Паспорт проекту

**Онлайн-платформа для управління проєктами**

# 1. Опис продукту проєкту: якості, можливості, переваги

## Короткий опис функціональності продукту

Онлайн-платформа для управління проєктами призначена для ефективного управління завданнями, комунікацією між членами команди та контролю за виконанням проектів. Основні функції платформи включають:

* **Створення та управління проєктами**: користувачі можуть ініціювати нові проєкти, призначати завдання, встановлювати дедлайни та відстежувати їх виконання.
* **Інструменти планування**: можливість створювати план завдань, визначати пріоритети, встановлювати залежності між завданнями.
* **Моніторинг прогресу**: візуалізація виконання завдань через діаграми Ганта та канбан-дошки, що дозволяє бачити загальний стан проєкту в режимі реального часу.
* **Спільна робота та комунікація**: інтеграція з чатом, можливість залишати коментарі до завдань і проектів, обговорення в режимі реального часу.
* **Аналіз і звітність**: наявність функцій для створення звітів і аналітики про виконання завдань, продуктивність команди, зведені звіти за проектами.

## Особливості, які виділяють продукт серед конкурентів

* **Інтуїтивний та адаптивний інтерфейс**: Платформа розроблена для того, щоб бути максимально зручною та доступною для користувачів різного рівня. Дизайн адаптується під різні пристрої (ПК, планшети, мобільні телефони), що забезпечує зручність доступу в будь-якому середовищі.
* **Безпека та конфіденційність**: Використовуються передові технології для захисту даних, такі як HTTPS-з'єднання та шифрування паролів. Платформа відповідає міжнародним стандартам безпеки, таким як GDPR, що гарантує збереження конфіденційності користувачів.
* **Масштабованість**: Платформа спроектована таким чином, щоб підтримувати як малі, так і великі команди. Вона може масштабуватися, обробляючи великі об'єми даних і користувачів, що дозволяє легко адаптуватися до росту компаній.
* **Інтерактивні аналітичні інструменти**: На відміну від багатьох інших платформ, наша система пропонує гнучкі інструменти для візуалізації продуктивності та прогресу, що дозволяє користувачам легко оцінювати ефективність роботи.
* **Гнучкість налаштувань**: Платформа дозволяє користувачам налаштовувати робочі процеси під свої потреби, що важливо для різних типів проєктів — від стартапів до великих корпорацій.

## Технічні характеристики

* **Платформи**: Онлайн-платформа, доступна через веб-браузер (Chrome, Firefox, Safari, Edge). Мобільна версія також розробляється для Android та iOS.
* **Мови програмування**:
* **Frontend**: React.js для швидкої та ефективної побудови інтерфейсу, CSS для стилізації та адаптивного дизайну.
* **Backend**: Node.js із використанням Express.js для побудови серверної частини.
* **База даних**: MongoDB, що забезпечує високу гнучкість у зберіганні та обробці даних.
* **Безпека**: Шифрування AES-256 для захисту даних користувачів і використання двофакторної аутентифікації для доступу до аккаунтів.

## Ключові переваги для цільової аудиторії

* **Підвищення ефективності**: Інтуїтивно зрозумілий інтерфейс та автоматизація робочих процесів дозволяють заощаджувати час і знижувати кількість помилок.
* **Економія часу**: Завдяки інтеграціям з іншими сервісами, такими як Slack і Google Calendar, платформа дозволяє автоматично синхронізувати завдання і терміни, що забезпечує більшу ефективність.
* **Зниження витрат**: Використання централізованої платформи для всіх етапів управління проектами дозволяє знижувати витрати на зовнішні інструменти і забезпечує зручність у роботі.
* **Покращення командної взаємодії**: Завдяки вбудованим інструментам для комунікації та спільної роботи, члени команди можуть оперативно обмінюватися інформацією, що зменшує кількість непорозумінь і підвищує продуктивність.

Цей продукт стане ефективним інструментом для малих, середніх та великих підприємств, які прагнуть автоматизувати процеси управління проектами та підвищити командну взаємодію.

# 2. Орієнтована тривалість проєкту та можливі етапи

## Загальна тривалість проєкту: 4 місяці

Проєкт передбачає створення онлайн-платформи для управління проєктами в умовах обмеженого часу. Для забезпечення своєчасного виконання і запуску продукту, проєкт буде поділений на п’ять ключових етапів. Кожен етап матиме конкретні контрольні точки та віхи, що дозволить зберігати контроль над процесом та виконанням задач.

### **1. Аналіз і планування (1 місяць)**

**Тривалість:** 1 місяць (1 вересня – 30 вересня)

Цей етап передбачає збори вимог, вивчення потреб цільової аудиторії та аналіз ринку для виявлення конкурентних переваг продукту. Основні завдання:

* **Збір вимог:** виявлення специфікацій для функціоналу продукту.
* **Аналіз ринку та конкурентів:** оцінка подібних рішень на ринку для виділення унікальних функцій продукту.
* **Підготовка технічного завдання:** визначення технологій, вибір стеку програмування, підготовка попередньої архітектури.

**Контрольні точки:**

* Затвердження вимог та технічного завдання.
* Підготовка документації для проектування.

### **2. Проектування (1 місяць)**

**Тривалість:** 1 місяць (1 жовтня – 31 жовтня)

На цьому етапі буде розроблено технічне проектування, а також дизайн інтерфейсу користувача. Основні завдання:

* **Розробка архітектури системи:** визначення клієнт-серверної архітектури, проектування взаємодії між модулями.
* **Розробка дизайну інтерфейсу:** створення прототипів інтерфейсу та проведення UX/UI тестувань.
* **Вибір інструментів для розробки:** підготовка середовища для розробки, налаштування серверної частини.

**Контрольні точки:**

* Затвердження архітектури системи та інтерфейсу.
* Підготовка технічної документації для розробки.

### **3. Розробка (1,5 місяця)**

**Тривалість:** 1,5 місяця (1 листопада – 15 грудня)

Цей етап є основним для створення функціоналу продукту. Він включає в себе розробку серверної частини, фронтенду та інтеграцію API. Основні завдання:

* **Розробка базового функціоналу:** створення основних модулів платформи, таких як реєстрація користувачів, управління проектами та завданнями.
* **Інтеграція API:** інтеграція з популярними інструментами (Slack, Zoom, Microsoft Teams).
* **Розробка адміністративної панелі:** інтерфейс для адміністраторів для управління користувачами та проектами.

**Контрольні точки:**

* Завершення розробки базових функцій.
* Інтеграція з зовнішніми сервісами.
* Підготовка попередньої версії для тестування.

### **4. Тестування (1 місяць)**

**Тривалість:** 1 місяць (16 грудня – 15 січня)

На етапі тестування буде проведено перевірку функціональності платформи, виявлення помилок та оптимізація продуктивності. Завдання:

* **Функціональне тестування:** перевірка всіх модулів та сценаріїв роботи, тестування всіх інтеграцій та взаємодій.
* **Регресійне тестування:** після кожної ітерації тестування перевірка всіх функцій на наявність помилок у вже реалізованих компонентах.
* **Тестування безпеки:** перевірка на наявність уразливостей, захист від SQL-ін’єкцій, XSS та CSRF атак.

**Контрольні точки:**

* Завершення основного тестування функціональності.
* Завершення тестування на безпеку та стабільність роботи.
* Підготовка до впровадження та запуску.

### **5. Впровадження (0,5 місяця)**

**Тривалість:** 0,5 місяця (16 січня – 31 січня)

Цей етап включає в себе запуск продукту в реальному середовищі, навчання користувачів та технічну підтримку. Завдання:

* **Переведення на продакшн середовище:** розгортання продукту на сервері та його підключення до бази даних.
* **Навчання користувачів:** проведення тренінгів для команди та підтримки користувачів.
* **Запуск продукту:** публічний запуск платформи.

**Контрольні точки:**

* Запуск у реальному середовищі.
* Офіційний публічний запуск продукту.
* Підготовка до подальшої підтримки та вдосконалення.

## Оцінка термінів та контрольних точок

Загальна тривалість: **4 місяці**.

Ключові віхи: завершення кожного етапу є контрольними точками для подальшого руху проєкту.

Після завершення кожного етапу проводиться аналіз і уточнення планів для наступних етапів.

# 3. Необхідні ресурси проєкту

## 3.1 Технічне обладнання та інфраструктура

Для успішної реалізації проєкту необхідно забезпечити стабільну і надійну інфраструктуру. Включаючи сервери для хостингу, а також інструменти для забезпечення високої доступності та ефективної роботи платформи.

* **Сервери для хостингу:** Платформа буде розгорнута на хмарних сервісах, таких як **DigitalOcean** або **AWS**, що забезпечать потрібний рівень масштабованості та надійності.
* **Бази даних:** Використання **MongoDB** для зберігання даних користувачів, проектів і завдань. Цей вибір обумовлений необхідністю підтримки гнучких даних без строгих вимог до схеми бази даних.

