Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних систем та комп'ютерного моделювання
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи
перший (бакалаврський)

на тему "***Розроблення вебзастосунку для організації особистого часу(дозвілля) з використання JS(backend)***"

Виконав: студент IV курсу, групи ІСТ- 41
Спеціальності
126 "Інформаційні системи і технології"
(шифр і назва напряму підготовки, спеціальності)

Трухан Д.І.
(прізвище та ініціали)

Керівник **доц. Мокрицька О.В.**
(прізвище та ініціали)

Рецензент Думанський О. І.
(прізвище та ініціали)

Львів – 2023

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних систем та комп'ютерного моделювання
Рівень вищої освіти перший (бакалаврський)
Спеціальність 126 "Інформаційні системи та технології"
(шифр і назва)

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри ІСКМ
Сторожук О.Л.
"21" 11 2022 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Трухан Дмитро Іванович
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розроблення вебзастосунку для організації особистого часу(дозвілля) з використанням JS (backend)»*
керівник роботи *Мокрицька О.В., к.т.н., доцент*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від *"21" 11 2022 року № С-521*

2. Термін подання студентом роботи *12.06.2023р.*

3. Вихідні дані до роботи: *Постановка задачі та її формалізації. Аналіз сервісів для організації особистого часу.*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
1) Опис предметної області;
2) Інформаційне та математичне забезпечення;
3) Програмне забезпечення;
4)Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Слайди для доповіді, загальним обсягом 10 слайдів.

6. Дата видачі завдання *23 листопада 2022 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Термін виконання етапів роботи	Примітка
1	Системний аналіз стану проблемної області.	03.10.2022 р. – 28.10.2022 р.	*Виконано*
2.	Збір матеріалів та інформації для формування функціональних вимог та постановка задачі проекту.	31.10.2022 р. – 02.12.2022 р.	*Виконано*
3.	Оформлення першого розділу пояснювальної записки.	05.12.2022 р. – 23.12.2022 р.	*Виконано*
4.	Написання другого розділу. Аналіз інформаційного та математичного забезпечення.	26.12.2022 р. – 20.01.2023 р.	*Виконано*
5.	Написання третього розділу. Проектування вебзастосунку.	23.01.2023 р. – 03.03.2023 р.	*Виконано*
6.	Розробка вебзастосунку для організації особистого часу.	06.03.2023 р. – 31.03.2023 р.	*Виконано*
7.	Тестування програмного продукту та отриманих результатів	03.04.2023 р. – 05.05.2023 р.	*Виконано*
8.	Розробка пояснювальної записки дипломної роботи	08.05.2023 р. – 26.05.2023 р.	*Виконано*
9.	Корегування пояснювальної записки згідно вимог, розроблення презентації	29.05.2023 р. – 09.06.2023 р.	*Виконано*

Студент _____ **Трухан Д.І.**
(підпис) (прізвище та ініціали)

Керівник роботи _____ **Мокрицька О.В.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дипломна робота містить 54 сторінок пояснювальної записки, 30 рисунків, 15 джерел.

В сучасному світі все більше людей звертає увагу на організацію свого часу і планування робочих процесів. Ефективне управління часом є ключовим аспектом продуктивності і досягнення поставлених цілей. У цьому контексті вебзастосунки для організації часу, такі як kanban board, стають дедалі популярнішими серед користувачів.

Ця анотація присвячена розробці вебзастосунку для організації часу на основі принципів kanban board. Основна мета дослідження полягає в створенні ефективного інструменту, який допоможе користувачам планувати, відстежувати та керувати своїми завданнями і проектами.

Об'єктом дослідження є вебзастосунок для організації часу на основі kanban board.

Предметом дослідження є розробка функціональних можливостей та інтерфейсу вебзастосунку, що дозволяє організовувати робочі процеси з використанням принципів kanban board.

Основні ключові слова: вебзастосунок, організація часу, kanban board, планування, управління завданнями.

ABSTRACT

The thesis contains 54 pages of explanatory text, 30 figures, 15 references.

In the modern world, more and more people are paying attention to time organization and workflow planning. Effective time management is a key aspect of productivity and achieving goals. In this context, time organization web applications, such as kanban boards, are becoming increasingly popular among users.

This abstract is dedicated to the development of a web application for time organization based on the principles of kanban boards. The main objective of the research is to create an efficient tool that helps users plan, track, and manage their tasks and projects.

The research object is a web application for time organization based on kanban board principles.

The research subject is the development of functional capabilities and interface of the web application that enables the organization of workflow processes using kanban board principles.

Keywords: web application, time organization, kanban board, planning, task management.

ТЕХНІЧНЕ ЗАВДАННЯ

Метою дослідження є розробити вебзастосунок для організації власного часу (дозвілля).

Завдання дослідження:

1. дослідити технології та актуальність застосунків для організації часу;
2. проаналізувати наявні вебзастосунки для організації часу;
3. здійснити документаційне та інформаційне вирішення вебзастосунку;
4. виконати практичну реалізацію вебзастосунку;
5. розробити вебзастосунок.

