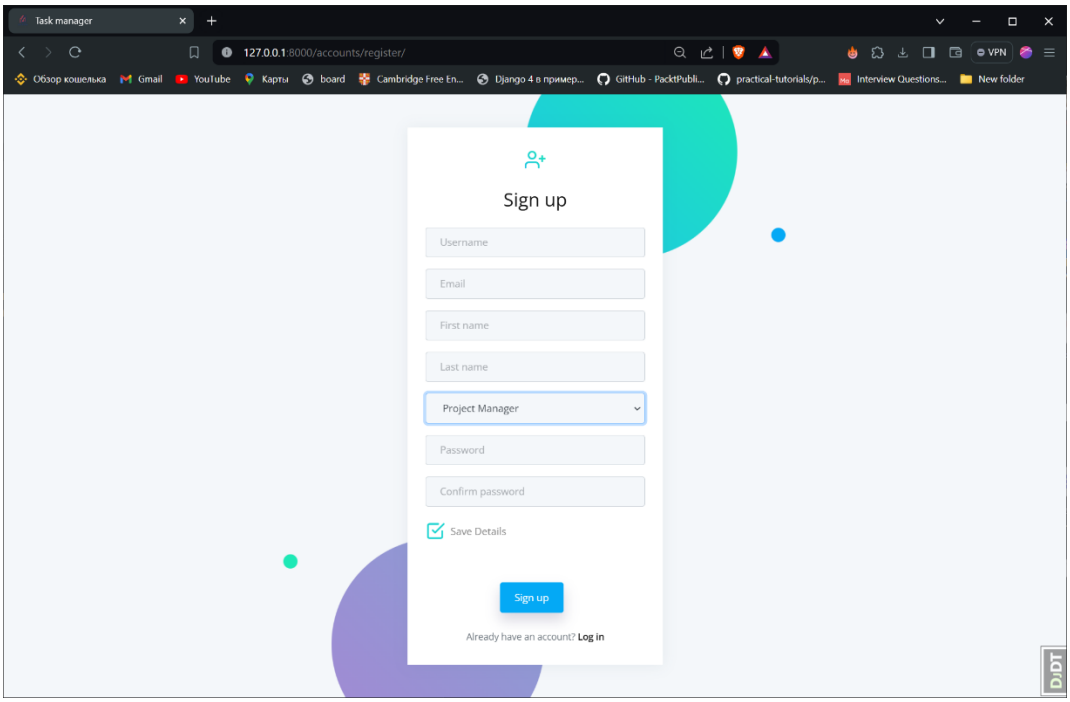

Рис 3.1 – Сторінка реєстрації

Реєстрація: Користувач заповнює форму реєстрації, вказуючи свої дані: ім'я, прізвище, електронну адресу, логін та пароль. Після успішної реєстрації користувач отримує доступ до системи.

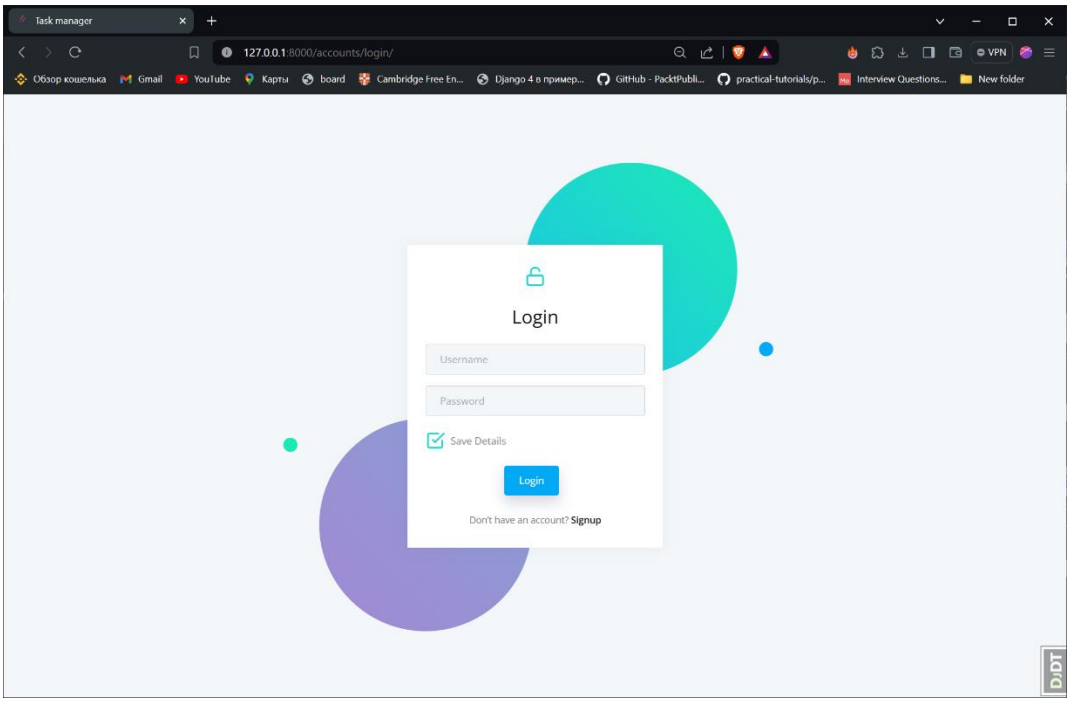

Рис 3.2 – Сторінка авторизації

Авторизація: Користувач вводить своє ім'я користувача та пароль для входу до системи. Після успішного входу користувач перенаправляється на головну сторінку.

3.5.3 Головна сторінка

На головній сторінці розміщено кількість проєктів, до яких належить користувач та кількість задач, створених користувачем. Якщо користувач не авторизован, на сторінці буде відображатися пропозиція авторизуватися. На рис. 3.3 – 3.4 зображена головна сторінка, якщо користувач автороизований, або ні.

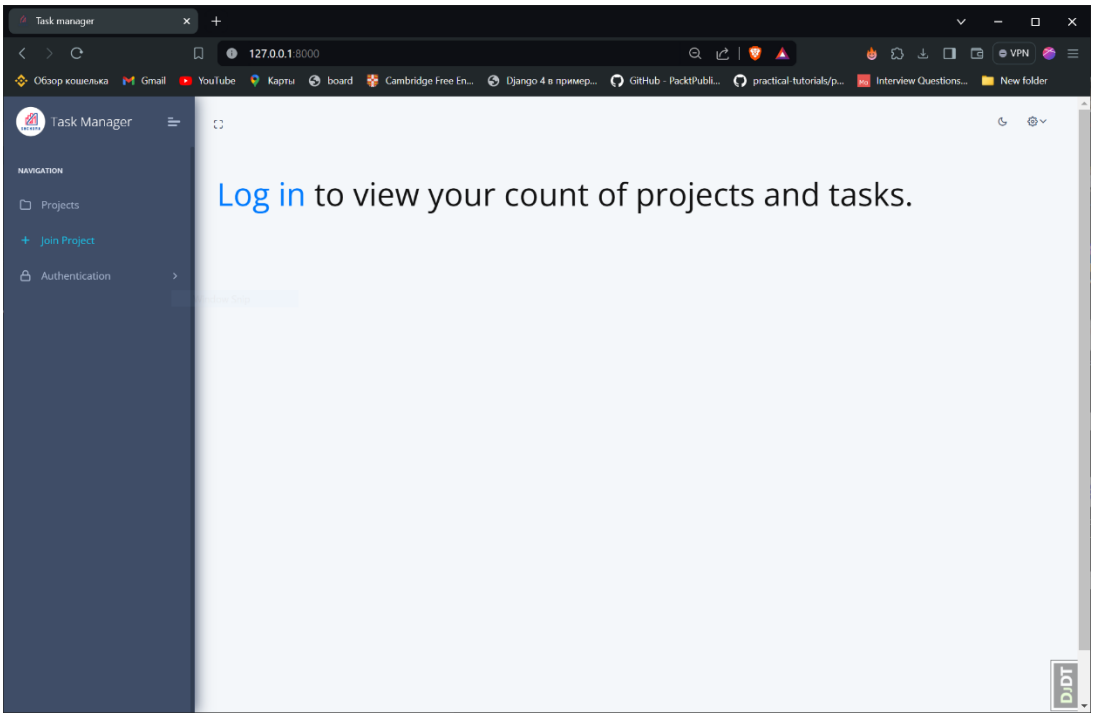

Рис. 3.3 – Головна сторінка, коли користувач не автороизований

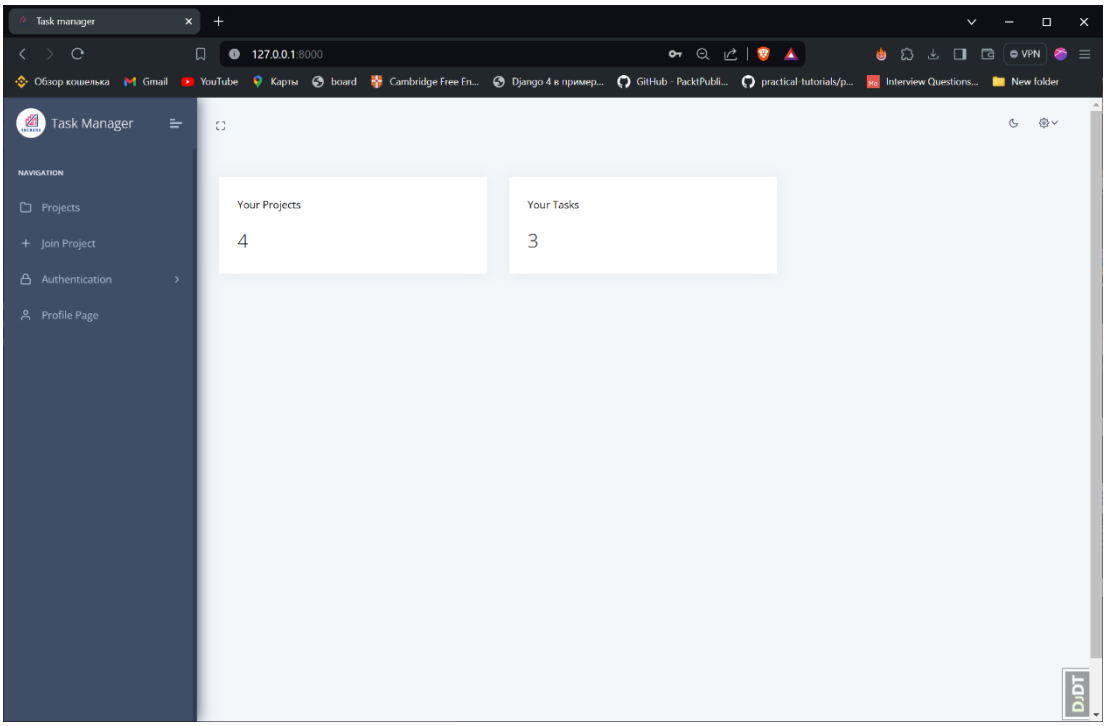

Рис. 3.4 – Головна сторінка, коли користувач авторизований

Якщо користувач не авторизований, у нього є можливість через випадаючий список зробити авторизацію, або зареєструватися (рис. 3.5).

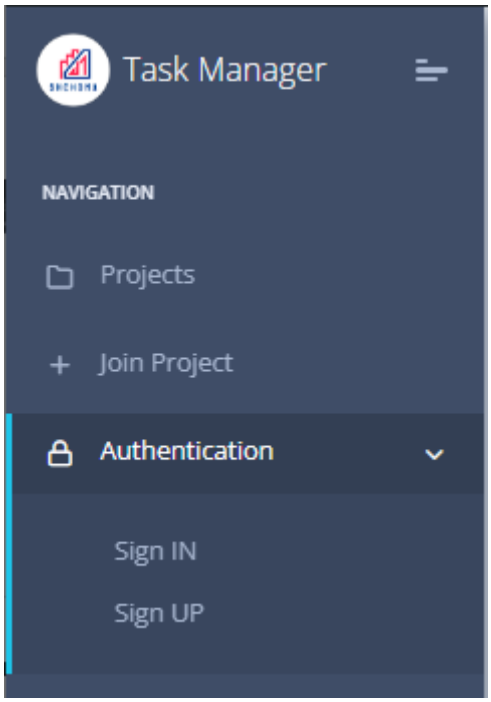

Рис. 3.5 – Навігаційна панель, коли користувач не авторизований

При авторизації користувач можу вийти із аккаунта натиснув на Log out (рис. 3.6).

