МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»
Завідувач кафедри
_____
«___» _____ 2022 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття ступеня вищої освіти «бакалавр»

з теми: «Розробка програмного забезпечення вебсайту компанії "Filiatix"»

Виконав: студент 4151 групи

Журавльов І. С.
(підпис, ПІБ)

Керівник роботи:
Викладач, к.т.н.
(посада, науковий ступень вчене звання)
Пухалевич А. В.
(підпис, ПІБ)

Миколаїв – 2022 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту: Журавльов Іван Сергійович _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: розробка програмного забезпечення вебсайту компанії «Filiatix». _____
_____

Керівник роботи: Пухалевич А.В. _____

Затверджені наказом ректора №256-уч від «30» травня 2022 р.

2. Термін подання роботи: 10.06.2022 р. _____

3. Вихідні дані по роботі: _____
_____

4. Перелік питань, що належать до розробки (найменування розділів) _____
Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Опис предметної галузі та постановка задачі; Проєкт програмного забезпечення (ескізний, технічний та робочий); Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів Тема дипломної роботи (слайд 1); Мета дипломної роботи (слайд 2); Функції програмного забезпечення (слайди 3–5); Постановка задачі (слайд 6); Use case діаграма (слайд 7); Діаграма діяльності для варіанту використання «Авторизація» (слайд 8); Діаграма діяльності для варіанту використання «Створення завдання» (слайд 8); Діаграма діяльності для варіантіу використання «Додавання коментаря до завдання» (слайд 9); Логічна модель даних (слайд 10); Фізична модель даних (слайд 11); Діаграма класів (слайд 12); Результат роботи (слайди 13–17); Висновки (слайд 18).

_____
_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 11.05.2022 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу Вступ	24.03.2022	ПП*
2.	Опис та аналіз предметної галузі	28.03.2022	ПП*
3.	Постановка задачі	31.03.2022	ПП*
4.	Ескізний проєкт програмного забезпечення	27.04.2022	
5.	Технічний проєкт програмного забезпечення	06.05.2022	
6.	Робочий проєкт програмного забезпечення	13.05.2022	
7.	Підготовка розділу Результати розробки програмного забезпечення	17.05.2022	
8.	Підготовка розділу з охорони праці	18.05.2022	ПП*
9.	Підготовка розділу Висновки	19.05.2022	
10.	Оформлення списку використаних джерел та додатків	20.05.2022	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	10.06.2022	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	12.06.2022	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	14.06.2022	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді	24.06.2022	

* - за результатами переддипломної практики (ПП), яка була з 21.03.2022 до 08.05.2022 р.

Студент _____ Журавльов І.С.
(підпис) (ПІБ)

Керівник роботи _____ Пухалевич А.В.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота «Розробка програмного забезпечення вебсайту компанії “Filiatix”» присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме – розробці програмного забезпечення відповідного вебсайту.

Програмне забезпечення написане з використанням мови програмування Node js, мови програмування JavaScript, мови розмітки HTML, фреймворку Vue js, мови стилю сторінок CSS та СУБД MySQL.

Кваліфікаційна робота містить аналіз предметної області та постановку практичної задачі інженерії програмного забезпечення, проєкт на розробку програмного забезпечення, результати розробки програмного забезпечення та демонстрацію його роботи, частину тексту коду програмного забезпечення вебсайту компанії «Filiatix» та розділ з охорони праці.

Робота виконана на 115 сторінках друкованого тексту, містить 5 додатків, 50 рисунків та список використаних джерел з 16 найменувань.

ABSTRACT

The qualification work «Software development of the “Filiatix” company website» is devoted to solving the practical problem of software engineering, characterized by the complexity and uncertainty of the conditions, namely, the development of software for the corresponding site.

The software is written using the Node js programming language, JavaScript programming language, HTML markup language, framework Vue js, CSS page style language and MySQL DBMS.

The qualification work contains an analysis of the subject area and the formulation of a practical problem of software engineering, a software development project, the results of software development and a demonstration of its work, a part of the text of the software code of the «Filiatix» website and a section about labor protection.

Work carried 115 pages of printed text, contains 50 pictures, 5 appendices and a list of references from 16 items.

ЗМІСТ

ВСТУП

З часів свого виникнення Інтернет почав розвиватися швидкими темпами, постійно вводяться різноманітні інновації: розвиток браузерів та пристроїв, які здатні використовувати Інтернет, зростає. Зараз для кожної компанії, яка має ієрархічну структуру та нараховує декілька робітників, дуже важливо відстежувати продуктивність співробітників. Саме тому для таких компаній вкрай актуально мати свій вебсайт для ефективного відстеження всіх активних та наявних робочих процесів.

Створення вебсайту компанії «Filiatix» має практичну цінність, тому, що це допоможе вести контроль (трекінг) робочого процесу в мережі Інтернет і є дієвим засобом відстеження процесів роботи. Завдання вебсайту компанії – безперебійне надання різноманітної інформації стосовно робочих процесів користувачам та керівництву в online-режимі.

Мета роботи: вирішення практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме – розробка програмного забезпечення вебсайту компанії «Filiatix».

Для створення вебсайту необхідно вирішити наступні завдання:

1. Виконати аналіз предметної області та аналіз наявних рішень для розробки вебсайтів для трекінгу.
2. Ознайомитися з основними технологіями для розробки сучасних вебсайтів.
3. Виконати постановку задача на розробку і створення вебсайту.
4. Розробити проєкт програмного забезпечення вебсайту компанії «Filiatix».
5. Виконати завдання створення вебсайту компанії «Filiatix».
6. Виконати тестування програмного забезпечення вебсайту компанії «Filiatix».
7. Розглянути питання з охорони праці.

РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ СТВОРЕННЯ ВЕБСАЙТІВ. ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБСАЙТУ КОМПАНІЇ «FILIATIX»

1.1 Аналіз предметної області компанії «Filiatix»

1.1.1 Опис сфери компанії «Filiatix»

Компанія «Filiatix» спеціалізується на плануванні процесів та менеджментів інших компаній. Зазвичай для відстеження та контролю робочого процесу існують спеціальні професії – менеджери. Але якщо багато працівників виконують роботу віддалено, ефективність менеджера стає меншою, оскільки для повноцінного функціонування йому необхідний безпосередній зв’язок з робітниками. Також, якщо компанія нечисленна, велика ймовірність того, що такі посади як менеджер будуть відсутні. Саме через розвиток компаній та постійного пошуку покращення умов і підвищення швидкості виконання процесів виникли різні допоміжні інструменти та методології. Нині найзручнішим і, як правило, безкоштовним інструментом є різного роду тасктрекери (дошки завдань), що активно допомагають з найшвидшим інформуванням про активні робочі процеси з актуальними статусами, структурованим поданням наявних задач та зв’язками між працівниками, якого так не вистачає менеджерам або команді працівників. Дуже важливим фактором, коли водночас багато людей виконують свою роботу, є повна класифікація наявних даних для більш швидкої орієнтації на дошці завдань.

Тасктрекер *(дошка завдань)* – це інструмент управління Agile-проєктами, який допомагає наочно представити завдання, обмежити обсяг незавершеної роботи та досягти максимальної ефективності (або швидкості).

Вона може допомогти робітникам підрозділів упорядкувати повсякденну роботу. За допомогою карток та стовпців на дошці Kanban команди з технічних питань та сервісні команди можуть зрозуміти, який обсяг роботи слід взяти на себе, та виконати його, дотримуючись принципів безперервного вдосконалення.

Дошки завдань використовують картки та стовпці, щоб дозволити командам ефективно візуалізувати свої робочі процеси та керувати ними.

Розглянемо основні компоненти детальніше (рисунок 1.1), використовуючи такі фреймворки як React, Angular або Ionic.

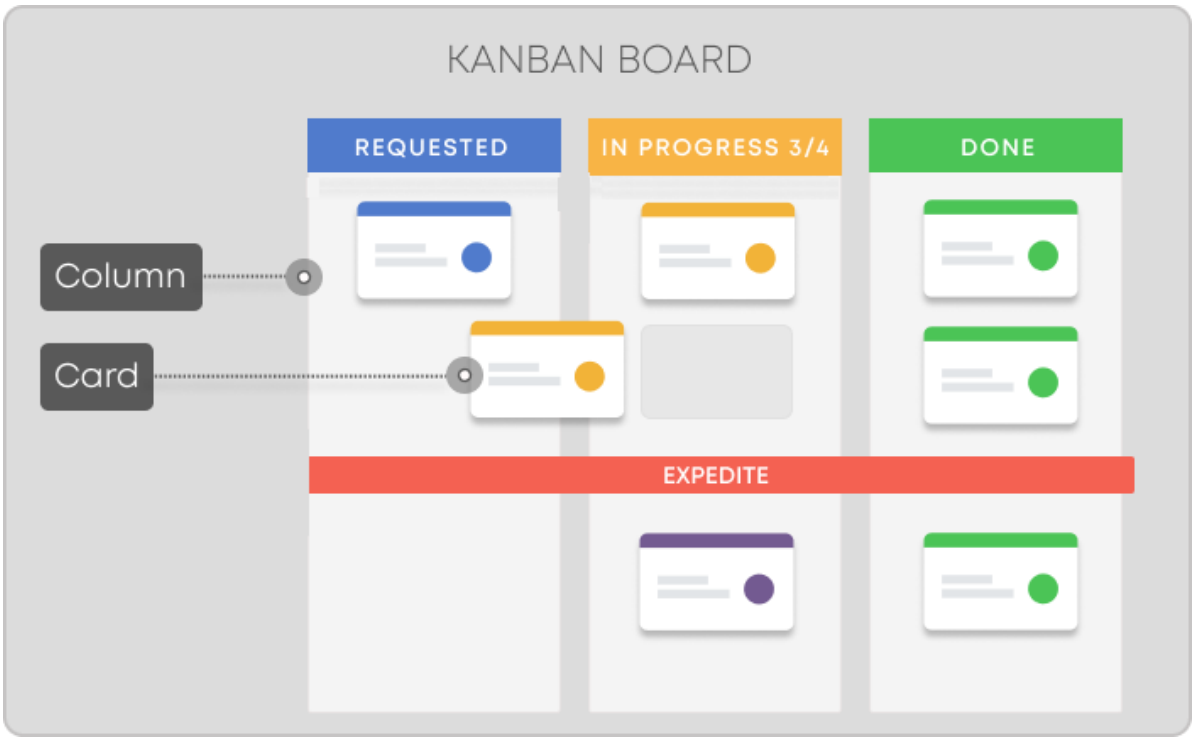

Рисунок 1.1 - Основні компоненти

1. **Картки** – це візуальне представлення завдань. Кожна картка містить інформацію про завдання та його статус, наприклад: термін виконання, правонаступник, опис тощо;
2. **Стовпці** – кожен стовпець на дошці представляє інший етап вашого робочого процесу. Картки проходять робочий процес до повного завершення.

1.1.2 Опис методів вирішення завдання з теми розробки програмного забезпечення вебсайту компанії «Filiatix»

Базовий метод (HTML, CSS)

HTML *(англ. HyperText Markup Language – мова розмітки гіпертекстових документів)* – стандартна мова розмітки вебсторінок в Інтернеті [1]. Більшість вебсторінок створюються за допомогою мови HTML (або XHTML). Документ HTML обробляється браузером та відтворюється на екрані у звичному для користувача вигляді. У більшості випадків автор документа суворо визначає зовнішній вигляд документа. У разі користуванням HTML читач, ґрунтуючись на можливостях Web-браузера, може, певною мірою, керувати зовнішнім виглядом документа (але не його змістом). HTML дозволяє відзначити, де в документі повинен бути заголовок або абзац за допомогою тега HTML, а потім надає Web-браузеру інтерпретувати ці теги.

HTML-теги можуть бути умовно розділені на дві категорії:

1) теги, що визначають, як буде зображатися Web-браузером тіло документа в цілому;

2) теги, що описують загальні властивості документа, такі як заголовок чи автор документа.

CSS *(англ. Cascading Style Sheets або скорочено CSS)* – спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних [2].

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів. Таблицю стилів CSS можна вмонтувати безпосередньо в HTML-сторінку – це внутрішня таблиця стилів. Або ж її можна створити в окремому файлі, і вже потім приєднати посилання на нього до потрібної HTML-сторінки – це зовнішня таблиця стилів. Зовнішню

таблицю необхідно підключити до основного HTML-документу за допомогою спеціальних тегів.

Модель клієнт-сервер (Client-Server)

Приблизно 7 років тому були лише клієнт і сервер. Клієнт, тобто браузер користувача, надсилає запит на сервер, а сервер відповідає HTML-файлом із CSS та JS, який завантажується у браузер. Кожного разу, коли користувач натискає кнопку або вносить зміни, браузер надсилає запит HTTP, а сервер обробляє запити та надсилає відповідь HTML та перезавантажує сторінку. Це була модель клієнт-сервер, де інтерфейс був просто HTML, CSS і JS, а бекендсервер виконував важку роботу [3].

Односторінкові програми (SPA)

Односторінковий додаток *(SPA, англ. single page application)* – це програма, якій не потрібно перезавантажувати сторінку під час її використання, вона працює у браузері [3]. Програма розділена на різні компоненти, які можна використовувати повторно. Коли відбуваються зміни, то замість перезавантаження всієї сторінки повторно зображаються лише окремі компоненти. Крім того, під час переходу від одного розділу до іншого замість перезавантаження сторінки та маршрутизації компоненти приховуються та відображаються інші компоненти.

Головна перевага односторінкових додатків – їхня швидкість. Файли HTML, CSS і Javascript попередньо завантажуються як пакет у браузер, тому програму не потрібно перезавантажувати, і вона дуже відчутна для користувача.

REST API

Опис REST API [4]

Була приведена модель клієнт-сервер, де сервер виконує всю роботу. Але з такими фреймворками, як Vue, React і Angular, фронтенд також тепер може виконувати дуже складні завдання, які раніше були неможливими.

Тож усе, що сервер повинен зробити – це надати дані цим інтерфейсним додаткам, і вони тепер вирішуватимуть складні завдання, а сервер може підключатися до бази даних і передавати дані. Більшість API сьогодні використовують JSON – легкий формат обміну даними для позначення об’єктів JavaScript для цього.

Сервери використовують кінцеві точки, які відповідають запитами JSON до інтерфейсу.

Мобільні та прогресивні вебсайти

70% користувачів сьогодні переглядають вебсайти за допомогою мобільних браузерів [5]. Тому необхідно створювати вебсайти, зважаючи на мобільний досвід. Адаптивний вебсайт можливий за допомогою медіа-запитів і фреймворків, таких як Bootstrap, Taiwind, Matirial UI.

Прогресивний вебсайт поєднує в собі найкраще з обох світів – веб та мобільні додатки. **Прогресивний вебсайт** – це тип програмного забезпечення, що постачається через Інтернет, створений з використанням поширених вебтехнологій, включно з HTML, CSS та JavaScript [6]. Він призначений для роботи як на Web, так і на мобільних платформах, враховуючи всі функції мобільних пристроїв, які не можуть виконувати традиційні вебсайти.

1.1.3 Аналіз існуючих вебсайтів, які надають змогу користуватися онлайн трекером завдань

На сьогоднішній день існує декілька вебсайтів, які надають змогу користуватися онлайн трекером завдань. Одним з таких вебсайтів є дошка завдань Trello.

При запуску, можна створити стовпці «Беклог», «На черзі», «В процесі» та «Готово» (рисунок 1.2).

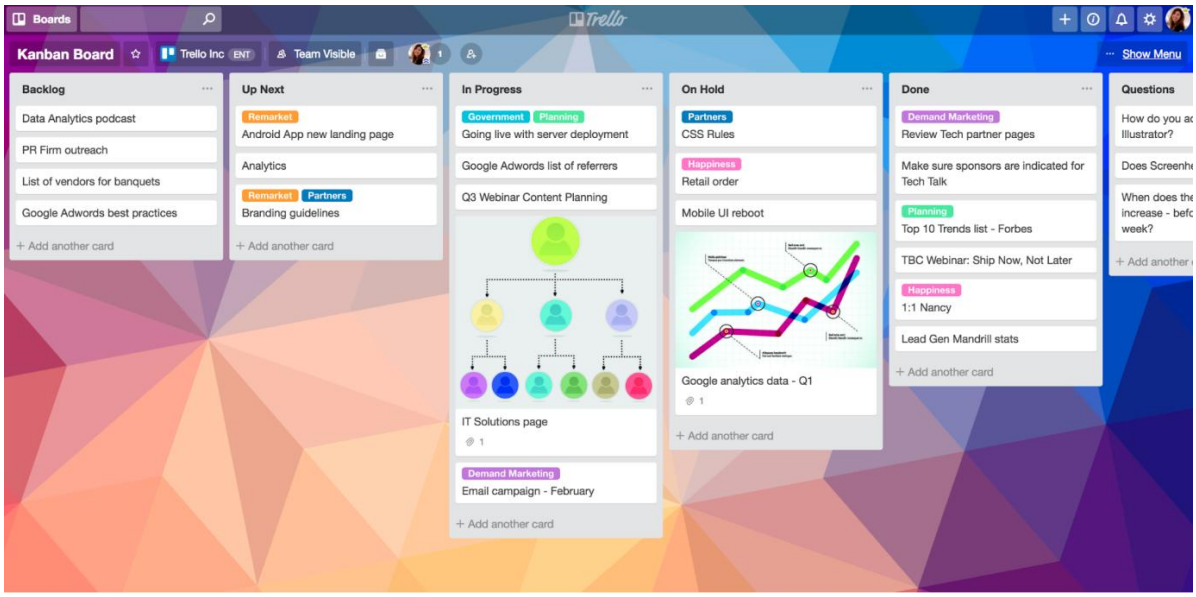

Рисунок 1.2 - Стовпці які можна створити

Кожне завдання представлене у вигляді картки, яка переміщується зі списку до стовпця у міру того, як завдання потрапляє до черги, над ним працюють та його виконують.

Проаналізувавши цей вебсайт, ми можемо бачити, що дана дошка для тасктрекінгу надає змогу:

- Створити завдання (рисунок 1.3);
- Додати колонку (рисунок 1.4);
- Змінювати положення завдання (рисунок 1.5).

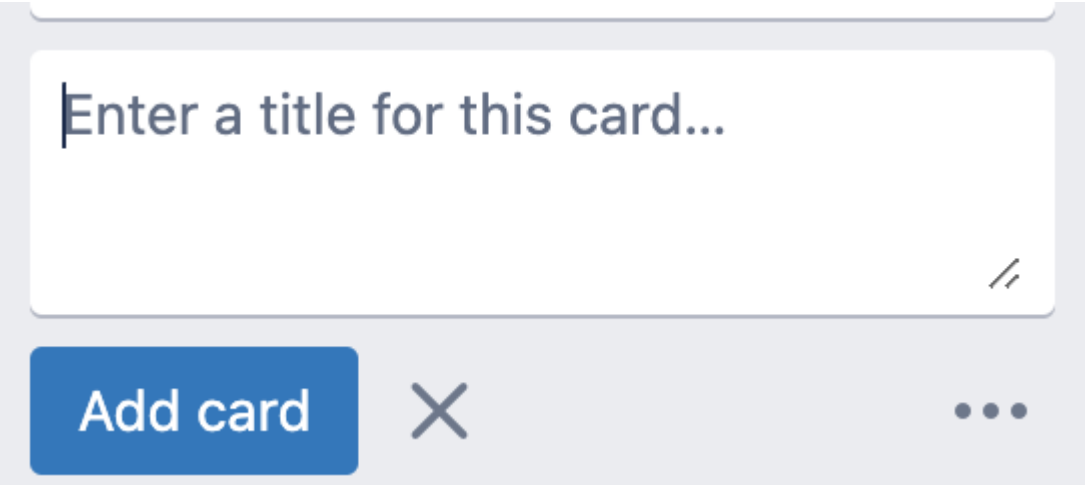

Рисунок 1.3 – Процес створення завдання

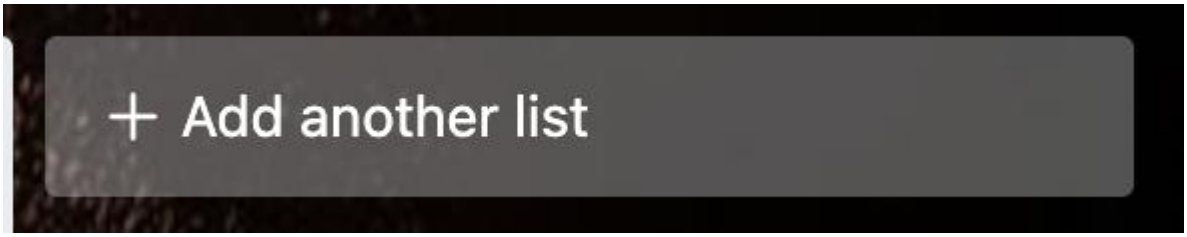

Рисунок 1.4 – Процес додавання колонки

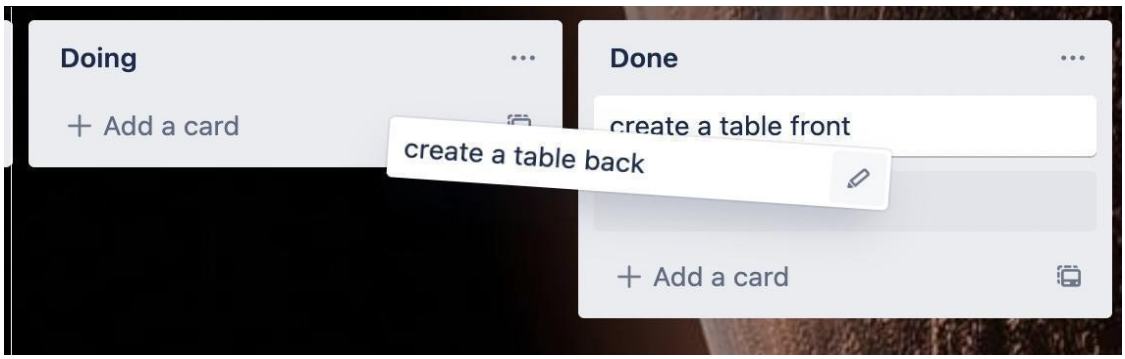

Рисунок 1.5 – Процес зміни положення завдання

Також даний сервіс має і свої недоліки, наприклад:

- Складний інтерфейс. На перший погляд не дуже зрозуміло як додати завдання, також не зрозуміло як перемикатися між власними проєктами;
- Не реалізовано пошук завдань за ключовими словами;
- Відсутні показники статистики відносно проєктів.

1.2 Постановка задачі на розробку програмного забезпечення вебсайту компанії «Filiatix»

1.2.1 Цілі і завдання дипломної роботи з урахуванням реальних можливостей і термінів виконання

Отже, проаналізувавши предметну область, існуючі вебсайти для ведення трекінгу завдань, їх переваги та недоліки, можна зробити висновок, що розробка програмного забезпечення вебсайту компанії «Filiatix» має практичну цінність.

Загалом програмне забезпечення вебсайту компанії «Filiatix» з боку реалізації повинно мати 3 ролі: користувач, адміністратор, суперадміністратор.

Програмне забезпечення вебсайту компанії «Filiatix» повинне забезпечувати користувача можливістю виконання перерахованих нижче функцій:

- Перегляд проєктів в який користувач був доданий;
- Перегляд завдань за фільтрами в проєктах до яких користувач був доданий;
- Додавати завдання в проєкти до яких користувач був доданий;
- Залишати коментарі в проєкти до яких користувач був доданий;
- Редагування існуючих завдань в проєктах до яких користувач був доданий.

Програмне забезпечення повинне забезпечувати адміністратора можливістю виконання перерахованих нижче функцій:

- Перегляд проєктів, до яких адміністратор був доданий, або які були створені адміністратором;
- Перегляд завдань за фільтрами в проєктах, до яких адміністратор був доданий, або які були створені адміністратором;

- Додавати завдання в проєкти, до яких адміністратор був доданий, або які були створені адміністратором;
- Залишати коментарі в проєктах, до яких адміністратор був доданий, або які були створені адміністратором;
- Створювати користувачів з роллю адміністратора або користувача;
- Додавати користувачів до проєктів, до яких адміністратор був доданий, або які були створені адміністратором;
- Редагування існуючих завдань в проєктах, до яких адміністратор був доданий, або які були створені адміністратором;
- Змінювати порядок стовпців в проєктах, до яких адміністратор був доданий, або які були створені адміністратором.

Програмне забезпечення повинне забезпечувати для суперадміністратора можливість виконання перерахованих нижче функцій:

- Перегляд усіх проєктів;
- Перегляд завдань за фільтрами в будь-яких проєктах;
- Додавання завдання в будь-які проєкти;
- Створювання користувачів з будь-якою роллю;
- Можливість залишати коментарі в будь-які проєкти;
- Додавання користувачів до будь-яких проєктів;
- Редагування будь-яких завдань;
- Можливість змінювати порядок стовпців в будь-яких проєктах.

1.2.2 Вимоги до програмного забезпечення вебсайту компанії «Filiatix»

- Зрозумілий інтерфейс;
- Наявність пошуку завдань за ключовими словами;
- Наявні показники статистики відносно проєктів.

Докладніше вимоги до програмного забезпечення наведено в технічному завданні у Додатку А.

РОЗДІЛ 2 ПРОЄКТ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБСАЙТУ КОМПАНІЇ «FILIATIX»

2.1 Ескізний проєкт програмного забезпечення вебсайту компанії «Filiatix»¶

Ескізний проєкт – проєктна документація, що дає загальне уявлення про будову та принцип дії виробу, містить принципові конструктивні рішення, а також дані, що визначають його відповідність для певного призначення.¶

UML (англ. Unified Modeling Language) – уніфікована мова моделювання, використовується у парадигмі об'єктноорієнтованого програмування. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю. UML був створений для визначення, візуалізації, проєктування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

2.1.1 Побудова моделі варіантів використання

Діаграма варіантів використання – в UML, діаграма, на якій зображено відношення між акторами системи та варіантами використання доступними їм в системі. Діаграма варіантів використання являє собою граф, який складається з множини акторів, варіантів використання обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами.

Проаналізувавши функціональні вимоги до програмного забезпечення, були виділені основні варіанти використання:

- авторизація;
- перегляд усіх завдань, що належать до проєкту;
- редагування завдання;
- створення завдання;
- додавання коментаря до завдання;
- створення проєкту;
- створення користувача, адміністратора, суперадміністратора;
- надання доступу користувачу, адміністратору до проєкту;
- перегляд усіх користувачів що мають доступ до проєкту.

Діаграма варіантів використання вебсайту компанії «Filiatix» показана на рисунку 2.1

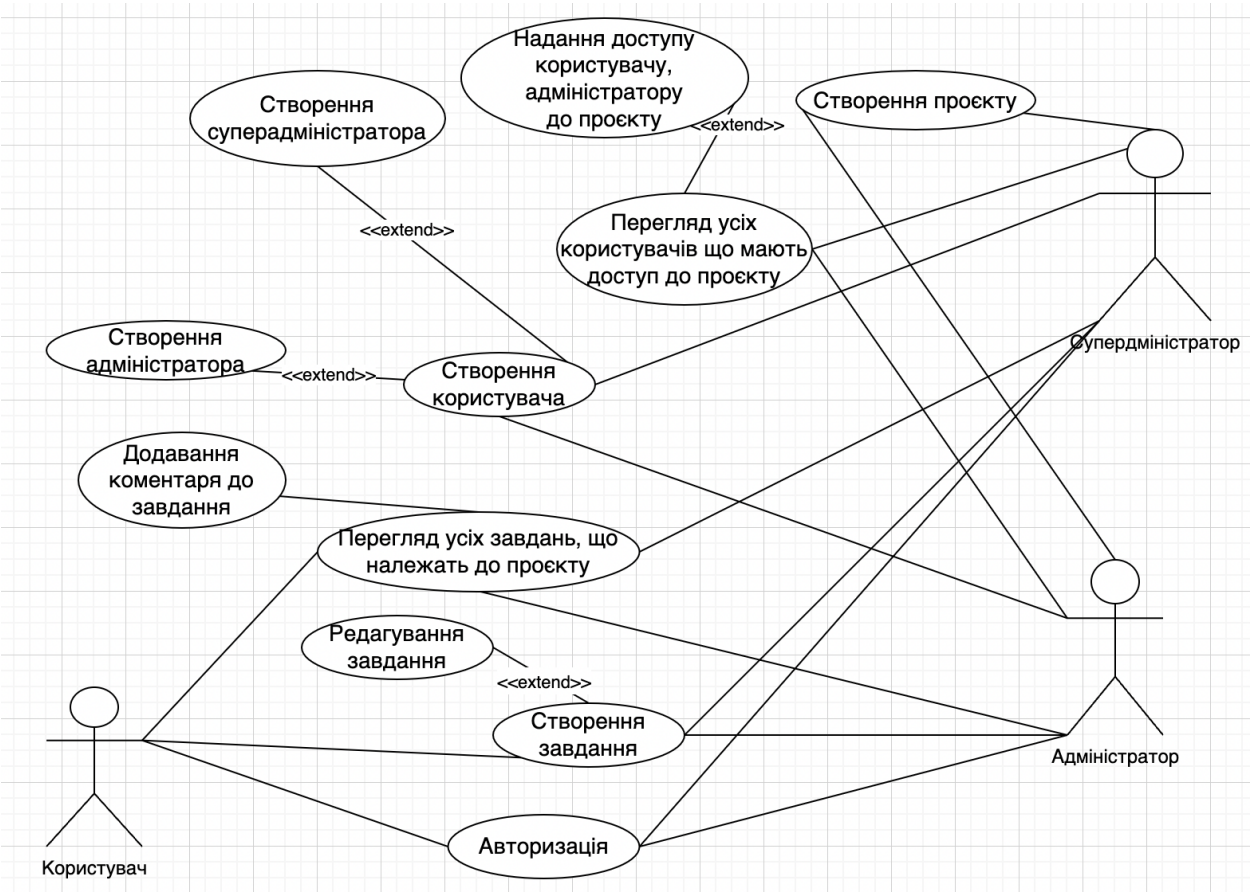

Рисунок 2.1 – Діаграма варіантів використання

2.1.2 Специфікації варіантів використання

Проаналізувавши діаграму варіантів використання, проведемо специфікації для кожного з варіантів використання. Нижче представлений реєстр варіантів використання, який зображений на таблиці 2.1.