## 3.2 Інструменти для розробки

Розробка платформи буде здійснюватися з використанням сучасних інструментів і технологій, що дозволять забезпечити швидкість і якість роботи.

* **IDE та редактори коду:**
* **Visual Studio Code** для фронтенд- та бекенд-розробки, оскільки цей редактор підтримує роботу з різними мовами програмування і має великий набір плагінів.
* **WebStorm** або інші IDE для розробки на JavaScript та для роботи з Node.js.
* **Фреймворки та технології для розробки:**
* **Frontend:** Використовуватиметься **React** для створення інтерфейсу користувача, що забезпечить гнучкість і адаптивність. Для стилізації застосовується **CSS**.
* **Backend:** Серверна частина буде реалізована з використанням **Node.js** та **Express.js**, що дозволить зручно організувати обробку запитів і обслуговування API.

## 3.3 Ліцензії на програмне забезпечення

Для розробки платформи буде необхідно використовувати ліцензійне програмне забезпечення та інструменти:

* **MongoDB:** Використання MongoDB на безкоштовній основі або з придбанням ліцензії для більш масштабованих рішень.
* **Visual Studio** та інші IDE можуть вимагати ліцензій для розширених функцій, хоча основні інструменти доступні безкоштовно.

Ці інструменти забезпечать ефективну розробку та інтеграцію компонентів системи, що дозволить зберігати високу якість продукту на всіх етапах його реалізації.

# 4. Функціональний обсяг

## 4.1 Перелік основних функцій

Онлайн-платформа для управління проєктами буде включати набір основних функціональних можливостей, які дозволяють користувачам ефективно організовувати та керувати проєктами, завданнями і командною роботою:

* **Реєстрація та авторизація користувачів:** Користувачі матимуть змогу реєструватися на платформі та входити через безпечний процес аутентифікації з використанням **JSON Web Token (JWT)** для безпечного доступу.
* **Управління проектами:** Користувачі зможуть створювати нові проєкти, редагувати та видаляти існуючі, а також організовувати завдання в межах кожного проекту.
* **Створення і управління завданнями:** Кожен проєкт включатиме можливість додавати, редагувати та видаляти завдання. Завдання можуть містити опис, дедлайни, пріоритети, та статуси.
* **Візуалізація прогресу:** Інтерфейс користувача включатиме різні способи візуалізації прогресу проєктів та завдань, наприклад, **канбан-дошка** та **діаграма Ганта**.
* **Комунікація між користувачами:** Буде реалізовано інтегровану систему чату для комунікації між членами команди. Користувачі зможуть обговорювати завдання, залишати коментарі та отримувати сповіщення.
* **Нагадування та сповіщення:** Сповіщення про зміни в проєктах, нові завдання, а також наближення дедлайнів будуть відправлятись користувачам через електронну пошту та вбудовану систему сповіщень.
* **Інтеграція з календарями:** Користувачі зможуть синхронізувати свої завдання з календарем (наприклад, Google Calendar), що дозволить автоматично створювати події на основі термінів виконання завдань.
* **Звіти та аналітика:** Платформа надасть користувачам можливість отримувати детальні звіти про стан завдань, продуктивність команди, витрачений час та інші параметри через інтерактивні панелі аналітики.