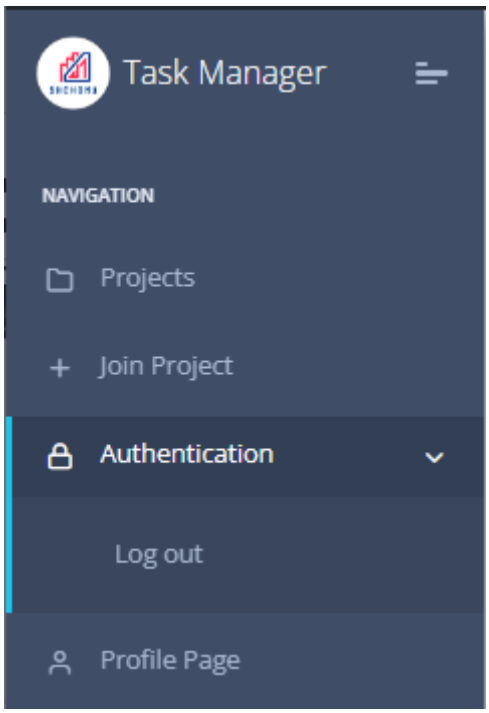

Рис. 3.6 – Навігаційна панель, коли користувач авторизований

3.5.2 Взаємодія з проєктами

Користувач переходить на сторінку Проєкти та переглядає список усіх проєктів, які користувач створив, або до яких приєднанний. На цій сторінці користувач має можливість створити новий проєкт, подивитись на певний проєкт, зробити пошук та використовувати пагінацію. На рис. 3.7 зображена сторінка списку проєктів.

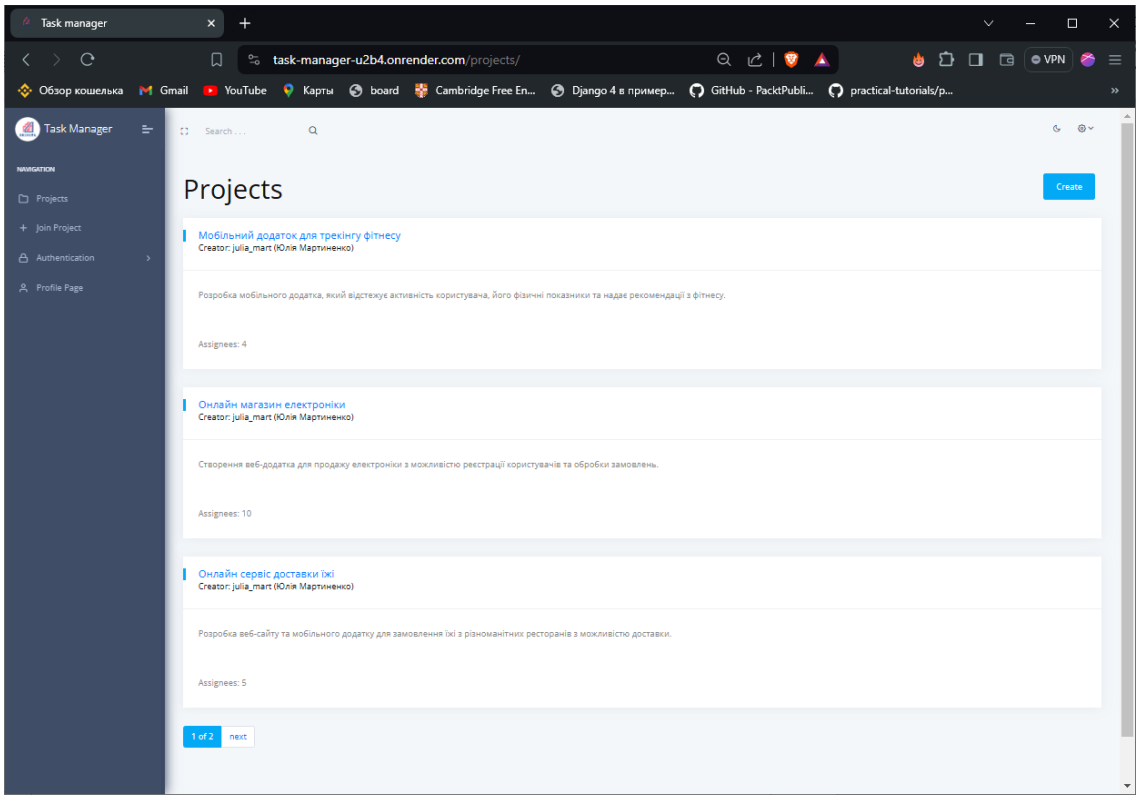

Рис. 3.7 – Сторінка списку проєктів

Користувач має можливість створювати новий проєкт. Для цього йому потрібно натиснути на клавішу `Create`. Після цього користувач перенаправиться до сторінки створення проєкту. На рис. 3.8 зображена сторінка створення проєкту.

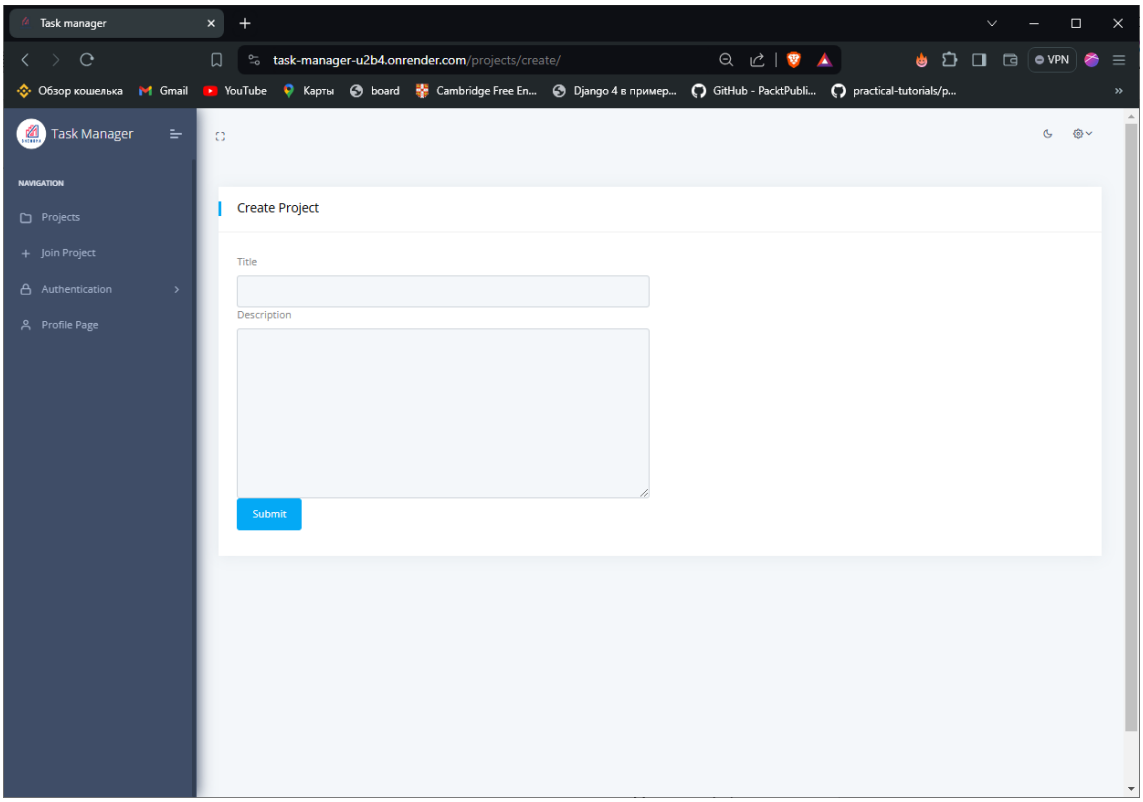

Рис 3.8 – Сторінка створення проєкту

У кожного проєкта є посилання на його детальний вигляд. На цій сторінці присутня назва, опису проєкту та кількість учасників. На рис. 3.9 зображена сторінка детального вигляду одно із проєктів.

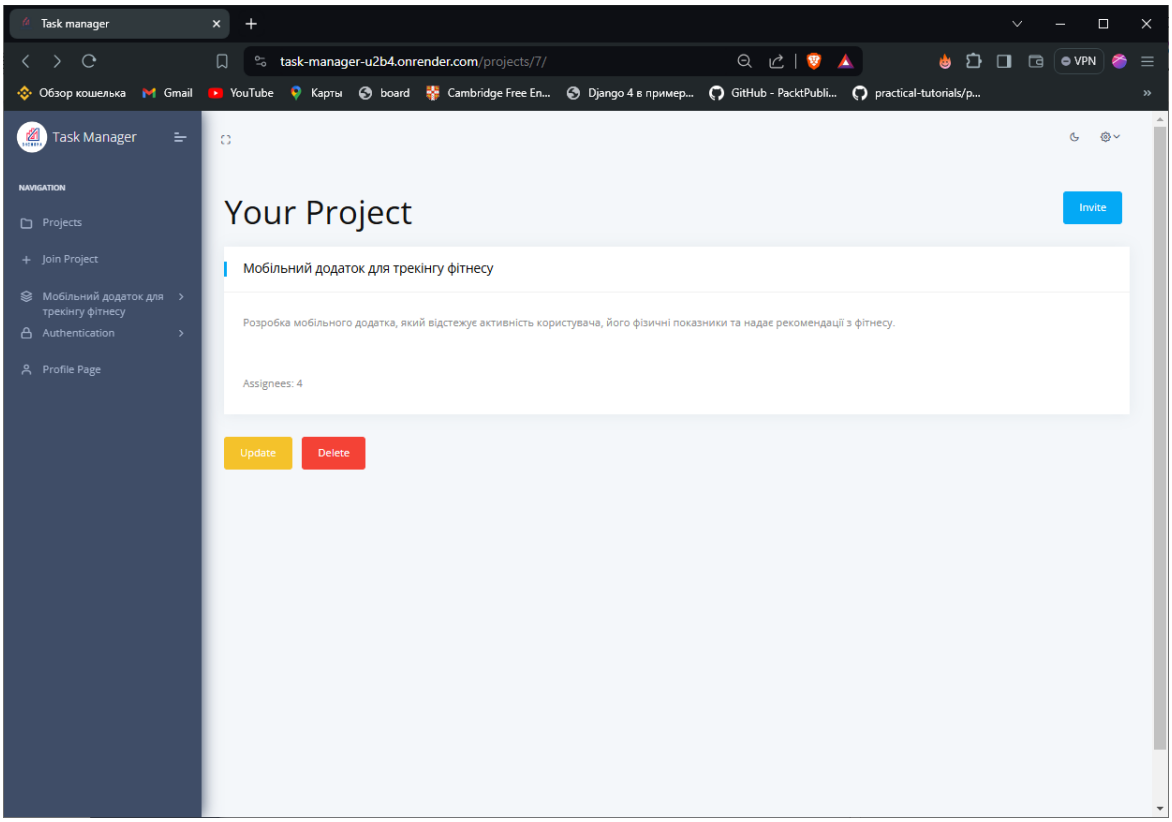

Рис. 3.9 – Детальна сторінка проєкта

На детальній сторінці проєкта присутні кнопки Update, Create, Invite. Кнопка Update – перенаправляє користувача на сторінку редагування проєкта, після редагування користувачу потрібно натиснути кнопку Submit для підтвердження змін (рис. 3.10). Кнопка Delete – перенаправляє користувача на сторінку видалення проєкта, де користувачу потрібно підтвердити видалення проєкта (рис. 3.11). Кнопка Invite – створює код для приєданнаня до поточного проєкта (рис. 3.12).