Таблиця 2.1 – Реєстр варіантів використання

Код	Основна дійова особа (актор)	Назва варіанту використання	Короткий опис
1	2	3	4
A1	Користувач	Авторизація	Прецедент дозволяє ідентифікувати користувача як користувача, суперадміністратора чи адміністратора
A2	Користувач	Перегляд усіх завдань що відносяться до проєкту	Прецедент дозволяє переглянути усі завдання що відносяться до проєкту
A3	Користувач	Редагування завдання	Прецедент дозволяє редагувати завдання
A4	Користувач	Створення завдання	Прецедент дозволяє створити завдання
A5	Користувач	Додавання коментаря до завдання	Прецедент дозволяє додати коментар до завдання
B1	Користувач (адміністратор)	Створення проєкту	Прецедент дозволяє створити проєкт

1	2	3	4
B2	Користувач (адміністратор)	Створення користувача	Прецедент дозволяє створити користувача з роллю користувача та адміністратора
B3	Користувач (адміністратор)	Надання доступу до проєкту	Прецедент дозволяє надати доступ користувачам або амінiстраторам до проєкту який був створений поточним користувачам або був доданий адміністратором
B4	Користувач (адміністратор)	Перегляд усіх користувачів що мають доступ до проєкту	Прецедент дозволяє переглянути усіх користувачів що мають доступ до проєкту
C1	Користувач (суперадміністратор)	Створення проєкту	Прецедент дозволяє створити проєкт
C2	Користувач (суперадміністратор)	Створення користувача	Прецедент дозволяє створити користувача з роллю користувача, суперадміністратора та адміністратора

1	2	3	4
С3	Користувач (суперадміністратор)	Надання доступу до проєкту	Прецедент дозволяє надати доступ будь-яким користувачам до будь-якого проєкту
С4	Користувач (суперадміністратор)	Перегляд усіх користувачів що мають доступ до проєкту	Прецедент дозволяє переглянути усіх користувачів що мають доступ до проєкту

Конкретизація варіантів використання.

А1. Авторизація

Основна дійова особа: Користувач.

Інші учасники прецеденту: відсутні.

Передумови: Користувач зайшов в систему.

Короткий опис

Прецедент дозволяє ідентифікувати користувача як користувача, суперадміністратора чи адміністратора.

А2. Перегляд усіх завдань, що належать до проєкту

Основна дійова особа: Користувач.

Інші учасники прецеденту: відсутні.

Передумови: Користувач ініціював перегляд усіх завдань, що належать до проєкту.

Короткий опис

Прецедент дозволяє переглянути список усіх завдань, що належать до конкретного проєкту.

А3. Редагування завдання

Основна дійова особа: Користувач.

Інші учасники прецеденту: відсутні.

Передумови: Завдання було вже створене та користувач ініціював редагування завдання.

Короткий опис

Прецедент дозволяє відредагувати конкретне завдання

А4. Створення завдання

Основна дійова особа: Користувач.

Інші учасники прецеденту: відсутні.

Передумови: Користувач ініціював створення завдання.

Короткий опис

Прецедент дозволяє створити завдання

А5. Додавання коментаря до завдання

Основна дійова особа: Користувач.

Інші учасники прецеденту: відсутні.

Передумови: Користувач ініціював додавання коментарю до завдання.

Короткий опис

Прецедент дозволяє додати коментар до завдання

В1. Створення проєкту

Основна дійова особа: Адміністратор.

Інші учасники прецеденту: відсутні.

Передумови: Адміністратор ініціював створення проєкту.

Короткий опис

Прецедент дозволяє створити проєкт

В2. Створення користувача

Основна дійова особа: Адміністратор.

Інші учасники прецеденту: відсутні.

Передумови: Адміністратор ініціював створення користувача.

Короткий опис

Прецедент дозволяє створити користувача з роллю користувач, адміністратор

В3. Надання доступу до проєкту

Основна дійова особа: Адміністратор.

Інші учасники прецеденту: відсутні.

Передумови: Адміністратор ініціював надання доступу до проєкту.

Короткий опис

Прецедент дозволяє надати доступ до проєкту користувачу який не мав доступу до проєкту

В4. Перегляд усіх користувачів що мають доступ до проєкту

Основна дійова особа: Адміністратор.

Інші учасники прецеденту: відсутні.

Передумови: Адміністратор ініціював перегляд усіх користувачів що мають доступ до проєкту.

Короткий опис

Прецедент дозволяє переглянути усіх користувачів що мають доступ до проєкту

С1. Створення проєкту

Основна дійова особа: Суперадміністратор.

Інші учасники прецеденту: відсутні.

Передумови: Суперадміністратор ініціював створення проєкту.

Короткий опис

Прецедент дозволяє створити проєкт

С2. Створення користувача

Основна дійова особа: Суперадміністратор.

Інші учасники прецеденту: відсутні.

Передумови: Суперадміністратор ініціював створення користувача.

Короткий опис

Прецедент дозволяє створити користувача з роллю користувач, адміністратор або суперадміністратор

С3. Надання доступу до проєкту

Основна дійова особа: Суперадміністратор.

Інші учасники прецеденту: відсутні.

Передумови: Суперадміністратор ініціював надання доступу до проєкту.

Короткий опис

Прецедент дозволяє надати доступ до проєкту будь-якому користувачу

С4. Перегляд усіх користувачів що мають доступ до проєкту

Основна дійова особа: Суперадміністратор.

Інші учасники прецеденту: відсутні.

Передумови: Суперадміністратор ініціював перегляд усіх користувачів що мають доступ до проєкту.

Короткий опис

Прецедент дозволяє переглянути усіх користувачів що мають доступ до проєкту.

2.1.3 Сценарії варіантів використання

На даній стадії процесу розробки програмного забезпечення створюються діаграми діяльності. Доцільно створити діаграми діяльності для основних варіантів використання.

На рисунку 2.2 наведено діаграму діяльності для варіанту використання «Авторизація»:

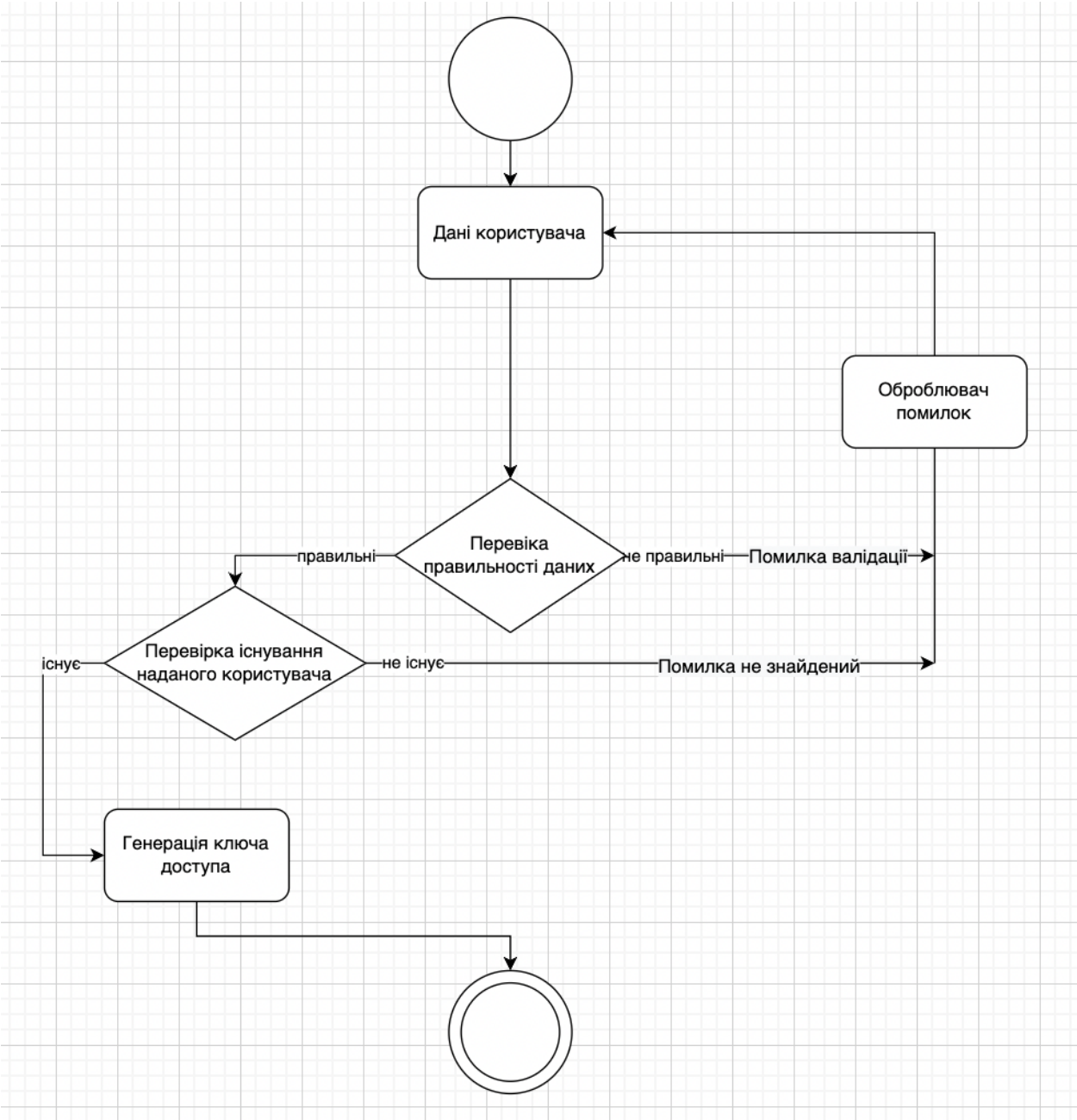

Рисунок 2.2 – Діаграма діяльності для варіанту використання «Авторизація»

На рисунку 2.3 наведено діаграму діяльності для варіанту використання «Створення завдання»:

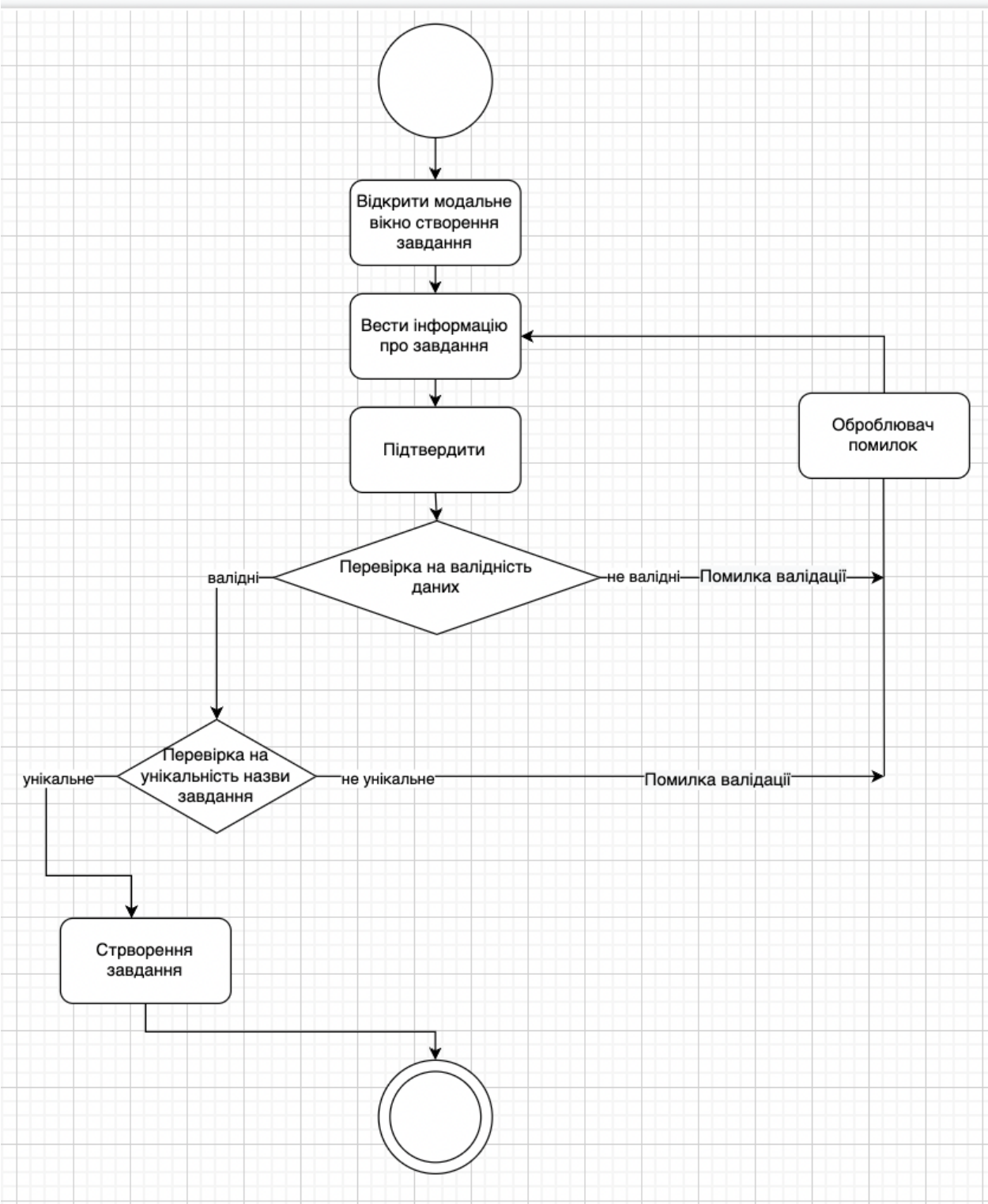

Рисунок 2.3 – Діаграма діяльності для варіанту використання «Створення завдання»

На рисунку 2.4 наведено діаграму діяльності для варіанту використання «Створення користувача»:

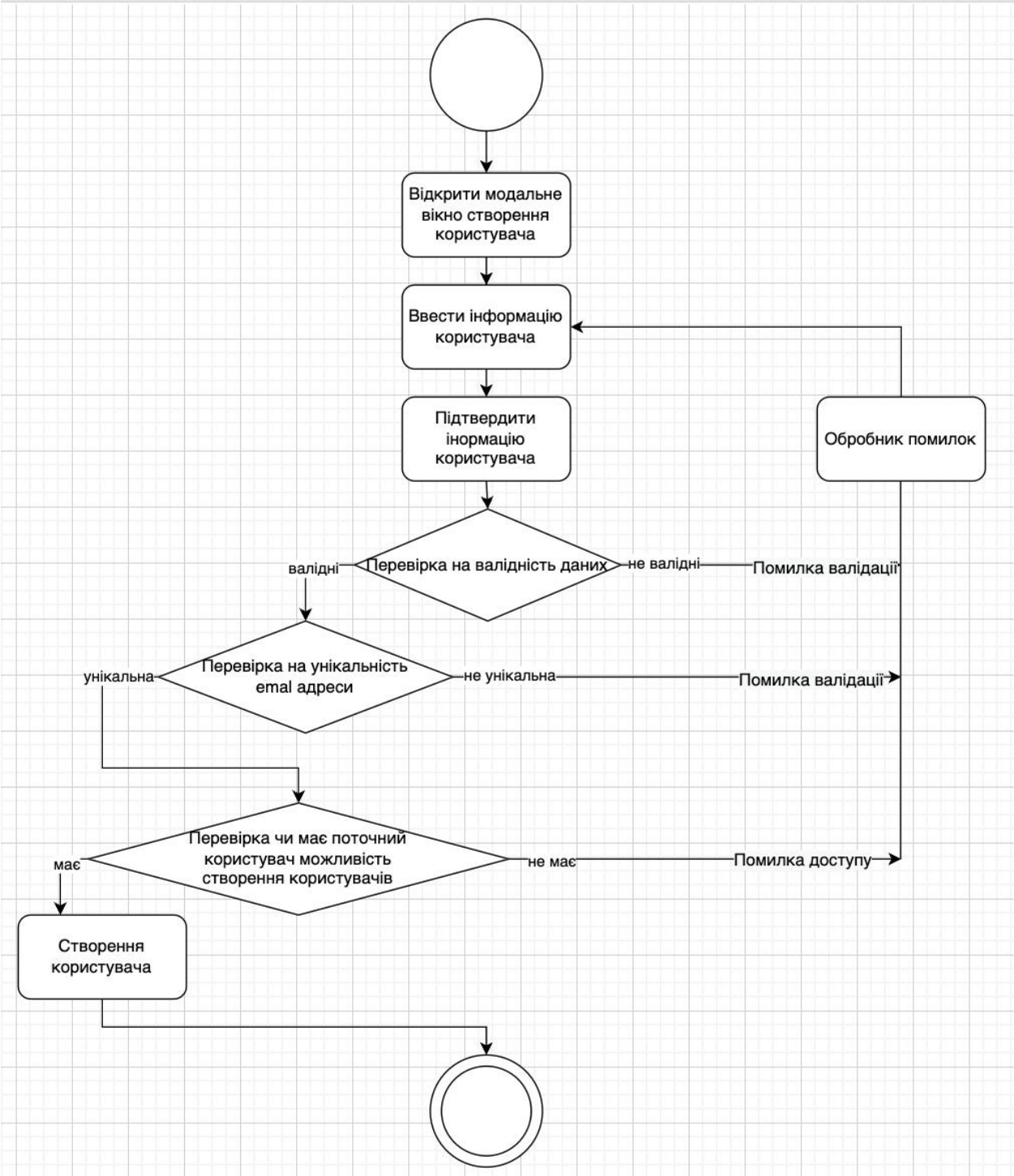

Рисунок 2.4 – Діаграма діяльності для варіанту використання «Створення користувача»

На рисунку 2.5 наведено діаграму діяльності для варіанту використання «Додавання коментаря до завдання»:

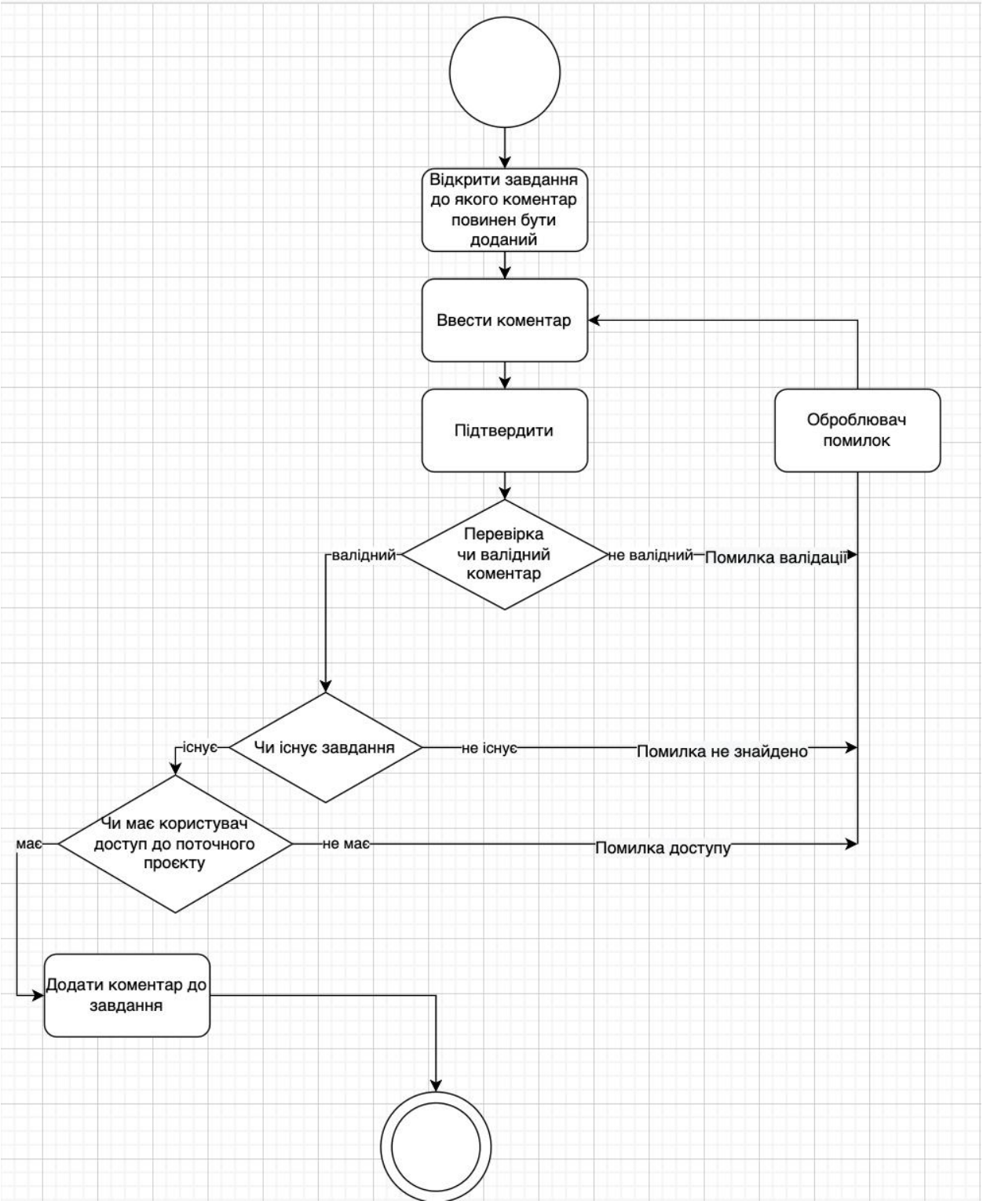

Рисунок 2.5 – Діаграма діяльності для варіанту використання «Додавання коментаря до завдання»

2.1.4 Інформаційна модель

Інформаційна модель програмного забезпечення представлена на рисунку 2.6.

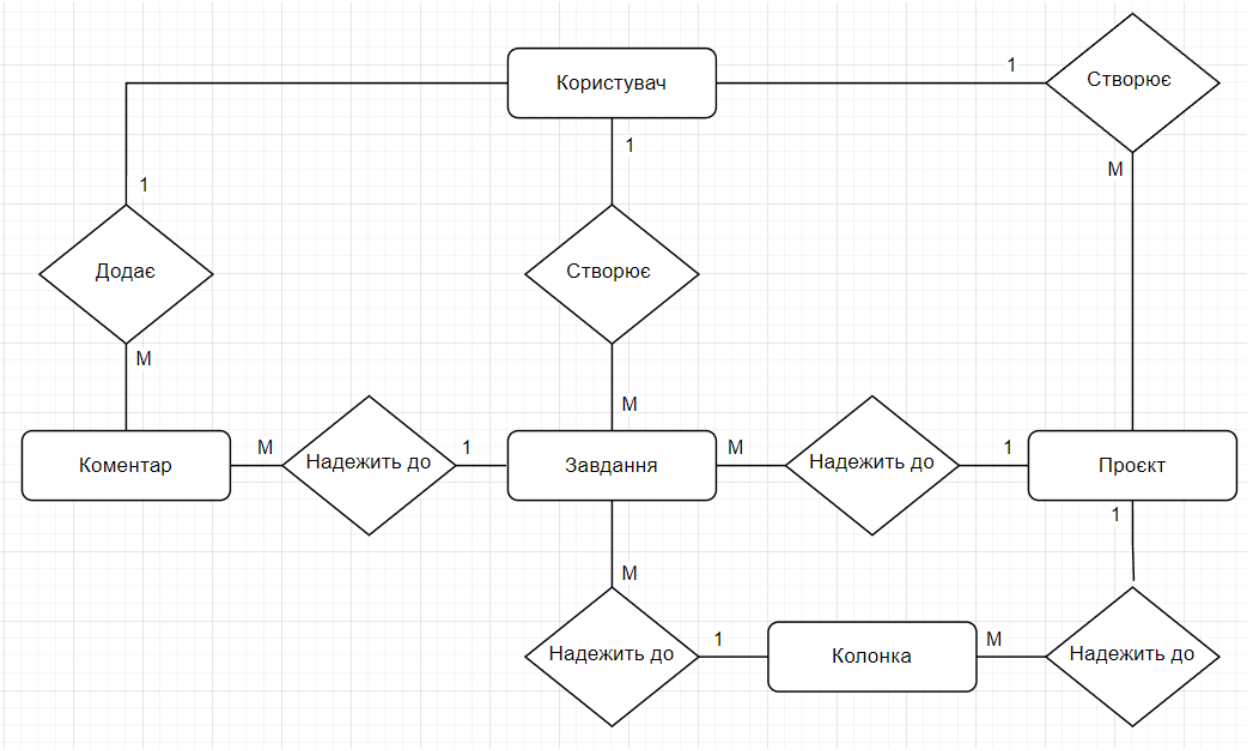

Рисунок 2.6 – Інформаційна модель

2.1.5 Розробка інтерфейсу програмного забезпечення

У цьому розділі представлені схеми взаємодії користувача з вебсайтом. При переході на головну сторінку вебсайту, він запрошує ввести дані до облікового запису (авторизуватися). Головна сторінка з авторизацією зображена на рисунку 2.7.

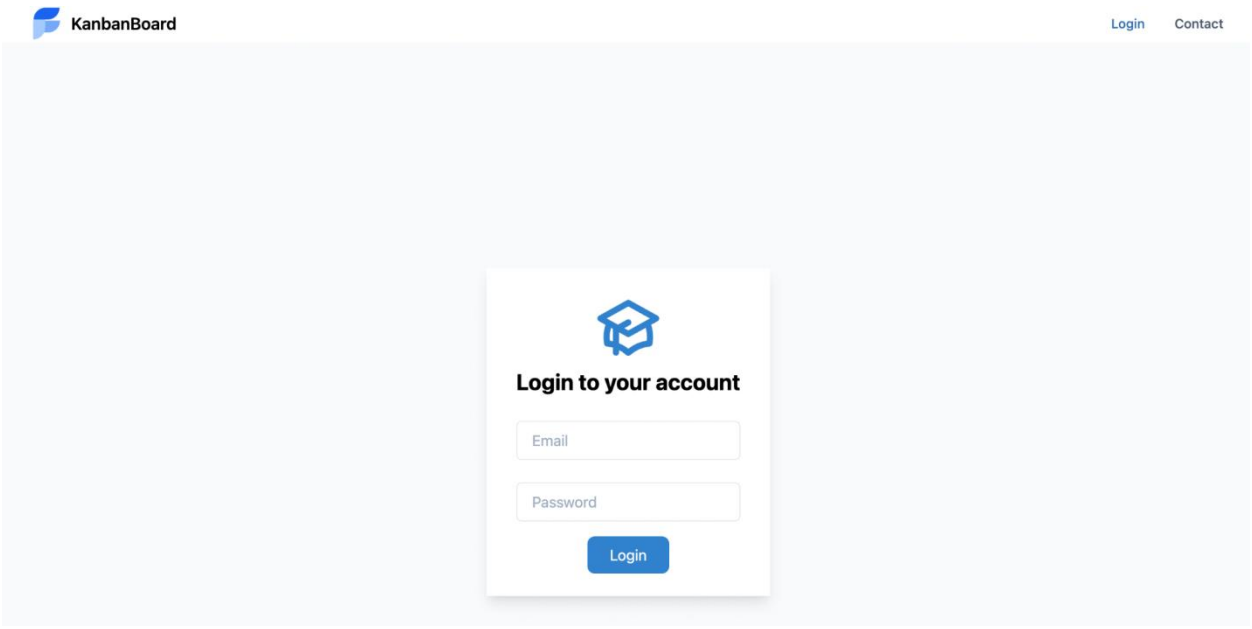

Рисунок 2.7 – Інтерфейс авторизації

Авторизація користувача зображена на рисунку 2.8.

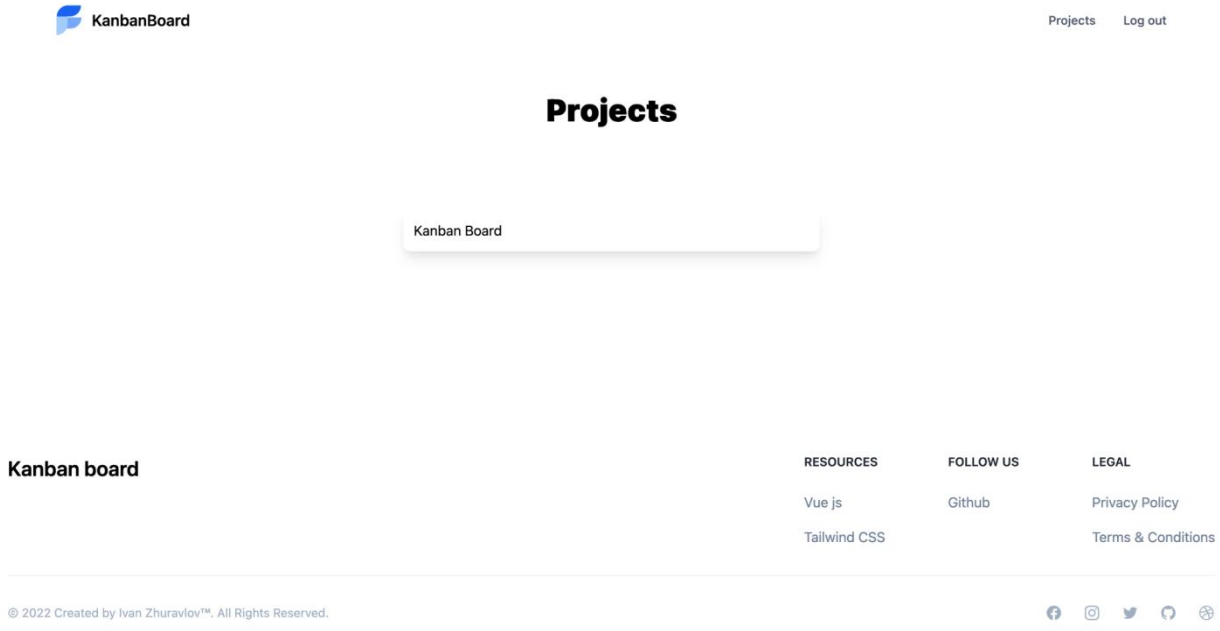

Рисунок 2.8 – Результат успішної авторизації користувача з роллю користувач

Авторизація адміністратора зображена на рисунку 2.9.