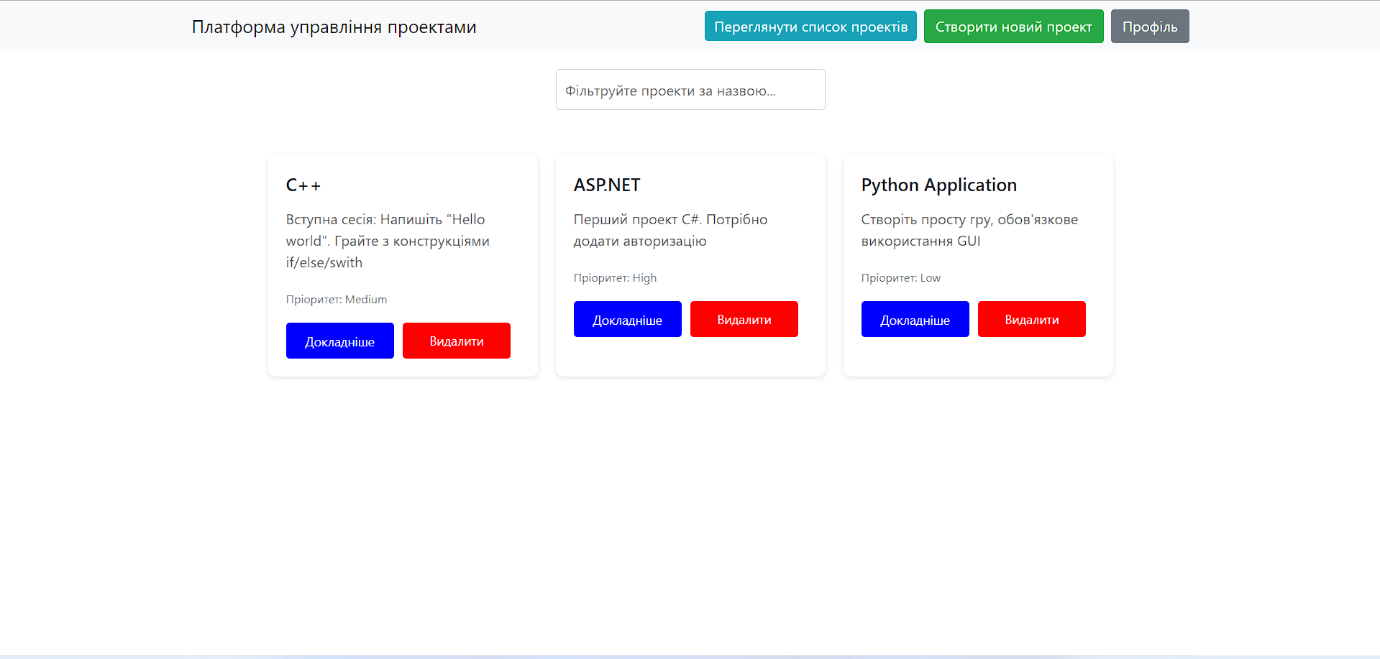


Рис. 4.1 Головна сторінка

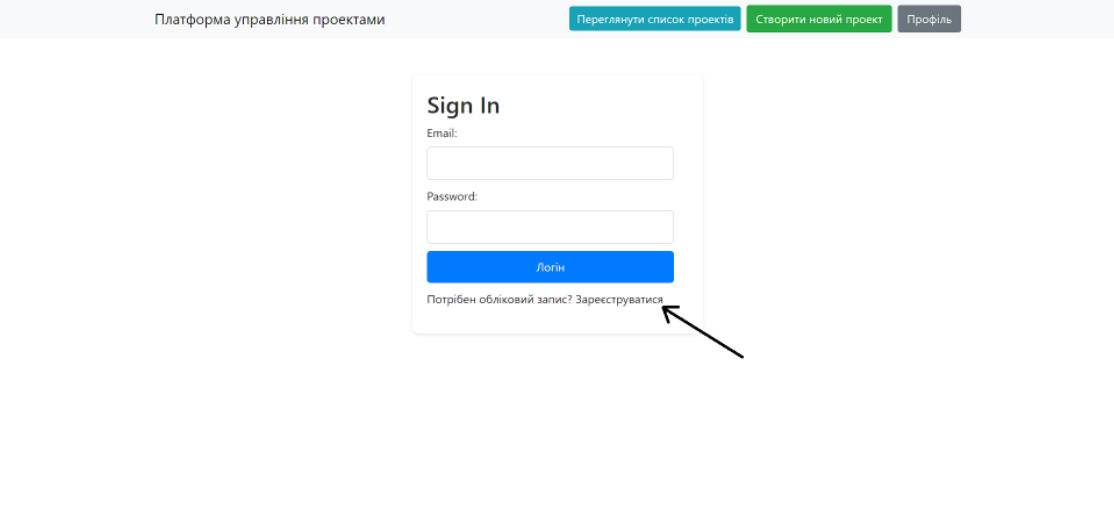


Рис. 4.2 Сторінка входу

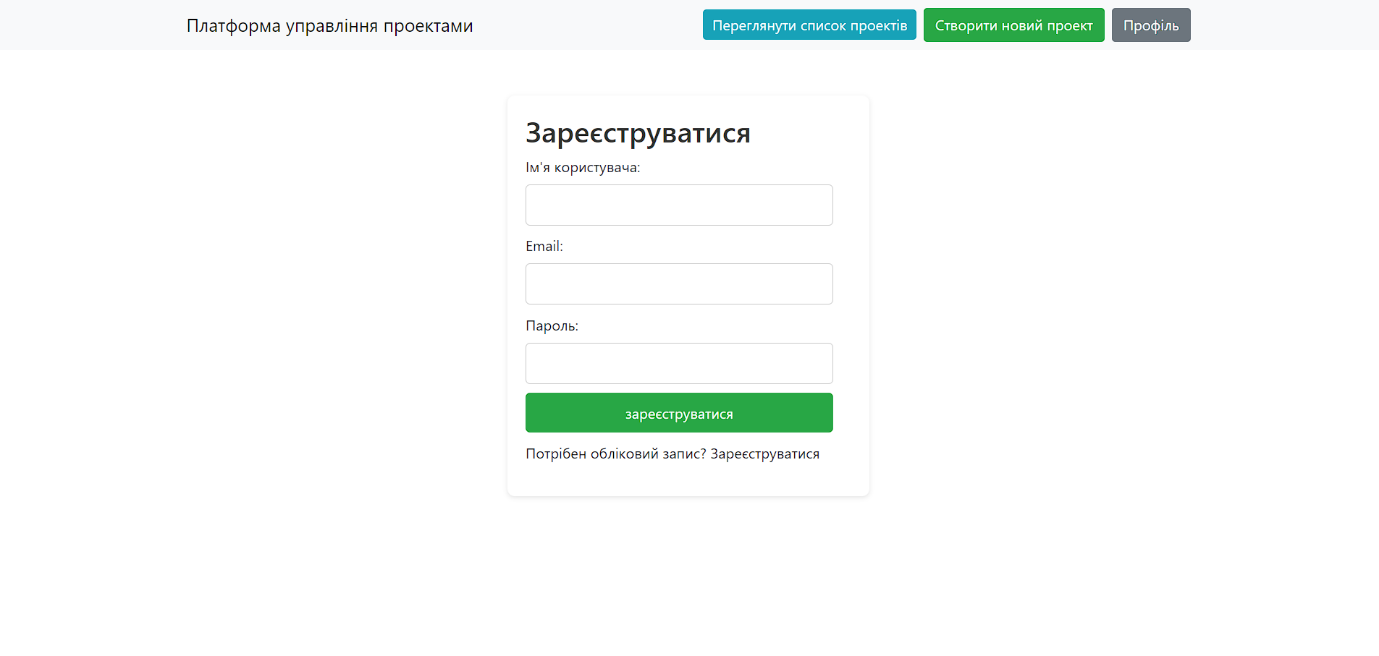


Рис. 4.3 Сторінка реєстрації

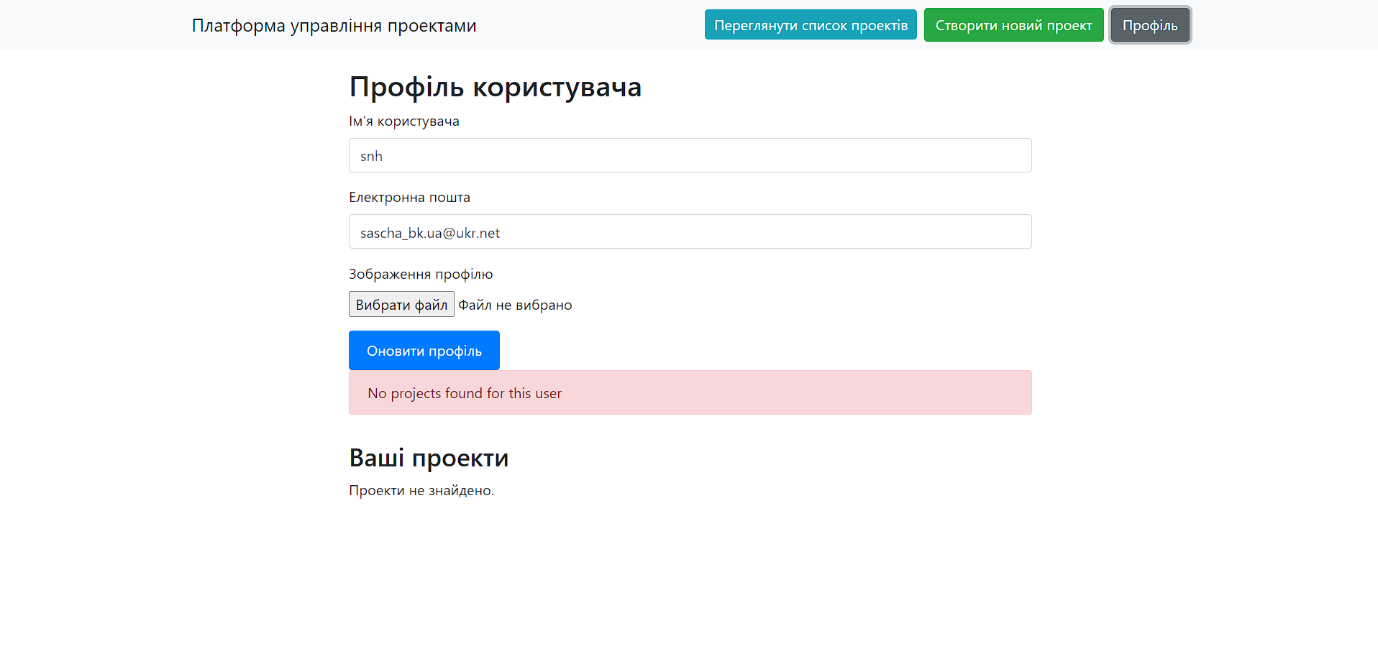


Рис. 4.4 Сторінка профілю користувача

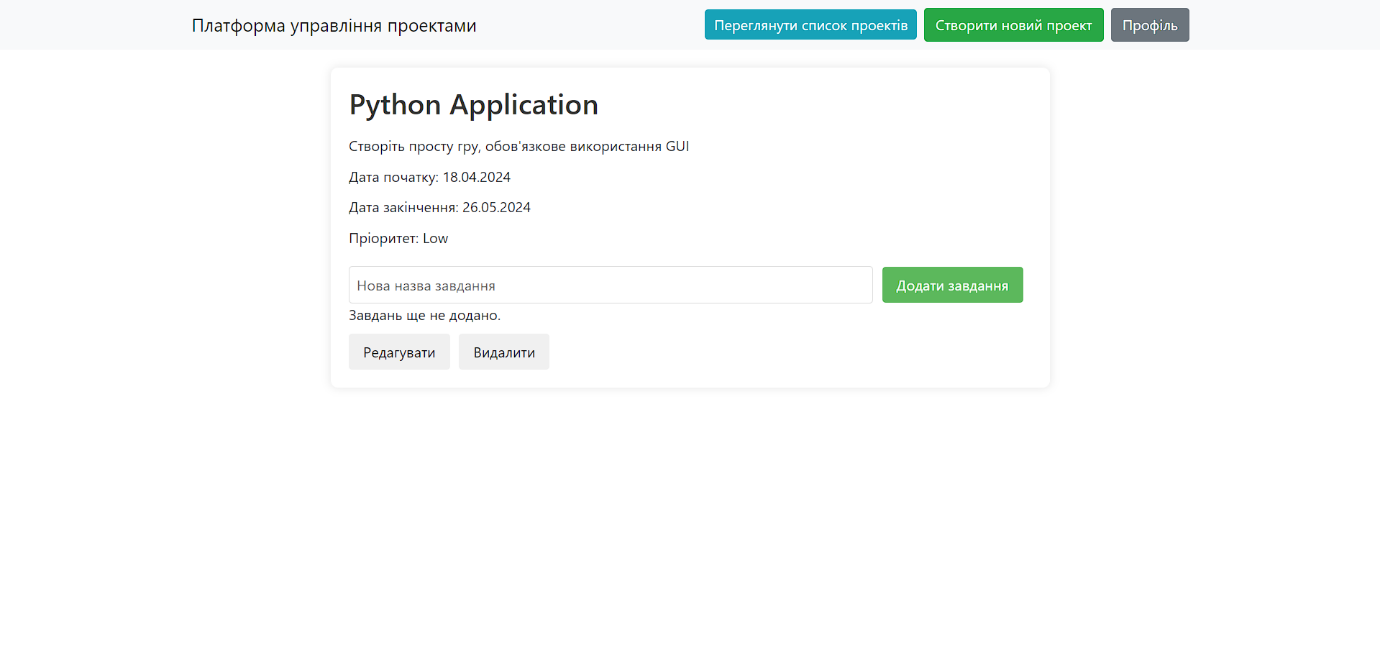


Рис. 4.5 Сторінка деталей проєкту

### 4.1.1 Створення та управління проєктами

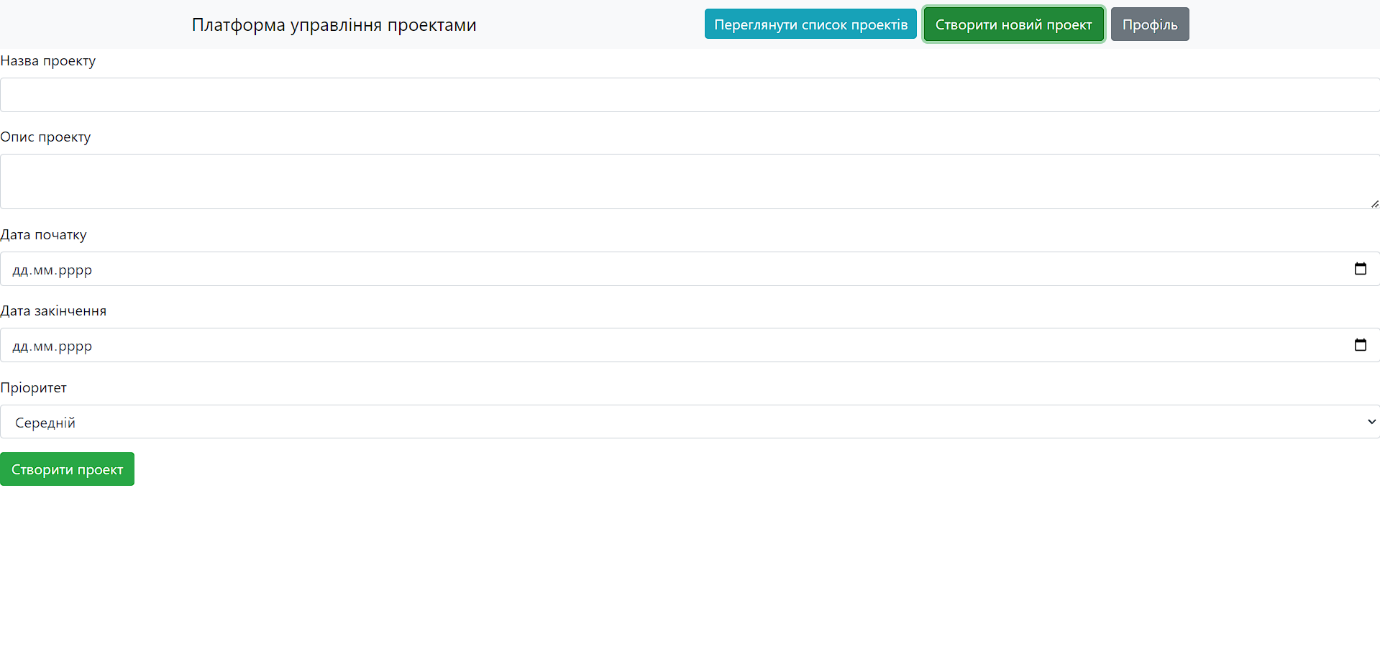


Рис. 4.6

Створення проєктів:

* **Форма створення:** Користувачі можуть створювати нові проєкти за допомогою форми, де потрібно вказати назву, опис, дати початку та завершення, а також пріоритет проєкту.