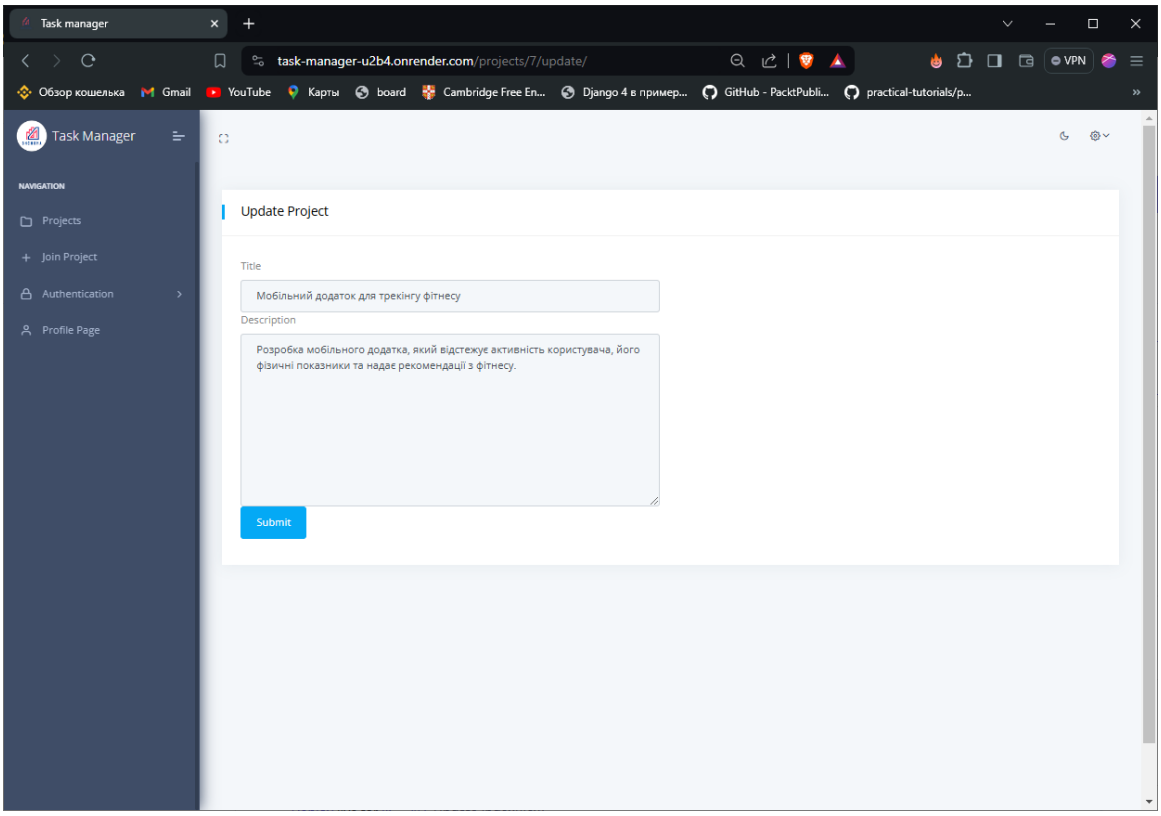

Рис. 3.10 – Сторінка редагування проєкта

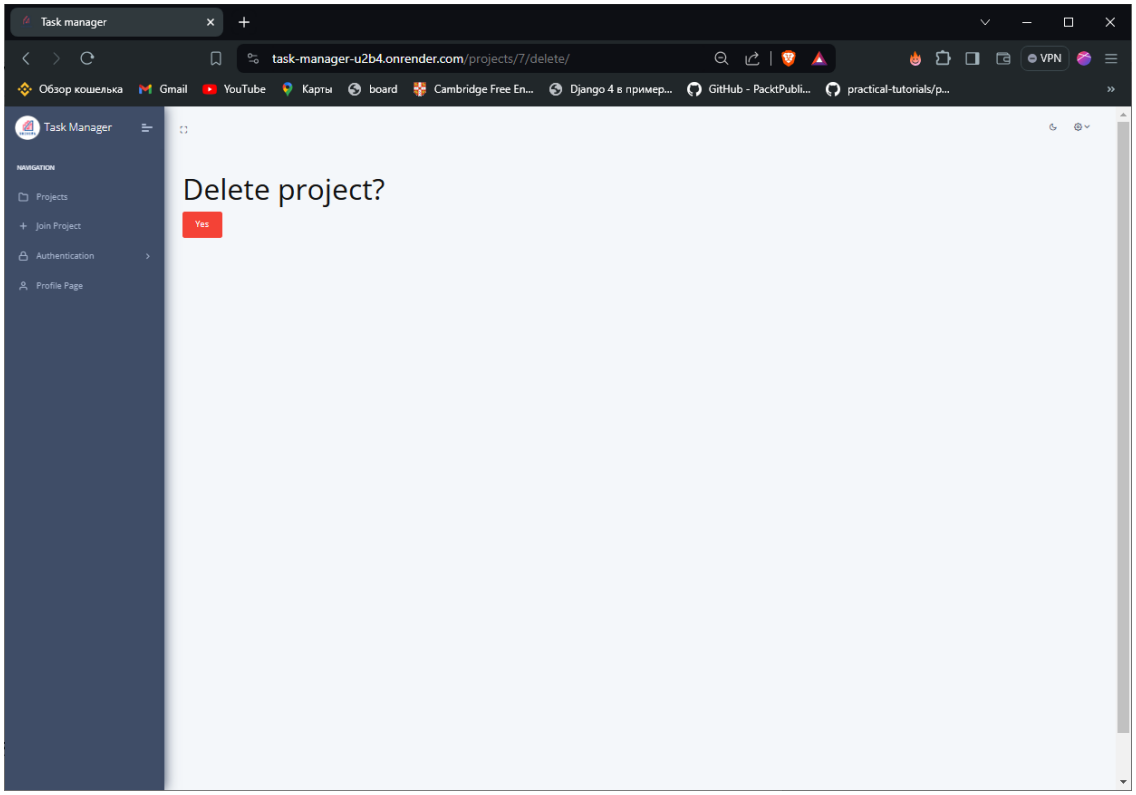

Рис. 3.11 – Сторінка видалення проєкта

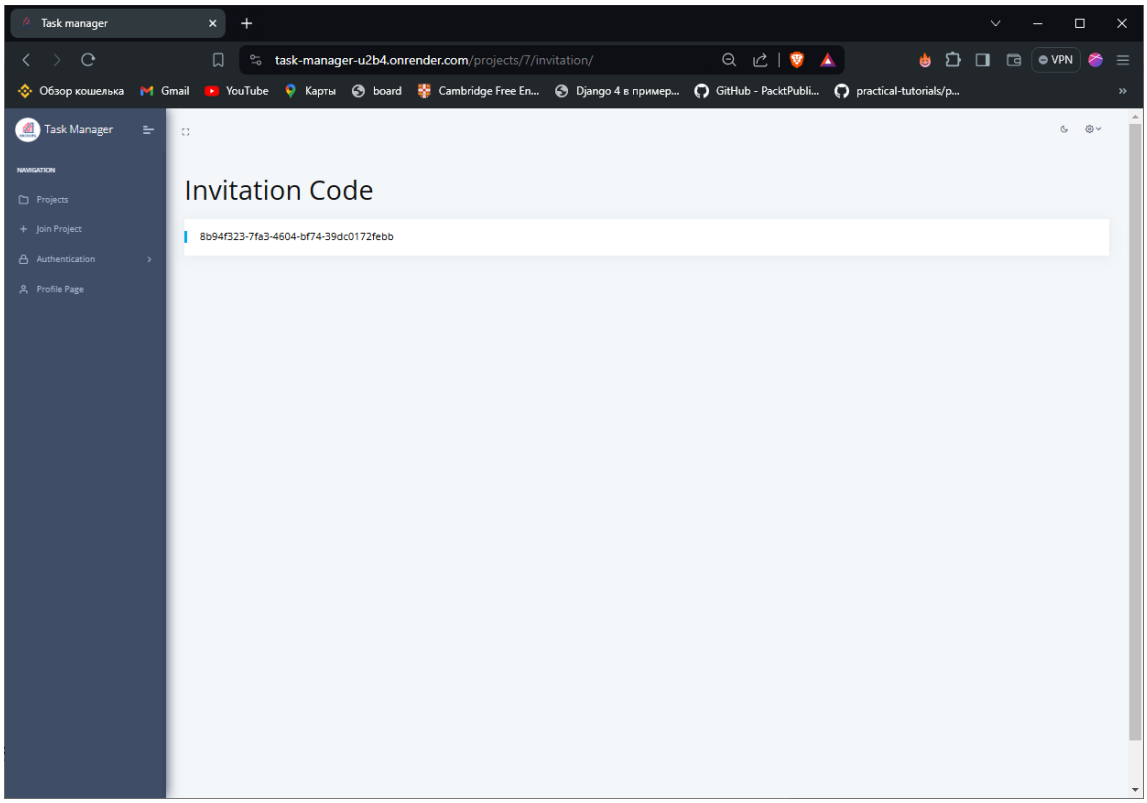

Рис. 3.12 – Сторінка згенерованого коду проєкта

Після згенерованого коду, власник проєкту може надіслати його іншим користувачам. Користувачу потрібно ввести цей код у вкладці Join Project та натиснути кнопку Join (рис. 3.13).

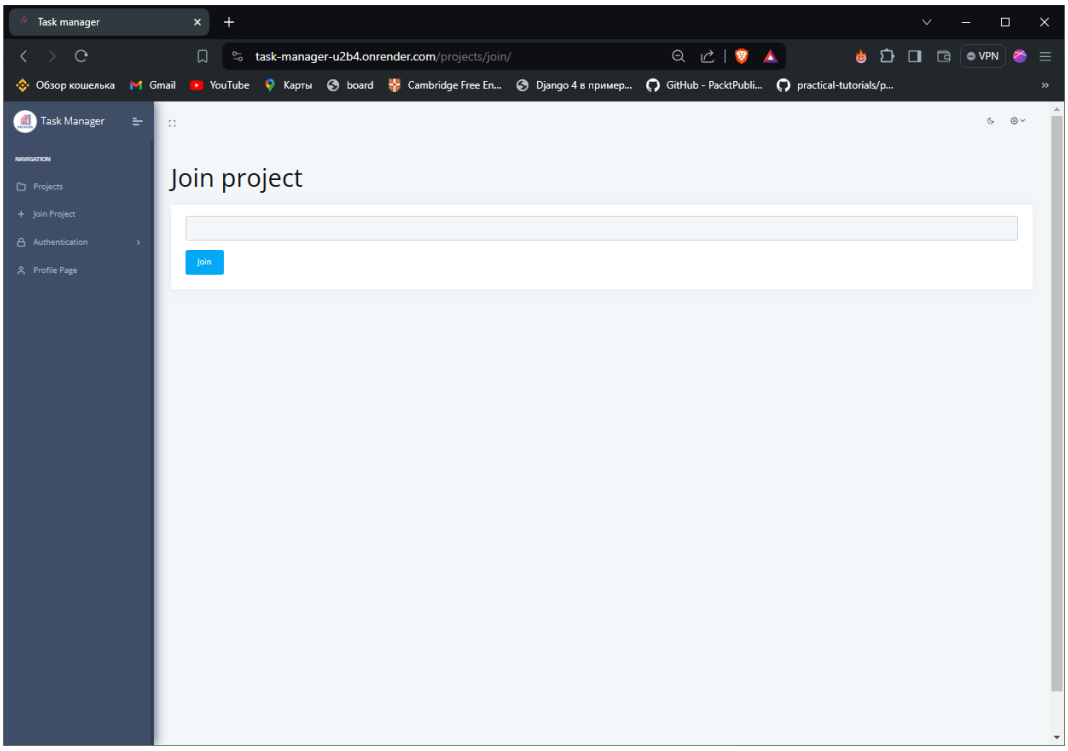

Рис. 3.13 – Сторінка форми для заходження до проєкту

В навігації проєкта можна отримати доступ до Project Tasks (задачі проєкта), Your Tasks (задачі користувача), Chat (чат проєкта) (рис. 3.14)

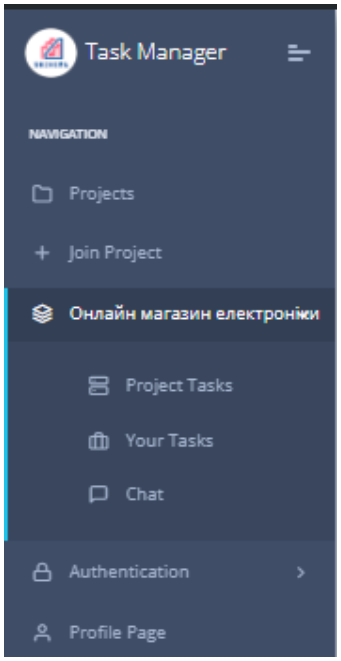

Рис 3.14 – Навігаційна панель проєкту

Chat – це місце, де користувачі можуть спілкуватися на рахунок проєкта (рис. 3.15).