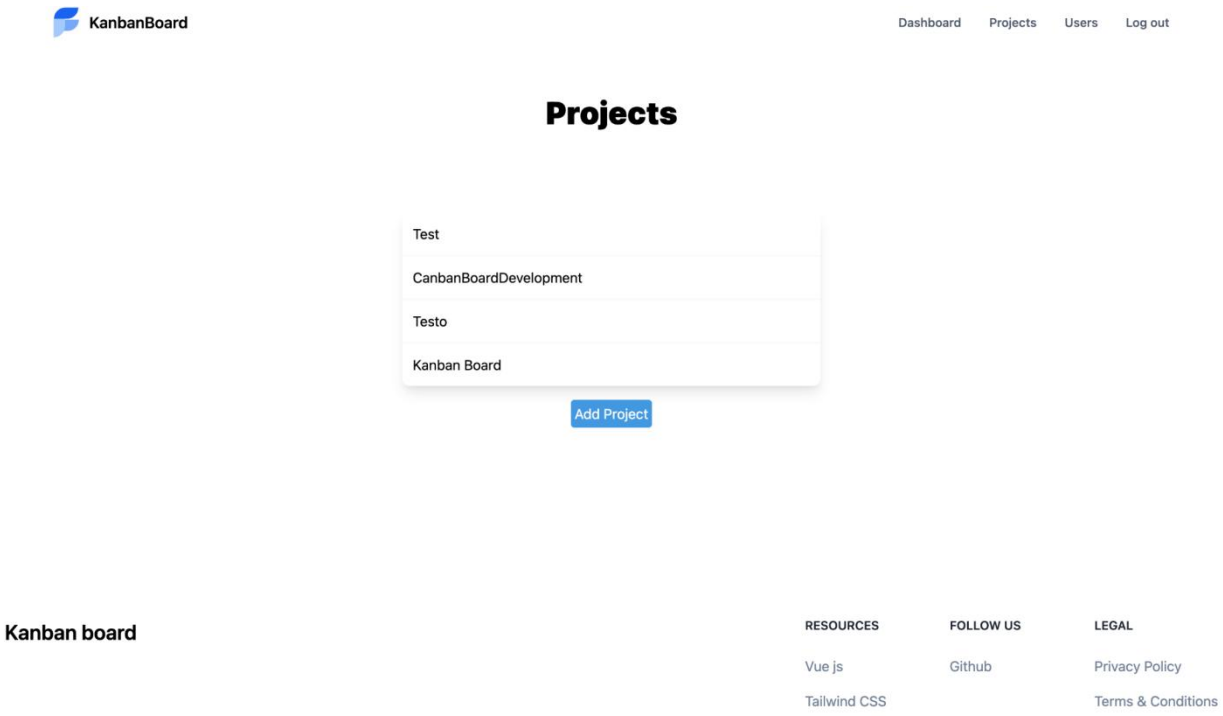

Рисунок 2.9 – Результат успішної авторизації користувача з роллю адміністратор

Авторизація суперадміністратора зображена на рисунку 2.10.

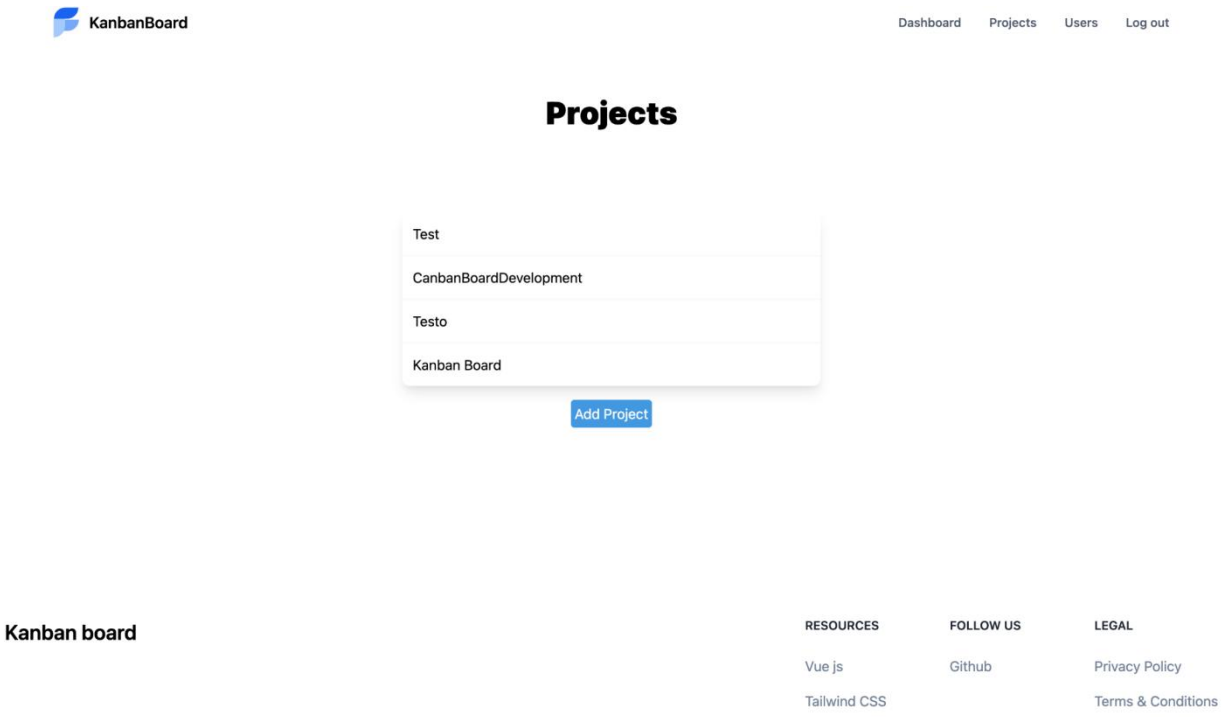

Рисунок 2.10 – Результат успішної авторизації користувача з роллю суперадміністратор

Процес створення завдання зображений на рисунку 2.11.

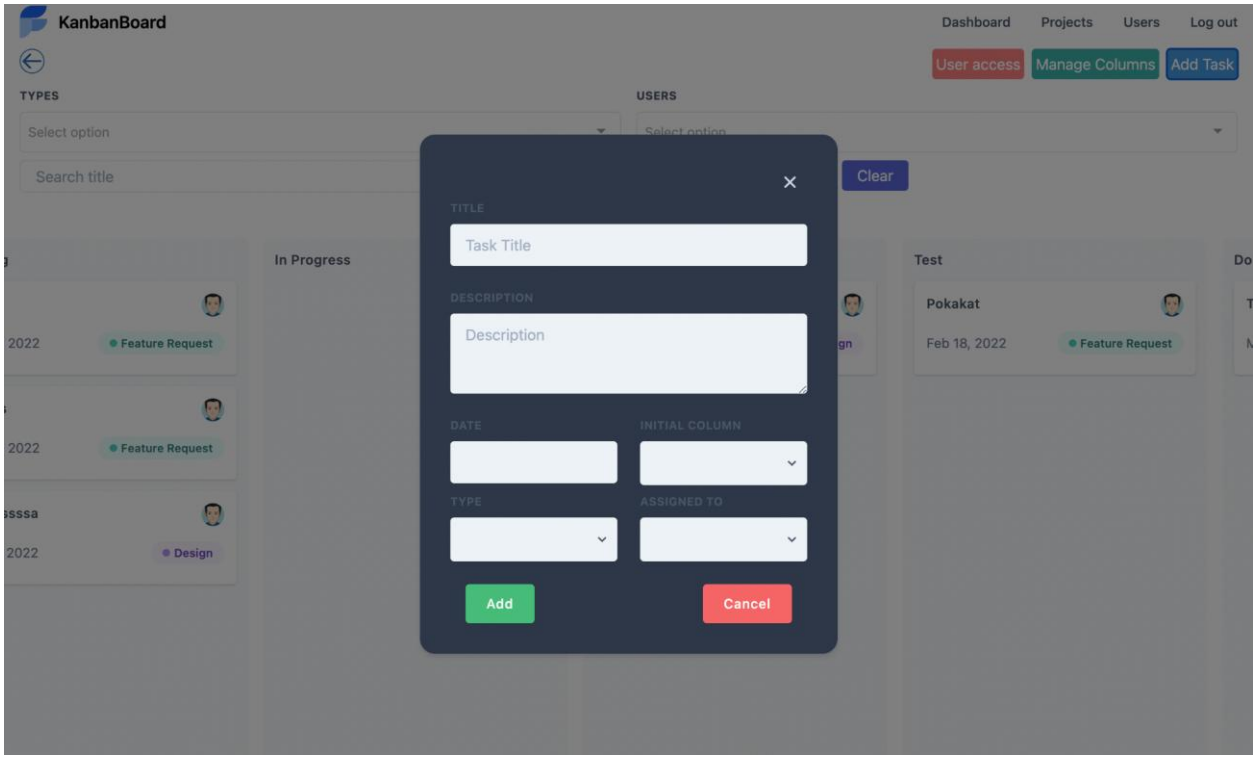

Рисунок 2.11 – Процес створення завдання

2.2 Технічний проєкт програмного забезпечення вебсайту компанії «Filiatix»

Проаналізувавши ескізний проєкт, був розроблений технічний проєкт програмного забезпечення шляхом побудови статичної та динамічної моделей. Статична модель представлена діаграмою класів, а динамічна – діаграмою взаємодії.

2.2.1 Логічна модель даних

Логічна модель даних представлена на рисунку 2.12.

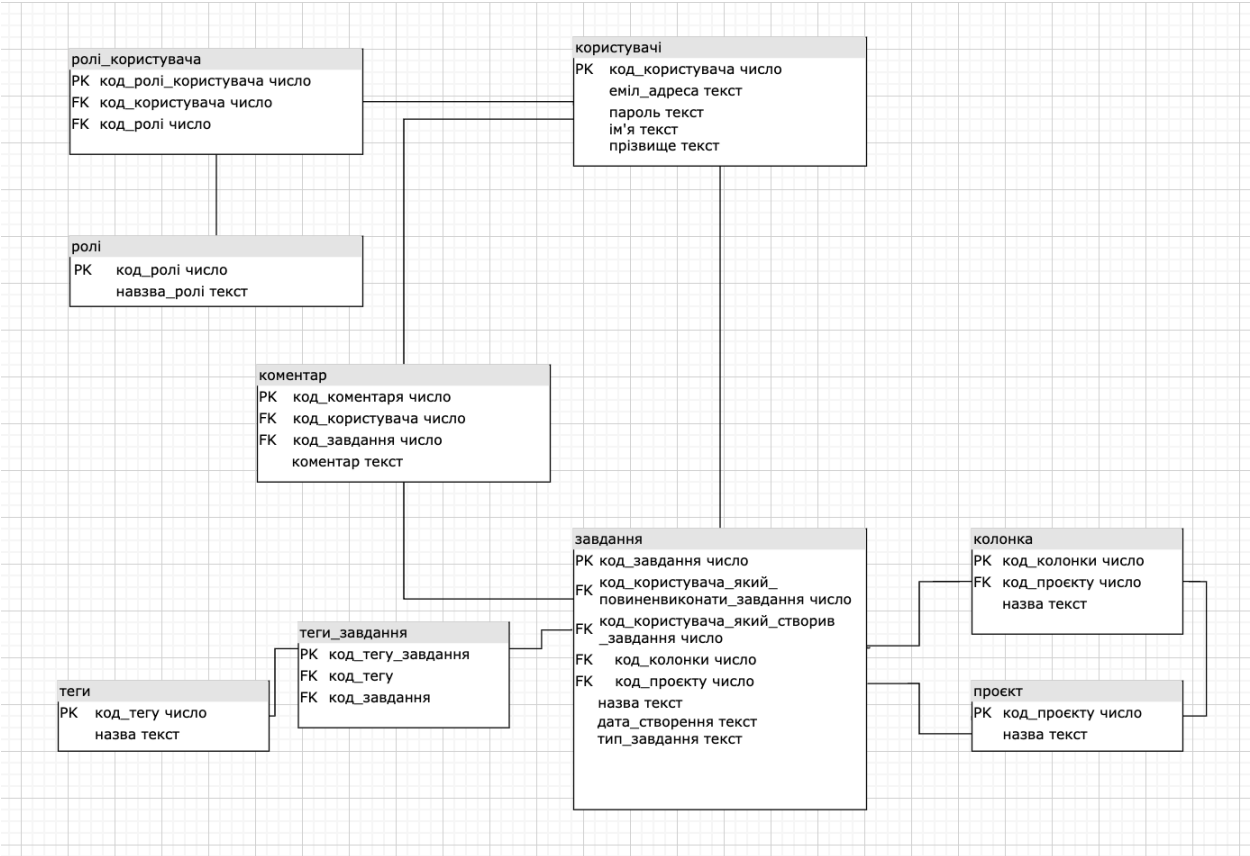

Рисунок 2.12 – Логічна модель даних

2.2.2 Структура класів

На основі моделі варіантів використання, інформаційної моделі та моделі програмного інтерфейсу була розроблена статична модель, представлена діаграмою класів, яка наведена на рисунку 2.13.

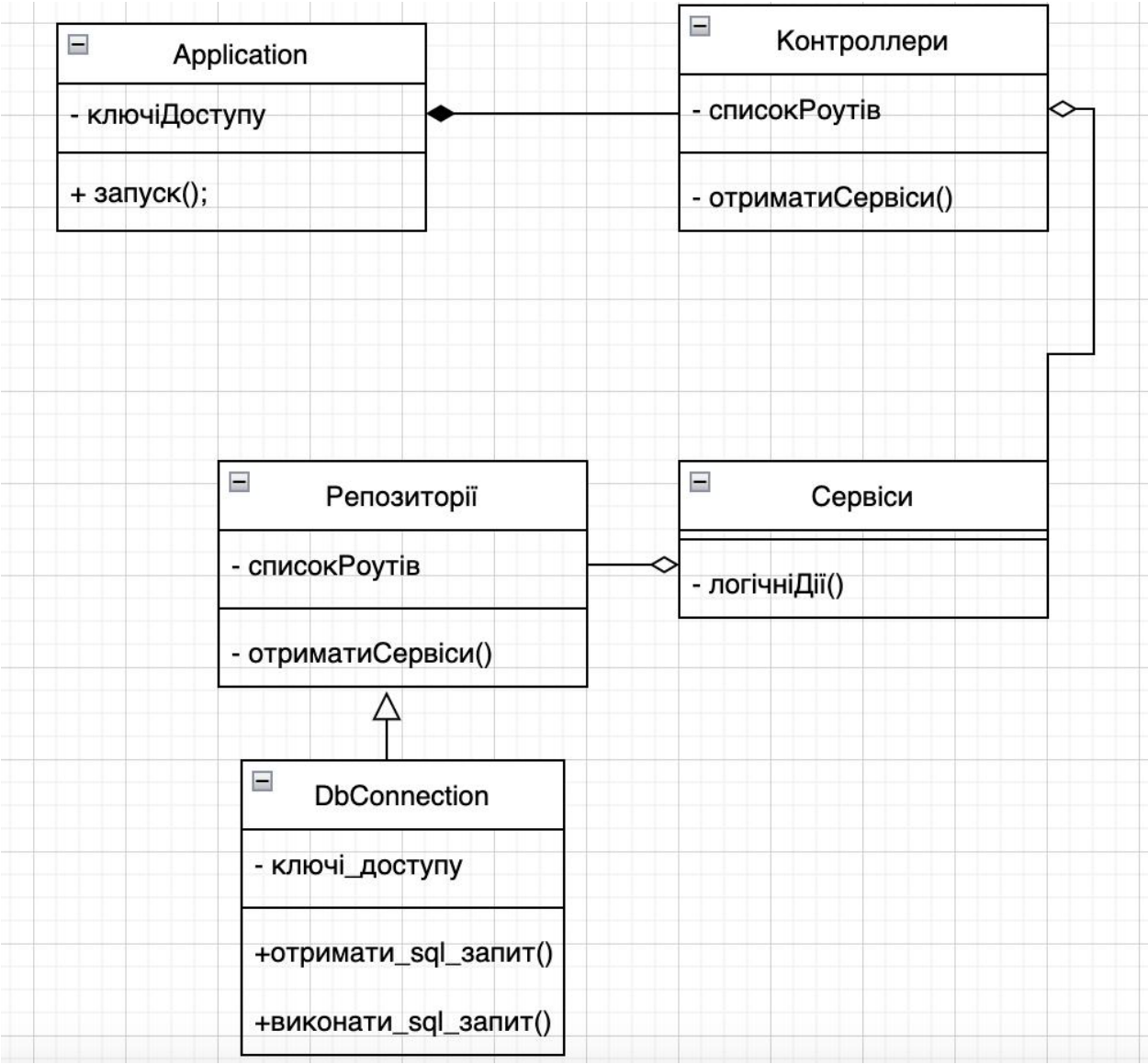

Рисунок 2.13 – Діаграма класів

2.2.3 Специфікації класів

Клас: «Application»;

Відповідальність: Головний клас, відповідає за працездатність програми;

Атрибути: accessKeys;

Нижче представлені функції класу:

- run() – головна функція, запускає роботу програми.

Клас: «Controllers»;

Відповідальність: Клас, що описує роботу об'єкта controllers;

Атрибути: listOfRoutes;

Нижче представлені функції класу:

- getServices() – функція, що дозволяє отримати інформацію про сервіси які підв’яжуться під контроллери.

Клас: «Services»;

Відповідальність: Клас, що описує роботу об’єкту services;

Нижче представлені функції класу;

- logicalActions() – функція, що дозволяє виконувати різноманітні дії пов’язані з логікою проєкту.

Клас: «Repository»;

Відповідальність: Клас що описує роботу об’єкту repository;

Атрибути: listOfRoutes;

Нижче представлені функції класу:

- getServices() – функція, що дозволяє отримати інформацію про сервіси які підв’яжуться під контролери.

Клас: «DbConnection»;

Відповідальність: Клас відповідальний за роботу із базою даних;

Атрибути: database_cridentials;

Нижче представлені функції класу:

- get_sql_query() – функція, що дозволяє отримати sql запити до таблиць;
- execute_sql() – функція, що дозволяє виконувати sql запити.

2.2.4 Динамічна модель програмного забезпечення

На основі діаграми варіантів використання та побудованої статичної моделі розробимо динамічну модель програмного забезпечення. На рисунку 2.14 зображена діаграма послідовності для варіанту використання «створення завдання».

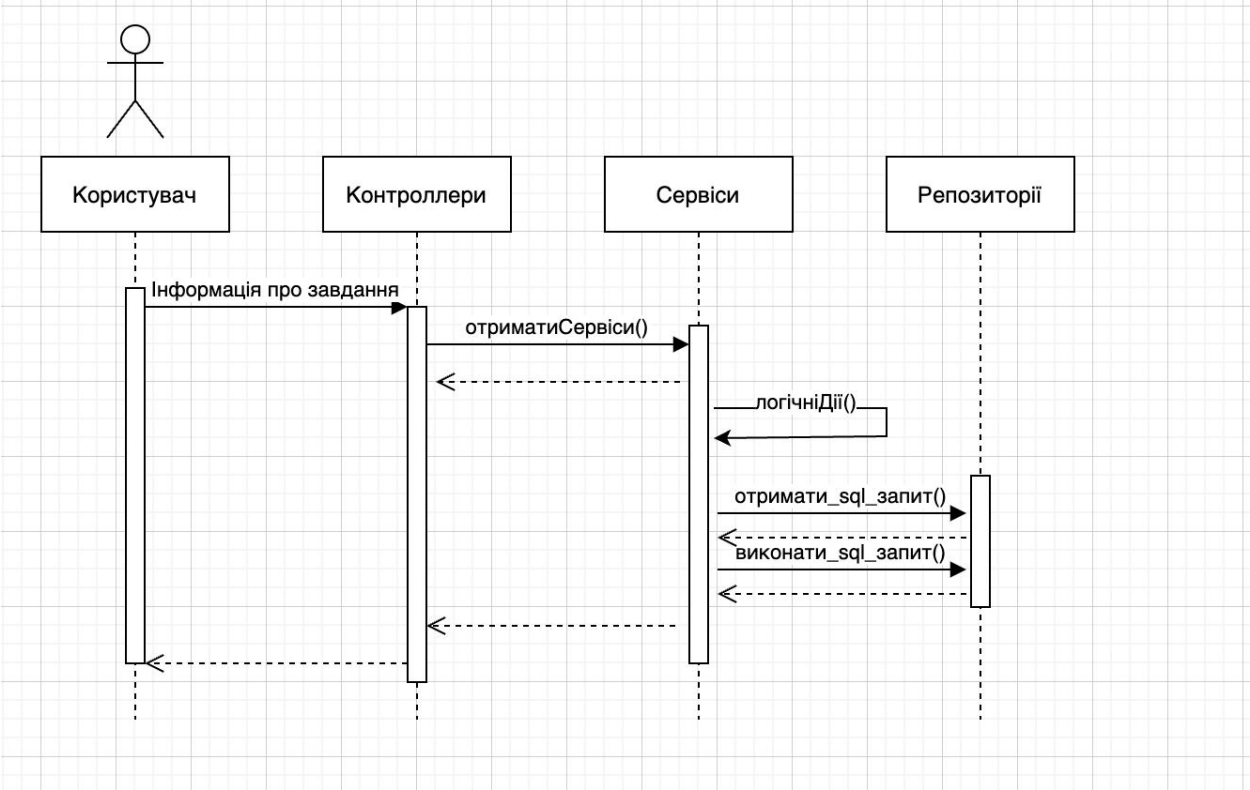

Рисунок 2.14 – Динамічна модель програмного забезпечення

2.3 Робочий проєкт програмного забезпечення вебсайту компанії «Filiatix»

2.3.1 Вибір мови програмування та системи керування бази даних

Для розробки даного програмного забезпечення мовою програмування було вирішено використати мову програмування Javascript, Node js та бібліотеку Vue js.

JavaScript – мультипарадигмальна мова програмування. Підтримує об'єктноорієнтований, імперативний та функціональний стилі. Є реалізацією специфікації ECMAScript. JavaScript зазвичай використовується як вбудована мова для програмного доступу до об’єктів програм.

Інтерпретатор JavaScript зчитує код файлу по кожному рядку окремо та по черзі. При цьому він відразу аналізує та перетворює цей код, прямо протягом «роботи» програми, і не зупиняє її виконання. На основі вже наявного AST, інтерпретатор створює байт-код. На цьому етапі жодної оптимізації не проводиться.

Переваги JavaScript:

- *Швидкість*. Оскільки JavaScript є «інтерпретованою» мовою, він скорочує час, який необхідний іншим мовам програмування, наприклад, Java для компіляції. JavaScript також є сценарієм на боці клієнта, який прискорює виконання програми, оскільки економить час, необхідний для підключення до сервера;
- *Простота мови.* JavaScript легко зрозуміти та вивчити. Структура схожа на більшість популярних мов такі як Java, C++. Також JavaScript дуже гнучка мова, заощаджує розробникам багато часу на розробку динамічного вмісту для Інтернету;
- *Популярність*. Оскільки всі сучасні браузери підтримують JavaScript, його можна побачити майже скрізь. Усі відомі компанії використовують JavaScript як інструмент, включаючи Google, Amazon, PayPal тощо;
- *Сумісність*. JavaScript ідеально співпрацює з іншими мовами програмування, тому багато розробників віддають перевагу йому при розробці багатьох програм. Ми можемо вбудувати його в будь-яку вебсторінку або в сценарій іншої мови програмування;
- *Багаті інтерфейси*. JavaScript надає розробникам різноманітні інтерфейси для створення привабливих вебсторінок. Компоненти або повзунки перетягуванням можуть надати вебсторінкам розширений інтерфейс. Це призводить до покращення інтерактивності користувача на вебсторінці;

- *Універсальність*. Тепер JavaScript можна розробляти як на зовнішньому, так і на внутрішньому плані. Внутрішня розробка використовує NodeJS, тоді як багато бібліотек допомагають у розробці інтерфейсу, як-от AngularJS, ReactJS, Vue js тощо;

Недоліки JavaScript:

- *Підтримка браузера*. У різних браузерах браузер по-різному інтерпретує JavaScript. Таким чином, перед публікацією код необхідно запустити на різних платформах. Старіші браузери не підтримують деякі нові функції;
- *Відсутність відлагодження*. Хоча деякі редактори HTML підтримують відлагодження, вони не настільки ефективні, як інші редактори, наприклад, C/C++, Java. Крім того, оскільки браузер дуже часто не показує більшість помилок, розробнику важко виявити проблему;
- *Єдине спадкування.* JavaScript підтримує лише одинарне, а не множинне. Для деяких програм може знадобитися ця характеристика об'єктноорієнтованої мови;
- *Повільна порозрядна функція*. JavaScript зберігає число як 64-розрядне число з плаваючою комою, а оператори працюють з 32-розрядними побітовими операндами. Таким чином, JavaScript перетворює число в 32-розрядні цілі числа зі знаком, оперує ними і перетворює їх назад у 64-розрядні числа JavaScript. Це безперервне перетворення займає більше часу для перетворення числа в ціле число. Це збільшує час, необхідний для виконання сценарію, і зменшує його швидкість;
- *Зупинка рендеру (відтворення)*. Одна помилка коду може зупинити відтворення всього коду JavaScript на вебсайті. Для користувача це виглядає так, ніби JavaScript відсутній. Однак браузери вкрай терпимі до цих помилок;

Node або Node.js – програмна платформа, заснована на движку V8 (що транслює JavaScript в машинний код), що перетворює JavaScript з вузькоспеціалізованої мови на мову загального призначення. Node.js додає можливість JavaScript взаємодіяти з пристроями введення-виводу через свій

API, написаний на C++, підключати інші зовнішні бібліотеки, написані різними мовами, забезпечуючи виклики до них із JavaScript-коду.

Переваги Node js:

- *Багата екосистема*. Одне слово – npm, стандартний менеджер пакетів Node.js, він також служить ринком інструментів JavaScript з відкритим кодом, що відіграє важливу роль у розвитку цієї технології. На даний момент в реєстрі npm доступно близько 836 000 бібліотек, а щотижня публікується понад 10 000 нових бібліотек, екосистема Node.js досить багата. Та ж статистика вказує, що 97 відсотків сучасних вебсайтів складаються з модулів npm. І це є доказом його безперечної популярності серед розробників. Завдяки такій різноманітності безкоштовних інструментів, доступних за кілька натискань, існує величезний потенціал для використання Node.js. У той же час програмне забезпечення з відкритим кодом набуває все більшої популярності, оскільки дозволяє створювати нові рішення, зменшуючи загальні витрати на розробку та час виходу на ринок;
- *Сильна корпоративна підтримка*. Як згадувалося вище, розробка Node.js була підтримана Joyent. У 2015 році було створено Node.js Foundation, щоб «забезпечити широке впровадження та допомогти прискорити розвиток Node.js». Засновниками організації стали IBM, Microsoft, PayPal, Fidelity і SAP. Список організацій, які використовують Node.js у виробництві, постійно зростає. Наразі до нього входять майже триста відомих компаній, таких як PayPal, Medium, Trello, Uber та Zendesk;
- *Безперебійна підтримка JSON.* Хоча інші серверні технології, такі як PHP і Ruby on Rails, можуть використовувати формат JSON для спілкування, Node.js робить це без перетворення між двійковими моделями і використовує JavaScript. Це особливо зручно, коли вам потрібно створити RESTful API для підтримки баз даних NoSQL, наприклад MongoDB – літера M у стеку MEAN. Це безперебійне спілкування з одним із основних стандартів передачі даних є ще однією перевагою екосистеми JavaScript;

Недоліки Node js:

- *Вузькі місця в продуктивності з важкими обчислювальними завданнями.* Найбільшим недоліком Node.js навіть зараз є його нездатність обробляти завдання, пов'язані з процесором. Але щоб зрозуміти коріння цієї проблеми, нам потрібен контекст. Почнемо з основ: із безпосередньо JavaScript. **Node.js** – це середовище виконання, яке виконує JavaScript на стороні сервера. Будучи мовою програмування інтерфейсу, JavaScript використовує один потік для швидкої обробки завдань. Для його роботи не потрібне використання потоків, оскільки завдання в JavaScript є легкими і займають мало ЦП;
- *Пекельна проблема зворотного дзвінка (Callback hell issue)*. Через свою асинхронну природу Node.js значною мірою покладається на зворотні виклики, функції, які запускаються після завершення кожного завдання в черзі. Збереження ряду завдань у черзі у фоновому режимі, кожна зі своїм зворотним викликом, може призвести до так званого «пекла зворотного виклику», що безпосередньо впливає на якість коду. Кажучи простими словами, це «ситуація, коли зворотні виклики вкладені в інші зворотні виклики на кілька рівнів, що потенційно ускладнює розуміння та підтримку коду»;
- *Незрілість оснастки*. Хоча основні модулі Node.js досить стабільні і їх можна вважати зрілими, у реєстрі npm є багато інструментів, які або поганої якості, або не задокументовані/перевірені належним чином. Більше того, сам реєстр недостатньо добре структурований, щоб пропонувати інструменти на основі їх рейтингу чи якості. Тому може бути важко знайти найкраще рішення для ваших цілей, не знаючи, що шукати;

Vue.js – JavaScript фреймворк з відкритим вихідним кодом для створення інтерфейсів користувача. Легко інтегрується у проєкти з використанням інших JavaScript-бібліотек. Може функціонувати як фреймворк для розробки односторінкових програм у реактивному стилі.

Як і будь-яка популярна технологія, Vue.js викликає суперечки у всьому спільноті програмістів. Загалом існують 3 основні JavaScript фреймворки: Vue, React, Angular. І є причини, чому Vue став другим найпопулярнішим фреймворком у 2021 році. **Ці причини**:

- *Маленький розмір*. Vue завантажений zip з фреймворком важить 18 КБ. Маючи легку вагу, фреймворк не тільки швидко завантажує та встановлює бібліотеку, але й позитивно впливає на ваше SEO та UX;
- *Віртуальний рендеринг і продуктивність DOM*. **Об'єктна модель документа (DOM)** – це представлення сторінок HTML з його стилями, елементами та вмістом сторінки як об'єктами. Об'єкти, що зберігаються у вигляді деревоподібної структури, генеруються браузером під час завантаження сторінки.

Коли користувач взаємодіє зі сторінкою, об'єкти змінюють свій стан, тому браузер повинен оновлювати інформацію та відображати її на екрані. Але оновлення всього DOM є громіздким. Заради швидкості Vue.js використовує віртуальну DOM: це як копія оригінальної DOM, яка визначає, які елементи оновлювати, без повторного відтворення всього DOM. Такий підхід робить рендеринг сторінки досить швидким і покращує продуктивність програми;

- *Однофайлові компоненти та читабельність.* Кожна частина вашої майбутньої програми/вебсторінки у Vue є компонентом. Компоненти представляють інкапсульовані елементи вашого інтерфейсу. У Vue.js компоненти можна писати на HTML, CSS та JavaScript, не розділяючи їх на окремі файли.

Поділ коду програми насправді є архітектурним підходом, який називається Component Based Architecture (CBA), і він також використовується в Angular і React, але саме Vue з самого початку підштовхує нас до такого підходу. Такий архітектурний підхід має масу **переваг:**

- *Можливість повторного використання компонентів.* Інкапсульовані компоненти – це в основному фрагменти коду, які можна повторно використовувати як шаблони для подібних системних елементів.
- *Читабельність коду*. Оскільки всі компоненти зберігаються в окремих файлах (а кожен компонент є лише одним файлом), код легше читати та розуміти, що полегшує його обслуговування та виправлення.
- *Добре підходить для юніттестування*. **Юніттестування** – це дія контролю якості, спрямована на перевірку того, як найменші частини програми працюють самостійно. Наявність компонентів значно спрощує це завдання.
- *Можливості інтеграції та гнучкість.* Важливим аспектом будь-якої нової технології є можливість інтеграції з існуючими програмами. З Vue.js це просто, тому що він покладається лише на JavaScript і не вимагає ніяких інших інструментів для роботи.

Vue також дозволяє шаблони за бажанням: використовуючи HTML, JS або JSX (розширення синтаксису JavaScript). Серед компонентів і легкої природи Vue можна використовувати майже в будь-якому проєкті. І ми раді повідомити, що перехід з React або Angular не буде великою проблемою, оскільки внутрішня організація Vue, в основному, є поєднанням двох.