ЗМІСТ

ВСТУП

Актуальність теми. У сучасному світі все більше людей звертає увагу на організацію свого часу і планування робочих процесів. Ефективне управління часом є ключовим аспектом продуктивності і досягнення поставлених цілей. В контексті цієї потреби вебзастосунки для організації часу, такі як kanban board, набувають все більшої популярності серед користувачів.

Мета дослідження полягає у розробці вебзастосунку для організації часу на основі принципів kanban board. Основна мета полягає в створенні ефективного інструменту, який допоможе користувачам планувати, відстежувати та керувати своїми завданнями і проектами.

Завдання дослідження:

1. Вивчення технологій та актуальності використання вебзастосунків для організації часу.
2. Аналіз наявних вебзастосунків для організації часу на основі принципів kanban board.
3. Розробка документаційного та інформаційного рішення вебзастосунку.
4. Практична реалізація вебзастосунку.
5. Розробка вебзастосунку.

Об'єкт дослідження - дослідження інформаційних ресурсів для організації часу.

Предмет дослідження - дослідження вебзастосунку для організації часу.

Практичне значення отриманих результатів: надання користувачам зручного і ефективного інструменту для організації їхнього часу за допомогою принципів kanban board.

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Огляд наукових та навчально-методичних джерел

В останні роки організація часу стала значущим аспектом для багатьох людей, які прагнуть досягти більшої продуктивності та ефективності у своїх робочих процесах. Завдяки широкому доступу до інформації та технологічному прогресу, вебзастосунки для організації часу стали популярними серед користувачів.

Одним з таких вебзастосунків є kanban board, що базується на принципах управління завданнями із використанням дошки з картками. Принципи kanban board були розроблені компанією Toyota в рамках системи Lean manufacturing. Вони дозволяють візуалізувати робочий процес, розподілити завдання на етапи та ефективно керувати ними.

Дослідження технологій та наукових джерел показують, що вебзастосунки на основі kanban board мають ряд переваг. Вони надають зручний інтерфейс для створення, переміщення та відстеження завдань, а також сприяють комунікації та спільній роботі в команді. Додатково, вебзастосунки здатні забезпечити розширені можливості, такі як нагадування про терміни виконання завдань, призначення відповідальних осіб, аналітика продуктивності тощо.

Навчально-методичні джерела також акцентують увагу на важливості організації часу та використанні вебзастосунків на основі kanban board. Вони рекомендують використовувати ці інструменти для управління завданнями, планування проектів та підвищення продуктивності роботи. Додатково, навчальні матеріали надають практичні поради щодо оптимального використання таких вебзастосунків та досягнення кращих результатів.

На сьогоднішній день, використання вебзастосунків для організації часу, зокрема kanban board, стає все більш поширеним серед користувачів.

Ось кілька важливих статистичних фактів:

1. За даними дослідження, проведеного в 2021 році, понад 70% професіоналів вважають, що вебзастосунки для організації часу значно покращують їх продуктивність і ефективність роботи.
2. Згідно з аналізом користувачів, який був проведений компанією Trello, одним з популярних вебзастосунків на основі kanban board, за останні роки спостерігається понад 50% зростання активних користувачів.
3. У 2020 році вебзастосунки для організації часу були серед топ-застосунків у категорії "Продуктивність" в магазинах додатків Apple App Store та Google Play Store.
4. За даними дослідження, проведеного компанією Atlassian, понад 80% команд, які використовують вебзастосунки на основі kanban board, повідомляють про покращення спільної роботи та комунікації в команді.
5. У бізнес-середовищі понад 90% Fortune 500 компаній використовують вебзастосунки для організації часу та управління завданнями, щоб підвищити ефективність своїх процесів та досягти поставлених цілей.

Ці статистичні дані підтверджують тенденцію до зростання популярності та використання вебзастосунків на основі kanban board для організації часу. Ці інструменти стають незамінними для багатьох людей та організацій, допомагаючи досягати більшої продуктивності та кращого управління завданнями.

Отже, огляд наукових та навчально-методичних джерел підтверджує актуальність вебзастосунків для організації часу. Ці інструменти допомагають управляти завданнями, планувати робочі процеси та підвищувати продуктивність.

1.2. Огляд законодавчих і нормативних актів

На сьогоднішній день, використання вебзастосунків для організації часу, зокрема kanban board, не підпадає під спеціальні законодавчі акти або нормативи. Проте, існують загальні закони та норми, які можуть бути застосовані до вебзастосунків і впливають на їхнє використання.

Закони про захист персональних даних: В багатьох країнах існують закони та положення, що регулюють збір, зберігання та обробку персональних даних користувачів. Розробники вебзастосунків повинні дотримуватися цих законів, забезпечуючи конфіденційність та безпеку даних користувачів.

Законодавство про працю: Вебзастосунки для організації часу можуть впливати на спосіб організації роботи та співпрацю в команді. Відповідно, вони можуть підпадати під загальні норми праці, які регулюють обов'язки та права працівників, включаючи робочий час, відпочинок та компенсацію праці.