* **Кнопка створення:** Після заповнення форми користувач натискає кнопку "Створити проєкт", що додає новий проєкт до списку.

Управління проєктами:

* **Редагування:** Користувачі можуть редагувати існуючі проєкти, змінюючи їх назву, опис, дати та пріоритет.
* **Видалення**: Проєкти можна видаляти, якщо вони більше не потрібні.

Створення та управління проєктами дозволяє користувачам тримати всі свої проєкти організованими та актуальними, забезпечуючи ефективне управління їх робочими процесами.

### 4.1.2 Додавання та керування завданнями

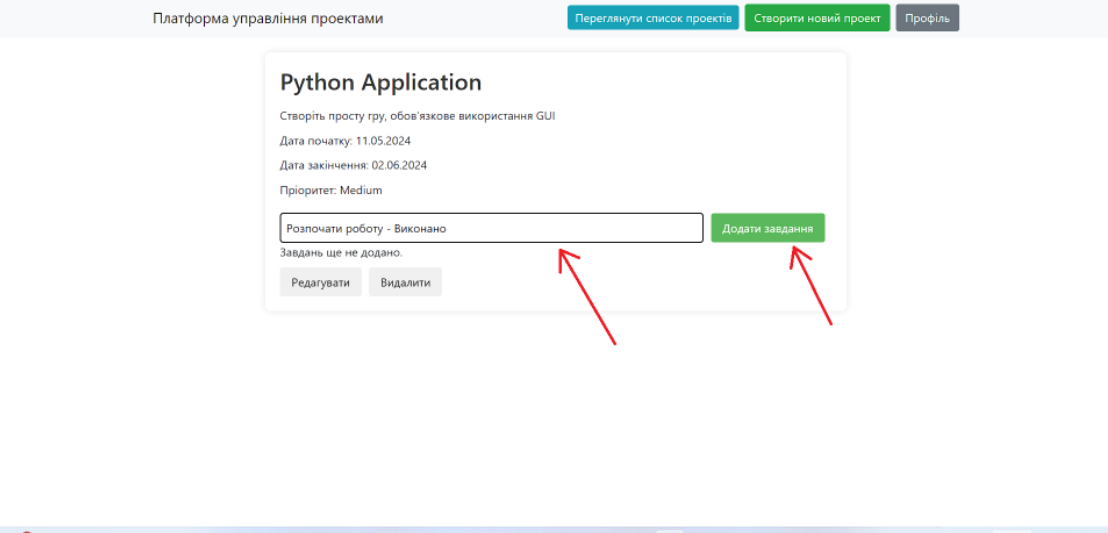


Рис. 4.7

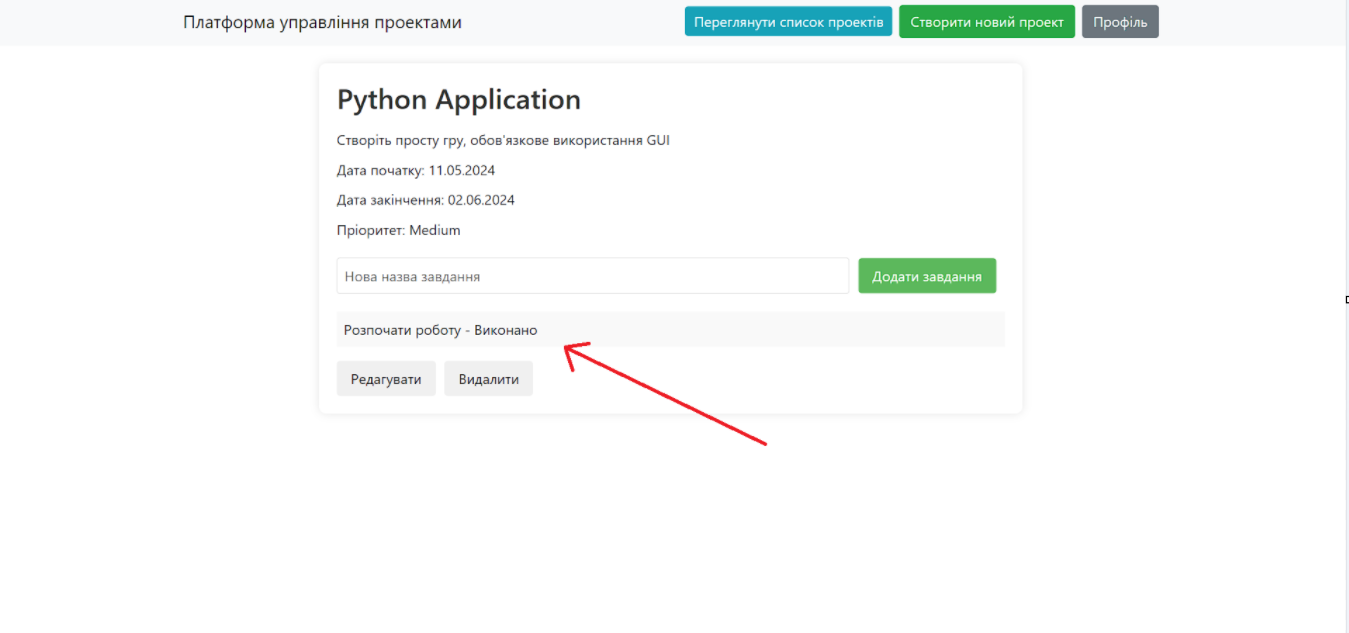


Рис. 4.8

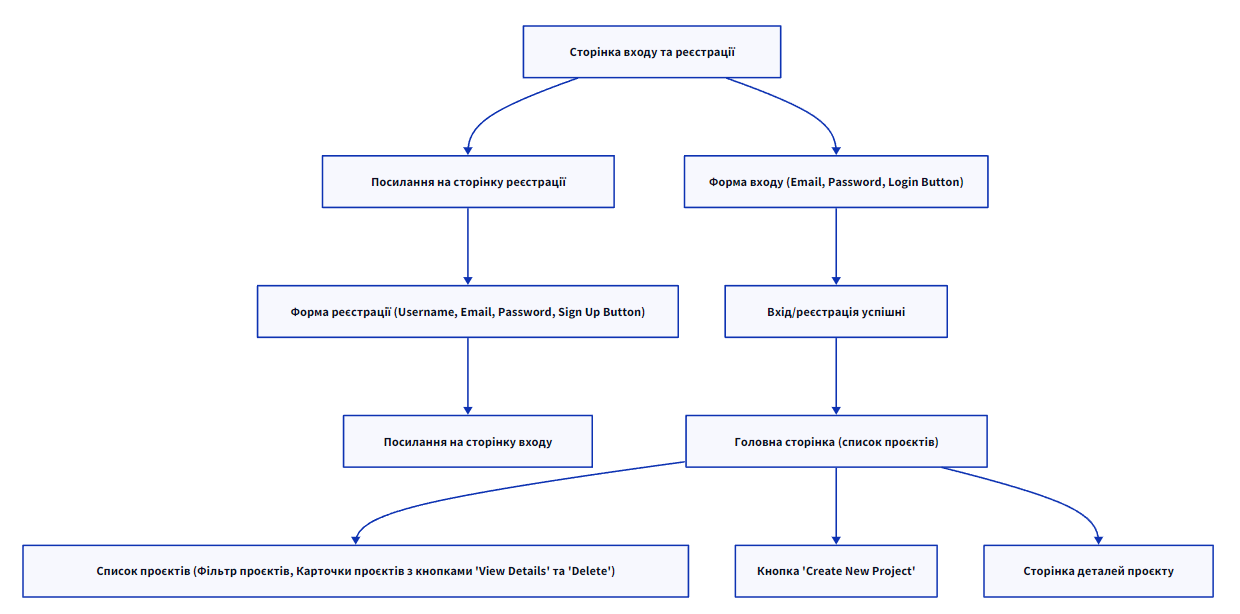
Додавання завдань:

* **Форма додавання завдань:** Користувачі можуть додавати завдання до своїх проєктів за допомогою форми, де потрібно вказати назву завдання.
* **Кнопка додавання:** Після заповнення форми користувач натискає кнопку "Додати завдання", що додає нове завдання до проєкту.

Керування завданнями:

* **Список завдань:** Кожен проєкт має список завдань, де відображаються всі додані завдання.
* **Редагування та видалення:** Користувачі можуть редагувати та видаляти завдання, підтримуючи актуальність та організацію проєктів.

Додавання та керування завданнями дозволяє користувачам детально планувати свої проєкти та ефективно контролювати виконання кожного етапу



Мал.2.9

## 4.2 Мінімальні функції для початкової версії

Мінімальний функціонал для початкової версії проєкту, який буде реалізовано на етапі першого релізу:

* **Реєстрація та авторизація користувачів**.
* **Створення та редагування проєктів**.
* **Створення, редагування та видалення завдань**.
* **Візуалізація завдань у вигляді канбан-дошки**.
* **Нагадування та сповіщення про важливі оновлення та дедлайни**.

Ці функції забезпечать базовий функціонал для ефективного управління проектами на платформі.

## 4.3 Можливості для подальшого масштабування та розширення функціоналу

Для подальшого розвитку платформи передбачено низку додаткових функцій та можливостей для розширення:

* **Масштабовані звіти та аналітика:** Можливість генерації детальних звітів по кожному проекту, аналіз витрат часу та прогресу команди.
* **Розширене управління правами доступу:** Платформа дозволить вводити різні ролі користувачів з різними рівнями доступу до функцій (наприклад, адміністраторам, керівникам проектів та звичайним користувачам).
* **Інтеграція з іншими сервісами:** Можливість додаткових інтеграцій з іншими популярними інструментами для командної роботи, такими як **Slack**, **Microsoft Teams**, **Zoom**.
* **Інтерфейс для мобільних пристроїв:** Розробка адаптивного мобільного додатку для роботи з платформою з будь-якого пристрою.
* **Вдосконалення інтерфейсу користувача:** Впровадження нових методів візуалізації даних, таких як графіки, статистика по завданнях тощо.

Ці розширення дозволять адаптувати платформу до потреб великих команд та організацій і підвищити її функціональність у міру росту користувачів.

# 5. Організаційна структура проєкту і функціональні ролі

## 5.1 Ключові ролі та обов'язки

Для успішної реалізації проекту створення онлайн-платформи для управління проектами, визначено наступні ключові ролі:

**Project Manager (PM)**

**Обов'язки:**

* Керівництво та координація всіх етапів проекту.
* Визначення цілей та вимог проекту, а також контроль за їх виконанням.
* Планування та контроль ресурсів, термінів та бюджету.
* Управління ризиками та їх мінімізація.
* Забезпечення комунікації між усіма учасниками проекту та стейкхолдерами.
* Ведення документації проекту та звітність.

**Team Lead**

**Обов'язки:**

* Керівництво командою розробників, надаючи технічні консультації та підтримку.
* Визначення архітектури системи та технологій, які будуть використовуватися.
* Розподіл завдань серед розробників та контроль їх виконання.
* Проведення код-рев'ю для забезпечення якості коду.
* Взаємодія з Project Manager для вирішення технічних питань та коригування термінів виконання завдань.

**Frontend Developer**

**Обов'язки:**

* Розробка та оптимізація користувацького інтерфейсу платформи.
* Впровадження адаптивного дизайну для різних пристроїв (ПК, мобільні пристрої).
* Використання технологій React та CSS для створення інтуїтивно зрозумілого інтерфейсу.
* Тестування інтерфейсу для забезпечення безперебійної роботи на різних платформах.

**Backend Developer**

**Обов'язки:**

* Розробка серверної частини платформи, включаючи API та інтеграцію з базою даних.
* Вибір та налаштування серверного середовища, оптимізація роботи серверів.
* Забезпечення безпеки даних, включаючи шифрування та захист від атак.
* Тестування та оптимізація продуктивності серверної частини.

**QA Engineer (Тестувальник)**

**Обов'язки:**

* Розробка та виконання тестів для перевірки функціональності платформи.
* Проведення функціональних та регресійних тестів після кожного етапу розробки.
* Виявлення дефектів та передача їх розробникам для виправлення.
* Підготовка звітів про тестування та контроль за їх усуненням.

**Business Analyst**

**Обов'язки:**

* Збір та аналіз вимог від стейкхолдерів і кінцевих користувачів.
* Підготовка технічних завдань для розробників.
* Аналіз ринку та конкурентів для виявлення можливих переваг продукту.
* Співпраця з Project Manager для коригування функціоналу на основі отриманих даних.

## 5.2 Опис організаційної структури

Організаційна структура команди є функціональною, де кожен учасник має чітко визначену роль і зону відповідальності, але всі працюють у тісній співпраці для досягнення спільної мети.

* **Project Manager** є центральною фігурою в команді, яка координує роботу всіх учасників. Він взаємодіє з усіма розробниками, тестувальниками та бізнес-аналітиками для контролю прогресу та вирішення організаційних питань.
* **Team Lead** зосереджений на технічних аспектах проекту, контролюючи розробку програмного забезпечення, вибір технологій та технічних рішень.
* **Frontend Developer** та **Backend Developer** працюють під керівництвом Team Lead, взаємодіють для забезпечення функціональності платформи, зокрема для забезпечення гармонійної роботи клієнтської та серверної частин.
* **QA Engineer** займається забезпеченням якості продукту, виконує тестування на всіх етапах і взаємодіє з усіма членами команди для виявлення та виправлення помилок.
* **Business Analyst** проводить постійний аналіз вимог і адаптує їх до потреб проекту, працюючи разом із командою для коригування функціоналу згідно з вимогами кінцевих користувачів.

## 5.3 Розподіл функцій та зон відповідальності

**Project Manager**:

* Планування та контроль графіків.
* Оцінка ризиків та управління ними.
* Звітування перед стейкхолдерами.

**Team Lead**:

* Керівництво розробниками.
* Технічне консультування та розробка архітектури.
* Проведення код-рев'ю.

**Frontend Developer**:

* Розробка інтерфейсу користувача.
* Реалізація UX/UI дизайну.
* Тестування клієнтської частини.

**Backend Developer**:

* Розробка серверної частини.
* Налаштування бази даних.
* Забезпечення безпеки платформи.

**QA Engineer**:

* Проведення тестування.
* Виявлення помилок та недоліків.
* Забезпечення стабільної роботи платформи.

**Business Analyst**:

* Збір вимог від користувачів.
* Аналіз конкурентів.
* Формулювання технічних завдань для розробників.

# 6. Концептуальне проектування

## 6.1 Загальна архітектура проєкту

Концептуальне проектування передбачає створення детального уявлення про основні компоненти продукту, його архітектуру та інтерфейс. Для розробки онлайн-платформи для управління проектами передбачається використання двох основних частин: **серверної частини** та **клієнтської частини**.

* **Серверна частина:** Серверна частина продукту відповідатиме за обробку запитів від клієнтської частини, управління даними та логіку взаємодії з базою даних. Для цього передбачається використання мови програмування **Python** або **Node.js**, які добре підходять для швидкої розробки API та серверних рішень.
* Сервер оброблятиме:
* Аутентифікацію користувачів.
* Створення, редагування та видалення проектів, завдань.
* Управління правами доступу.
* Інтерфейс для обміну даними між користувачами.
* Для забезпечення високої продуктивності серверної частини буде використано **Docker** для контейнеризації додатку та можливості масштабування.
* **Клієнтська частина:** Клієнтська частина платформи повинна бути доступною як через веб-браузери, так і через мобільні додатки (у майбутньому). Для цього використовується **React** для створення веб-інтерфейсу з гарним UX/UI дизайном. Веб-інтерфейс передбачає зручне управління проектами, завданнями та чітке відображення результатів роботи.
* Клієнтська частина включатиме:
* Інтерфейс для створення проектів та завдань.
* Візуалізація статусу завдань, графіків і діаграм.
* Функції комунікації між користувачами (чат, повідомлення).