- *Легко навчатися*. Інструмент може отримати масове застосування лише тоді, коли його легко зрозуміти, як це може бути у випадку з вивченням Vue.js. Щоб почати кодування, Vue не вимагає глибоких знань про бібліотеки, JSX і TypeScript, як це часто буває з іншими інтерфейсними технологіями. Все, що вам потрібно для початку, це базові знання HTML, CSS і JavaScript, звичайно.

Недоліки Vue.js:

- *Мовний бар'єр*. Впровадження Vue на таких підприємствах, як Xiaomi і Alibaba, допомогло популяризувати фреймворк і створило попит на ринку праці. Оскільки Vue.js стає все більш популярним у Китаї, значна частина

його вмісту та обговорень, як не дивно, китайська. Це проблема для інженерів, які володіють лише англійською та українською мовами;

- *Відсутність підтримки масштабних проєктів.* Vue.js досі молодий фреймворк. Розмір спільноти та команди розробників все ще не може зрівнятися з більш зрілим Angular. Також не користується фінансовою підтримкою великих підприємств. Для того, щоб технологія була прийнята у великомасштабних проєктах, технологія повинна бути стабільною та сильно підтримуваною, щоб проблеми можна було швидко вирішувати. Хоча у Vue не так багато проблем з цим, і навіть є попит від таких підприємств, як IBM і Adobe, він все ще використовується у відносно невеликих проєктах;
- *Обмежені ресурси*. Хоча екосистема досить широка, і є всі необхідні інструменти для початку розробки за допомогою Vue, вона все ще не така велика, як React або Angular. Щоб бути більш точним, порівняння кількості плагінів, доступних для React і Vue.js: різниця в сотнях;
- *Брак досвідчених розробників*. **Vue.js** – відносно молода технологія, яка тільки почала набирати популярність. Доведеться почекати кілька років, поки його масове впровадження на ринок праці заповнений досвідченими розробниками Vue.js.

MySQL – вільна реляційна система управління базами даних. Розробку та підтримку MySQL здійснює корпорація Oracle, яка отримала права на торгову марку разом із поглиненою Sun Microsystems, яка раніше придбала шведську компанію MySQL AB.

Ця СУБД має низку **переваг**:

- *Знижена загальна вартість володіння* MySql одна з найпопулярніших систем управління базами даних з відкритим вихідним кодом, яка дозволяє керувати реляційною базою даних. Оскільки MySql з відкритим вихідним кодом, ви можете використовувати MySql безкоштовно і, якщо хочете, ви можете налаштувати його вихідний код відповідно до ваших

вимог. Більшість компаній віддають перевагу MySQL, оскільки їм не потрібно нічого платити за цей чудовий продукт;

- *Портативність*. MySQL є міжплатформним сервером баз даних. Він може працювати на різних платформах, таких як Linux, Solaris, Windows тощо. Це хороший вибір для тих проєктів, які орієнтовані на декілька платформ, зокрема вебсайти. Фактично, MySQL є частиною відомого стека серверів LAMP (Linux Apache MySQL PHP), який використовується у всьому світі для розробки вебсайтів. MySql підтримує багато платформ з різними мовами, такими як C, C++, PHP, PERL, JAVA, Python тощо;
- *Безперебійне підключення*. Для підключення до сервера MySQL доступні різні безпечні та безперебійні механізми підключення. Ці з'єднання включають іменовані канали, сокети TCP/IP і сокети UNIX;
- *Безпека даних.* MySQL є визнаною у всьому світі найбільш безпечною та надійною системою керування базами даних, яка використовується в популярних вебсайтах, включаючи WordPress, Drupal, Joomla, Facebook та Twitter. Дані, захищені паролем, і перевага цих паролів полягає в тому, що вони зберігаються в зашифрованому вигляді і не можуть зламати ці складні алгоритми шифрування.

Недоліки:

- *MySQL важко масштабувати*. MySQL не розроблений для масштабування, навіть його неможливо масштабувати, як і Facebook, але потрібні серйозні інженерні зусилля, щоб зробити це можливим, зазвичай вам потрібно багато зусиль, щоб він запрацював;
- *MySQL не призначений для даних великого розміру*. MySQL добре працює в більшості малих і середніх програм, але коли розмір даних зростає, продуктивність знижується. Коли дані зростають, тільки простий і індексований запит отримує хорошу продуктивність, для складного запиту він легко стає повільним, іноді навіть не в змозі виконати запит за можливий час очікування. Вам потрібно ретельно розробити свій SQL-запит, щоб він залишався доступним;

- *Не повна стандартність SQL*. MySQL не повністю відповідає стандарту SQL-92, MySQL не підтримує деякі стандартні функції і має деякі розширення, які не належать до стандартного SQL;
- *MySQL належить Oracle*. MySQL є продуктом з відкритим вихідним кодом, але тепер його придбала Oracle, яка має повний контроль над програмним забезпеченням, багато розробників нервують через ситуацію. Деякі з них звернулися до MariaDB.

Коли Oracle придбала Sun Microsystems, MySQL, яка належить Sun, також була продана Oracle. База даних Oracle в основному використовується в підприємствах і великих корпораціях, вона має очевидне домінування в цій області, але MySQL все ще залишається одним із конкурентів. Oracle опублікувала офіційну обіцянку підтримувати конкурентоспроможність MySQL, але ця обіцянка може не дотриматися.

Ці та інші особливості мов роблять розгортання та розробку додатків надзвичайно швидким.

2.3.2 Фізична модель даних

На етапі створення фізичної моделі детальне проєктування виконується з використанням діаграми компонентів. Діаграма компонентів описує особливості фізичного представлення системи, дозволяє визначити архітектуру розроблюваної системи, встановивши залежності між програмними компонентами. Компоненти відповідають представленню робочих екземплярів одиниць коду. На рисунку 2.15 наведена діаграма компонентів:

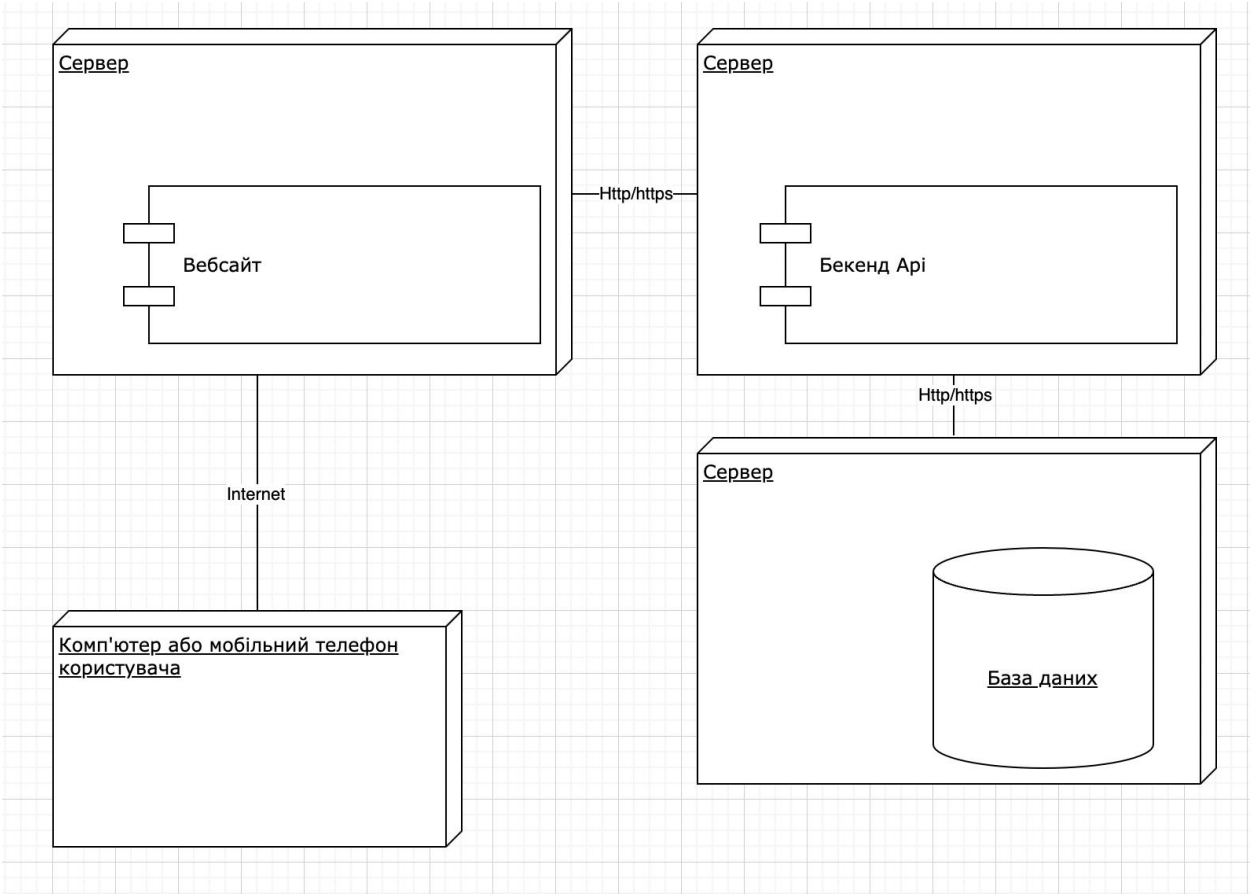

Рисунок 2.15 – Діаграма компонентів

Фізична модель даних представлена на рисунку 2.16.

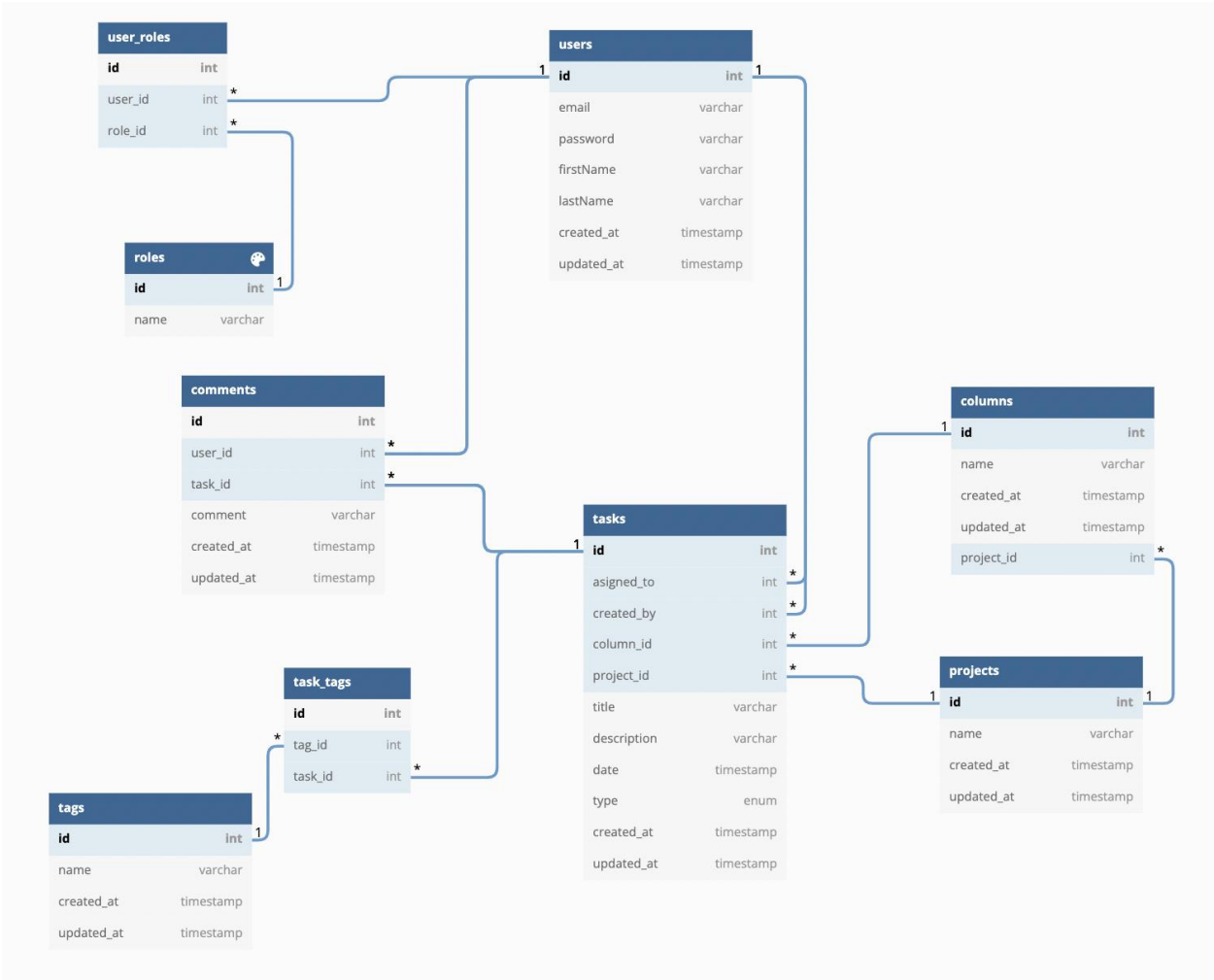

Рисунок 2.16 – Фізична модель даних

2.3.3 Тестування варіантів використання

Для тестування програмного забезпечення були обрані лише основні функції, а саме такі варіанти використання, як: «Створення завдання», «Авторизація».

Тестування варіанта використання «Створення завдання»

При створенні завдання, користувачеві необхідно заповнити 5 полів: Назва завдання, опис завдання, тип завдання, дата завдання, колонка завдання в якій спочатку буде завдання. Правильними вхідними даними для даних полів є:

- Назва завдання – текст;

- Опис завдання – текст;
- Тип завдання – текст, який передбачає відповідність з можливими типами завдань у системі;
- Дата завдання – дата (timestamp);
- Колонка завдання – номер, який передбачає відповідність з можливими колонками завдань у проєкті.

Для даного варіанту використання створимо 3 тести:

Тест 1.

Вхідні дані:

- Назва завдання – пустий текст;
- Опис завдання – текст;
- Тип завдання – текст, який передбачає відповідність з можливими типами завдань у системі;
- Дата завдання – дата (timestamp);
- Колонка завдання – номер, який передбачає відповідність з можливими колонками завдань у проєкті.

Очікуваний результат:

- На етапі введення назви завдання, програма повинна повідомити про помилку, та запропонувати користувачеві ввести назву;
- На етапі введення інформації завдання не повинно виникнути ніяких помилок.

Результат тестування представлений на рисунку 2.17:

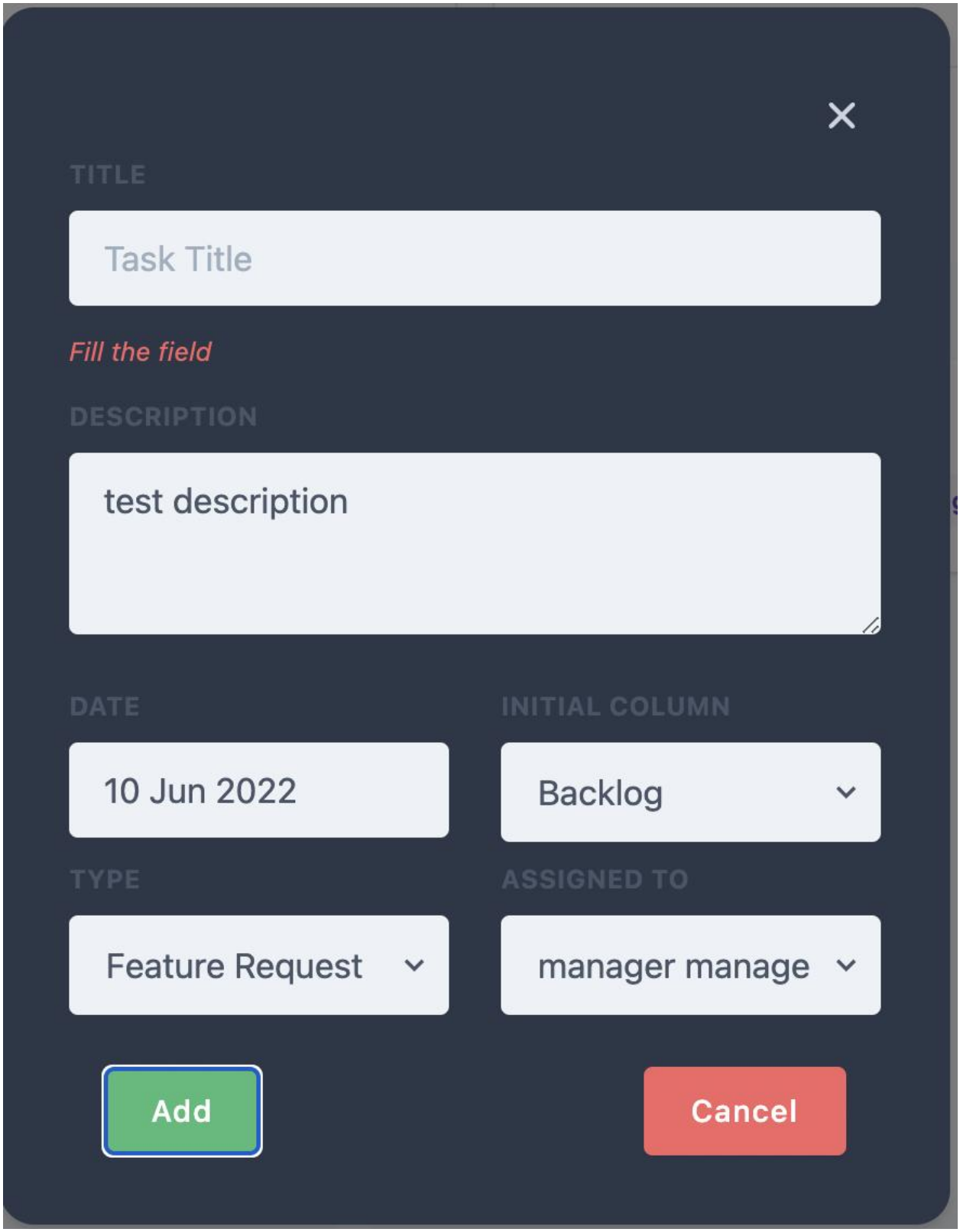

Рисунок 2.17 – Результат тестування тесту 1 варіанту використання «Створення завдання»

Тест 2.

Вхідні дані:

- Назва завдання – текст;
- Опис завдання – текст;
- Тип завдання – текст, який передбачає відповідність з можливими типами завдань у системі;
- Дата завдання – дата (timestamp);
- Колонка завдання – відсутня.

Очікуваний результат:

- На етапі введення назви завдання, програма повинна повідомити про помилку, та запропонувати користувачеві ввести колонку завдання;
- На етапі введення інформації завдання не повинно виникнути ніяких помилок.

Результат тестування представлений на рисунку 2.18:

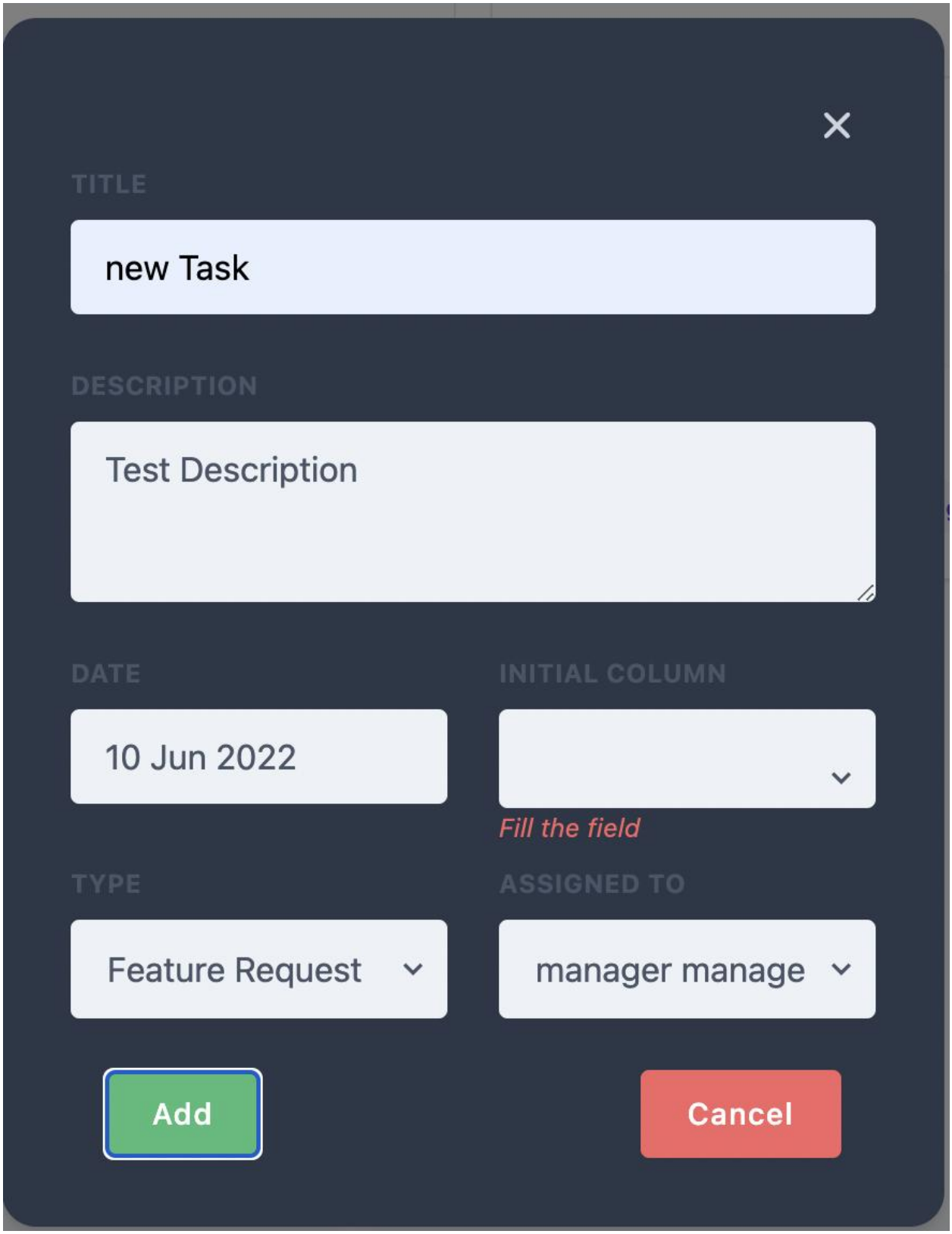

Рисунок 2.18 – Результат тестування тесту 2 варіанту використання «Створення завдання»

Тест 3.

Вхідні дані:

- Назва завдання – відсутня;
- Опис завдання – відсутня;
- Тип завдання – відсутня;
- Дата завдання – відсутня;
- Колонка завдання – відсутня.

Очікуваний результат:

- На етапі введення назви завдання, програма повинна повідомити про помилку, та запропонувати користувачеві ввести повну інформацію;
- На етапі введення інформації завдання не повинно виникнути ніяких помилок.

Результат тестування представлений на рисунку 2.19:

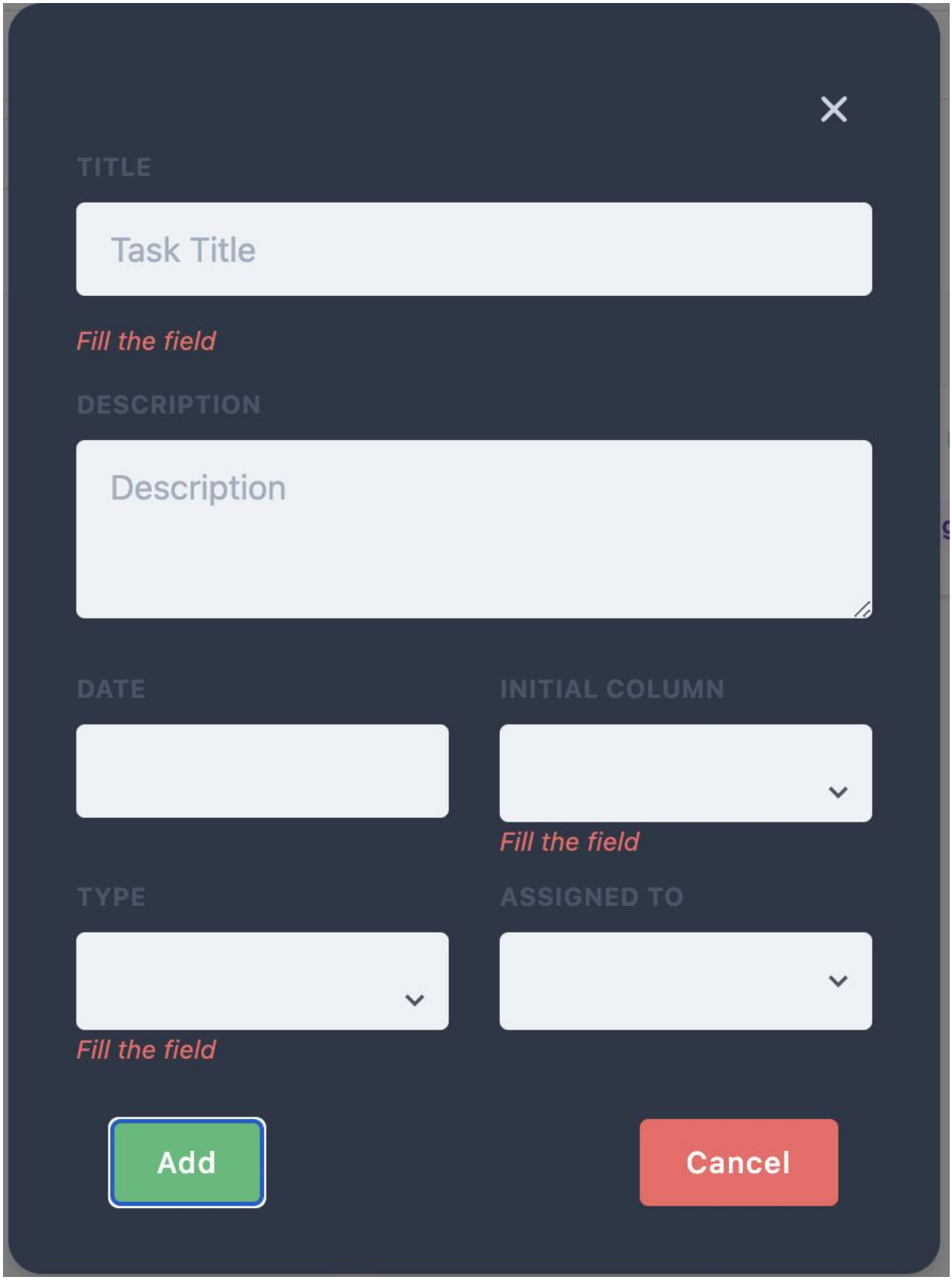

Рисунок 2.19 – Результат тестування тесту 3 варіанту використання «Створення завдання»

Отже, в ході тестування варіанту використання «Створення завдання» не було знайдено ніяких помилок.

Тестування варіанта використання «Авторизація»

У вебсайті можна авторизуватися як «користувач», «адміністратор», «суперадміністратор».

При авторизації користувачеві необхідно заповнити 2 поля: Логін та пароль. Правильними вхідними даними для даних полів є:

- Логін – текст;
- Пароль – текст.

Для даного варіанту використання створимо 3 тести:

Тест 1.

Вхідні дані:

- Логін – неіснуючий логін;
- Пароль – неіснуючий пароль.

Очікуваний результат:

- На етапі введення назви завдання, програма повинна повідомити про помилку, та запропонувати користувачеві ввести існуючий логін та пароль;
- На етапі введення інформації авторизації не повинно виникнути ніяких помилок.

Результат тестування представлений на рисунку 2.20:

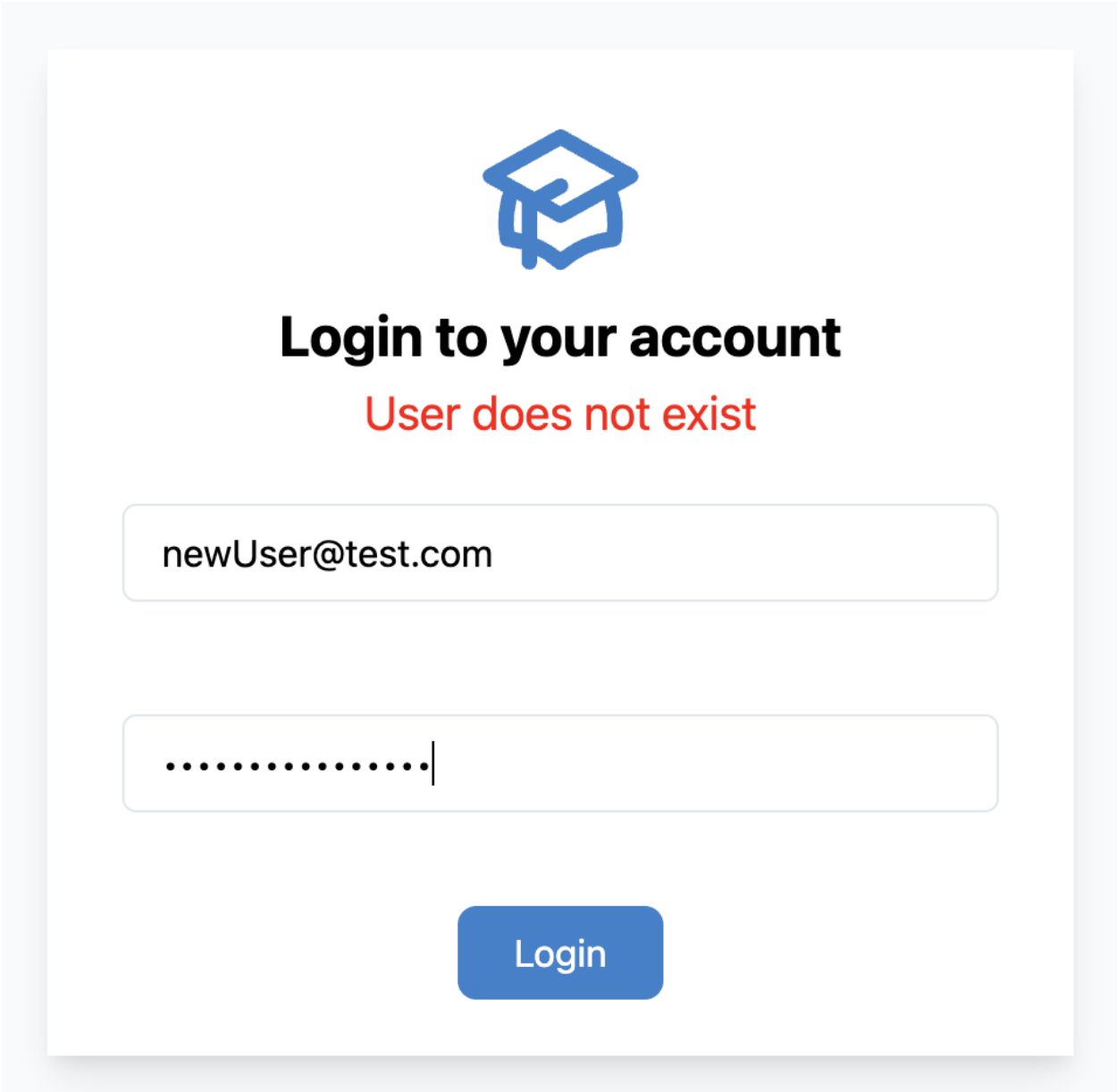

Рисунок 2.20 – Результат тестування тесту 1 варіанта використання «Авторизація»

Тест 2.