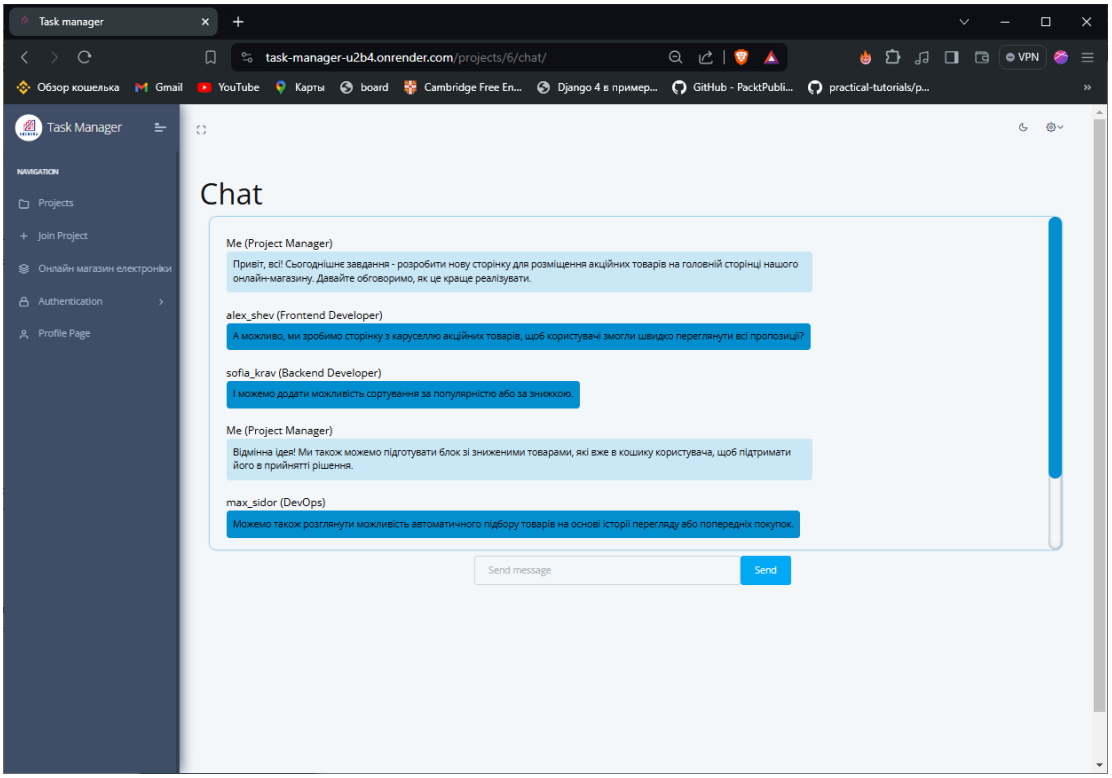

Рис. 3.15 – Сторінка чату проєкта

3.5.2 Взаємодія з задачами

Project Tasks – це список усіх задач користувачів поточного проєкту (рис. 3.16). Цей список виглядає, як таблиця, у котрій назва, автор, позиція автора, статус завдання, дедлайн. Також на цій сторінці присутня пагінація та пошук за назвою.

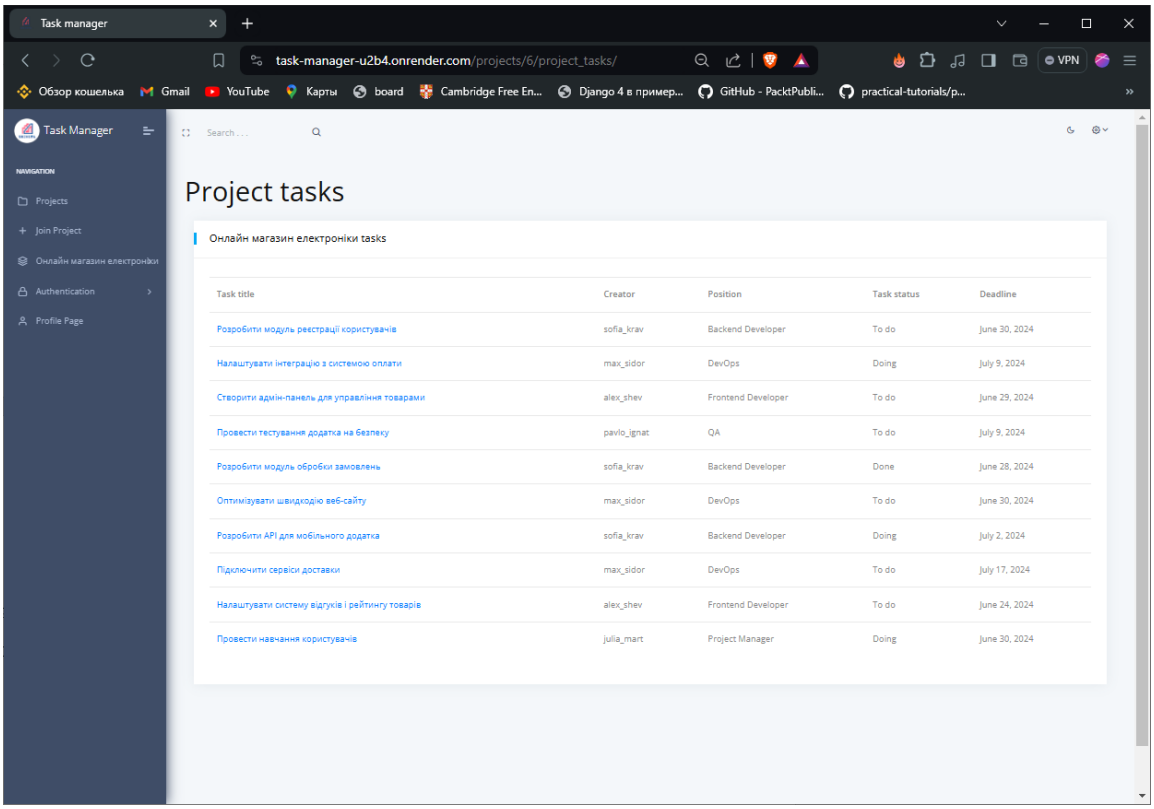

Рис 3.16 – Сторінка задач проєкту

Your Tasks – це список задач створених користувачем (рис. 3.17). На цій сторінці знаходяться три поля. To do – задачі, котрі потрібно зробити. Doing – задачі, котрі зараз виконуються. Done – задачі, котрі вже виконані.

Задачі можуть мати різний колір за рахунок пріорітету. Якщо задача знаходиться в Done, то має зелений колір. Якщо у задачі минув дедлайн або задача знаходиться в Done більше одного дня, то вона автоматично видаляється.

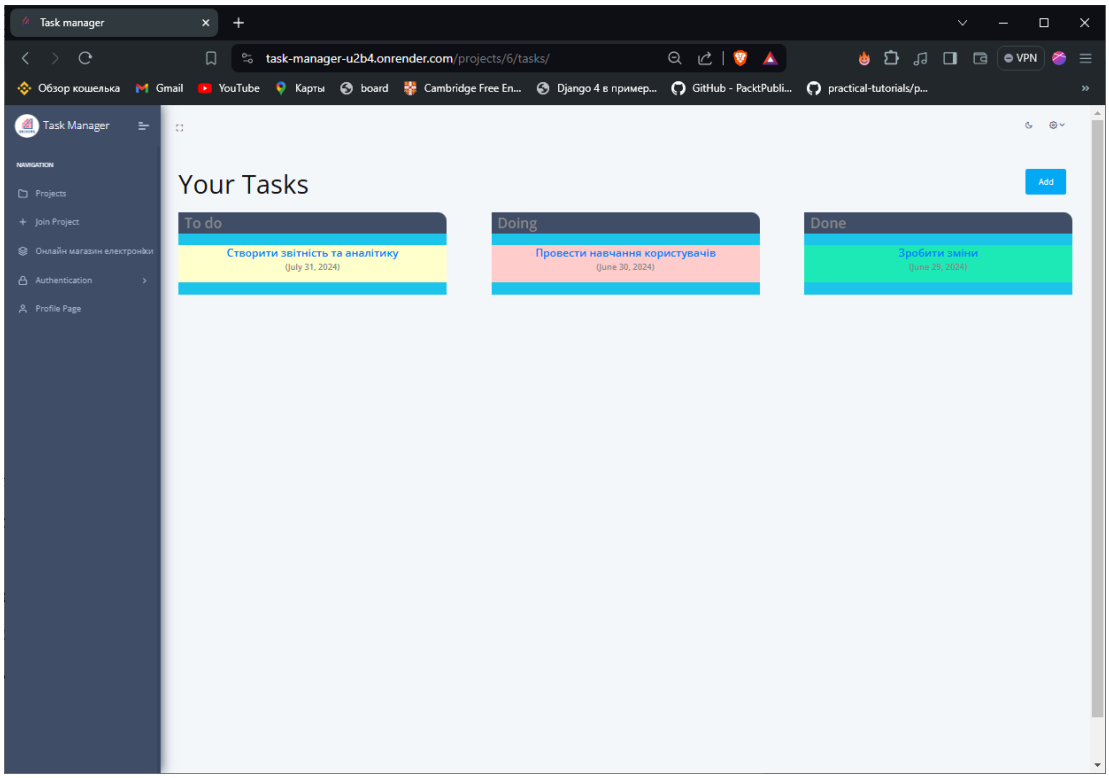

Рис 3.17 – Сторінка задач користувача

На сторінці задач користувача є кнопка Add, яка дозволяє додати нову задачу (рис. 3.18)

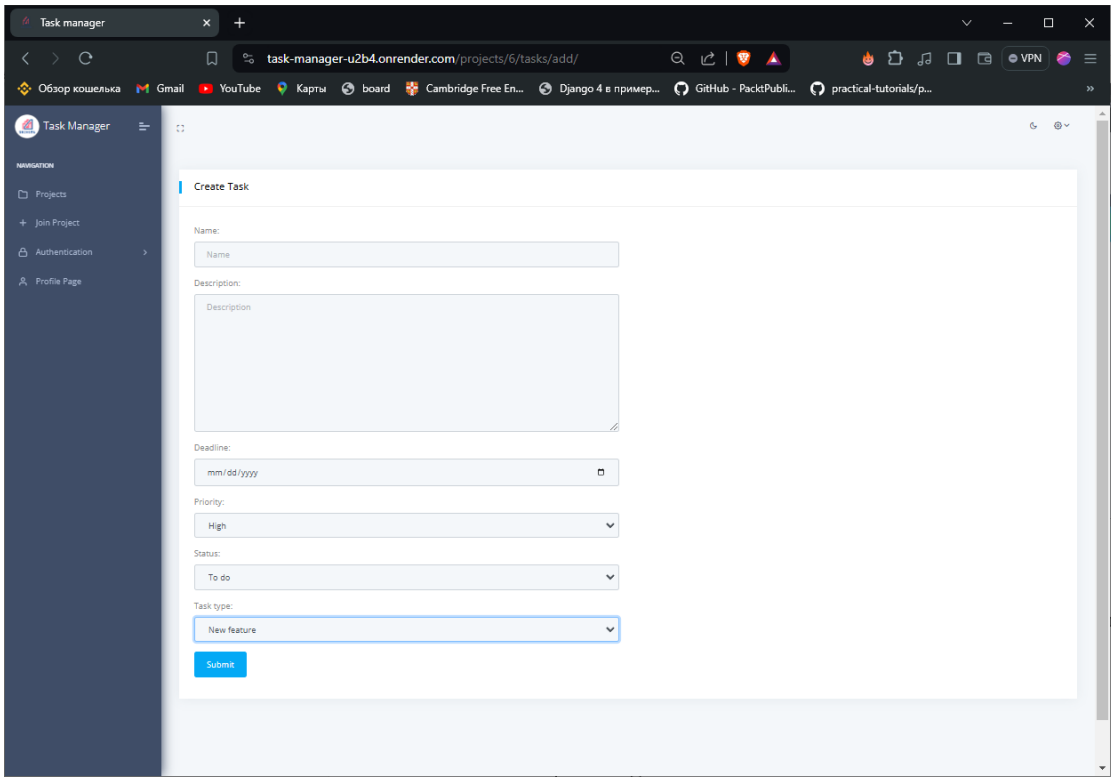

Рис. 3.18 – Сторінка створення задачі

На рис. 3.19 зображена детальна сторінка задачі.