## 6.2 Ключові компоненти системи

* **База даних:** Для зберігання даних проектів, завдань, користувачів та іншої інформації буде використано **реляційну базу даних** (наприклад, **PostgreSQL** або **MySQL**). Вибір залежатиме від вимог щодо обробки великих обсягів даних і необхідності у складних запитах.
* База даних повинна включати:
* Таблиці для користувачів, проектів, завдань.
* Зв'язки між користувачами та проектами (ролі, права доступу).
* Логи подій для відстеження змін в системі.
* **API та інтерфейси взаємодії:** Для забезпечення безпечної та зручної взаємодії між клієнтською та серверною частинами буде розроблено **REST API** або **GraphQL** для обміну даними між фронтендом і бекендом. API забезпечить:
* Аутентифікацію та авторизацію користувачів.
* Операції CRUD (Create, Read, Update, Delete) для управління проектами та завданнями.
* Механізми обміну даними в реальному часі, наприклад, за допомогою **WebSockets** для чатів чи сповіщень.
* **Інтерфейс користувача (UI):** Інтерфейс платформи буде створений із урахуванням простоти і зручності використання. Дизайн буде сучасним і мінімалістичним, з орієнтацією на швидкість роботи. Веб-інтерфейс включатиме:
* Панель навігації з доступом до основних функцій.
* Створення та перегляд проектів/завдань.
* Візуальні елементи для відображення прогресу (діаграми, списки).

## 6.3 Технічні рішення для масштабованості та безпеки

* **Масштабованість:** Платформа повинна бути спроектована так, щоб підтримувати можливість її масштабування. Для цього буде використано такі технічні рішення:
* **Контейнеризація** через **Docker** для забезпечення гнучкості в розгортанні і масштабуванні додатку.
* Використання **микросервісної архітектури** для розподіленої обробки запитів, що дозволить оптимізувати ресурси та збільшити продуктивність.
* **Балансувальники навантаження** для розподілу запитів на різні сервери в разі збільшення навантаження.
* **Безпека:** Для забезпечення високого рівня безпеки будуть використані такі підходи:
* **Шифрування даних** при передачі через HTTPS та зберігання чутливих даних за допомогою **AES**.
* **Оновлення та патчі** для всіх компонентів системи для усунення вразливостей.
* **Аутентифікація через JWT** для безпечної авторизації користувачів.
* Використання **ролей та прав доступу** для обмеження доступу до чутливої інформації в залежності від рівня користувача (адміністратор, менеджер, користувач).

## 6.4 Висновок

Концептуальне проектування продукту включає в себе добре сплановану архітектуру, де кожен компонент взаємодіє з іншими для забезпечення безпеки, масштабованості та зручності для кінцевих користувачів. Комбінація правильних технологій дозволить створити надійну та швидку платформу, яка буде готова до подальшого розвитку та масштабування.

# 7. Огляд методології

## 7.1 Вибір підходу до розробки

Для цього проєкту було обрано методологію **Agile** з використанням фреймворку **Scrum**. Цей підхід дозволяє гнучко реагувати на зміни вимог і забезпечує постійну взаємодію з клієнтом протягом всього циклу розробки. Agile дозволяє швидко адаптуватися до нових умов і підтримує невеликі, але часті випуски продукту, що ідеально підходить для сучасних проєктів у сфері розробки програмного забезпечення.

* **Scrum** передбачає роботу у спринтах (короткі цикли розробки, зазвичай 2–4 тижні), що дозволяє забезпечити регулярний прогрес та надавати клієнтам працюючі функціональні версії продукту. У результаті цього підходу ми можемо швидко тестувати і оцінювати продукт на кожному етапі, а також коригувати його відповідно до зворотного зв'язку.

## 7.2 Процес управління проєктом та взаємодія з клієнтами

Процес управління проєктом у межах методології Agile і Scrum включатиме:

**Початкова фаза (аналіз і планування):**

* Під час цієї фази визначаються вимоги та очікування клієнта.
* Формується список функціональних можливостей та пріоритетів для першої версії продукту.
* Клієнт бере участь у обговоренні критеріїв приймання, щоб забезпечити правильне розуміння продукту.

**Щоденні зустрічі (Daily Standups):**

* Кожен учасник команди надає короткий звіт про виконану роботу, проблеми та плани на день.
* Визначаються пріоритети для спринтів та обговорюються проблеми.

**Спринти:**

* Кожен спринт триватиме 2–3 тижні і включатиме розробку, тестування і впровадження частини функціоналу.
* Наприкінці кожного спринту команда проводить демонстрацію результатів клієнту, щоб отримати зворотний зв'язок.

**Ретроспективи:**

* Після кожного спринту проводиться зустріч для аналізу того, що було зроблено добре і що можна покращити.
* Це дозволяє постійно вдосконалювати процес розробки і адаптувати команду до нових викликів.

## 7.3 Етапи виконання робіт за методологією

Проєкт буде розбитий на кілька основних етапів за методологією Scrum, що допоможе підтримати чіткий графік роботи та контролювати хід виконання. Важливими етапами є:

**Підготовка та планування (Sprint 0):**

* Підготовка дорожньої карти проєкту.
* Формування беклогу (список завдань та функціональностей).
* Розподіл задач на спринти.

**Розробка:**

* Кожен спринт починається з планування, де команда обирає пріоритетні задачі з беклогу для реалізації.
* Протягом спринту команда працює над функціональними модулями продукту.

**Тестування:**

* Тестування буде проводитися в кожному спринті на етапі розробки.
* Регулярне проведення юніт-тестів і інтеграційних тестів для перевірки коректності роботи функцій.
* Тестування буде виконуватись як вручну, так і за допомогою автоматизованих тестів.
* У кожному спринті передбачається інтеграція та тестування нових функцій на тестовому середовищі.

**Взаємодія з клієнтами:**

* В кінці кожного спринту команда проводить презентацію клієнту, демонструючи досягнуті результати.
* Клієнт надає зворотний зв'язок, який дозволяє коригувати наступні кроки в розробці.

**Фінальний етап:**

* Після завершення основної розробки продукту, команда переходить до етапу стабілізації, де фокус робиться на виправленні багів, покращенні продуктивності та підготовці продукту до запуску.

## 7.4 Тестування та зворотний зв'язок

Тестування є невід’ємною частиною Agile-процесу і буде проводитися паралельно з розробкою:

* **Юніт-тести:** Проводитимуться автоматизовані юніт-тести для перевірки кожної функціональної одиниці продукту.
* **Інтеграційні тести:** Забезпечать правильність взаємодії між компонентами системи.
* **Функціональні тести:** Перевірка основних функцій, таких як створення проектів, завдань, комунікація між користувачами.
* **Зворотний зв'язок від клієнта:** Після кожного спринту буде проводитись обговорення з клієнтом, щоб зібрати зворотний зв'язок та уточнити можливі зміни.

Завдяки гнучкому підходу в тестуванні та постійним взаємодіям із клієнтом, ми можемо швидко реагувати на зміни та коригувати процес розробки, забезпечуючи вищу якість кінцевого продукту.

## 7.5 Висновок

Використання методології Agile та фреймворку Scrum дозволить забезпечити ефективне управління проєктом, гнучкість у прийнятті рішень і постійний зворотний зв'язок від клієнта. Це дозволить досягти високої якості продукту при збереженні адаптивності та здатності оперативно реагувати на зміни в вимогах.

# 8. Реалізація та заключне тестування

## 8.1 Кроки по розробці основних компонентів продукту

Розробка продукту буде проходити в кілька етапів, починаючи з базових функцій і поступово переходячи до більш складних компонентів. Ось ключові етапи розробки:

**Розробка архітектури системи**:

* Створення загальної архітектури системи, яка включає серверну та клієнтську частини.
* Визначення основних компонентів (база даних, API, інтерфейс користувача).
* Інтеграція серверних і клієнтських частин для забезпечення безперебійної роботи системи.

**Розробка функціоналу користувацького інтерфейсу**:

* Розробка інтуїтивно зрозумілого інтерфейсу для кінцевих користувачів.
* Врахування UX/UI принципів для підвищення зручності використання.
* Інтеграція з серверною частиною для забезпечення повноцінного функціоналу.

**Розробка серверної логіки**:

* Впровадження бекенд-логіки, яка обробляє запити клієнтів і взаємодіє з базою даних.
* Розробка API для комунікації між фронтендом та бекендом.
* Забезпечення масштабованості та ефективної роботи серверної частини при високому навантаженні.

**Розробка бази даних**:

* Проектування структури бази даних, визначення таблиць та зв'язків між ними.
* Реалізація механізмів для зберігання та обробки даних.
* Оптимізація запитів до бази даних для забезпечення високої продуктивності.