Вхідні дані:

- Логін – текст;
- Пароль – пусте поле.

Очікуваний результат:

- На етапі введення назви завдання, програма повинна повідомити про помилку, та запропонувати користувачеві ввести існуючий логін та пароль;

- На етапі введення інформації авторизації не повинно виникнути ніяких помилок.

Результат тестування представлений на рисунку 2.21:

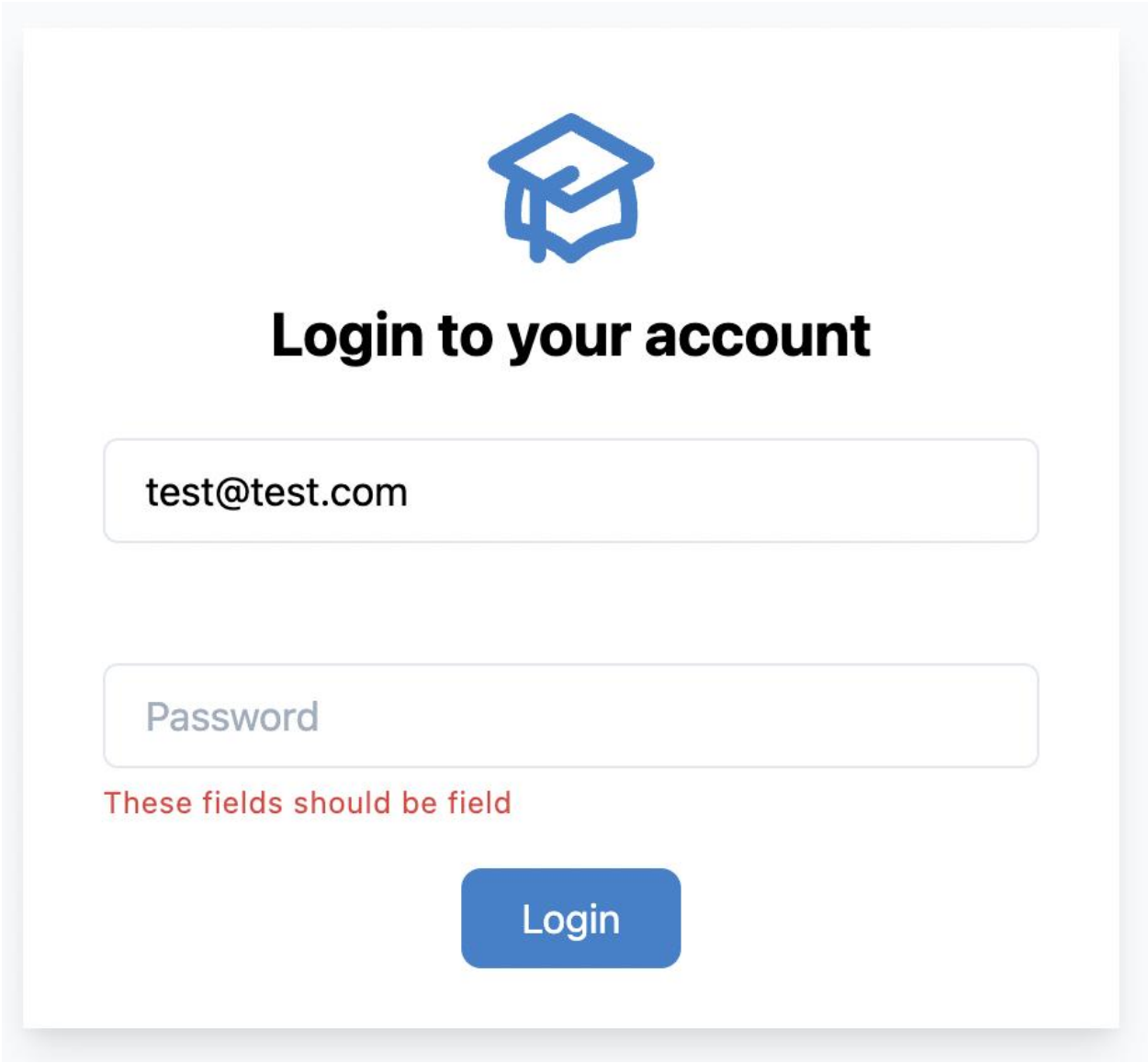

Рисунок 2.21 – Результат тестування тесту 2 варіанту використання «Авторизація»

Тест 2.

Вхідні дані:

- Логін – пусте поле;
- Пароль – пусте поле.

Очікуваний результат:

- На етапі введення назви завдання, програма повинна повідомити про помилку, та запропонувати користувачеві ввести існуючий логін та пароль;
- На етапі введення інформації авторизації не повинно виникнути ніяких помилок.

Результат тестування представлений на рисунку 2.22:

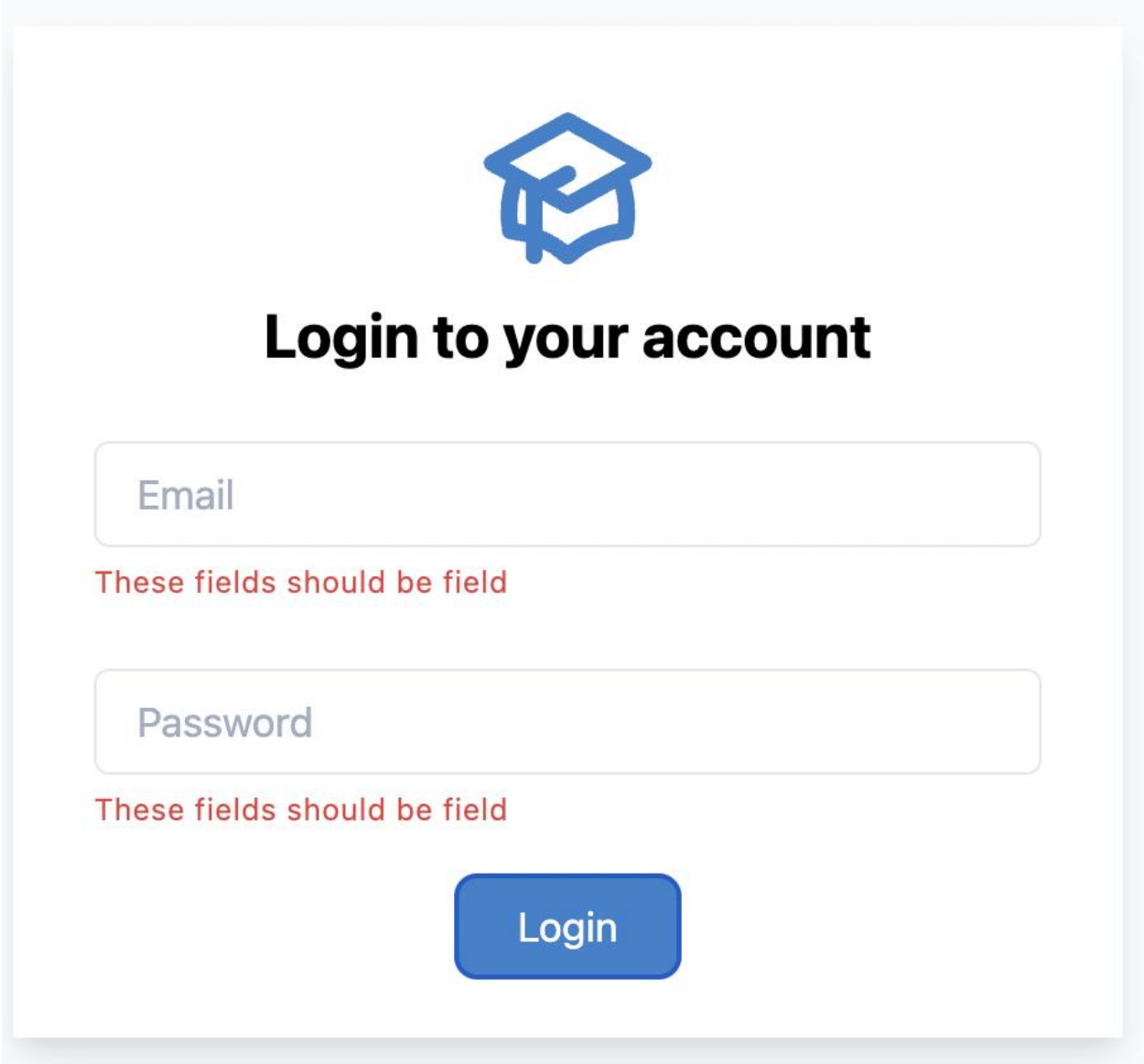

Рисунок 2.22 – Результат тестування тесту 3 варіанта використання «Авторизація»

Отже, в ході тестування варіанту використання «Авторизація» не було знайдено ніяких помилок.

2.3.4 Випробування програми

Програму і методику випробувань розробленого програмного забезпечення наведено в Додатку В. На основі інструкцій, що наведені в цьому додатку, проведемо випробування розробленого програмного забезпечення.

Коли користувач заходить на головну сторінку вебсайту. Головна сторінка зображена на рисунку 2.23:

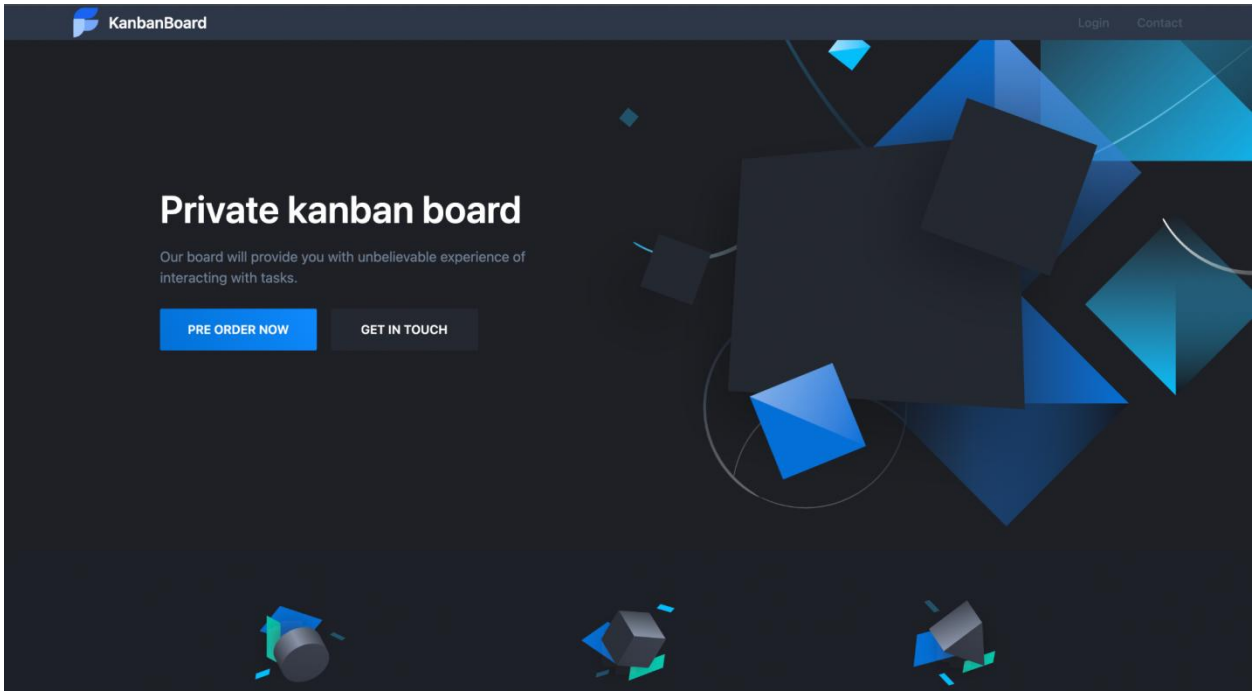

Рисунок 2.23 – Головна сторінка

Щоб почати взаємодію з основною частиною вебсайту йому потрібно увійти до свого профілю, який заздалегідь був створений адміністратором або суперадміністратором. Сторінка логіну зображена на рисунку 2.24:

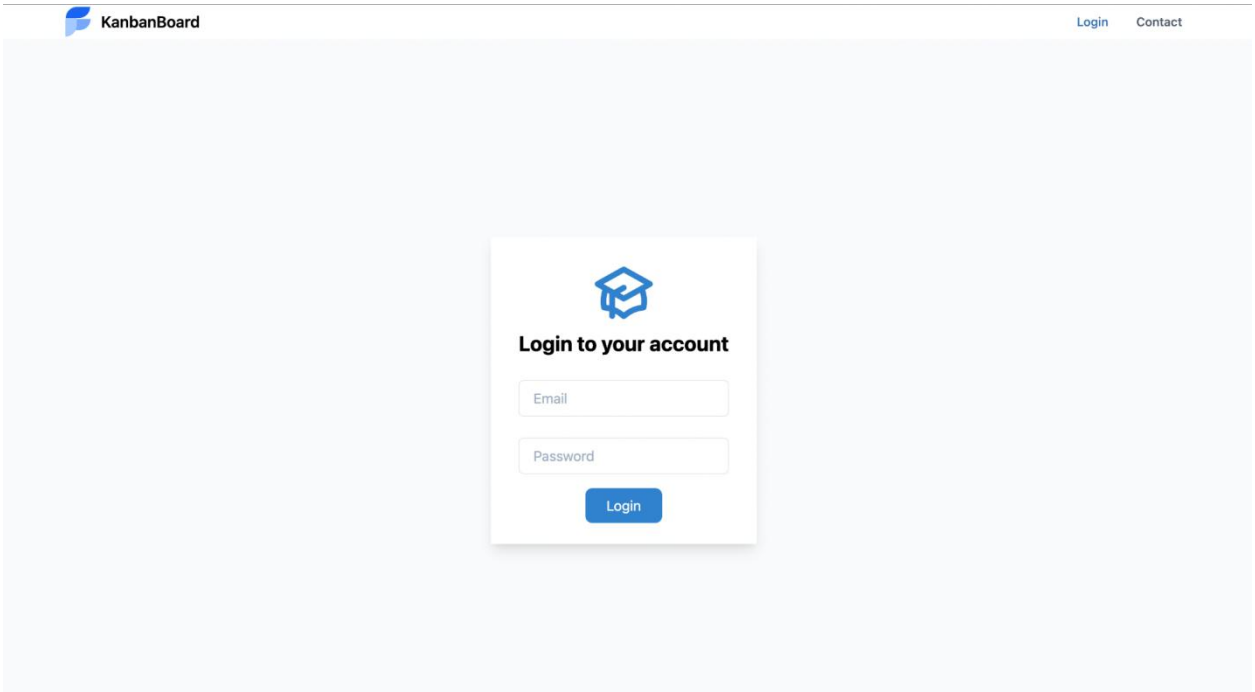

Рисунок 2.24 – Сторінка логіну

Після того як користувач авторизувався він має доступ до основного функціоналу вебсайту. Якщо користувач має роль адміністратора або суперадміністратора він має можливість створити проєкт. Створення проєкту зображено на рисунку 2.25:

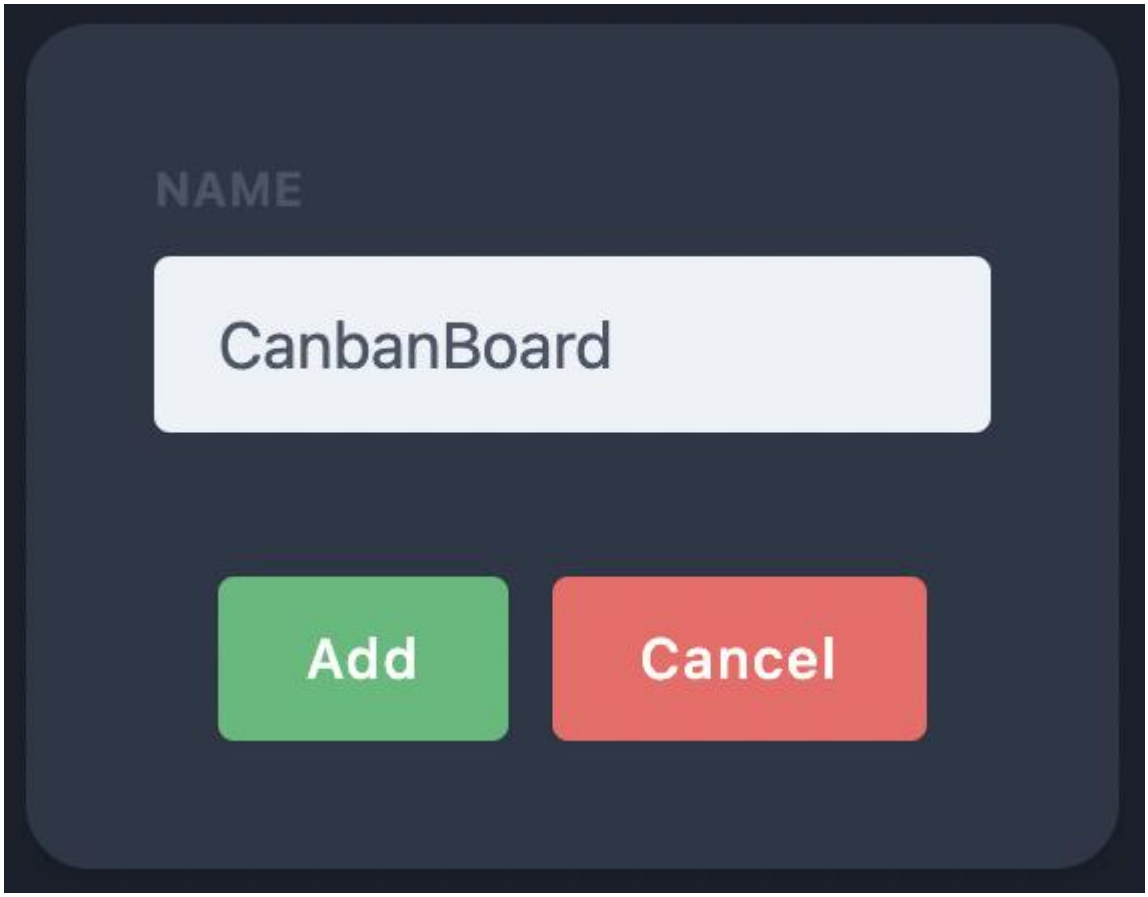

Рисунок 2.25 – Створення проєкту

Після того як необхідний проєкт був створений або користувач був доданий до необхідного проєкту, він має можливість створити завдання. Процес створення завдання та результат зображені на рисунках 2.26 та 2.27 відповідно:

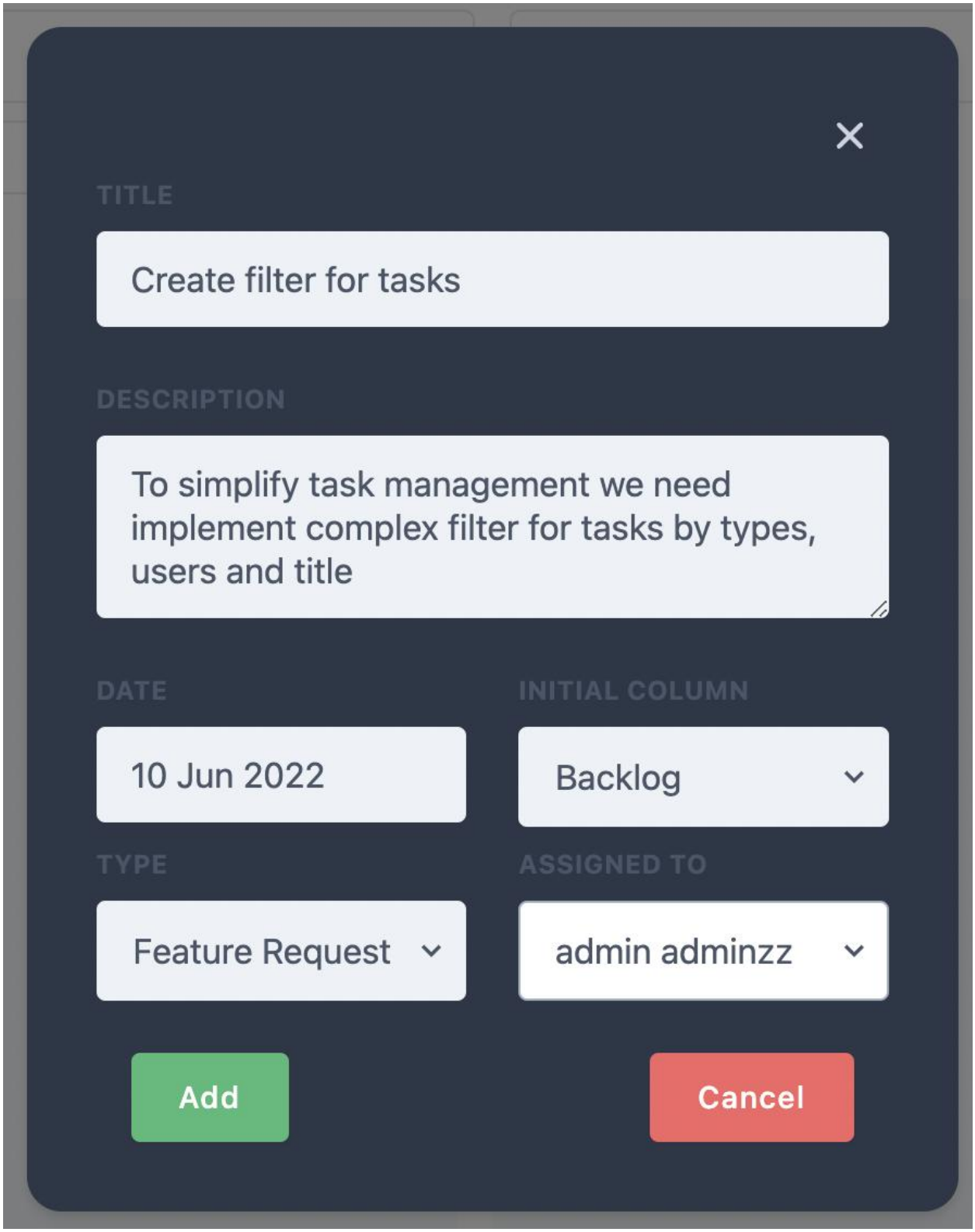

Рисунок 2.26 – Створення завдання

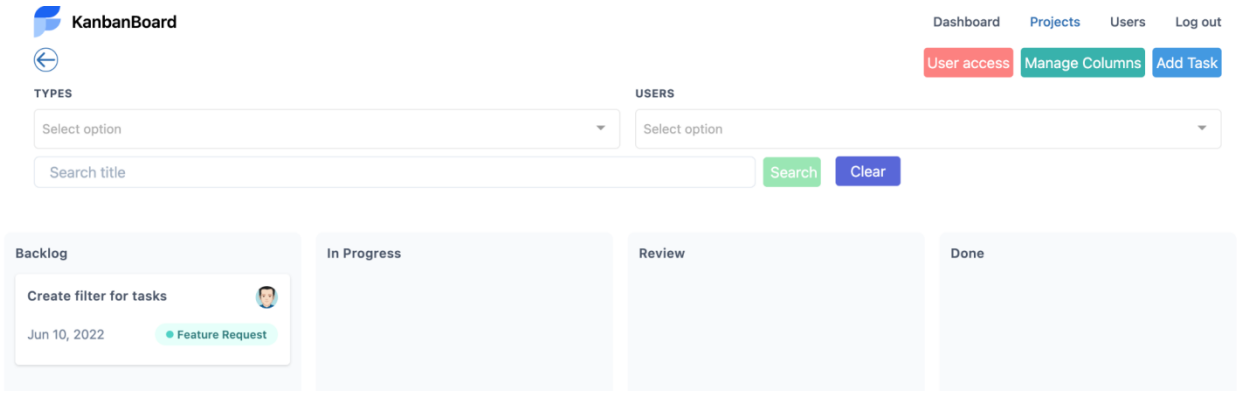

Рисунок 2.27 – Результат створення завдання

Після того як завдання було створено або якщо користувач починає роботу над завданням ми маємо змогу відредагувати завдання. Відредагувати завдання можемо декількома способами, обидва способи відрізняються. Перший спосіб редагування інформації про завдання, а другий зміна колонки завдання. Процес редагування завдання та результат зображені на рисунках 2.28, 2.29, 2.30 та 2.31 відповідно:

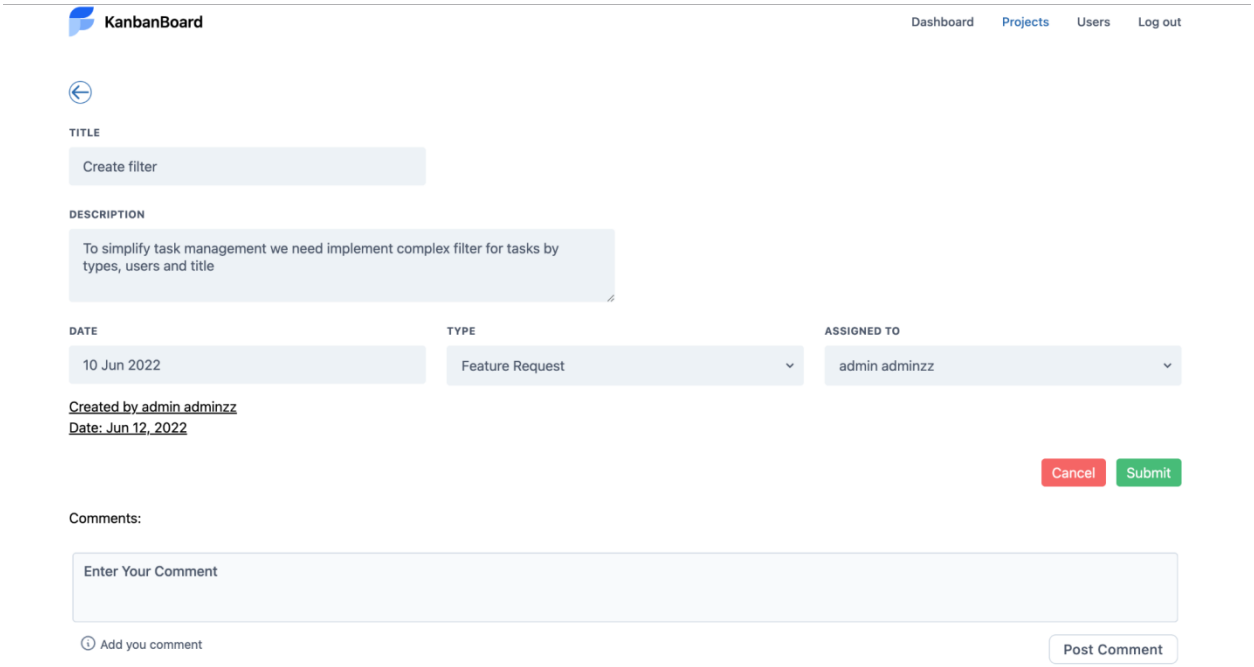

Рисунок 2.28 – Процес редагування завдання

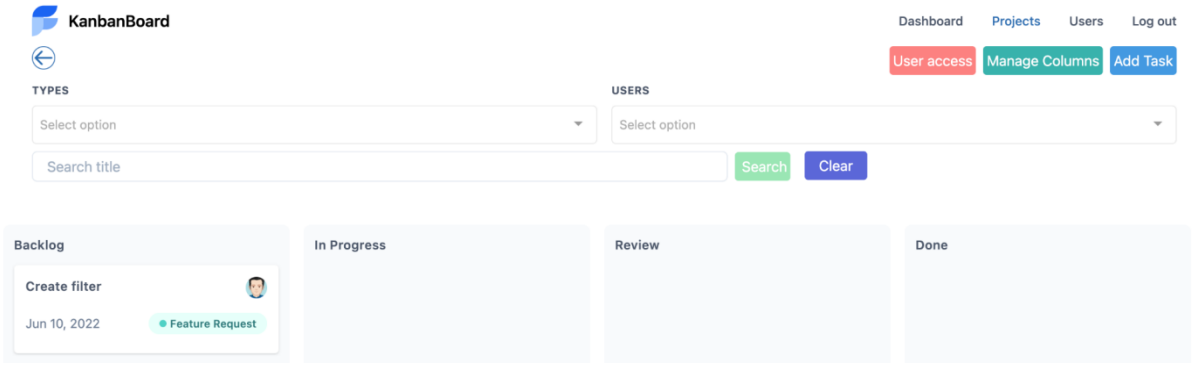

Рисунок 2.29 – Результат редагування завдання

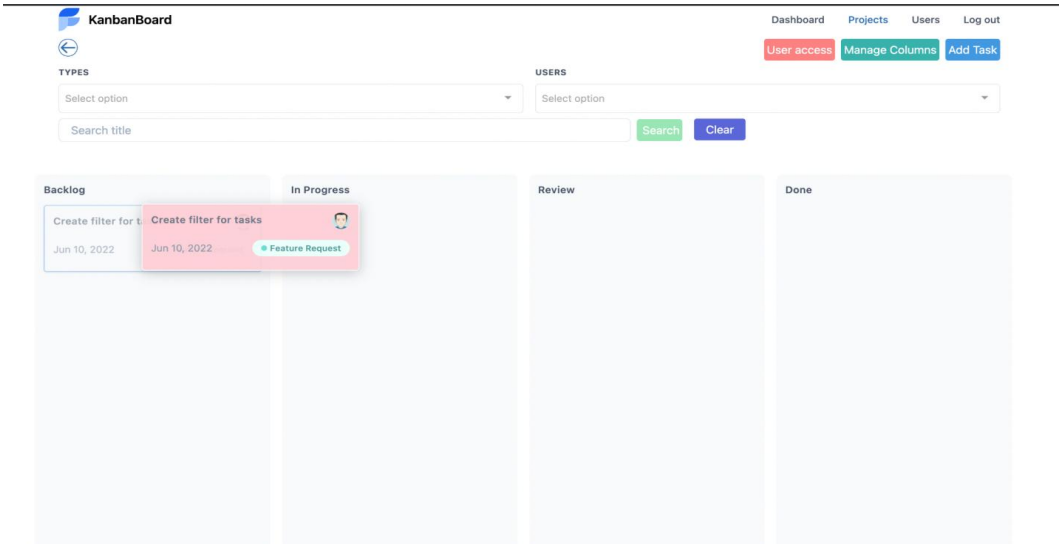

Рисунок 2.30 – Процес редагування колонки завдання

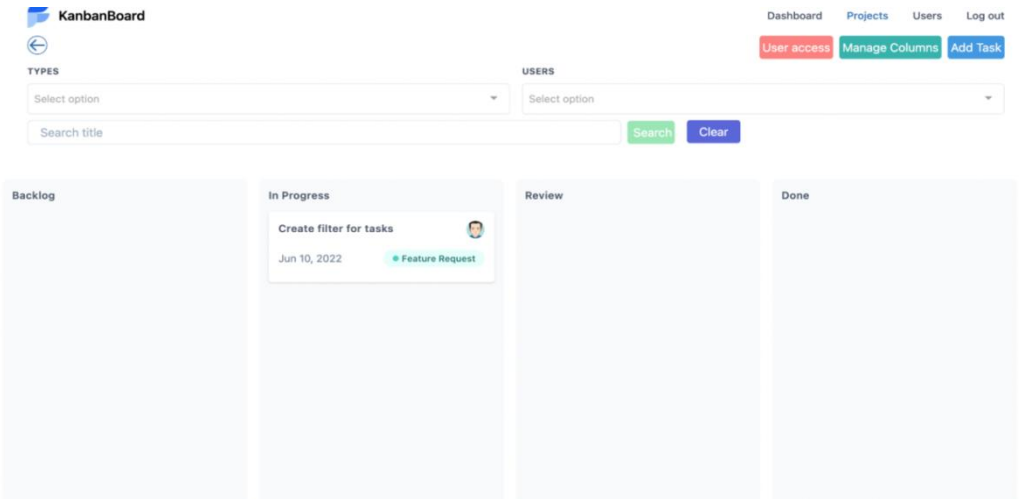

Рисунок 2.31 – Результат редагування колонки завдання

Отже, програма пройшла всі випробування та в ході випробування не було знайдено жодних помилок.