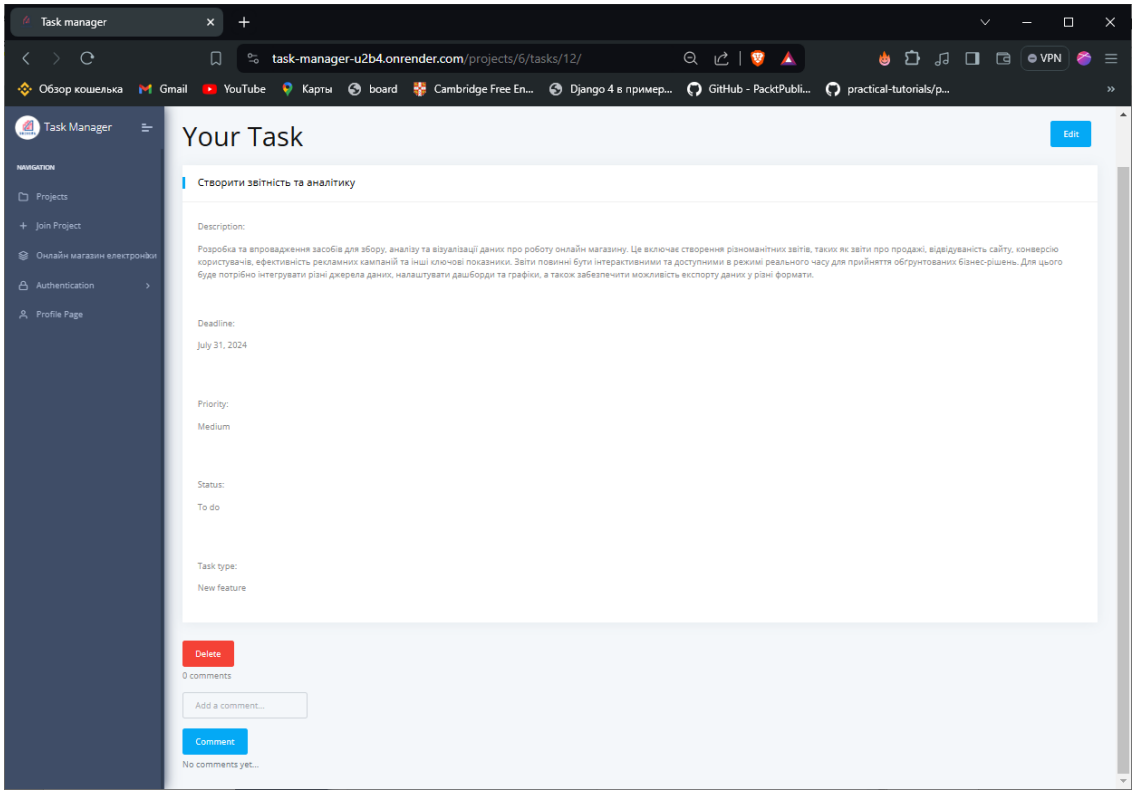

Рис. 3.19 – Детальна сторінка задачі

У детальній сторінці задачі є форма для відправлення коментаря (рис. 3.20).

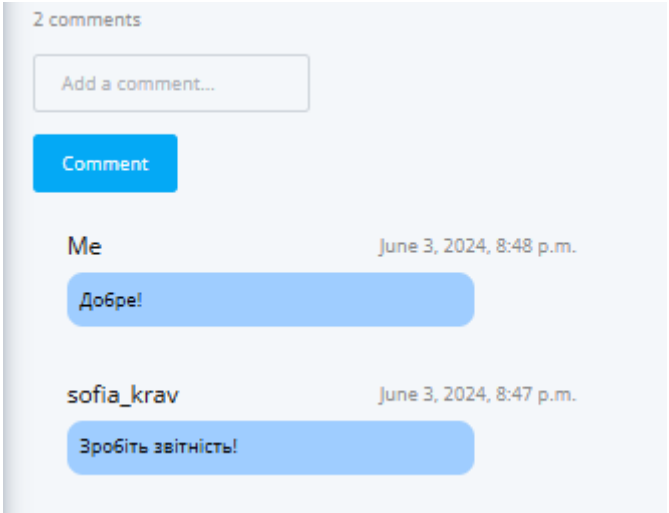

Рис. 3.20 – Секція коментарів на сторінці задачі

Також на детальній сторінці завдання присутні кнопки Delete, Edit. Delete – видалєю поточну задачу. Edit – перенаправляє користувача до сторінки редагування задачі (рис. 3.21).

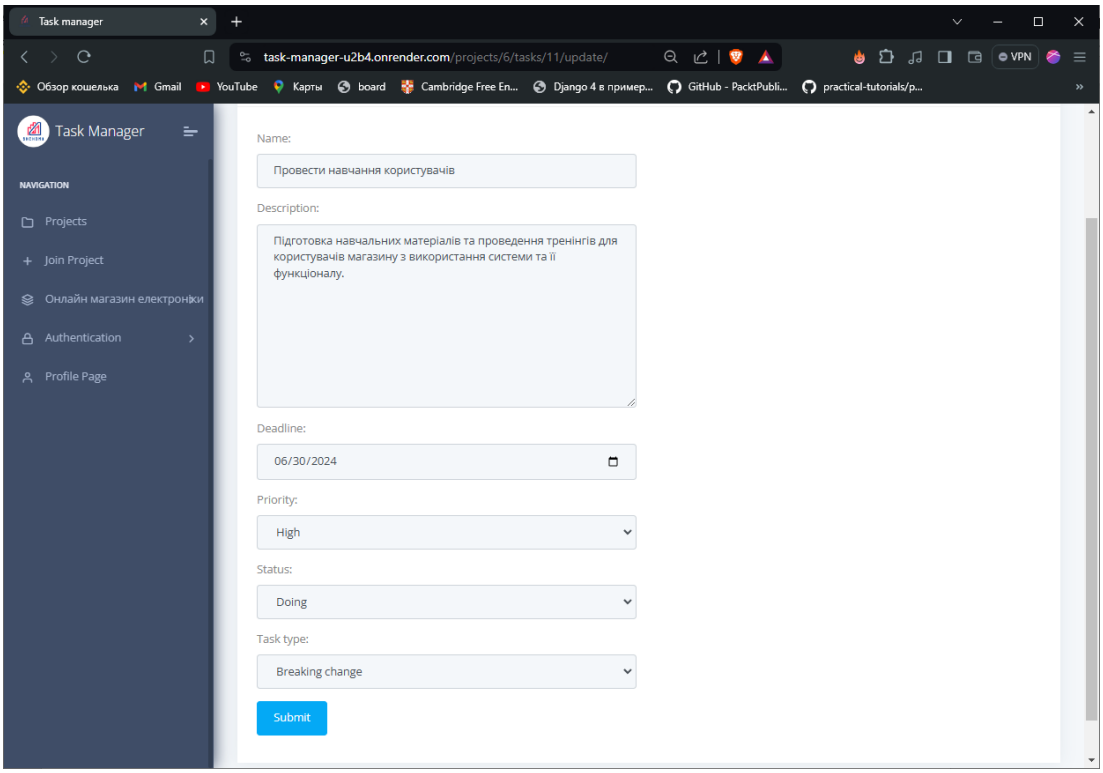

Рис. 3.21 – Сторінка редагування задачі

3.5.3 Профіль

У правій частині веб-додатку є шестерня, при натискані відкривається меню, де можна вийти з аккаунту, зайти у свій профіль, змінити пароль (рис. 3.22).

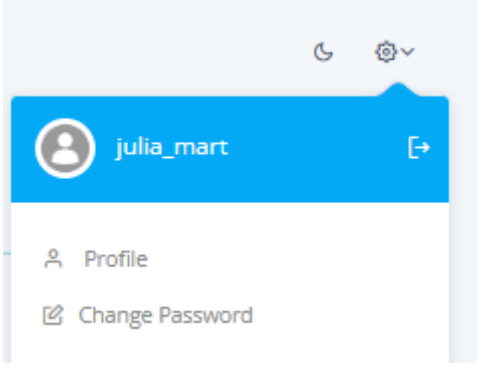

Рис. 3.22 – Меню налаштування

Change Password – це посилання на сторінку зміни пароля. Для зміни паролю потрібно вести свій поточний пароль та новий (рис. 3.23).

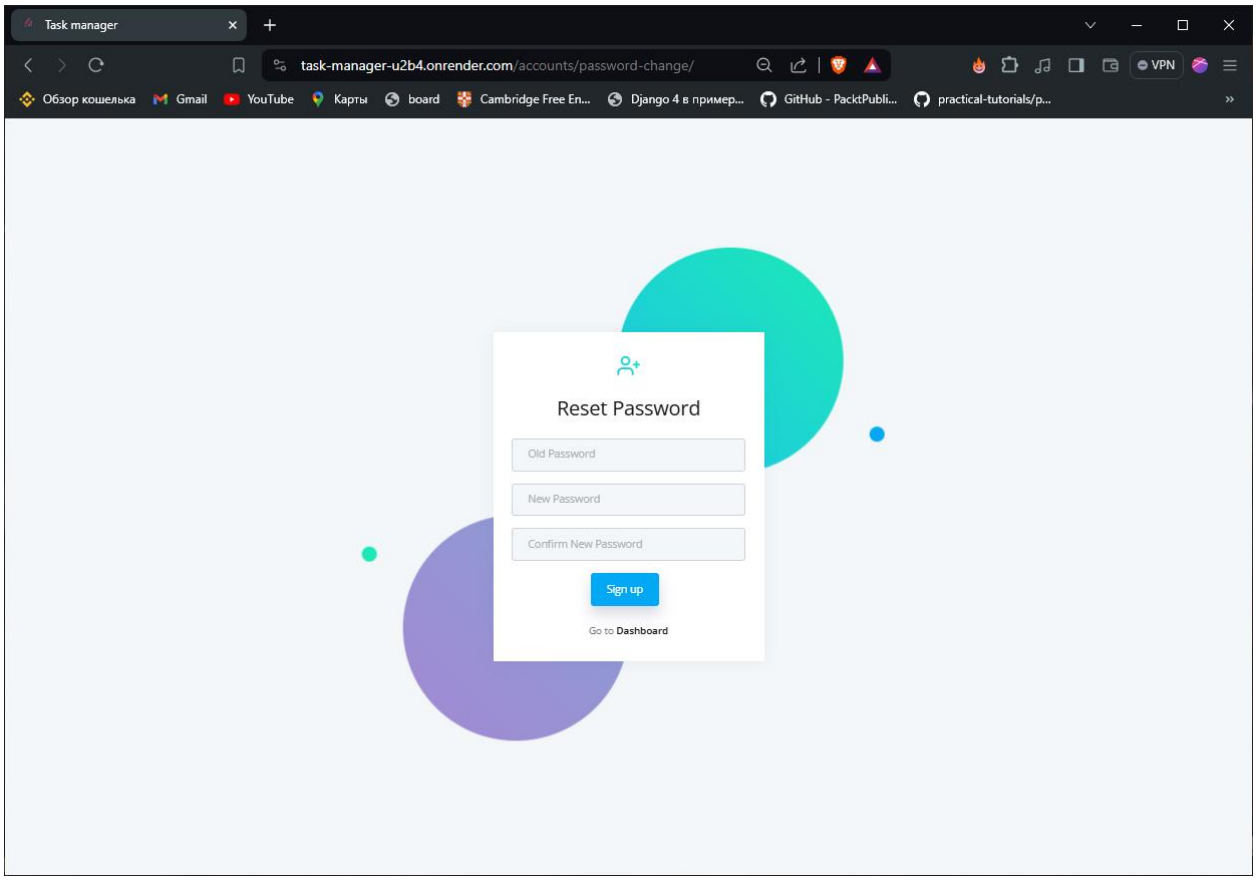

Рис. 3.23 – Сторінка зміни паролю

На рис. 3.24 зображена сторінка профілю користувача. На цій сторінці зображена інформація про користувача.