**Інтеграція сторонніх сервісів**:

* Визначення необхідних сторонніх бібліотек і сервісів (наприклад, платіжні системи, сервіси аутентифікації).
* Інтеграція сторонніх API для додавання додаткових функцій (наприклад, інтеграція з календарем, система повідомлень).

## 8.2 Тестування функціоналу на кожному етапі

Тестування є ключовою частиною процесу розробки і буде проводитися на кожному етапі, щоб переконатися, що продукт відповідає вимогам та працює без помилок:

**Юніт-тести**:

* Проводитимуться на рівні окремих компонентів або функцій продукту.
* Кожен окремий модуль (функція, клас) буде тестуватися окремо для перевірки правильності виконання його логіки.
* Це дозволить виявити помилки на ранньому етапі розробки, ще до інтеграції компонента в систему.

**Інтеграційні тести**:

* Тестування взаємодії між різними компонентами системи: база даних, серверна логіка, клієнтський інтерфейс.
* Перевірка того, як компоненти працюють разом і чи немає помилок у їх взаємодії.
* Тестування API, яке дозволяє перевірити правильність взаємодії між фронтендом і бекендом.

**Функціональні тести**:

* Перевірка основного функціоналу продукту: реєстрація користувачів, створення завдань, комунікація між користувачами, планування.
* Важливо, щоб усі функціональні вимоги були виконані на кожному етапі розробки.

**UI/UX тести**:

* Тестування зручності інтерфейсу: перевірка, чи легко користувачам орієнтуватися в додатку, чи відповідають елементи інтерфейсу їх функціям.
* Залучення реальних користувачів для тестування з точки зору зручності та інтуїтивності.

**Тестування продуктивності та навантаження**:

* Визначення максимального навантаження, яке система може витримати.
* Тестування на великих обсягах даних для виявлення потенційних проблем із швидкістю або масштабованістю.

**Безпека**:

* Тестування на вразливості безпеки, перевірка на наявність SQL-ін'єкцій, XSS-атак і інших потенційно небезпечних моментів.

## 8.3 Плани для виконання заключного тестування перед запуском

Заключне тестування є важливою частиною процесу розробки, що дозволить переконатися в готовності продукту до запуску. Для цього буде виконано кілька важливих етапів:

**Функціональне тестування**:

* Перевірка всіх функцій, щоб переконатися, що вони працюють коректно відповідно до вимог.
* Виконання всіх можливих сценаріїв використання продукту для переконання, що система працює без помилок.

**Тестування зворотного зв'язку**:

* Останнє тестування продукту із залученням кінцевих користувачів для отримання їхнього зворотного зв'язку.
* Проведення тестування на різних пристроях і браузерах, щоб упевнитися, що продукт працює на всіх платформам.

**Тестування продуктивності**:

* Проведення стрес-тестів для перевірки, як система реагує на високі навантаження.
* Оптимізація роботи системи на основі результатів тестів.

**Тестування безпеки**:

* Перевірка системи на вразливості, особливо перед запуском у виробниче середовище.
* Використання інструментів для виявлення та усунення можливих загроз безпеці.

**Тестування інтерфейсу**:

* Перевірка відповідності UI/UX вимогам і забезпечення зручності користування додатком.

**Фінальний огляд продукту**:

* Перегляд усіх аспектів продукту командою для впевненості у відсутності помилок.
* Оцінка готовності продукту до випуску.

Після успішного завершення заключного тестування та виправлення всіх знайдених помилок продукт буде готовий до запуску.

# 9. Експлуатація та підтримка

## 9.1 Опис процесу моніторингу продуктивності та виправлення помилок

Після запуску продукту необхідно забезпечити постійний моніторинг його продуктивності, щоб виявляти потенційні проблеми та забезпечити безперебійну роботу системи.

**Моніторинг продуктивності**:

* Відстеження показників системи в реальному часі, таких як швидкість роботи, навантаження на сервери, час відгуку користувацького інтерфейсу.
* Використання інструментів моніторингу (наприклад, New Relic, Prometheus) для збору даних про продуктивність та виявлення потенційних "вузьких місць".
* Перевірка сервісів та процесів, які можуть мати вплив на загальну ефективність системи, включаючи бази даних, API та сторонні інтеграції.

**Виправлення помилок**:

* Виявлення та документування помилок, які можуть виникнути під час експлуатації.
* Впровадження процесу обробки помилок та їх пріоритетизація: критичні помилки (що блокують роботу продукту) і менш важливі (що не впливають на основну функціональність).
* Регулярне виправлення помилок за допомогою патчів, а також робота над покращенням загальної стабільності продукту.

**Виправлення дефектів**:

* У разі виявлення дефектів (наприклад, багів у програмному забезпеченні), проводиться швидка локалізація проблеми та розробка виправлення, яке після тестування буде імплементоване в основну версію продукту.

## 9.2 Регулярні оновлення та поліпшення

Регулярні оновлення продукту дозволяють забезпечити його актуальність, додавати нові функції та покращувати вже наявні. Оновлення можуть бути як дрібними, так і великими, в залежності від потреб користувачів та змін на ринку.

**Планові оновлення**:

* Регулярні оновлення для покращення функціональності, виправлення помилок і впровадження нових функцій на основі відгуків користувачів.
* Підготовка плану випуску нових версій продукту: визначення основних змін, тестування, а також сповіщення користувачів про оновлення.
* Перевірка сумісності з різними версіями операційних систем, браузерів, пристроїв.

**Поліпшення безпеки**:

* Постійне оновлення компонентів системи, які можуть бути уразливими до атак (наприклад, оновлення бібліотек, патчі безпеки).
* Виправлення потенційних вразливостей безпеки, які могли бути виявлені після запуску продукту.

**Масштабування системи**:

* Підготовка системи до масштабування у разі збільшення кількості користувачів або навантаження на сервіс.
* Оновлення архітектури для забезпечення стабільної роботи в умовах високих навантажень.

## 9.3 Підтримка користувачів і технічна підтримка

Підтримка користувачів є важливою частиною післязапускової експлуатації продукту. Вона включає в себе не тільки технічні питання, але й допомогу з використанням продукту, навчання та інші аспекти взаємодії з кінцевими користувачами.

**Технічна підтримка**:

* Створення каналу технічної підтримки для користувачів (через чат, телефон або електронну пошту).
* Забезпечення своєчасної відповіді на запити користувачів та вирішення технічних проблем.
* Визначення рівнів підтримки (наприклад, стандартна підтримка, екстрена підтримка для критичних проблем).

**Система зворотного зв’язку**:

* Збір зворотного зв'язку від користувачів через анкети, форми відгуків або соціальні мережі.
* Аналіз зворотного зв'язку для покращення функціональності продукту, усунення неполадок та впровадження нових можливостей.

**Навчання користувачів**:

* Розробка документації та керівництв користувача, щоб допомогти користувачам легко освоїти продукт.
* Вебінари, відеоуроки та інші ресурси для навчання користувачів.

**Аналіз та звітування про роботу продукту**:

* Регулярний збір статистики використання продукту для виявлення можливих проблем або уподобань користувачів.
* Створення звітів про основні показники ефективності та виконання цілей продукту, щоб відстежувати прогрес та виявляти області для поліпшення.

**Оновлення інформації для користувачів**:

* Підтримка актуальності інтерфейсу, додавання нових функцій, виправлення зауважень користувачів.
* Повідомлення користувачів про оновлення продукту, нові функціональні можливості та важливі зміни.

# 10. Можливі ризики

## 10.1 Технічні ризики

**Невідповідність технологій**:

* Використання застарілих або невідповідних технологій може призвести до складнощів при інтеграції з іншими системами або обмеження функціональності продукту.
* Ризик вибору платформи або фреймворку, який не забезпечить необхідної масштабованості або безпеки на етапі експлуатації.
* Потенційні проблеми з оновленням програмного забезпечення або сумісністю з новими версіями операційних систем.

**План на випадок ризику**:

* Перед початком розробки провести детальний аналіз вибору технологій і платформ, які будуть відповідати вимогам проекту.
* Проводити тестування та моніторинг у реальних умовах використання, щоб мінімізувати потенційні неполадки з технологіями.
* Включити у план розробки можливість оновлення та інтеграції з новими технологіями, якщо це необхідно.

**Збої в інфраструктурі (сервери, бази даних, хостинг)**:

* Ризик відмови серверної інфраструктури або проблем з хостингом може призвести до тимчасового припинення роботи продукту.
* Можливі проблеми з доступністю даних або повна втрата даних у разі технічних збоїв.

**План на випадок ризику**:

* Використовувати надійні хостингові компанії з гарною репутацією та забезпеченням відмовостійкості.
* Створення плану резервного копіювання і відновлення даних.
* Регулярне тестування на можливі збої в інфраструктурі для попередження аварійних ситуацій.