РОЗДІЛ 3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБСАЙТУ КОМПАНІЇ «FILIATIX»

В процесі виконання дипломної роботи був розроблений та описаний проект, програмне забезпечення вебсайту компанії «Filiatix» для контролю робочого процесу, а також методика тестування для проекту, завдяки якої було протестоване розроблене програмне забезпечення. Розроблене програмне забезпечення задовольняє вимогам, які поставлені у технічному завданні.

Програмне забезпечення розроблено мовою програмування Nodejs та Javascript. Текст програми складається із 10666 рядків і займає 452 Mb дискового простору. Проведене тестування програмного забезпечення продемонструвало повну працездатність програми, відповідність її характеристик та виконуваних функцій технічному завданню, готовність до впровадження. Випробування були виконані згідно з програмою та методикою випробувань, яку наведено у Додатку В. Програмне забезпечення сервера вимагає встановлення інтерпретатора Nodejs та Javascript. Для роботи з даним вебсайтом користувач повинен лише встановити будь-який сучасний браузер на свій пристрій. Інструкція користувача для роботи з даним телеграм ботом наведена в Додатку Д.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ

Поняття охорони праці визначається ст. 1 Закону України «Про охорону праці». «**Охорона праці** – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці».

4.1 Фактори ризику

4.1.1 Навантаження на зір

Однією з причин підвищеного навантаження на органи зору є необхідність ретельного розглядання дрібних об’єктів на близькій відстані від ока. Теорія акомодації каже, що фокусування на рівновіддалені від очей об’єкти відбувається лише шляхом зміни форми кришталика, але останні дослідження показали, що при цьому також відбувається й подовження самого очного яблука. Залежність подовження очного яблука щодо відстані до об’єкта спостереження є нелінійною. Для менших відстаней, деформація є значною, а для великих відстаней, як 40-50 см деформація незначна. Тривалі підвищені деформації за межами стандартної пружності очного яблука призводять до накопичення залишкових деформацій що призводить до виникнення і розвитку короткозорості. Окрім явищ, пов’язаних із зоровою діяльністю, довготривала робота за комп’ютером має інші фактори, що значно збільшують навантаження на очі користувача.

По-перше, доводиться тривалий час дивитися безпосередньо на джерело яскравого світла – монітор (дисплей). Також, доводиться досить часто переходити з одного способу читання на інший.

По-друге, відбувається оновлення поверхні дисплея та точок зображення з певною частотою. Зазвичай шляхом інерційності зору це явлення виявляється майже непомітним, проте, навантаження на орган зору буде тим вище, чим нижче частота оновлення зображення.

4.1.2 Особливості робочої пози

Травми, спричинені повторюваними навантаженнями За останніми дослідженнями, довготривала та інтенсивна робота за такими пристроями як клавіатура та комп'ютерна миша може стати джерелом тяжких професійних захворювань. Ці захворювання, зумовлені так названою травмою повторюваних навантажень (ТВП), й розвиваються поступово. Легкий біль в руці, який вчасно не вилікували, може в результаті призвести навіть до інвалідності.

Профільні фахівці вважають, що природним та сталим положенням кистей людини є вертикальне, а не долонею вниз, як зазвичай при роботі з комп'ютером. Також при роботі за клавіатурою (мишею) однакові рухи повторюються зі сталою періодичністю, протягом довготривалого часу.

Захворювання, спричинені травмами повторюваних навантажень, містять хвороби м'язів, сухожиль, нервових закінчень рук. Найчастіше за все страждають кисті, зап'ястя та плечі, але може бути порушена робота й шийних областей рук. У людей, які багато працюють з обчислювальною технікою (ОТ), захворювання зазвичай настають як результат довготривалої роботи за незручно або неправильно розташованою клавіатурою, наприклад, при занадто високому чи низькому розташуванні поверхні столу або погано підібраному кріслі.

Тривала гіподинамія.

Будь-яка поза при довготривалій фіксації веде до застою крові у внутрішніх органах і капілярах, та є шкідливою для опорно-рухового апарату людини. Тому дуже важливим є правильно підібрати позу та намагатися

періодично її змінювати, щоб запобігти застою крові у внутрішніх органах і капілярах.

Не фізіологічне положення різних частин тіла.

Фізіологічним положення тіла людини є так названий ембріональний стан, якщо повністю розслабитися в солоній воді, ви зможете з легкістю випробувати його на собі. В тому стані коли м'язи всього тіла повністю розслаблені та на них впливає тільки природний тонус спокою, тіло приходить в конкретне положення (так названий ембріональний стан). Для спинного та шийного відділу у вертикальному положенні фізіологічне положення інше, в якому дуже виражені шийний та поперековий вигини хребта, як пряма вертикальна лінія, яка проходить через потилицю та куприк.

Людині правильну поставу необхідно вивчити самостійно своїм тілом, а саме за допомогою його контролю протягом довготривалого часу, а згодом правильна постава буде підтримуватися автоматично. Простіше за все – встати до рівної стіни та дуже щільно притиснути до неї свої потилицю, лопатки, сідниці, ікри та п'яти. Прийти до ідеалу взагалі досить складно, в процесі роботи за комп'ютером, чи іншою технікою особливо, але без сумніву цього варто прагнути – хоча б для певних частин тіла людини.

4.2 Вимоги до апаратури та робочого місця

4.2.1 Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями:

- робочі місця працівників з екранними пристроями мають бути спроєктовані так і мати такі розміри, щоб працівники мали простір для зміни робочого положення та рухів;
- для забезпечення безпеки та захисту здоров'я працівників усе випромінювання від екранних пристроїв має бути зведене до гранично допустимого рівня (вплив на людину факторів довкілля - шуму, вібрації,

забруднювачів, температури тощо, який не спричиняє соматичних або психічних розладів, а також змін стану здоров'я, працездатності, поведінки, що виходять за межі пристосувальних реакцій) з погляду безпеки та охорони здоров'я працівників;

- організація робочого місця працівника з екранними пристроями має забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним, антропологічним, психофізіологічним вимогам, а також характеру виконуваних робіт;
- освітлення робочого місця працівника з екранними пристроями має створювати відповідний контраст між екраном і навколишнім середовищем (з урахуванням виду роботи) та відповідати вимогам ДСанПІН 3.3.2.007-98;
- мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями має підтримуватись на постійному рівні та відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99, затверджених постановою Головного державного санітарного лікаря України від 01 грудня 1999 року № 42 (далі - ДСН 3.3.6.042-99);
- робочий стіл або робоча поверхня повинні бути достатнього розміру та мати поверхню з низькою відбивною здатністю, допускати гнучкість під час розміщення екрана, клавіатури, документів і відповідного устаткування;
- робоче крісло має бути стійким і дозволяти працівнику з екранними пристроями легко рухатися та займати зручне положення;
- сидіння має регулюватися по висоті, спинка сидіння – як по висоті, так і по нахилу;
- слід передбачати підніжку для тих, кому це необхідно для зручності.

4.2.2 Вимоги до виробничих приміщень для експлуатації ВДТ ЕОМ та ПЕОМ

- об’ємно-планувальні рішення будівель та приміщень для роботи з ВДТ ЕОГМ і ПЕОМ мають відповідати вимогам цих Правил;
- розміщення робочих місць з ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено;
- площа не одне робоче місце має становити не менше ніж 6,0 м2, а об’єм не менше ніж 20,0 м3;
- приміщення для роботи з ВДТ повинні мати природне та штучне освітлення відповідно до СНиП II-4-79;
- природне освітлення має здійснюватися через світлові прорізи, орієнтовані переважно на північ чи північний схід і забезпечувати коефіцієнт природною освітленістю (КПО) не нижче ніж 1,5%. Розраховується КПО за методикою, викладеною в СНиП II-4-79;
- виробничі приміщення для роботи з ВДТ (операторські, диспетчерські) не повинні межувати з приміщеннями, в яких рівні шуму і вібрації перевищують допустимі значення (виробничі цехи, майстерні тощо) за СН 3223-85, СН 3044-84, ГР 2411-81, ГОСТ 12.1.003-83;
- звукоізоляція огороджувальних конструкцій приміщень з ВДТ має забезпечувати параметри шуму, що відповідають вимогам СН 3223-85, ГОСТ 12.1.003-83, ГОСТ 12.1.012-90 (дод.1);
- приміщення для роботи з ВДТ мають бути обладнані системами опалення, кондиціонування повітря, або припливно-витяжною вентиляцією відповідно до СНиП 2.04.05-91;
- нормовані параметри мікроклімату, іонного складу повітря, вмісту шкідливих речовин мають відповідати вимогам СН 4088-86, СН 2152-80, ГОСТ 12.1.005-88, ГОСТ 12.1.007-76 та (дод.2,3);
- віконні прорізи приміщень для роботи з ВДТ мають бути обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки);
- для внутрішнього оздоблення приміщень з ВДТ слід використовувати дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі 0,7-0,8, для стін 0,5-0,6;

- покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3-0,5. Поверхня підлоги має бути рівною, неслизькою, з антистатичними властивостями;
- забороняється для оздоблення інтер'єру приміщень ВДТ застосовувати полімерні матеріали (деревинно-стружкові плити, шпалери, що миються, рулонні синтетичні матеріали, шаруватий паперовий пластик тощо), що виділяють у повітря шкідливі хімічні речовини;
- полімерні матеріали для внутрішнього оздоблення приміщень з ВДТ можуть бути використані при наявності дозволу органів та установ державної санітарно-епідеміологічної служби;
- виробничі приміщення можуть обладнуватись шафами для зберігання документів, магнітних дисків, полицями, стелажами, тумбами тощо з урахуванням вимог до площі приміщень;
- у приміщеннях з ВДТ слід щоденно робити вологе прибирання;
- приміщення з ВДТ мають бути оснащені аптечками першої медичної допомоги;
- при приміщеннях з ВДТ мають бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного розвантаження. В кімнаті психологічного розвантаження слід передбачити встановлення пристроїв для приготування й роздачі тонізуючих напоїв, а також місця для занять фізичною культурою (СНиП 2.09.04.-87);
- вимоги для допоміжних приміщень повинні відповідати СНиП 2.09.04-87.

4.2.3 Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99

Терміни та визначення

1. *Виробниче приміщення* – замкнутий простір в спеціально призначених будинках та спорудах, в яких постійно (по змінах) або періодично (протягом частини робочого дня) здійснюється трудова діяльність людей.
2. *Робоча зона* – простір, в якому знаходяться робочі місця постійного або непостійного (тимчасового) перебування працівників.
3. *Робоче місце* – місце постійного або тимчасового перебування робочого в процесі трудової діяльності.
4. *Постійне робоче місце* – місце, на якому робочий знаходиться понад 50% робочого часу або більше 2-х годин безперервно. Якщо при цьому робота здійснюється в різних пунктах робочої зони, то вся ця зона вважається постійним робочим місцем.
5. *Непостійне робоче місце* – місце, на якому робочий знаходиться менш як 50% робочого часу або менше 2-х годин безперервно.
6. *Теплий період року* – період року, який характеризується середньодобовою температурою зовнішнього середовища вище +10 C^o.
7. *Холодний період року* – період року, який характеризується середньодобовою температурою зовнішнього повітря, що дорівнює +10 C^o і нижче.
8. *Категорія робіт* – розмежування робіт за важкістю на основі загальних енерговитрат організму.
9. Важкі фізичні роботи (категорія III) охоплюють види діяльності, при яких витрати енергії становлять 291-349 Вт (251-300 ккал/год.). До категорії III належать роботи, пов’язані з постійним переміщенням, перенесенням значних (понад 10 кг) вантажів, які потребують великих фізичних зусиль.

Загальні положення

Санітарні норми поширюються на умови мікроклімату в межах робочої зони виробничих приміщень підприємств, закладів, установ тощо, незалежно від їх форми власності та підпорядкування.

Цей документ регламентує нормативні величини оптимальних та допустимих показників мікроклімату та встановлює вимоги до методів вимірювання мікрокліматичних параметрів та їх оцінки.

Норми не поширюються на мікроклімат підземних та гірничих виробок, пересувних транспортних засобів, тваринницьких та птахівницьких ферм, приміщень для зберігання сільськогосподарської продукції, холодильників, складів і т. ін., а також приміщень, в яких параметри мікроклімату встановлюються відповідно до технологічних вимог.

Мікрокліматичні умови виробничих приміщень характеризуються такими
показниками:

- температура повітря;
- відносна вологість повітря;
- швидкість руху повітря;
- інтенсивність теплового (інфрачервоного) опромінення;
- температура поверхні.

4.3 Організаційні заходи

4.3.1 Режим роботи

Необхідно регулярно влаштовувати перерви під час робочого процесу, протягом цих перерв правильним рішенням буде походити, розім’ятися, виконати комплекс фізичних вправ.

Протягом роботи за монітором важливо слідкувати за морганням, намагатися робити це частіше. Моргання сприяє зволоженню рогівки ока та видаленню відмерлих клітин з поверхні очного яблука, також масажує очні

яблука та покращує їх кровообіг. Також можна промасувати закриті очі пальцями, від зовнішнього до внутрішнього кута й круговими рухами всередину-назовні. Корисною процедурою є обертання очима із закритими повіками.

4.3.2 Зорова гімнастика

Вправи варто виконувати сидячи чи стоячи, відвернувшись від екрану, зберігаючи ритмічне дихання, та підтримуючи максимальну амплітуду руху очей.

Вправа № 1

1. Заплющте очі, сильно напружуючи м’язи, на 2-4 секунди, потім розкрийте очі, розслабивши м’язи, подивіться вдалину протягом 3-6 секунд. Повторюйте 4-5 разів.
2. Подивіться на перенісся і затримайте погляд на 2-4 секунди. Не доводьте очей до втоми. Розплющте очі, подивіться вдалину на 3-6 секунд. Повторюйте 4-5 разів.
3. Не повертаючи голову, подивіться наліво та зафіксуйте погляд на 2-4 секунди, потім подивіться вдалину перед собою протягом 3-6 секунд. Схожим чином проводяться вправи з фіксацією погляду вліво, вгору та вниз. Повторюйте 3-4 рази.
4. Перенесіть погляд швидко по діагоналі: направо вгору, наліво вниз, потім прямо вдалину на 3-6 секунд; потім наліво вгору, направо вниз і подивитися вдалину на 3-6 секунд. Повторюйте 4-5 разів.

Вправа № 2

1. Тримайте голову в положенні прямо. Покліпайте очима, не напружуючи м’язи, протягом 10-15 секунд.
2. Не повертаючи голови (положення голови – прямо) й з заплющеними очима, подивіться направо на 2-4 секунди, а згодом наліво на 2-4 секунди і так само на прямо, на 2-4 секунди. Підніміть очі вгору на 2-4 секунди,

опустіть вниз на 2-4 секунди та переведіть погляд прямо перед собою на 3-6 секунд. Повторюйте 4-5 разів.

3. Подивіться на вказівний палець, віддалений від очей на відстань від 25 см до 30 см, протягом 2-4 секунди, потім переведіть погляд вдалину на 3-6 секунд. Повторюйте 4-5 разів.

4. Дотримуючись середнього темпу зробіть 3-4 кругових рухи в ліву сторону, стільки ж в праву сторону та, розслабивши очні м’язи, подивіться вдалечінь на 3-6 секунд. Повторюйте 1-2 рази.

ВИСНОВКИ

Під час кваліфікаційної роботи була досягнута мета, а саме – вирішена практична задача інженерії програмного забезпечення: розробка програмного забезпечення вебсайту компанії «Filiatix». Був виконаний аналіз предметної галузі, що містить аналіз предмета та об’єкту дослідження, аналіз конкурентів за результатами яких було обґрунтовано доцільність розробки проєкту, складено технічне завдання проєкту, вибрана стратегія розробки та створення вебсайту, а також було розглянуто основні положення та вимоги з охорони праці, був створений проєкт, а також розроблена методика тестування до нього. Після створення проєкту він був протестований за цією методикою. Також ми створили інструкцію для користувача та опис програми. Результатом роботи є створення програмного забезпечення вебсайту компанії «Filiatix».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке Html? URL: https://wikipedia.org/wiki/HTML. (дата звернення: 12.03.2022).
2. Що таке Css? URL: https://wikipedia.org/wiki/CSS. (дата звернення: 12.03.2022).
3. Класифікація вебсайтів. URL: https://sites.google.com/site/myinformacionnyetehnologii/cajt/klassifikacia-veb-sajtov. (дата звернення: 14.03.2022).
4. Що таке Rest api? URL: https://wikipedia.org/wiki/REST. (дата звернення: 12.03.2022).
5. Що таке канбан та чим він корисний? URL: https://worksection.com/blog/kanban.html. (дата звернення: 17.03.2022).
6. Текстовий Редактор. URL: https://uk.wikipedia.org/wiki/Текстовий_редактор. (дата звернення: 12.03.2022).
7. Канбан у виробництві. URL: https://up-pro.ru/encyclopedia/kanban-sistema/. (дата звернення: 18.03.2022).
8. Різні способи створення вебсайту. URL: https://www.tentononline.com/different-ways-to-build-a-website/. (дата звернення: 21.03.2022).
9. Як створити вебсайт. URL: https://www.websitebuilderexpert.com/building-websites/. (дата звернення: 24.03.2022).
10. MySql. URL: https://dev.mysql.com/doc/. (дата звернення: 12.03.2022).
11. Nest js documentation. URL: https://docs.nestjs.com/. (дата звернення: 18.04.2022).

12. Node js documentation. URL: https://nodejs.org/en/docs/. (дата звернення: 12.03.2022).

13. Sequelize. URL: https://sequelize.org/v5/. (дата звернення: 12.03.2022).

14. Vue js – Progressive JavaScript Framework. URL: https://v2.vuejs.org/v2/guide/. (дата звернення: 12.03.2022).

15. WebStorm. URL: https://www.jetbrains.com/webstorm/. (дата звернення: 12.03.2022).

16. 8 найкращих методологій для розробки програмного забезпечення. URL: https://www.uptech.team/blog/software-development-methodologies. (дата звернення: 24.03.2022).

ДОДАТОК А ТЕХНІЧНЕ ЗАВДАННЯ

1 Вступ

- Назва програмного продукту: «Kanban board».
- Призначення програми: Контроль робочого процесу в мережі Інтернет.
- Створення системи передбачає використання будь-якого браузера на комп’ютері або мобільному телефоні. Створення завдань, редагування завдань, реєстрація користувачів, створення проєктів, редагування проєктів, створення стовпців, редагування стовпців, перегляд проєктів, перегляд завдань за фільтрами, додавати завдання в проєкти, додавати коментарі до завдань.

2 Підстави для розробки

Підставами для розробки цього програмного забезпечення є тема кваліфікаційної роботи «Розробка програмного забезпечення вебсайту компанії “Filiatix”», затверджена наказом # від 30 квітня 2022.

3 Призначення розробки

3.1 Функціональне призначення

Функціональним призначенням програми є спрощення процесів відстеження та плануванні робочого процесу робітників.

3.2 Експлуатаційне призначення

Програма повинна експлуатуватися в мережі Інтернет такими людьми як:

- *Користувач* – користувач, який був створений адміністратором або суперадміністратор, щоб мати можливість взаємодіяти з системою.

- *Адміністратор* – користувач, що має можливість взаємодіяти з системою та створювати нових користувачів.
- Суперадміністратор – користувач, що має можливість взаємодіяти з системою та створювати нових користувачів та адміністраторів.

4 Вимоги до програми або програмного виробу

4.1 Вимоги до програмних інструментів

Вихідні коди програми повинні бути реалізовані на мові Node js версії 14.7.1. В якості середовища розробки програм повинна бути використана WebStorm IDE. В якості бази даних повинна використовуватися СУБД MySql версії 5.7.

4.2 Вимоги до складу виконуваних функцій

Програмне забезпечення повинне забезпечувати користувача можливістю виконання перерахованих нижче функцій:

- Перегляд проєктів в який користувач був доданий;
- Перегляд завдань за фільтрами в проєктах до яких користувач був доданий;
- Додавати завдання в проєкти до яких користувач був доданий;
- Залишати коментарі в проєкти до яких користувач був доданий;
- Редагування існуючих завдань в проєктах до яких користувач був доданий.

Програмне забезпечення повинне забезпечувати адміністратора можливістю виконання перерахованих нижче функцій:

- Перегляд проєктів, до яких адміністратор був доданий, або які були створені адміністратором;
- Перегляд завдань за фільтрами в проєктах, до яких адміністратор був доданий, або які були створені адміністратором;

- Додавати завдання в проєкти, до яких адміністратор був доданий, або які були створені адміністратором;
- Залишати коментарі в проєктах, до яких адміністратор був доданий, або які були створені адміністратором;
- Створювати користувачів з роллю адміністратора або користувача;
- Додавати користувачів до проєктів, до яких адміністратор був доданий, або які були створені адміністратором;
- Редагування існуючих завдань в проєктах, до яких адміністратор був доданий, або які були створені адміністратором;
- Змінювати порядок стовпців в проєктах, до яких адміністратор був доданий, або які були створені адміністратором.

Програмне забезпечення повинне забезпечувати для суперадміністратора можливість виконання перерахованих нижче функцій:

- Перегляд усіх проєктів;
- Перегляд завдань за фільтрами в будь-яких проєктах;
- Додавання завдання в будь-які проєкти;
- Створювання користувачів з будь-якою роллю;
- Можливість залишати коментарі в будь-які проєкти;
- Додавання користувачів до будь-яких проєктів;
- Редагування будь-яких завдань;
- Можливість змінювати порядок стовпців в будь-яких проєктах.

4.3 Вимоги до організації вхідних та вихідних даних

Введення інформації здійснюється за допомогою посилання даних з фронтенду на бекенд. Вихідна інформація відображається за допомогою роботи фронтенду. Уся інформація зберігається на сервері в відповідній базі даних.

4.4 Вимоги до надійності

4.4.1 Вимоги до забезпечення надійного функціонування програми

Надійне функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких наведено нижче:

- організацією безперебійного живлення технічних засобів;
- використанням ліцензійного програмного забезпечення.

4.4.2 Час відновлення після відмови

Час відновлення після відмови, викликаної збоєм електроживлення на сервері (іншими зовнішніми факторами), не фатальним збоєм операційної системи, не повинно перевищувати 10-ти хвилин.

Час відновлення після збою, викликаного несправністю технічних засобів, фатальним збоєм операційної системи, не повинно перевищувати час, необхідний для усунення несправностей технічних засобів і перевстановленням програмних засобів.

4.4.3 Відмови через некоректність дій користувача

Відмови програми внаслідок дій користувача неможливі, оскільки усі данні проходять валідацію з двох сторін (фронтенду та бекенду), у разі виникнення помилки валідації програми буде вказувати на помилки.

4.5 Вимоги до видів обслуговування

Програма не потребує проведення будь-якого обслуговування.

4.6 Вимоги до чисельності та кваліфікації персоналу

Мінімальна кількість персоналу, необхідного для функціонування програми, повинна складати одну особу – адміністратор. Адміністратор повинен мати базові навички володіння комп’ютером.

4.7 Вимоги до складу і параметрів технічних засобів

4.7.1 Backend

В склад технічних засобів повинен входити сервер з наступними параметрами:

- операційна система: Linux;
- процесор: Intel Celeron G4900, 3.1 GHZ, 2 ядра та більше;
- Інтернет: 3 Мбіт/с та більше;
- оперативна пам’ять: мінімум 4Гб;
- жорсткий диск: для нормальної роботи системі потрібно 500Мб вільного дискового простору.

4.7.2 Frontend

В склад технічних засобів повинен входити сервер з наступними параметрами:

- операційна система: Linux;
- процесор: Intel Celeron G4900, 3.1 GHZ, 2 ядра та більше;
- Інтернет: 3 Мбіт/с та більше;
- оперативна пам’ять: мінімум 2Гб;
- жорсткий диск: для нормальної роботи системі потрібно 500Мб вільного дискового простору.

4.8 Вимоги до інформаційної і програмної сумісності

4.8.1 Вимоги до інформаційних структур і методів розв’язку

Вимоги до інформаційних структур на вході та виході, а також до методів не висуваються.

4.8.2 Вимоги до вихідних кодів та мов програмування

Вихідні коди програми повинні бути реалізовані на мові Node js версії 14.7.1 та Javascript. В якості середовища розробки програм повинна бути використана WebStorm IDE. В якості бази даних повинна використовуватися СУБД MySql версії 5.7.

4.8.3 Вимоги до програмних засобів, що використовуються програмою

Вимоги до захисту не висуваються.

4.8.4 Вимоги до захисту

Вимоги до захисту не висуваються.

4.9 Вимоги до маркування

Доступ до програмного додатка здійснюється за допомогою Інтернету та сучасного браузера.

4.10 Вимоги до транспортування та збереження

Вимоги до транспортування та збереження не висуваються.

5 Вимоги до програмної документації

Попередній склад програмної документації повинен містити в собі:

- технічне завдання;
- текст програмних модулів;
- програма та методика випробовування;
- опис програми;
- інструкція користувача.

6 Техніко-економічні показники

Техніко-економічні показники не висуваються.

7 Стадії та етапи розробки

Стадії та етапи розробки представлено в таблиці 1.

Таблиця А.1 – Стадії та етапи розробки програмного забезпечення

№	Стадії розробки	Етапи розробки	Зміст робіт по етапах	Строки виконання
1	2	3	4	5
1	Технічне завдання	Розробка, узгодження та затвердження технічного завдання	Постановка задачі; визначення та уточнення вимог до технічних засобів; визначення вимог до програми; визначення стадій, етапів і термінів розробки програми і документації на неї; узгодження і затвердження технічного завдання.	Початок: 01.02.22 Закінчення: 21.02.22
2	Ескізний проєкт	Розробка та затвердження ескізного проєкту	Виявлення вимог; концепція майбутньої системи; складання ескізної частини складання програмного продукту	Початок: 01.03.22 Закінчення: 19. 03.22

Продовження таблиці А.1

1	2	3	4	5
3	Технічний проєкт	Розробка та затвердження технічного проєкту	Загальні системні вирішення, алгоритми задач, оцінка економічної ефективності автоматизованої системи та перелік заходів підготовки об’єкта для провадження	Початок: 01.05.21 Закінчення: 03.05.22
4	Робочий проєкт	Розробка та затвердження робочого проєкту	Деталізовані загальносистемні проєкті рішення, програми та інструкції для розв’язку завдань, розробка програмного забезпечення.	Початок: 04.05.22 Закінчення: 15.05.22

8 Порядок контролю та приймання

Тестування програмного продукту повинно проводитися у відповідності до узгодженої заздалегідь із замовником програмного продукту методики випробувань. Тестування проводиться в зазначені строки.

Контроль за кожним етапом розробки проводиться викладачем шляхом надсилання йому завершених розділів кваліфікаційної роботи. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником (викладачем).

Хід проведення приймально-здавальних випробувань документують за допомогою протоколу проведення випробувань.

На підставі протоколу проведення випробувань виконувач сумісно з замовником підписують акт приймання – здачі програмного забезпечення в експлуатацію.