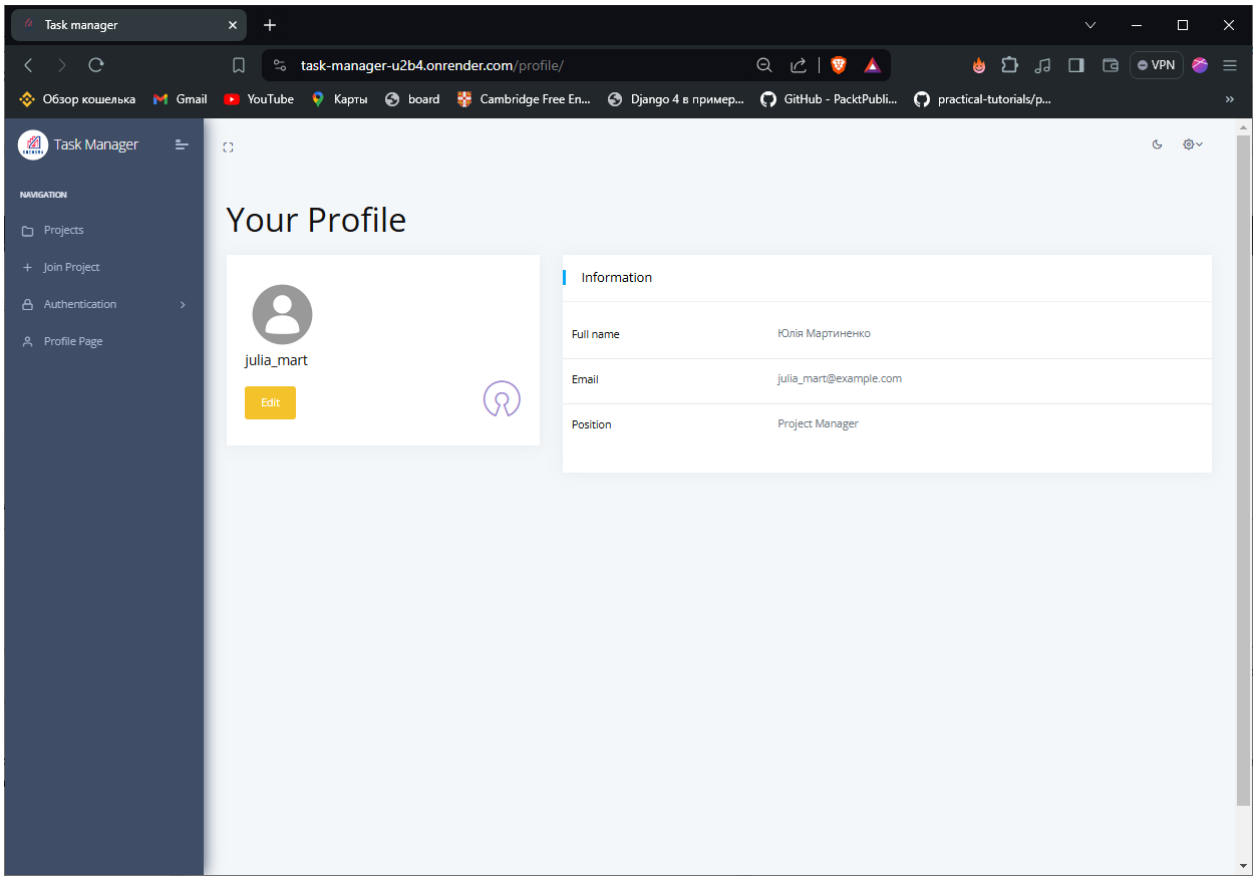

Рис. 3.24 – Сторінка профілю користувача

На сторінці профілю користувача є кнопка Edit (Редагування). Ця кнопка призначення для оновлення інформації користувача (рис. 3.25).

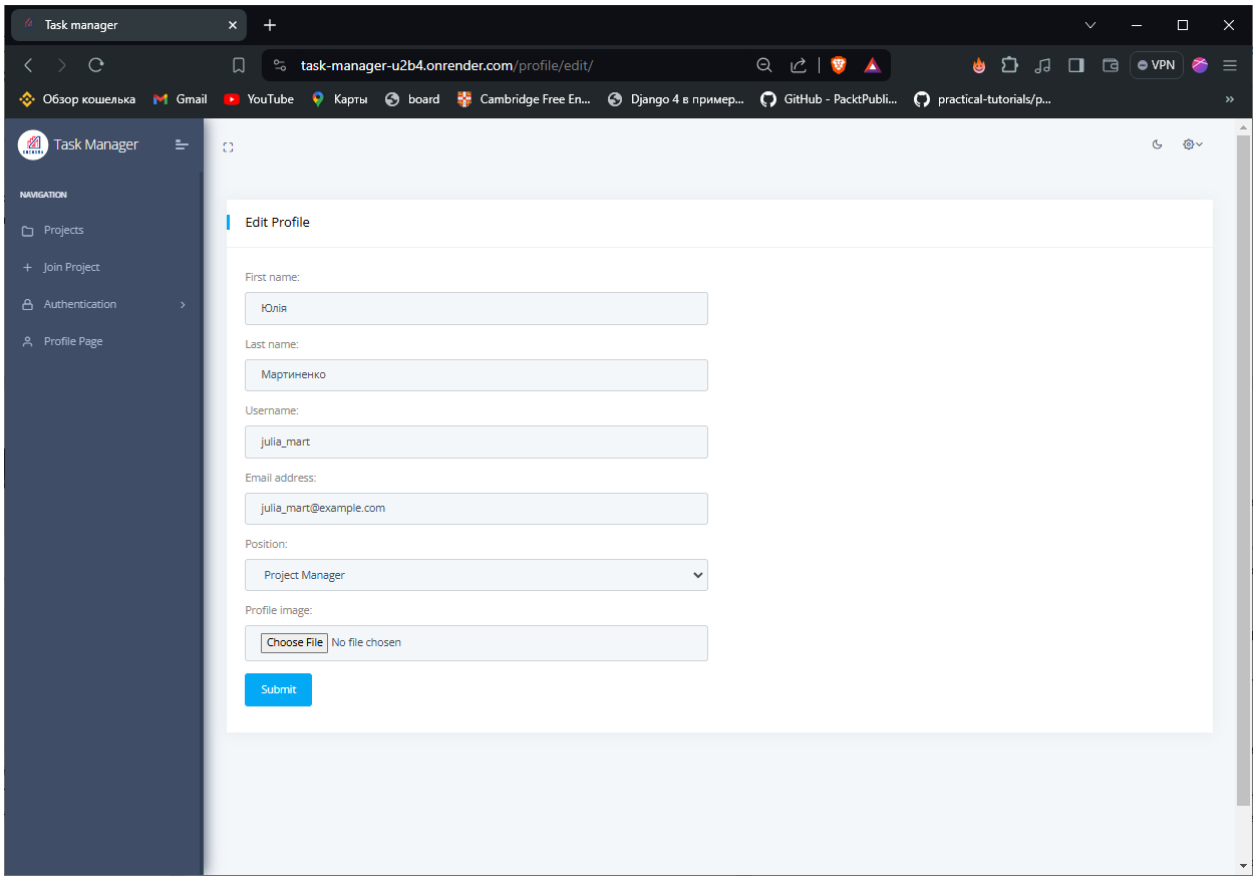

Рис. 3.25 – Сторінка редагування профілю користувача

3.6 Хостинг і розгортання системи

Для хостингу веб-додатка обрано Dashboard Render[7], що забезпечує зручне розгортання та управління додатками, а також інтеграцію з іншими сервісами. Адреса для доступу до проєкту: https://task-manager-u2b4.onrender.com.

Основні етапи розгортання:

1. **Налаштування середовища розробки:**
   - Вибір фреймворка для розробки.
   - Конфігурація середовища розробки з урахуванням вибраних технологій.
2. **Розгортання бази даних:**
   - Створення бази даних на платформі neon.tech[8].
   - Налаштування з'єднання з базою даних у додатку.
3. **Налаштування CI/CD:**

- Впровадження системи безперервної інтеграції та безперервного розгортання (CI/CD) для автоматизації процесу розгортання.
- Інтеграція з GitHub або іншим репозиторієм для автоматичного розгортання нових версій додатка.

4. **Конфігурація хостингу:**
    - Вибір відповідного тарифного плану на Dashboard Render з урахуванням вимог до продуктивності та масштабованості.
    - Налаштування домену та SSL-сертифікатів для забезпечення безпеки даних.

Висновки за розділом 3

У цьому розділі було детально описано процес реалізації веб-додатку для agile управління it-проєктами, включаючи використані засоби розробки, вимоги до технічного та програмного забезпечення, опис програмної реалізації та проєктування інтерфейсу. Описані також ключові елементи керівництва користувача, що допомагають користувачам ефективно взаємодіяти з системою. Наступні розділи будуть присвячені тестуванню та оцінці ефективності розробленого додатку.

ВИСНОВКИ

У даній роботі було виконано розробку веб-додатку для управління IT-проєктами за методологією Agile з використанням технологій Python, Django та PostgreSQL. Проведений аналіз та реалізація додатку дозволили досягти наступних результатів:

1. **Реалізація функціональності веб-додатку**:
    - Додаток реалізує наступні основні функції: авторизація користувачів, управління профілем, створення та управління проєктами, робота з задачами, адаптація інтерфейсу за допомогою Bootstrap. Це забезпечує зручність використання та ефективне управління проєктами для команд, що працюють за методологією Agile.
2. **Використання сучасних технологій**:
    - У розробці використано сучасні технології, такі як Python, Django та PostgreSQL, що забезпечує високу продуктивність та надійність роботи додатку.

Таким чином, виконана робота підтверджує можливість і доцільність розробки веб-додатку для Agile управління IT-проєктами, який відповідає сучасним вимогам та сприяє підвищенню ефективності та якості роботи команд. Подальший розвиток додатку може включати розширення функціональності та інтеграцію з іншими системами управління проєктами для досягнення ще вищої продуктивності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Agile software development [Електронний ресурс]. – режим доступу: URL: https://en.wikipedia.org/wiki/Agile_software_development (Дата звернення - 14.05.2024)
2. Agile vs. Waterfall [Електронний ресурс]. – режим доступу: URL: https://www.productplan.com/learn/agile-vs-waterfall/ (Дата звернення - 12.05.2024)
3. Kanban - A brief introduction | Atlassian [Електронний ресурс]. – режим доступу: URL: https://www.atlassian.com/agile/kanban (Дата звернення - 15.05.2024)
4. Jira (software) [Електронний ресурс]. – режим доступу: URL: https://en.wikipedia.org/wiki/Jira_(software) (Дата звернення - 15.05.2024)
5. Trello | Atlassian [Електронний ресурс]. – режим доступу: URL: https://www.atlassian.com/software/trello (Дата звернення - 17.05.2024)
6. Project Management Software | Microsoft Project [Електронний ресурс]. – режим доступу: URL: https://www.microsoft.com/en-us/microsoft-365/project/project-management-software (Дата звернення - 13.05.2024)
7. Render · The Easiest Cloud For All Your Apps [Електронний ресурс]. – режим доступу: URL: https://dashboard.render.com/ (Дата звернення - 15.05.2024)
8. Neon Serverless Postgres — Ship faster [Електронний ресурс]. – режим доступу: URL: https://neon.tech/ (Дата звернення - 15.05.2024)

ДОДАТКИ

Додаток А

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В. Н. Каразіна

Факультет комп'ютерних наук
Кафедра теоретичної та прикладної системотехніки
Рівень вищої освіти (освітньо-кваліфікаційний рівень) **бакалавр**
Галузь знань: **12 – Інформаційні технології**
Спеціальність: **123 – Комп'ютерна інженерія.**

ЗАТВЕРДЖУЮ
Завідувач кафедри теоретичної
та прикладної системотехніки
_____д.т.н., проф. Шматков С. І.
«21» грудня 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

ЩОМА ДМИТРО РУСЛАНОВИЧ

(прізвище, ім'я, по батькові студента)

1. Тема роботи **«WEB-ДОДАТОК AGILE УПРАВЛІННЯ IT-ПРОЄКТАМИ»**

керівник роботи Булавін Дмитро Олексійович, доцент, к.т.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «03» травня 2024року № 4101-5/909

2. Строк подання студентом роботи 31 травня 2024року

3. Перелік питань, які потрібно розробити

1) Визначення вимог до функціоналу веб-додатка на основі методів Agile.
2) Проектування архітектури веб-додатка з урахуванням принципів масштабованості та гнучкості Agile.
3) Розроблення функціоналу для створення та управління завданнями, ітераціями та спринтами в межах веб-додатка.