## 10.2 Ризики на етапі тестування та впровадження

**Недостатнє тестування або проблеми під час інтеграції**:

* Тестування може не охопити всі можливі варіанти використання продукту, що може призвести до виявлення серйозних проблем на етапі впровадження.
* Можливі труднощі при інтеграції з іншими системами або сторонніми сервісами.

**План на випадок ризику**:

* Забезпечити комплексне тестування продукту на всіх етапах розробки: юніт-тести, інтеграційні тести, тестування користувацького інтерфейсу.
* Визначити список критичних компонентів системи для детального тестування перед запуском.
* Проводити тестування на реальних даних та у різних сценаріях, щоб знизити ймовірність виявлення критичних помилок на етапі запуску.

**Затримки в тестуванні та впровадженні**:

* Можливі затримки в процесі тестування через велику кількість непередбачуваних багів або неполадок в інфраструктурі.
* Затримки в інтеграції з іншими системами, що може призвести до відкладання запуску.

**План на випадок ризику**:

* Розробка чіткої стратегії тестування та впровадження з визначенням часових рамок для кожного етапу.
* Підготовка запасного плану для швидкого вирішення проблем та затримок.
* Постійний моніторинг процесу тестування та інтеграції для своєчасного виявлення проблем.

## 10.3 Фінансові і організаційні ризики

**Зміни в складі команди**:

* Втрата ключових співробітників або зміна складу команди може призвести до затримок у виконанні завдань або втрати знань і досвіду.

**План на випадок ризику**:

* Забезпечення наявності резервних кадрів та розподіл відповідальності серед кількох членів команди для запобігання залежності від окремих осіб.
* Розробка планів по забезпеченню безперервності роботи (наприклад, задокументовані робочі процеси та код, що дозволяє іншим членам команди швидко підхопити проект).
* Проведення регулярних внутрішніх нарад і обговорень, щоб всі члени команди були в курсі прогресу та змін.

**Конфлікти в команді або організаційні труднощі**:

* Внутрішні конфлікти або недостатня комунікація між учасниками проекту можуть уповільнити процес розробки або створити проблеми при досягненні цілей.

**План на випадок ризику**:

* Створення чіткої організаційної структури і визначення ролей та відповідальності.
* Регулярні зустрічі команди для обговорення прогресу та вирішення поточних проблем.
* Підтримка здорової атмосфери в команді через корпоративну культуру і відкритий діалог.

## 10.4 Зниження інтересу користувачів

**Неактуальність або відсутність запиту на продукт**:

* Ризик того, що продукт не буде відповідає вимогам цільової аудиторії або не задовольнятиме потреби ринку.

**План на випадок ризику**:

* Проведення маркетингових досліджень та тестування продукту на ранніх етапах розробки.
* Пошук альтернативних функціональних можливостей, які могли б підвищити інтерес до продукту.
* Внесення змін до стратегії маркетингу та розвитку продукту на основі зворотного зв’язку від користувачів.

# 11. Терміни

## 11.1 План дій та графік реалізації

**Етап 1: Аналіз та планування (1 тиждень)  
Терміни**: 1–7 вересня

* **Завдання**:
* Оцінка вимог до проекту та визначення технічних вимог.
* Розробка початкової документації (опис продукту, технічні характеристики).
* Формування команди проекту та визначення відповідальних осіб.
* **Відповідальні особи**: Project Manager, технічний керівник.

**Етап 2: Проектування (2 тижні)  
Терміни**: 8–21 вересня

* **Завдання**:
* Опис концептуальної архітектури продукту, визначення компонентів системи.
* Розробка макетів інтерфейсу користувача та прототипу.
* Планування та підготовка технічної інфраструктури для тестування.
* **Відповідальні особи**: Архітектор системи, дизайнер UX/UI, технічний керівник.

**Етап 3: Розробка основних компонентів (6 тижнів)  
Терміни**: 22 вересня – 3 листопада

* **Завдання**:
* Розробка основних функцій продукту, програмування серверної та клієнтської частини.
* Інтеграція бази даних, розробка функціональності.
* Проведення внутрішніх тестів для перевірки працездатності системи.
* **Відповідальні особи**: Розробники, тестувальники.

**Етап 4: Тестування та виправлення помилок (2 тижні)  
Терміни**: 4–17 листопада

* **Завдання**:
* Проведення інтеграційних тестів, функціональних тестів.
* Виправлення виявлених помилок та багів.
* Тестування продукту на відповідність вимогам безпеки та масштабованості.
* **Відповідальні особи**: Тестувальники, розробники.

**Етап 5: Пілотний запуск (1 тиждень)  
Терміни**: 18–21 листопада

* **Завдання**:
* Організація пілотного запуску для обмеженої групи користувачів.
* Збір зворотного зв’язку та внесення коригувань до продукту.
* **Відповідальні особи**: Project Manager, маркетолог, технічна підтримка.

## 11.2 Деталізація етапів

**Аналіз і планування**:

* Формування команди проекту, визначення обов'язків.
* Розробка плану проекту, визначення функціональних вимог.

**Проектування**:

* Розробка архітектури програмного продукту.
* Дизайн інтерфейсу та UX, планування бази даних і безпеки.

**Розробка**:

* Написання коду для основних компонентів.
* Інтеграція з базою даних, розробка інтерфейсу користувача.
* Визначення тестових сценаріїв та їх реалізація.

**Тестування**:

* Тестування функціональності продукту, виявлення та виправлення багів.
* Проведення стрес-тестів для перевірки масштабованості та ефективності.

**Запуск**:

* Пілотне тестування, аналіз результатів, підготовка до повного запуску.

## 11.3 Визначення відповідальних осіб

**Project Manager**:

* Загальна організація та управління проектом, контроль термінів та бюджету.
* Взаємодія з клієнтами і надання звітності.

**Розробники**:

* Розробка і програмування функцій продукту.
* Виконання технічних завдань, тестування компонентів.

**Тестувальники**:

* Виявлення помилок і багів, проведення тестування всіх компонентів системи.

**UX/UI дизайнер**:

* Розробка інтерфейсу користувача, створення макетів і прототипів.

**Маркетолог**:

* Підготовка маркетингових матеріалів, пілотний запуск і просування продукту серед користувачів.

## 11.4 Прогнозовані терміни для тестування та впровадження

**Тестування**:

* Включає різні етапи тестування — юніт-тести, інтеграційні тести, функціональні тести — що відбуватимуться під час всього етапу розробки.
* Заключне тестування планується на 4–17 листопада.

**Пілотний запуск**:

* Пілотне тестування: 18–21 листопада. Це останній етап перед остаточним впровадженням.

**Остаточне впровадження**:

* Після пілотного запуску, на основі отриманих відгуків, буде виконано останні коригування та підготовка до фінального впровадження.

# Рекомендований розподіл балів від Team-Lead’а

Внесок технічних членів команди:

**1. Коцюба Максим (Team Lead)**

* Як Team Lead, Максим координував усі етапи проекту, керував командою, визначав пріоритети та забезпечував виконання завдань вчасно. Він активно взаємодіяв з усіма учасниками проєкту, допомагав в ухваленні технічних рішень та злитті гілок у системі контролю версій.  
  **Рекомендовані бали**:

**2. Березюк Максим (Project Manager)**

* Максим відповідальний за організацію процесів управління проектом, планування та забезпечення відповідності термінів та вимог. Він тісно співпрацював із клієнтами, щоб визначити пріоритети і переконатися, що проект йде за планом.  
  **Рекомендовані бали**:

**3. Тарасов Максим (Business Analyst)**

* Максим займався аналізом вимог замовника та користувачів, допомагав формулювати технічні завдання та забезпечував ефективне збирання даних для розробки продукту.  
  **Рекомендовані бали**:

**4. Бортківський Олександр (Тестувальник)**

* Олександр проводив ретельне тестування продукту, виявляючи помилки на різних етапах розробки та забезпечуючи високу якість продукту перед запуском. Він активно взаємодіяв із розробниками для виправлення виявлених проблем.  
  **Рекомендовані бали**:

**5. Ткач Степан (Тестувальник)**

* Степан також займався тестуванням продукту та перевіркою функціональних можливостей, він зосереджувався на виявленні багів та перевірці інтеграційних компонентів.  
  **Рекомендовані бали**:

**6. Пантюхін Денис (Розробник)**

* Денис активно працював над серверною частиною продукту, забезпечуючи інтеграцію з базами даних, а також займався оптимізацією коду та підтримкою стабільної роботи додатку.  
  **Рекомендовані бали**:

**7. Панзига Володимир (Розробник)**

* Володимир працював над фронтендом, створюючи інтерфейси користувача та інтегруючи їх із бекендом. Його завдання включали поліпшення користувацького досвіду і швидкості роботи додатку.  
  **Рекомендовані бали**:

## Загальний розподіл балів

* Коцюба Максим (Team Lead): 75
* Березюк Максим (Project Manager): 75
* Тарасов Максим (Business Analyst): 75
* Бортківський Олександр (Тестувальник): 75
* Ткач Степан (Тестувальник): 60
* Пантюхін Денис (Розробник): 75
* Панзига Володимир (Розробник): 75