ДОДАТОК Б ТЕКСТ ПРОГРАМИ

task.controller.ts

```
import { Body, Controller, HttpCode, Post, Get, Delete, Param, ParseIntPipe, Patch } from '@nestjs/common';
import { ApiBearerAuth, ApiOperation, ApiResponse, ApiTags } from '@nestjs/swagger';
import { ErrorResponseDto } from '../../core';
import { TasksService } from '../services/task.service';
import {
  CurrentUserHasRoles,
  Policies,
  PoliciesEnum,
  UserHasAccessToTheProject,
  UserHasAccessToTheProjectFromTask
} from 'src/auth';
import { TaskModel } from 'src/database/models';
import { CreateTaskDto, UpdateTaskDto } from '../dtos';
import { UserRolesEnum } from '../../users';
import { TaskTypes } from '../../database/enums';

@ApiTags('tasks')
@Controller('tasks')
export class TaskController {
  constructor(private readonly tasksService: TasksService) {
  }

  @ApiOperation({
    summary: 'Get tasks',
  })
  @ApiResponse({
    status: 200,
    description: 'Tasks sent',
  })
```

```
@ApiResponse({
  status: 403,
  description: 'Forbidden',
  type: ErrorResponseDto,
})
@HttpCode(200)
@Get('/')
@ApiBearerAuth()
@Policies(PoliciesEnum.AUTHENTICATED)
async getTasks(): Promise<any> {
  return this.tasksService.getTasks()
}

@ApiOperation({
  summary: 'Get tasks types',
})
@ApiResponse({
  status: 200,
  description: 'Types sent',
})
@ApiResponse({
  status: 403,
  description: 'Forbidden',
  type: ErrorResponseDto,
})
@HttpCode(200)
@Get('/types')
@ApiBearerAuth()
@Policies(PoliciesEnum.AUTHENTICATED)
async getTasksTypes(): Promise<TaskTypes[]> {
  return this.tasksService.getTaskTypes()
}

@ApiOperation({
  summary: 'Get a task',
})
@ApiResponse({
  status: 200,
  description: 'Task sent',
```

```
  })
  @ApiResponse({
    status: 403,
    description: 'Forbidden',
    type: ErrorResponseDto,
  })
  @HttpCode(200)
  @Get('/:taskId')
  @ApiBearerAuth()
  @Policies(PoliciesEnum.AUTHENTICATED)
  @UserHasAccessToTheProjectFromTask('taskId')
  async getOneTask(
    @Param('taskId', ParseIntPipe) taskId: number
  ): Promise<any> {
    return this.tasksService.getOneTask(taskId)
  }

  @ApiOperation({
    summary: 'Create task',
  })
  @ApiResponse({
    status: 201,
    type: TaskModel
  })
  @Post('/')
  @ApiBearerAuth()
  @Policies(PoliciesEnum.AUTHENTICATED)
  @UserHasAccessToTheProject('project_id')
  createTask(
    @Body() createTaskPayload: CreateTaskDto
  ) {
    return
this.tasksService.createTask(createTaskPayload)
  }

  @ApiOperation({
    summary: 'Delete a task',
  })
  @ApiResponse({
```

```
    status: 204,
    description: 'Task deleted',
  })
  @ApiResponse({
    status: 404,
    description: 'Task not found'
  })
  @Delete('/:taskId')
  @ApiBearerAuth()
  @Policies(PoliciesEnum.AUTHENTICATED)
  @CurrentUserHasRoles(UserRolesEnum.ADMIN,
UserRolesEnum.SUPER_ADMIN)
  @UserHasAccessToTheProjectFromTask('taskId')
  async deleteProject(
    @Param('taskId') taskToDelete: number,
  ): Promise<number> {
    return this.tasksService.deleteTask(+taskToDelete)
  }

  @ApiOperation({
    summary: 'Update a task',
  })
  @ApiResponse({
    status: 204,
    description: 'Task updated',
  })
  @ApiResponse({
    status: 404,
    description: 'Task not found'
  })
  @Patch('/:taskId')
  @ApiBearerAuth()
  @Policies(PoliciesEnum.AUTHENTICATED)
  @UserHasAccessToTheProjectFromTask('taskId')
  async updateTask(
    @Param('taskId') taskId: number,
    @Body() payload: UpdateTaskDto
  ) {
```

```
    return     this.tasksService.updateTaskById(taskId,
payload)
  }

  @ApiOperation({
    summary: 'Change task column'
  })
  @ApiResponse({
    status: 204,
    description: 'Task column changed',
  })
  @ApiResponse({
    status: 404,
    description: 'Task or column not found'
  })
  @Patch('/:taskId/:columnId')
  @ApiBearerAuth()
  @Policies(PoliciesEnum.AUTHENTICATED)
  @UserHasAccessToTheProjectFromTask('taskId')
  async changeTaskColumn(
    @Param('taskId', ParseIntPipe) taskId: number,
    @Param('columnId', ParseIntPipe) columnId: number,
  ) {
    return this.tasksService.changeTaskColumn({ taskId,
columnId })
  }
}
```

TaskPage.vue

```
<template>
  <div>
    <Loader v-if="loading" />
    <Modal v-if="error" :text="errorText" :click="() =>
{closeModal()}" />
    <div class="bg-white rounded border-gray-200 px-2
sm:px-4 py-2.5 dark:bg-gray-800 mt-12">
      <div class="container flex justify-start mx-
auto">
        <GoBackArrow />
```

```
      </div>
    </div>
    <!--  Edit Task section-->
    <div class="container mx-auto mt-6 mb-6">

      <div class="flex flex-wrap -mx-3 mb-3 ">

        <div class="w-full md:w-1/3 px-3 mb-6 md:mb-0
">
          <label class="block uppercase tracking-wide
text-gray-700 text-xs font-bold mb-2" for="grid-first-
title">
            Title
          </label>
          <input
              :disabled="!edit"
              v-model="task.title"
              :class="`${!edit? 'cursor-not-allowed' :
''} appearance-none block w-full bg-gray-200 text-gray-
700 rounded py-3 px-4 mb-3 leading-tight focus:outline-
none focus:bg-white`"
              id="grid-first-title"
              type="text"
              placeholder="Task Title"
          >
        </div>

      </div>

      <div class="flex flex-wrap -mx-3 mb-3 ">

      <div class="w-full md:w-1/2 px-3 mb-6 md:mb-0 ">
        <label class="block uppercase tracking-wide
text-gray-700 text-xs font-bold mb-2"
for="description">
          Description
        </label>
        <textarea
            :disabled="!edit"
```

```
            v-model="task.description"
            id="description"
            rows="3"
            :class="`${!edit?  'cursor-not-allowed'  :
''} appearance-none block w-full bg-gray-200 text-gray-
700 rounded py-3 px-4 mb-3 leading-tight focus:outline-
none focus:bg-white`"
            placeholder="Description"
        />
      </div>

      </div>

      <div class="flex flex-wrap -mx-3 mb-3 ">

        <div class="w-full md:w-1/3 px-3 mb-6 md:mb-0
">
          <label class="block uppercase tracking-wide
text-gray-700 text-xs font-bold mb-2">
            Date
          </label>
          <Datepicker
            :disabled="!edit"
            v-model="task.date"
            :input-class="`${!edit?        'cursor-not-
allowed' : ''} appearance-none block w-full bg-gray-200
text-gray-700   rounded   py-3   px-4   leading-tight
focus:outline-none focus:bg-white`"
          />
        </div>

        <div class="w-full md:w-1/3 px-3 mb-6 md:mb-0
">
          <label class="block uppercase tracking-wide
text-gray-700 text-xs font-bold mb-2" for="type">
            Type
          </label>
          <div class="relative">
```

```
            <select                    v-model="task.type"
:disabled="!edit"    :class="`${!edit?    'cursor-not-
allowed' : ''} block appearance-none w-full bg-gray-200
border border-gray-200 text-gray-700 py-3 px-4 pr-8
rounded leading-tight focus:outline-none focus:bg-white
focus:border-gray-500`" id="type">
              <option  v-for="(type,  idx)  in  types"
:key="idx" :value="type">{{type}}</option>
            </select>
            <div  class="pointer-events-none  absolute
inset-y-0  right-0  flex  items-center  px-2  text-gray-
700">
              <svg   class="fill-current   h-4   w-4"
xmlns="http://www.w3.org/2000/svg"  viewBox="0  0  20
20"><path  d="M9.293  12.95l.707.707L15.657  8l-1.414-
1.414L10 10.828 5.757 6.586 4.343 8z"/></svg>
            </div>
  <!--                  <p  v-if="validationErrors.type"
class="text-red-500                                text-xs
italic">{{validationErrors.type}}</p>-->

          </div>
        </div>

        <div class="w-full md:w-1/3 px-3 mb-6 md:mb-0
">
          <label  class="block  uppercase  tracking-wide
text-gray-700       text-xs       font-bold       mb-2"
for="assigned_to">
            Assigned To
          </label>
          <div class="relative">
            <select           v-model="task.assigned_to"
:disabled="!edit"    :class="`${!edit?    'cursor-not-
allowed' : ''} block appearance-none w-full bg-gray-200
border border-gray-200 text-gray-700 py-3 px-4 pr-8
rounded leading-tight focus:outline-none focus:bg-white
focus:border-gray-500`" id="assigned_to">
```

```
              <option   v-for="(user)   in   usersList"
:key="user.id"       :value="user.id">{{user.firstName}}
{{user.lastName}}</option>
            </select>
            <div   class="pointer-events-none   absolute
inset-y-0  right-0  flex  items-center  px-2  text-gray-
700">
              <svg    class="fill-current    h-4    w-4"
xmlns="http://www.w3.org/2000/svg"   viewBox="0   0   20
20"><path   d="M9.293   12.95l.707.707L15.657   8l-1.414-
1.414L10 10.828 5.757 6.586 4.343 8z"/></svg>
            </div>
          </div>
        </div>

      </div>

      <p class="underline">
        Created   by   {{task.createdByUser.firstName}}
{{task.createdByUser.lastName}}
      </p>
      <p class="underline">
        Date: {{formDate(task.created_at)}}
      </p>
    </div>

    <div class="container flex justify-end mx-auto mb-
6">
      <button  v-if="!edit"  class='bg-blue-500  text-
white p-1 pl-3 pr-3 rounded overflow-hidden hover:bg-
indigo-500 duration-300' @click="openEdit">
        Edit
      </button>
      <button v-if="edit" class='mr-3 bg-red-500 text-
white p-1 pl-3 pr-3 rounded overflow-hidden hover:bg-
red-800 duration-300' @click="closeEdit">
        Cancel
      </button>
```

```
      <button  v-if="edit"  class='bg-green-500  text-
white p-1 pl-3 pr-3 rounded overflow-hidden hover:bg-
green-800 duration-300' @click="editTask">
        Submit
      </button>
    </div>
    <!--  End Edit Task section-->

    <!--  Comments section-->
    <div class="container  mx-auto mt-6 mb-6">
      <p class="mb-2">Comments:</p>
      <Comments :commentsList="task.comments"/>
      <div class="flex flex-wrap -mx-3 mb-3 mt-3">
        <AddComment                    :taskId="task.id"
:addParentComment="addComment"/>
      </div>
    </div>
    <!-- End Comments section-->

  </div>
</template>

<script>
import Loader from "@/components/Loader";
import Modal from "@/components/modals/Modal";
import GoBackArrow from "@/components/GoBackArrow";
import AddComment from "@/components/AddComment";
import Comments from "@/components/Comments";
import Datepicker from 'vuejs-datepicker';
import    {    getDateFormatFromISOString    }    from
"@/helpers/functions";
import { mapGetters } from "vuex";

export default {
  name: "TaskPage",
  computed: {
    ...mapGetters({
      bearerToken: "auth/bearerToken",
    })
```

```
},
components: {
  Loader,
  Modal,
  Datepicker,
  AddComment,
  Comments,
  GoBackArrow,
},
data() {
  return {
    loading: false,
    task: {
      id: 0,
      firstName: '',
      lastName: '',
      createdByUser: {
        id: null,
        firstName: '',
        lastName: ''
      },
      comments: [],
      project_id: null,
    },
    error: false,
    errorText: '',
    types: [],
    usersList: [],
    edit: false,
  };
},
async mounted() {
  this.loading = true;
  await this.loadTask();
  await this.loadTypes();
  await this.loadUsersList();
  this.loading = false;
},
methods: {
```

```
async loadTask() {
  const taskId = this.$route.params.task_id;
  try {
    const task = await this.$axios.get(
        `/api/tasks/${taskId}`,
        {
          headers: {
            'Authorization': this.bearerToken
          }
        }
    )
    this.task = task.data
  } catch (e) {
    console.log(e)
    this.error = true
    this.errorText = e.response.data.error || e
  }
},
async loadTypes() {
  const taskTypes = await this.$axios.get(
      `/api/tasks/types`,
      {
        headers: {
          'Authorization': this.bearerToken
        }
      }
  )
  this.types = taskTypes.data
},
async loadUsersList() {
  const projectId = this.task.project_id;
  if (!projectId) return;
  const usersList = await this.$axios.get(
      `/api/projects/access-users/${projectId}`,
      {
        headers: {
          'Authorization': this.bearerToken
        }
      }
```

```
      )
      this.usersList = usersList.data
    },
    addComment(comment) {
      this.task.comments    =    [...this.task.comments,
comment]
    },
    async editTask() {
      console.log(this.task)
      const  {  id:  taskId,  title,  description,  date,
type, assigned_to } = this.task
      const payload = {
        title,
        description,
        date,
        type,
        assigned_to,
      }
      try {
        await  this.$axios.patch(`/api/tasks/${taskId}`,
payload, {
          headers: {
            'Authorization': this.bearerToken
          }
        })
        this.closeEdit()
      } catch (e) {
        console.log(e)
        this.error = true
        this.errorText = e.response.data.error || e
      }
    },
    openEdit() {
      this.edit = true
    },
    closeEdit() {
      this.edit = false
    },
    closeModal() {
```

```
      this.error = false;
      this.$router.push('/projects')
    },
    formDate(date) {
      return getDateFormatFromISOString(date)
    },
  }
}
</script>

<style scoped>

</style>
```

ДОДАТОК В ПРОГРАМИ ТА МЕТОДИКА ВИПРОБУВАНЬ

1. Об’єкт випробувань

   Об’єктом випробувань є програмне забезпечення вебсайту для контролю робочого процесу. Областю застосування програмного забезпечення є відстеження прогресу виконання певної частини роботи.

2. Мета випробувань

   Перевірка придатності програмного забезпечення до використання за призначенням, її відповідність заданим вимогам і документації. Перевірка працездатності та правильності роботи програмного забезпечення.

3. Вимоги до програми

   При проведенні випробувань, функціональні характеристики програми підлягають перевірці на відповідність вимогам, викладеним у пункті «Вимоги до складу виконуваних функцій» Технічного завдання.

4. Вимоги до програмної документації

   В комплект програмної документації повинні входити наступні документи:

   - технічне завдання;
   - інструкція користувача;
   - програма і методика випробувань;

- текст програми.

5. Склад та порядок випробувань

Вимоги до складу та параметрів технічних і програмних засобів вказані в Додатку А.

Порядок проведення випробувань:

- виконуються контрольні тести в довільному порядку;
- проводиться аналіз отриманих результатів і встановлюється відповідність програмного продукту вимогам і системним документам.

При виявлені помилок у роботі системи складається їх перелік і обговорюється термін їх виправлення розробником. Після цього замовник проводить повторне тестування в повному обсязі (можливе використання нових чи додаткових тестів).

6. Методи випробувань

Методом випробування розробленого програмного забезпечення є тестування чорної скриньки, яке представляє процес виконання програми з метою виявлення помилок. Кроки процесу задаються тестами.

Розглянемо тести для програмного забезпечення для відстеження та контролю робочого процесу.

- варіанта використання «Авторизація»;
- варіанта використання «Створення користувача»;
- варіанта використання «Створення завдання»;
- варіанта використання «Додавання коментаря до завдання».

ДОДАТОК Г ОПИС ПРОГРАМИ

1. Загальні відомості

1.1. Позначення і найменування програми

Найменування: Розробка програмного забезпечення вебсайту для для відстеження та контролю робочого процесу.

Призначення: контроль робочого процесу у форматі вебсайту.

1.2. Програмне забезпечення, необхідне для функціонування програми

1.2.1. Backend.

В склад технічних засобів повинен входити сервер з наступними параметрами:

- операційна система: Linux;
- процесор: Intel Celeron G4900, 3.1 GHZ, 2 ядра та більше;
- інтернет: 3 Мбит/с та більше;
- оперативна пам’ять: мінімум 4Гб;
- твердий диск: для нормальної роботи системі потрібно 150Мб вільного дискового простору.

1.2.2 Frontend.

В склад технічних засобів повинен входити комп’ютер, або смартфон з мінімальними характеристиками, який підтримує встановлення будь-якого браузера, що має підтримку Javascript.

Android:

- версія ОС: Android 2.2 +;
- 82 - інтернет: потрібно;
- потрібно вільного місця: 180 Mb.

IOS:

- версія ОС: iOS 9.0 +;
- інтернет: потрібно;

- потрібно вільного місця: 210 Mb.

Комп’ютер:

- ОС: Windows XP, 7, Vista, 8, 8.1, 10;
- процесор: Intel Celeron 1800 MHz;
- оперативна пам'ять: 256 MB ОЗУ;
- відеокарта: Intel HD Graphics;
- місце на диску: 128 MB.

1.3. Мови програмування

Вихідні коди програми повинні бути реалізовані на мові Node js версії 14.7.1 та Javascript. В якості середовища розробки програм повинна бути використана WebStorm IDE. В якості бази даних повинна використовуватися СУБД MySql версії 5.7.

2. Функціональне призначення

2.1. Класи розв'язуваних завдань та призначення програми

Функціональним призначенням програми є автоматизація відстеження та контролю робочого процесу для працівників менеджерів та користувачів за допомогою вебсайту.

2.2. Функціональні обмеження

Для роботи з вебсайтом необхідно мати підключення до мережі інтернет. Також необхідно мати будь-який браузер.

3. Опис логічної структури

3.1. Структура програми

Програмне забезпечення складається з наступних частин. Графічне зображення всіх компонентів можна побачити на рисунку Г.1:

- Сервер, на якому запущений Backend;
- Сервер, на якому запущений Frontend;
- Сервер, на якому запущена база даних;
- Пристрій користувача, із встановленим браузером та підключенням до мережі інтернет.

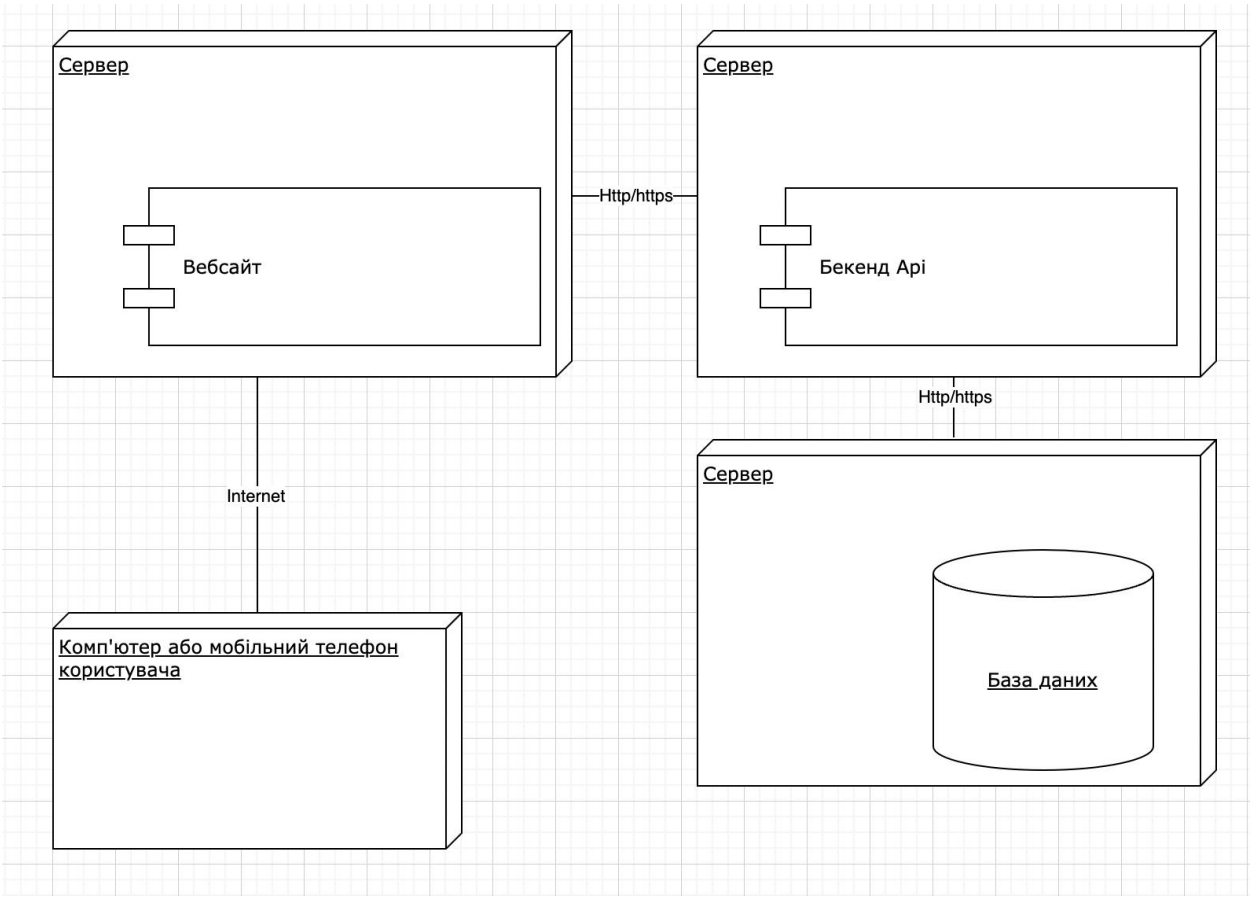

Рисунок Г.1 – Схема взаємодії

3.2. Алгоритм програми:

Програмне забезпечення для контролю робочого процесу має технологію «клієнт-сервер». Клієнтом виступає вебсайт(frontend), а сервером(backend) – комп’ютер із запущеним скриптом.

Робота ПЗ представлена взаємодією між клієнтською та серверною частинами, а також базою даних. Алгоритм роботи програми є наступним:

1) Користувач взаємодіє з вебсайтом;

2) Вебсайт через Інтернет відсилає запити серверу;

3) Сервер обробляє запит;

4) Скрипт звертається до бази даних (якщо це потрібно) та відсилає результат;

5) Вебсайт відображає отриманий результат у вигляді виконаних дій або у разі помилки повідомленнями про помилки.

3.3 Використовувані методи:

Програмне забезпечення відстеження та контролю робочого процесу реалізовані на мові Node js версії 14.7.1 та Javascript. В якості середовища розробки програм повинна бути використана WebStorm IDE. В якості бази даних повинна використовуватися СУБД MySql версії 5.7.

ДОДАТОК Д ІНСТРУКЦІЯ КОРИСТУВАЧА

При першому запуску програми на головній сторінці ми побачимо загальний опис вебсайту (рисунок Д.1).

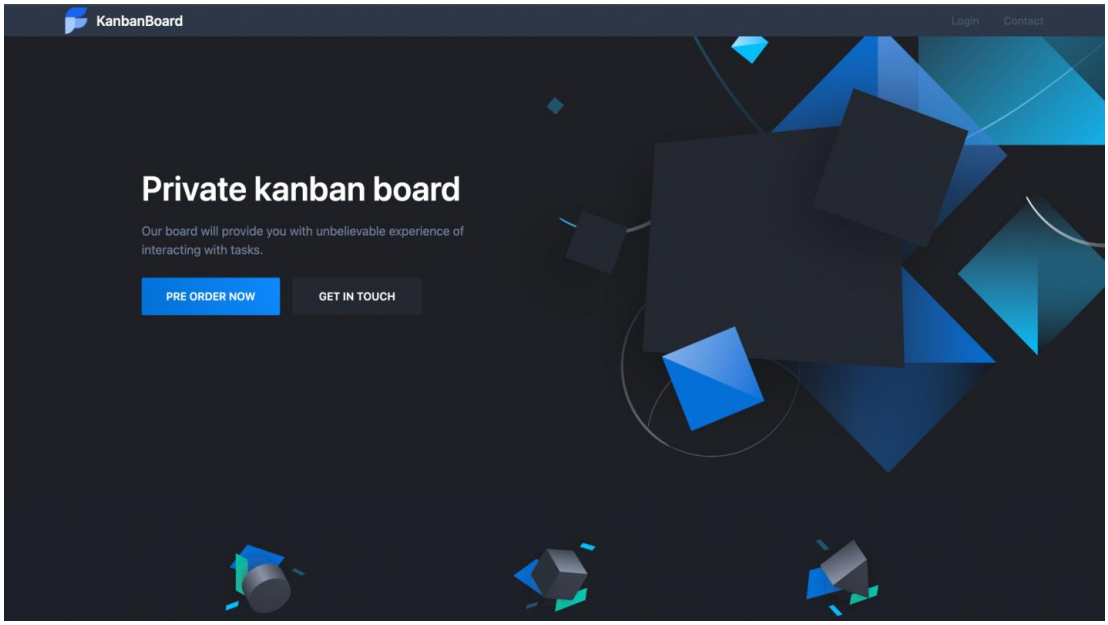

Рисунок Д.1 – Головна сторінка

Щоб розпочати користуватись повним функціоналом вебсайту необхідно авторизуватись. Сторінка авторизіції зображена на рисунку Д.2.

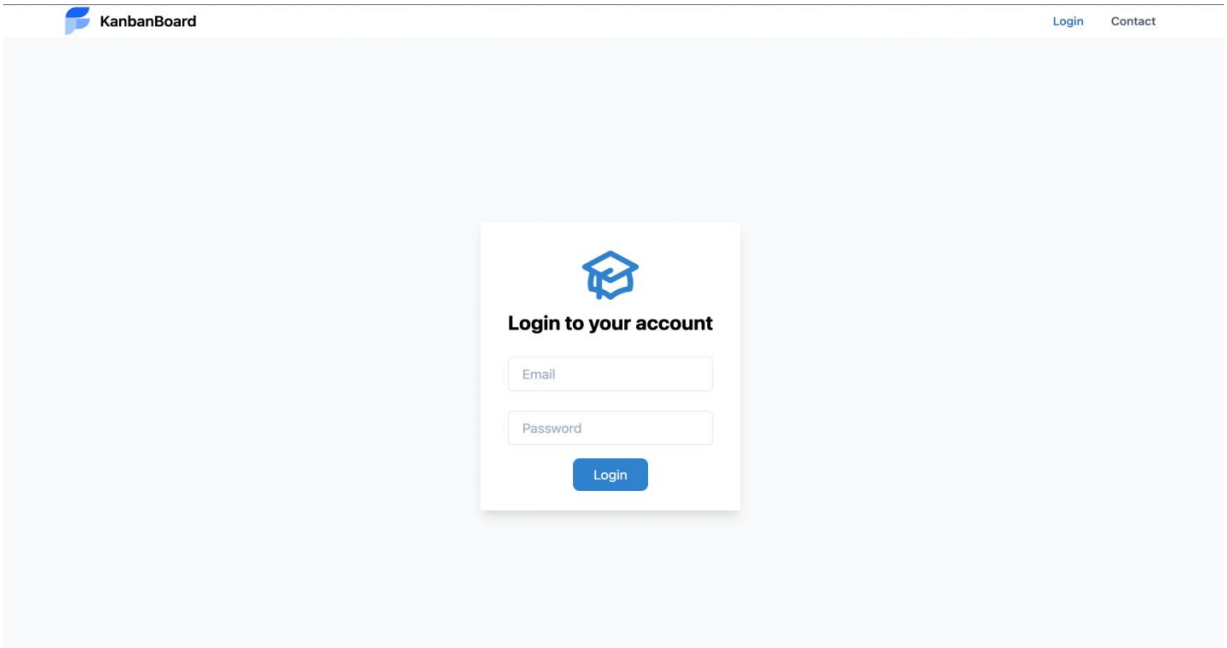

Рисунок Д.2 – Сторінка авторизації

Користувач може увійти під роллю яку йому надав адміністратор або суперадміністратор. В залежності від ролі користувач має змогу виконувати дії. Якщо користувач має роль адміністратор або суперадміністратор, він має можливість створити проєкт зображено на рисунку Д.3, у випадку якщо користувач має роль користувача він може бути тільки доданий у проєкт адміністратором або суперадміністратором.

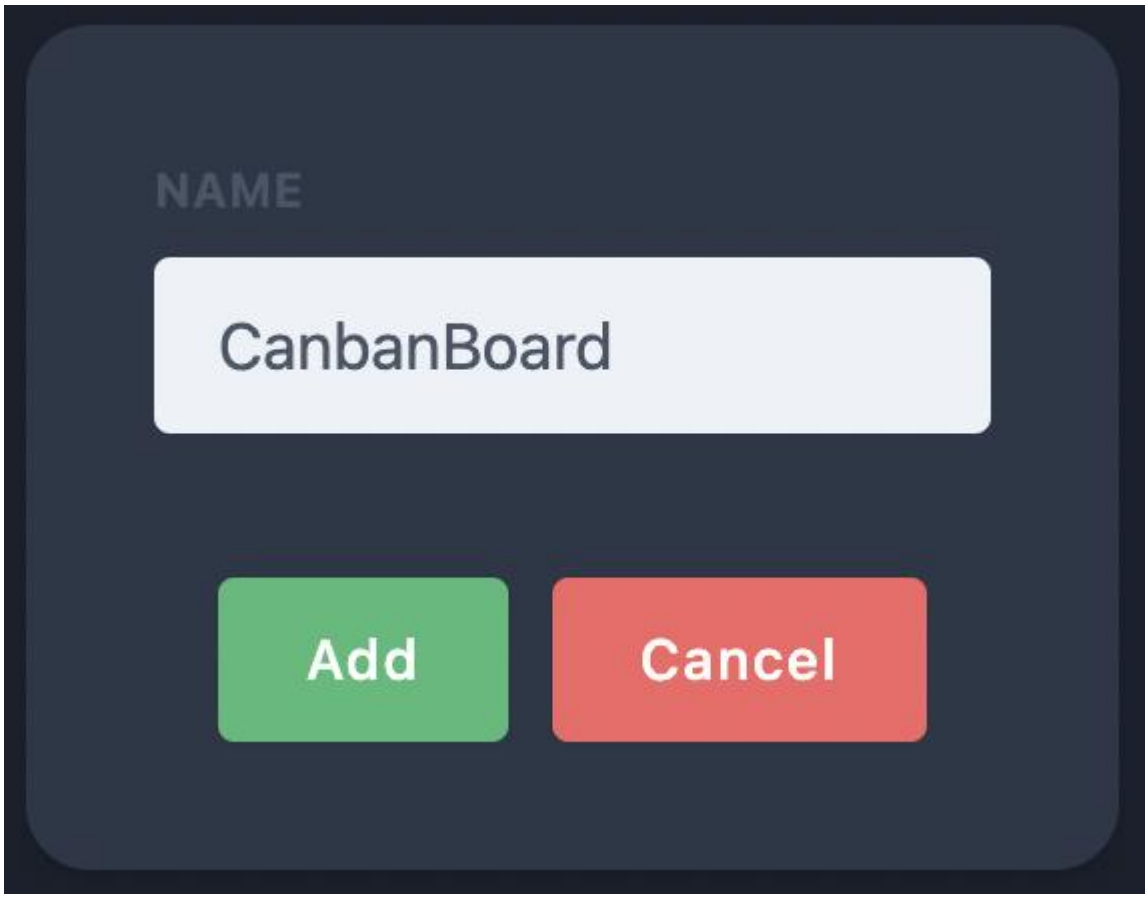

Рисунок Д.3 – Створення проєкту

Після того як користувач створив або був доданий до проєкту він має змогу створити завдання (рисунок Д.4).