4. План роботи

№ з/п	Назви етапів роботи	Термін виконання етапів роботи
1	Аналіз научної літератури	14.02.2024 - 20.02.2024
2	Аналіз наявних веб-застосунків для управління завданнями та проєктами.	21.02.2023 - 28.02.2024
3	Проектування веб-додатка	29.02.2024 - 13.03.2024
4	Розроблення функціоналу	14.03.2024 - 27.03.2024
5	Тестування та налагодження	28.03.2024 - 10.04.2024
6	Підготовка до випуску	11.04.2024 - 24.04.2024
7	Розробка пояснювальної записки.	25.04.2024 - 8.05.2024

5. Дата видачі завдання 14.02.2024

Студент Д. Р. Щома _____
ініціали, прізвище підпис

Керівник роботи Д. О. Булавін _____
ініціали, прізвище підпис

Додаток Б

Технічне завдання
на розробку програмного виробу
«WEB-ДОДАТОК AGILE УПРАВЛІННЯ ІТ-ПРОЄКТАМИ»

<table>
<tr><td>Назва розділу</td><td colspan="2">Назва і зміст підрозділу</td></tr>
<tr><td>1. Об'єкт випробувань.</td><td colspan="2">1.1. Найменування випробуваного програмного виробу.
1.2. Область його застосування</td></tr>
<tr><td>2. Мета випробувань.</td><td colspan="2">Перевірка функціональних можливостей, надійності та сумісності програмного виробу для забезпечення ефективного управління ІТ-проєктами.</td></tr>
<tr><td>3. Призначення розробки.</td><td colspan="2">3.1. Мета розробки програмного виробу
3.2. Призначення програмного виробу
3.3. Початкові дані для розробки</td></tr>
<tr><td>4. Технічні вимоги до програмного виробу.</td><td colspan="2">4.1. Вимоги до функціональних характеристик:
4.2. Вимоги до надійності</td></tr>
<tr><td>5. Вимоги до програмної документації.</td><td colspan="2">5.1 Склад програмної документації, що подається на випробування</td></tr>
<tr><td>6. Техніко-економічні показники.</td><td colspan="2">6.1. Засоби випробувань.
6.2. Порядок проведення випробувань.</td></tr>
<tr><td rowspan="5">7. Стадії і етапи розробки</td><td>Дата</td><td>Назва етапу</td></tr>
<tr><td>Від 14 лютого 2024
До 20 лютого 2024</td><td>Аналіз научної літератури.</td></tr>
<tr><td>Від 21 лютого 2024
До 28 лютого 2024</td><td>Аналіз наявних веб-застосунків для управління завданнями та проєктами.</td></tr>
<tr><td>Від 29 лютого 2024
До 13 березня 2024</td><td>Проектування веб-додатка.</td></tr>
<tr><td>Від 14 березня 2024</td><td>Розроблення функціоналу.</td></tr>
</table>

	До 27 березня 2024	
	Від 28 березня 2024 До 10 квітня 2024	Тестування та налагодження.
	Від 11 квітня 2024 До 24 квітня 2024	Підготовка до випуску.
	Від 25 квітня 2024 До 8 травня 2024	Розробка пояснювальної записки.
8. Порядок контролю і приймання	1. Перевірку ходу розробки програми виконувати раз в 3 тижні. 2. Захист розробленої моделі провести на засіданні Атестаційної комісії. 3. Пояснювальну записку подати в електронному вигляді в 1 примірнику.	

Виконавець

студент групи КІ-41

Щома Д. Р.

Замовник

Булавін Д. О

Додаток В

Програма і методика випробувань

програмного вироби

«WEB-ДОДАТОК AGILE УПРАВЛІННЯ ІТ-ПРОЄКТАМИ»

1. Об'єкт випробувань

1.1 Найменування випробуваного програмного виробу

WEB-ДОДАТОК AGILE УПРАВЛІННЯ ІТ-ПРОЄКТАМИ

1.2 Область його застосування

Управління ІТ-проєктами з використанням Agile-методологій.

2. Мета випробувань

Перевірка функціональних можливостей, надійності та сумісності програмного виробу для забезпечення ефективного управління ІТ-проєктами.

3. Загальні положення

3.1 Підстави для проведення випробувань

Підставою для проведення випробувань є наказ про призначення атестаційної комісії.

3.2 Місце і тривалість випробувань

Приймальні (приймально-здавальні) випробування проводяться на базі комп'ютерного класу кафедри в період роботи атестаційної комісії.

3.3 Обсяг випробувань

Приймальні випробування програмного вироби проводяться в обсязі відповідному цієї Програми і методики випробувань.

3.4 Організації, які беруть участь у випробуваннях

Приймальні випробування проводяться атестаційною комісією напередодні засідання (або в процесі засідання) за участю Замовника, Виконавця та інших осіб, присутніх на засіданні.

4. Вимоги до програми або програмного виробу

4.1. Вимоги до функціональних характеристик:

1) Забезпечення управління задачами (створення, редагування, видалення).

2) Відстеження прогресу по задачам.

3) Підтримка командної роботи (ролі).

4) Взаємодія з проєктами.

5) Взаємодія з профілем.

4.2. Вимоги до надійності:

1) Збереження даних при збої системи

4.3. Вимоги до умов експлуатації офісні приміщення.

4.4. Вимоги до складу і параметрів технічних засобів Персональний комп’ютер у повній комплектації (ноутбук)

4.5. Вимоги до інформаційної та програмної сумісності забезпечити сумісність з усіма обчислювальними засобами.

4.6. Вимоги до маркування та упаковки відсутні.

4.7. Вимоги до транспортування і зберігання відсутні.

4.8. Спеціальні вимоги не пред'являються.

5. Вимоги до програмної документації

5.1 Склад програмної документації, що подається на випробування

1) Технічне завдання на розробку програмного вироби (представлено в Додатку Б до пояснювальної записки до дипломної роботи).

2) Ця Програма і методика випробувань розробленого програмного вироби (представлена в Додатку В до пояснювальної записки до дипломної роботи).

3) Опис програмного вироби (представлено в розділі 3 пояснювальної записки до дипломної роботи).

6. Засоби і порядок випробувань

6.1 Засоби випробувань

Випробування проводяться на технічних засобах, яких персональний комп’ютер, ноутбук.

Випробування проводяться з використанням програмних засобів, таких як Pycharm.

6.2 Порядок проведення випробувань

Перевірка комплектності складу програмної документації здійснюється за критерієм наявності зазначеної в технічному завданні документації.

Перевірку здійснювати за критерієм відповідності вимогам єдиної системи програмної документації (ЄСПД).

6.2.1. Перевірка виконання програми.

Тест 1: Перевірка створення проєкту. По виду рисунку робиться висновок про працездатність моделі (рис. 6.1 – 6.2).

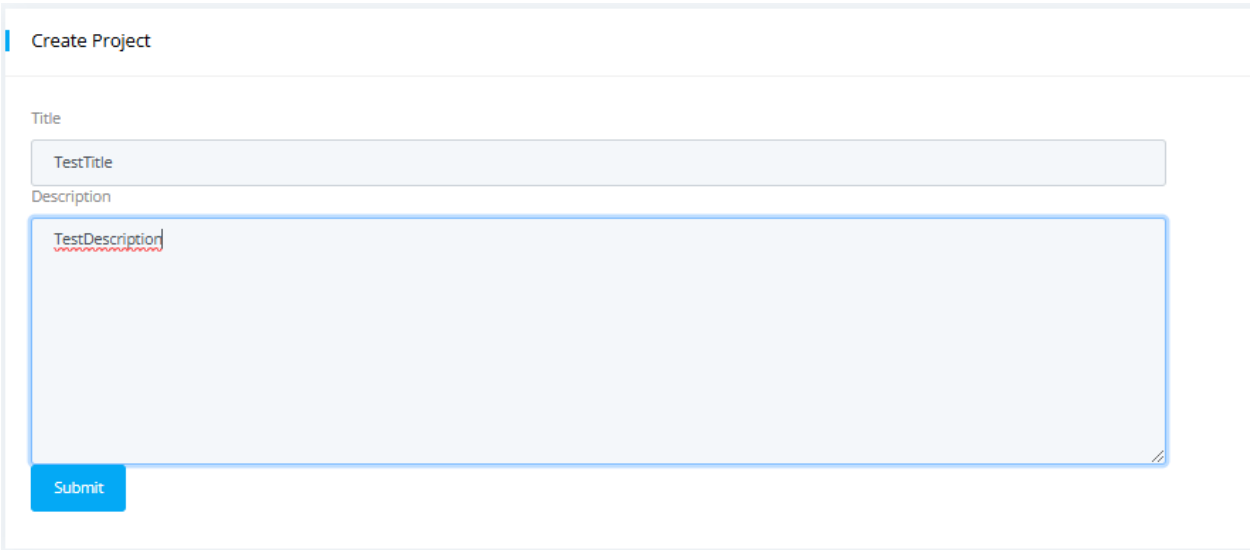

Рисунок 6.1 – Перевірка створення проєкту

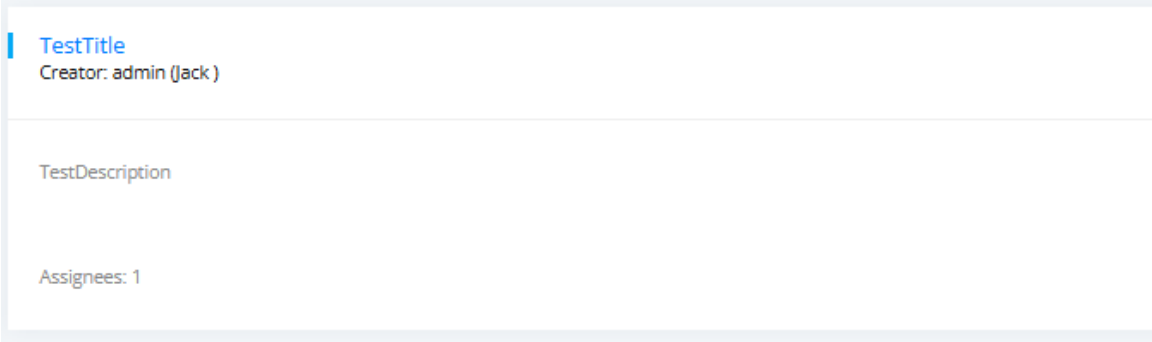

Рисунок 6.2 – Результат створення проєкту

Висновок: перевірка створення проєкту пройшла успішно.

Тест 2: Перевірка створення Python-програми з тесту Project моделі. По виду рисунку робиться висновок про працездатність моделі (рис. 6.3 – 6.4).

```python
Dmytro Shchoma
class ProjectModelTest(TestCase):
    Dmytro Shchoma
    @classmethod
    def setUpTestData(cls):
        worker = get_user_model().objects.create_user(
            username="Test3",
            password="test1234",
            first_name="Test1",
            last_name="Test2"
        )
        Project.objects.create(
            title="TestTitle",
            creator=worker
        )

    Dmytro Shchoma
    def test_project_str(self):
        project = Project.objects.get(id=1)
        self.assertEqual(str(project), second: "TestTitle - creator: Test3 (Test1 Test2)")

    Dmytro Shchoma
    def test_get_absolute_url(self):
        project = Project.objects.get(id=1)
        self.assertEqual(project.get_absolute_url(), second: "/projects/1/")
```

Рис 6.3 – Код Python-програми. Тестування Project model

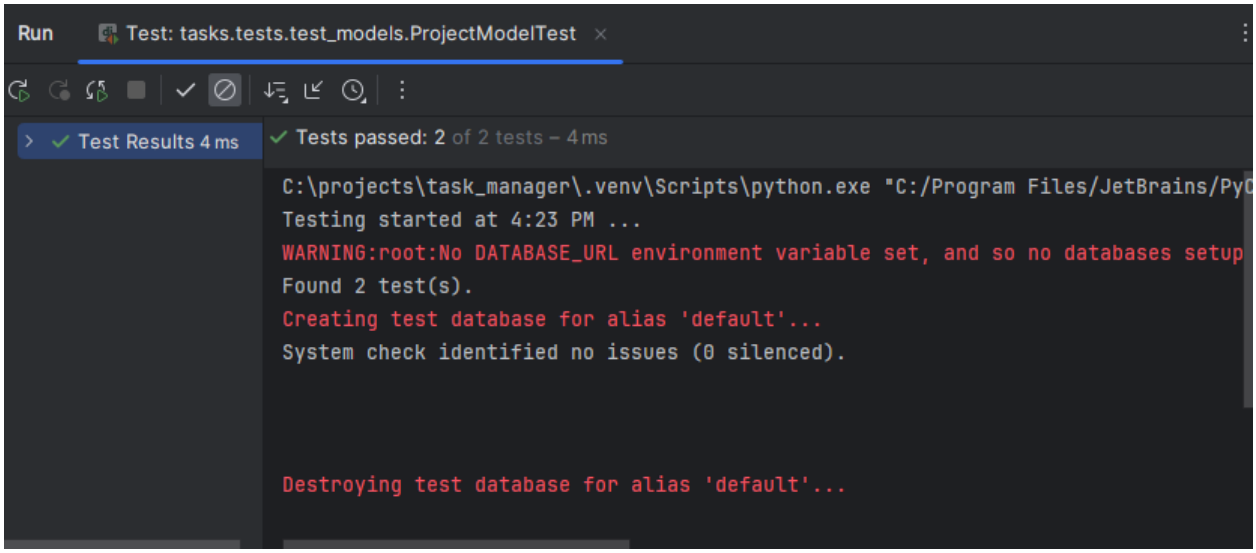

Рис 6.4 – Результат тестування Project model

Висновок: перевірка створення Python-програми з Project модулі пройшла успішно.

Висновки: при вдалому виконанні всіх 2 тестів випробування розробленого додатку вважаються успішними.

Виконавець

студент групи КІ-41

Щома Д.Р.