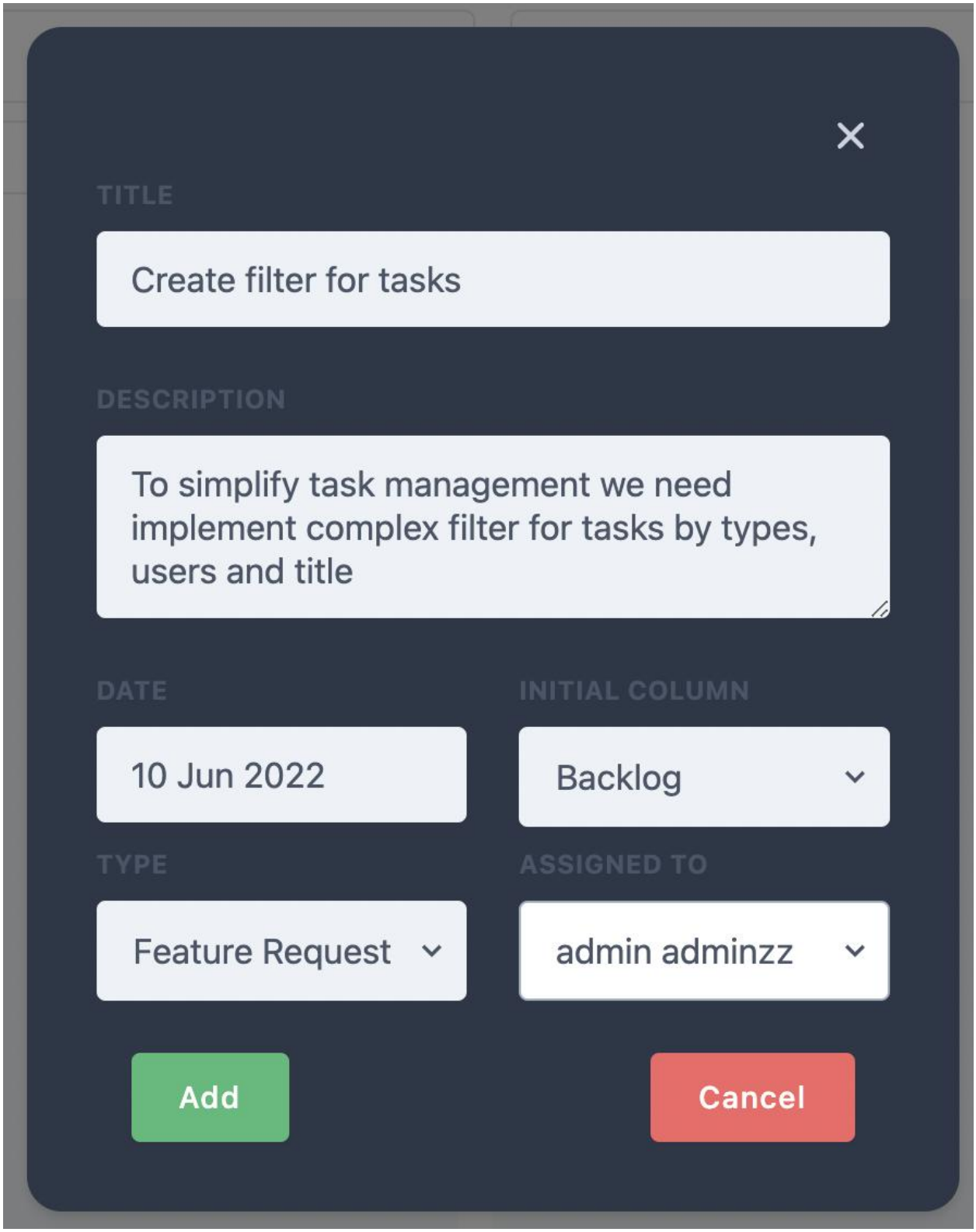

Рисунок Д.4 – Створення завдання

Після того як користувач створив завдання або необхідне завдання вже існувало, користувач коли починає роботу над завданням має змогу змінити статус завдання у проєкті (рисунок Д.5).

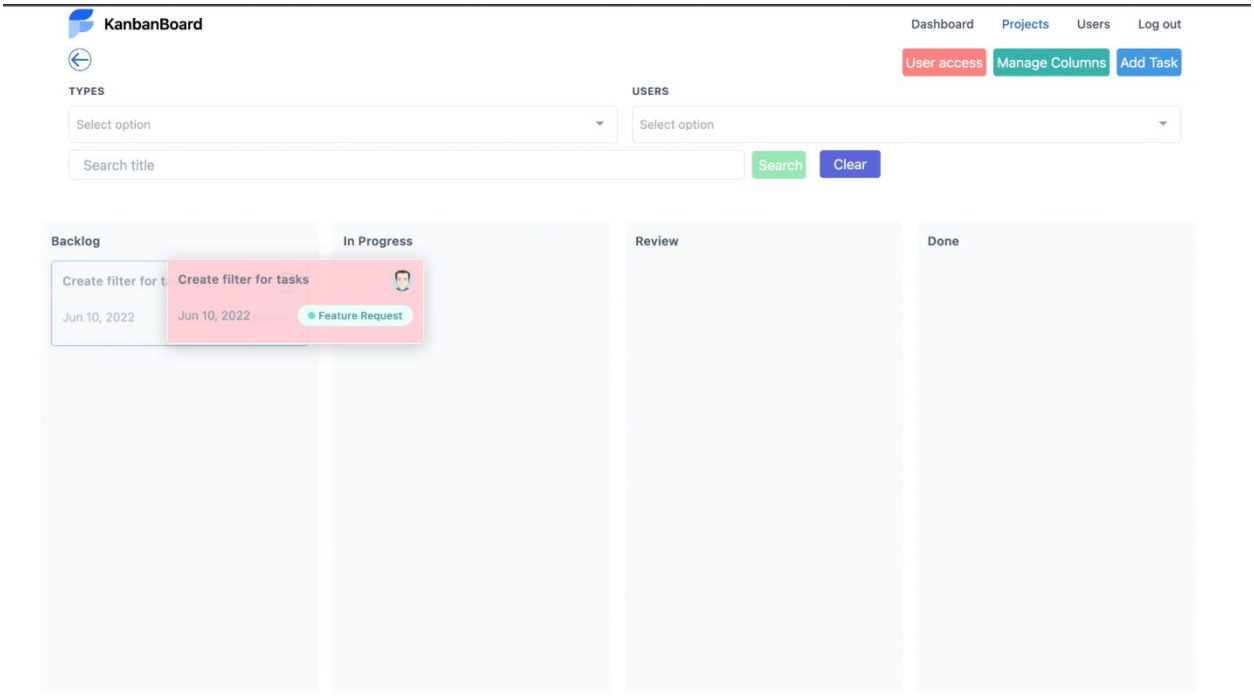

Рисунок Д.5 – Зміна статусу завдання

Якщо користувач впевнений що необхідно доповнити опис або відредагувати завдання, він має змогу це зробити (рисунок Д.6).

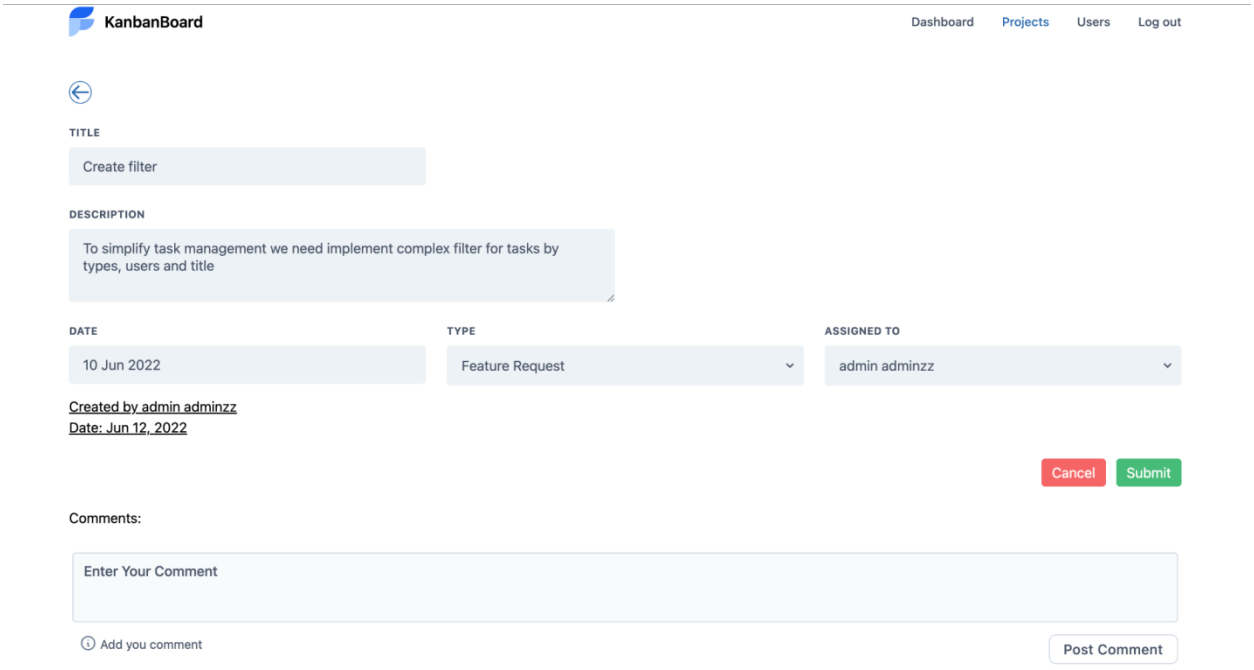

Рисунок Д.6 – Редагування завдання

Також якщо користувач має якісь зауваження стосовно завдання, він має змогу залишити коментар до завдання (рисунок Д.7).

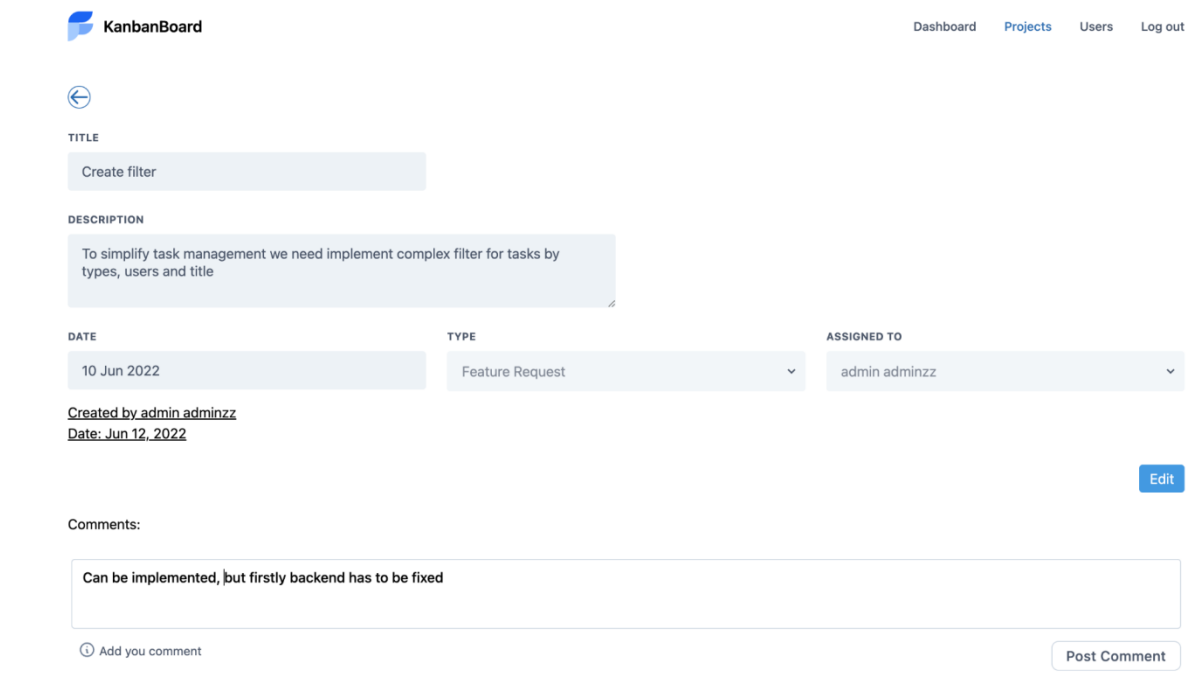

Рисунок Д.7 – Додавання коментаря

У залежності від використання кількість завдань буде зростати, тому для більшої зручності користувач має змогу пошуку та фільтрації завдань (рисунок Д.8).

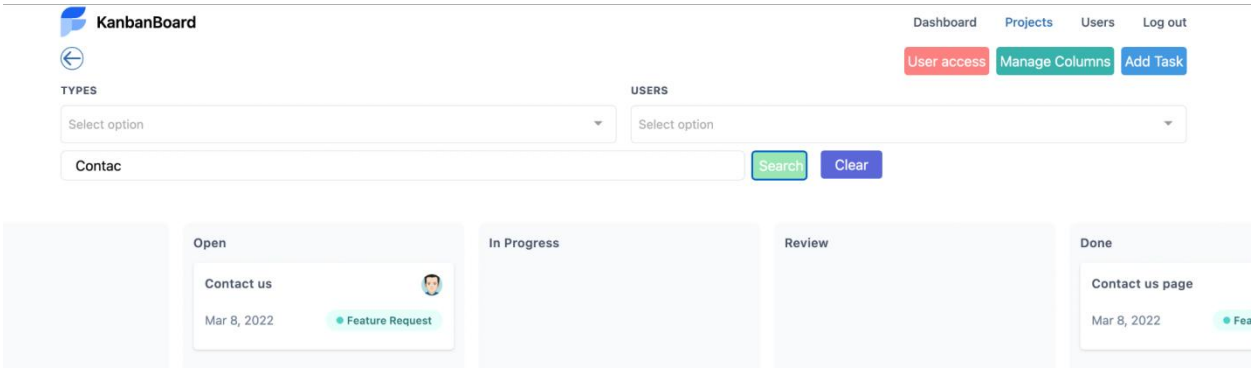

Рисунок Д.8 – Використання фільтру

Користувач має змогу переглянути усі проєкти до яких він був доданий, також якщо користувач має роль адміністратора, він має змогу створити проєкт. Якщо користувач має роль суперадміністратора, він має змогу переглядати усі проєкти та створювати нові (рисунок Д.9).

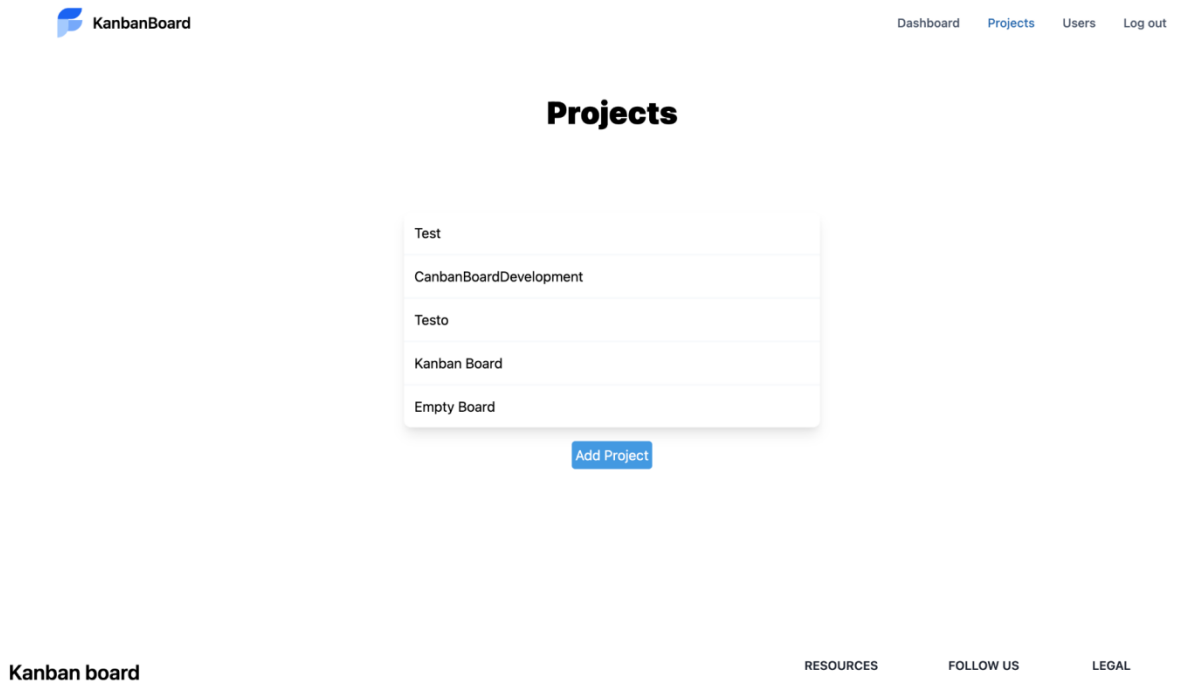

Рисунок Д.9 – Сторінка проєктів

Якщо користувач має роль суперадміністратора або адміністратора він має змогу додавати користувачів до проєкту (рисунок Д.10).

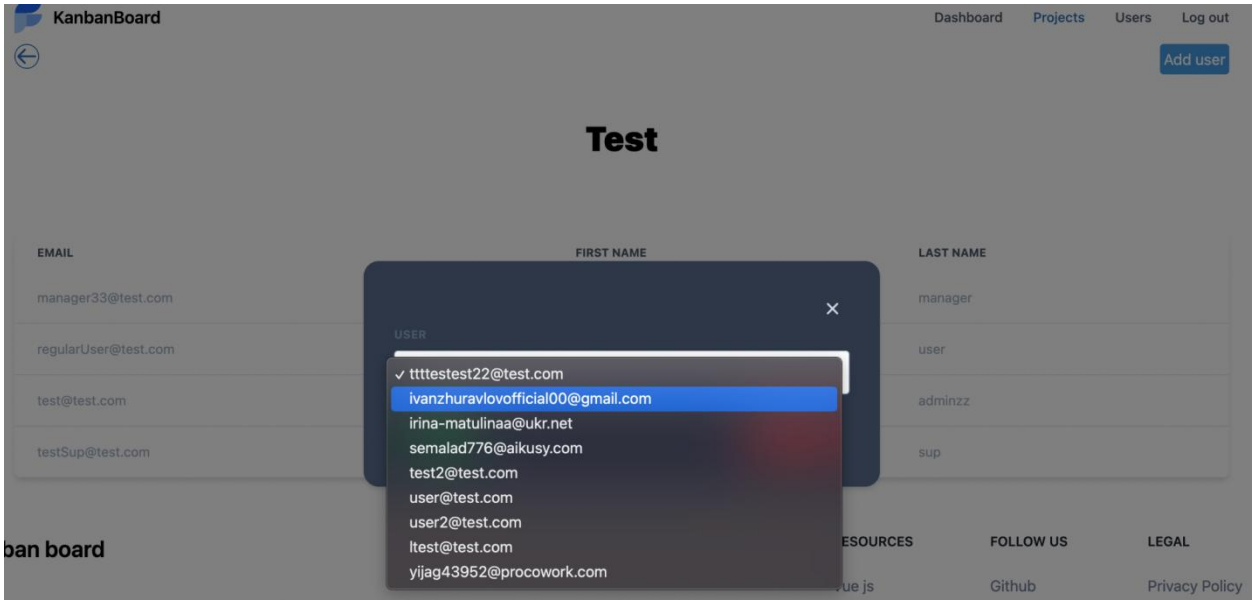

Рисунок Д.10 – Додавання користувача до проєкту

Якщо користувач має роль суперадміністратора або адміністратора він має змогу створювати нові колонки до проєкту (рисунок Д.11).

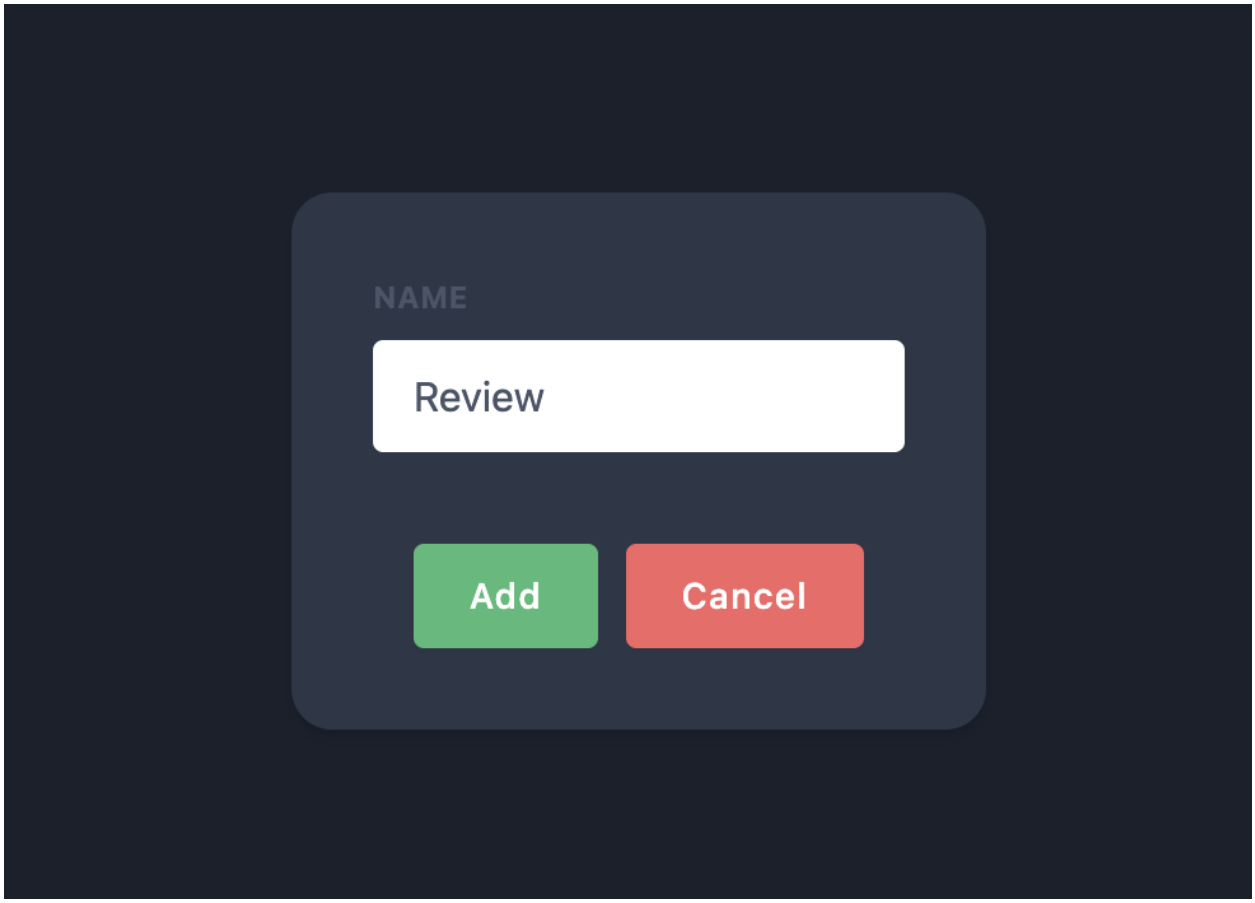

Рисунок Д.11 – Створення нової колонки для проєкту

Якщо користувач має роль суперадміністратора або адміністратора він має змогу змінювати порядок колонок у проєкті (рисунок Д.12).

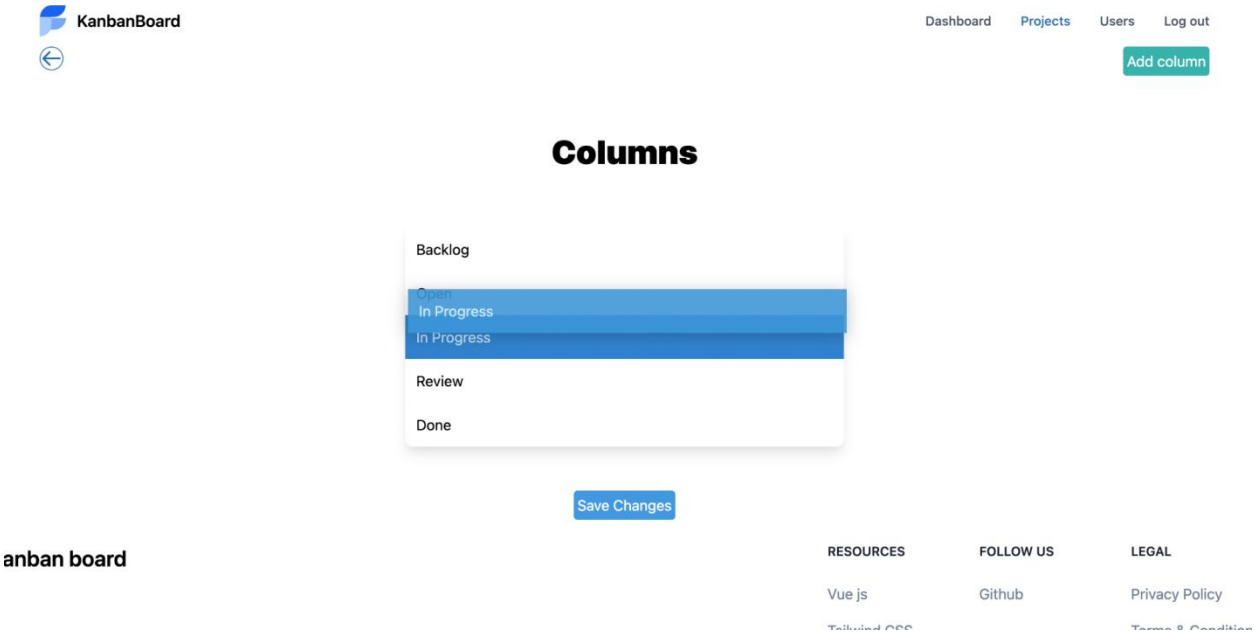

Рисунок Д.12 – Зміна порядку колонок

Якщо користувач має роль суперадміністратора або адміністратора він має змогу створити користувача. Якщо користувач має роль адміністратора він може створювати користувачів з роллю адміністратора та користувача, але якщо користувач має роль суперадміністратора він може створювати користувачів з будь-якою роллю (рисунок Д.13).

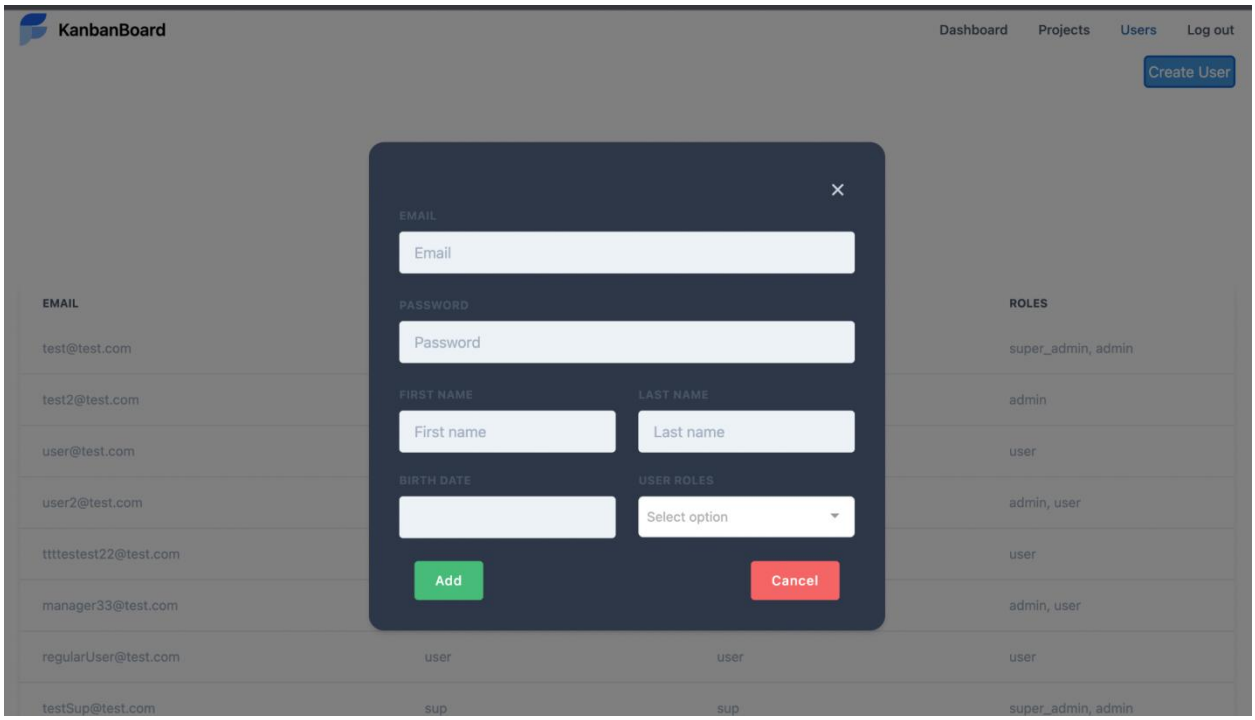

Рисунок Д.13 – Створення користувачів

Якщо користувач має роль суперадміністратора він має змогу переглянути статистику стосовно конверсії проєктів вебсайту (рисунок Д.14).

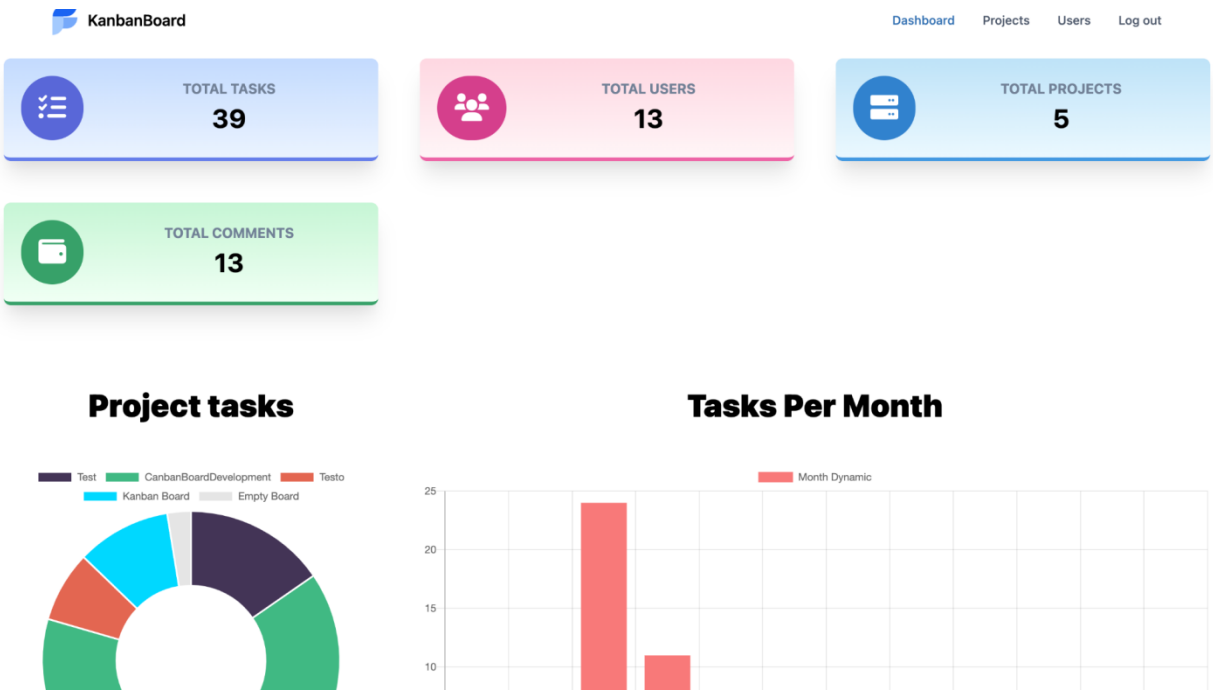

Рисунок Д.14 – Перегляд статистики вебсайту