Додаток Г

Лістінг

views.py

```
import uuid

from django.contrib import messages
from django.contrib.auth import logout
from django.contrib.auth.decorators import login_required
from django.contrib.auth.mixins import LoginRequiredMixin
from django.contrib.auth.views import LoginView, PasswordChangeView
from django.db.models import Q

from django.shortcuts import render, redirect, get_object_or_404
from django.urls import reverse_lazy
from django.views import generic, View
from django.http import HttpResponseForbidden

from tasks.forms import (
    RegistrationForm,
    LoginForm,
    ProjectForm,
    UserPasswordChangeForm,
    JoinProjectForm,
    ChatMessageForm,
    ProjectSearchForm,
    TaskForm,
    CommentForm,
    ProjectTaskSearchForm,
    ProfileForm
)
from tasks.models import (
    Task,
    Project,
    ChatMessage,
    TaskComment,
    Worker
)


class IndexView(generic.View):
    template_name = "tasks/index.html"

    def get(self, request, *args, **kwargs):
        if request.user.is_authenticated:
            user_projects = Project.objects.filter(
                Q(creator=request.user) | Q(assignees=request.user)
            ).distinct()

            project_ids = set(user_projects.values_list('id', flat=True))
            num_projects = len(project_ids)
            num_tasks = Task.objects.filter(creator=request.user).count()

            context = {
                "num_projects": num_projects,
                "num_tasks": num_tasks
            }

            return render(request, self.template_name, context)
```

```
        return render(request, self.template_name)


class UserLoginView(LoginView):
    template_name = "accounts/login.html"
    form_class = LoginForm


class LogoutView(View):
    @staticmethod
    def get(request, *args, **kwargs):
        logout(request)

        return redirect("/")


class UserRegistrationView(generic.CreateView):
    template_name = "accounts/signup.html"
    form_class = RegistrationForm
    success_url = reverse_lazy("tasks:login")


class UserPasswordChangeView(PasswordChangeView):
    form_class = UserPasswordChangeForm
    success_url = reverse_lazy("tasks:login")
    template_name = "accounts/change_password.html"


class TaskListView(LoginRequiredMixin, generic.ListView):
    model = Task
    template_name = "tasks/task_list.html"
    context_object_name = "task_list"

    def get_queryset(self):
        project = get_object_or_404(Project, id=self.kwargs["pk"])
        queryset = super().get_queryset().filter(project=project,
creator=self.request.user)

        return queryset

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        project = get_object_or_404(Project, id=self.kwargs["pk"])
        context["project"] = project
        context["show_tabs"] = True

        return context


class ProjectTaskListView(LoginRequiredMixin, generic.ListView):
    model = Task
    template_name = "tasks/project_task_list.html"
    context_object_name = "project_tasks"
    paginate_by = 10

    def get_queryset(self):
        project = get_object_or_404(Project, id=self.kwargs["pk"])
        queryset = super().get_queryset().filter(project=project)
        search_query = self.request.GET.get("title", None)
        if search_query:
            queryset = queryset.filter(name__icontains=search_query)

        return queryset
```

```
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        project = get_object_or_404(Project, id=self.kwargs["pk"])
        context["project"] = project
        context["search_form"] = ProjectTaskSearchForm()
        context["show_tabs"] = True
        context["show_search"] = True

        return context

    def dispatch(self, request, *args, **kwargs):
        project = get_object_or_404(Project, id=self.kwargs["pk"])
        if request.user in project.assignees.all() or request.user ==
project.creator:
            return super().dispatch(request, *args, **kwargs)

        return HttpResponseForbidden("You do not have permission to view this
page.")


class TaskDetailView(LoginRequiredMixin, generic.View):
    @staticmethod
    def get(request, pk, task_pk):
        project = get_object_or_404(Project, pk=pk)
        task = get_object_or_404(Task, id=task_pk)
        comments = TaskComment.objects.filter(task=task)
        if request.user not in project.assignees.all():
            return HttpResponseForbidden("You do not have permission to this
page.")
        form = CommentForm()
        context = {
            "form": form,
            "task": task,
            "project": project,
            "show_tabs": True,
            "comments": comments
        }

        return render(request, "tasks/task_detail.html", context)

    @staticmethod
    def post(request, pk, task_pk):
        project = get_object_or_404(Project, pk=pk)
        task = get_object_or_404(Task, id=task_pk)
        comments = TaskComment.objects.filter(task=task)
        if request.user not in project.assignees.all():
            return HttpResponseForbidden("You do not have permission to this
page.")
        form = CommentForm(request.POST)
        if form.is_valid():
            comment = form.save(commit=False)
            comment.sender = request.user
            comment.task = task
            comment.save()
        else:
            form = CommentForm()
        context = {
            "form": form,
            "task": task,
            "project": project,
            "show_tabs": True,
            "comments": comments
```

```
        }

        return render(request, "tasks/task_detail.html", context)


class TaskCreateView(LoginRequiredMixin, generic.CreateView):
    model = Task
    template_name = "tasks/task_form.html"
    form_class = TaskForm
    success_url = reverse_lazy("tasks:task-list")

    def form_valid(self, form):
        form.instance.creator_id = self.request.user.id
        form.instance.project_id = self.kwargs["pk"]
        self.success_url = reverse_lazy("tasks:task-list", kwargs={"pk":
self.kwargs["pk"]})

        return super().form_valid(form)


class TaskUpdateView(LoginRequiredMixin, generic.View):
    @staticmethod
    def get(request, pk, task_pk):
        project = get_object_or_404(Project, pk=pk)
        task = get_object_or_404(Task, id=task_pk)

        if request.user != task.creator:
            return HttpResponseForbidden("You do not have permission to this
page.")

        form = TaskForm(instance=task)

        context = {
            "form": form,
            "project": project,
            "task": task,
            "show_tabs": True
        }

        return render(request, "tasks/task_form.html", context)

    @staticmethod
    def post(request, pk, task_pk):
        project = get_object_or_404(Project, pk=pk)
        task = get_object_or_404(Task, id=task_pk)

        if request.user != task.creator:
            return HttpResponseForbidden("You do not have permission to this
page.")

        form = TaskForm(request.POST, instance=task)

        if form.is_valid():
            form.instance.creator_id = request.user.id
            form.instance.project_id = pk
            form.save()

            return redirect("tasks:task-list", pk=pk)

        context = {
            "form": form,
            "project": project,
            "task": task,
```

```
            "show_tabs": True
        }

        return render(request, "tasks/task_form.html", context)


class TaskDeleteView(LoginRequiredMixin, generic.View):
    @staticmethod
    def post(request, pk, task_pk):
        task = get_object_or_404(Task, id=task_pk)

        if request.user != task.creator:
            return HttpResponseForbidden("You do not have permission to this
page.")

        task.delete()

        return redirect("tasks:task-list", pk=pk)

    @staticmethod
    def get(request, pk, task_pk):
        project = get_object_or_404(Project, pk=pk)
        task = get_object_or_404(Task, id=task_pk)

        context = {
            "project": project,
            "task": task,
            "show_tabs": True
        }

        return render(request, "tasks/task_confirm_delete.html", context)


class ProjectListView(LoginRequiredMixin, generic.ListView):
    model = Project
    queryset = Project.objects.all()
    context_object_name = "project_list"
    template_name = "tasks/project_list.html"
    paginate_by = 3

    def get_queryset(self):
        user = self.request.user
        queryset = Project.objects.filter(Q(creator=user) |
Q(assignees=user)).distinct()

        form = ProjectSearchForm(self.request.GET)
        if form.is_valid():
            title = form.cleaned_data.get("title")
            queryset = queryset.filter(title__icontains=title)

        return queryset

    def get_context_data(self, *, object_list=None, **kwargs):
        context = super(ProjectListView, self).get_context_data(**kwargs)
        title = self.request.GET.get("title", "")
        context["search_form"] = ProjectSearchForm(
            initial={"title": title}
        )
        context["show_search"] = True

        return context
```

```
class ProjectDetailView(LoginRequiredMixin, generic.DetailView):
    model = Project
    template_name = "tasks/project_detail.html"
    context_object_name = "project"

    def dispatch(self, request, *args, **kwargs):
        if request.user == self.get_object().creator or request.user in
self.get_object().assignees.all():
            return super().dispatch(request, *args, **kwargs)

        return HttpResponseForbidden("You do not have permission to this
page")

    def get_context_data(self, **kwargs):
        context = super().get_context_data()
        context["show_tabs"] = True
        context["project"] = self.object

        return context


class ProjectCreateView(LoginRequiredMixin, generic.CreateView):
    model = Project
    template_name = "tasks/project_form.html"
    form_class = ProjectForm
    success_url = reverse_lazy("tasks:project-list")

    def form_valid(self, form):
        project = form.save(commit=False)
        project.creator = self.request.user
        project.save()
        project.assignees.add(self.request.user)

        return super().form_valid(form)


class ProjectUpdateView(LoginRequiredMixin, generic.UpdateView):
    model = Project
    form_class = ProjectForm
    success_url = reverse_lazy("tasks:project-list")

    def dispatch(self, request, *args, **kwargs):
        if self.get_object().creator != self.request.user:
            return HttpResponseForbidden("You do not have permission to this
project.")

        return super().dispatch(request, *args, **kwargs)


class ProjectDeleteView(LoginRequiredMixin, generic.DeleteView):
    model = Project
    success_url = reverse_lazy("tasks:project-list")

    def dispatch(self, request, *args, **kwargs):
        if self.get_object().creator != self.request.user:
            return HttpResponseForbidden("You do not have permission to this
project.")

        return super().dispatch(request, *args, **kwargs)


class GenerateCodeView(LoginRequiredMixin, generic.View):
    @staticmethod
```

```
    def get(request, pk):
        project = get_object_or_404(Project, id=pk)
        if request.user != project.creator:
            return HttpResponseForbidden("You do not have permission to this
project.")

        project.invitation_code = uuid.uuid4()
        project.save()
        context = {
            "project": project
        }

        return render(request, "tasks/project_invitation.html", context)


class JoinProjectView(LoginRequiredMixin, generic.View):
    @staticmethod
    def get(request):
        form = JoinProjectForm()

        return render(request, "tasks/project_join_form.html", {"form":
form})

    @staticmethod
    def post(request):
        form = JoinProjectForm(request.POST)
        if form.is_valid():
            invitation_code = form.cleaned_data.get("invitation_code")
            try:
                project =
Project.objects.get(invitation_code=invitation_code)
            except Project.DoesNotExist:
                messages.warning(request, "Invalid invitation code.")

                return redirect("tasks:project-join")
            if project.assignees.filter(id=request.user.id).exists():
                messages.warning(request, "You are already a member of this
project.")

                return redirect("tasks:project-join")
            project.assignees.add(request.user)

            return redirect(project.get_absolute_url())
        else:
            error_message = form.errors.get("invitation_code")
            if error_message:
                messages.warning(request, error_message)

            return render(request, "tasks/project_join_form.html", {"form":
form})


class ChatMessagesView(LoginRequiredMixin, generic.View):
    @staticmethod
    def get(request, pk):
        project = get_object_or_404(Project, id=pk)

        if request.user == project.creator or request.user in
project.assignees.all():
            messages_connected = ChatMessage.objects.filter(project=project)
            context = {
                "project": project,
                "messages": messages_connected,
```

```
                "message_form": ChatMessageForm(),
                "show_tabs": True
            }

            return render(request, "tasks/chat_messages.html", context)

        return HttpResponseForbidden("You do not have permission to view this
project.")

    @staticmethod
    def post(request, pk):
        project = get_object_or_404(Project, id=pk)

        if request.user == project.creator or request.user in
project.assignees.all():
            message_form = ChatMessageForm(request.POST)
            if message_form.is_valid():
                new_message = message_form.save(commit=False)
                new_message.project = project
                new_message.sender = request.user
                new_message.save()

                return redirect("tasks:project-chat", pk=pk)
            else:
                messages_connected =
ChatMessage.objects.filter(project=project)
                context = {
                    "project": project,
                    "messages": messages_connected,
                    "message_form": message_form,
                    "show_tabs": True
                }

                return render(request, "tasks/chat_messages.html", context)

        return HttpResponseForbidden("You do not have permission to view this
project.")


class ProfileDetailView(LoginRequiredMixin, generic.DetailView):
    worker = Worker
    success_url = reverse_lazy("tasks:profile-detail")
    template_name = "tasks/profile_detail.html"

    def get_object(self, queryset=None):
        return self.request.user


class ProfileUpdateView(LoginRequiredMixin, generic.UpdateView):
    worker = Worker
    success_url = reverse_lazy("tasks:profile-detail")
    template_name = "tasks/profile_form.html"
    form_class = ProfileForm

    def get_object(self, queryset=None):
        return self.request.user

    def form_valid(self, form):
        profile = form.save(commit=False)
        profile.user = self.request.user
        new_profile_image = self.request.FILES.get("profile_image")
        if new_profile_image:
            profile.profile_image = new_profile_image
```

```
        profile.save()

        return super().form_valid(form)
```