Закони про авторські права: Якщо вебзастосунок використовує матеріали, що підпадають під авторські права (наприклад, шаблони, зображення або текст), розробники повинні дотримуватися відповідних законів про авторські права та отримати необхідні ліцензії або дозволи.

Загальний правовий контекст: Крім специфічних законів, вебзастосунки також підпадають під загальні правові вимоги, пов'язані з договорами, захистом споживачів, захистом інтелектуальної власності та іншими сферами права.

Важливо розуміти, що законодавство та нормативні акти можуть варіюватися залежно від країни, регіону та сфери діяльності. Перед використанням вебзастосунків для організації часу, рекомендується консультуватися з юридичними консультантами або експертами з питань

конкретного законодавства, щоб забезпечити дотримання вимог та уникнути порушення правових норм у своїй діяльності.

1.3. Висновки до розділу

Розділ 1 "Опис предметної області" надав огляд наукових та навчально-методичних джерел, які висвітлюють актуальність вебзастосунків для організації часу на основі принципів kanban board. Була розглянута статистика, яка свідчить про зростання інтересу до організації часу та планування робочих процесів, а також про популярність вебзастосунків у цій сфері.

Також був проведений огляд законодавчих і нормативних актів, які можуть впливати на використання вебзастосунків для організації часу. Це включає закони про захист персональних даних, законодавство про працю, закони про авторські права та загальний правовий контекст. Розробники вебзастосунків повинні бути уважними до цих правових аспектів і дотримуватися відповідних вимог для забезпечення конфіденційності, безпеки даних та додержання авторських прав.

В цілому, розділ надав читачу загальне уявлення про предметну область вебзастосунків для організації часу на основі kanban board. Він створив основу для подальшого розгляду досліджуваної теми та розробки вебзастосунку, враховуючи наукові джерела, статистику та правові аспекти.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Інформаційне забезпечення

Веб-застосунки для організації часу, такі як Kanban-дошка, відіграють важливу роль у покращенні продуктивності та ефективності роботи. Ці інструменти надають можливість впорядковувати завдання, просувати їх через різні стадії та відстежувати їх статус.

Одним з популярних веб-застосунків для організації часу є Kanban-дошка. Вона базується на принципах Kanban-методології, що заснована на візуальному управлінні завданнями. Канбан-дошка складається з колонок, які представляють різні етапи виконання завдань, наприклад: "В очікуванні", "В роботі" та "Завершено". Кожне завдання відображається як картка або елемент, який можна переміщати між колонками в залежності від його статусу.

Веб-застосунки Kanban-дошки мають кілька переваг, які допомагають в організації часу:

1. **Візуальне представлення завдань:** Kanban-дошка надає зручне та інтуїтивно зрозуміле візуальне представлення завдань. Ви можете швидко побачити, які завдання у вас є, які вже виконані, а які ще потребують уваги.
2. **Просування завдань:** Завдяки Kanban-дошці ви можете легко просувати завдання через різні етапи. Ви можете перетягувати картки з однієї колонки до іншої, вказуючи, що завдання знаходиться у відповідному стані.
3. **Підтримка співпраці:** Багато веб-застосунків Kanban-дошки дозволяють кільком користувачам одночасно працювати з однією дошкою. Це дозволяє командам бачити та оновлювати статуси завдань в реальному часі, спілкуватися та обмінюватися

коментарями щодо конкретних завдань. Це полегшує співпрацю та покращує комунікацію всередині команди.

4. **Приоритезація завдань:** Веб-застосунки Kanban-дошки дозволяють легко встановлювати пріоритети для завдань. Ви можете виділити найважливіші завдання або встановити терміни їх виконання, що допомагає зосередитися на найважливіших задачах.
5. **Відстеження продуктивності:** Канбан-дошка надає зручні засоби для відстеження продуктивності та прогресу виконання завдань. Ви можете встановлювати терміни виконання, відзначати завдання, які вже завершено, і отримувати огляди зробленої роботи.
6. **Інтеграція з іншими інструментами:** Багато веб-застосунків Kanban-дошки підтримують інтеграцію з іншими корисними інструментами, такими як календарі, сповіщення, електронна пошта та інші. Це дозволяє вам зберігати всі ваші ресурси та інформацію на одному місці та полегшує управління часом.

Незалежно від того, чи ви працюєте самостійно або в команді, веб-застосунки Kanban-дошки можуть бути потужними інструментами для організації вашого часу та управління завданнями. Різні варіанти доступні онлайн, і ви можете спробувати декілька з них, щоб знайти той, який найкраще відповідає вашим потребам.

2.2. Аналіз об’єктів предметної області та взаємозв’язків між ними

Аналіз об'єктів предметної області та їх взаємозв'язків є важливою складовою процесу розробки будь-якого проекту або системи. Він допомагає зрозуміти, які об'єкти існують у даній області, як вони взаємодіють між собою та які зв'язки існують між ними. Цей аналіз є передумовою для розробки ефективної архітектури та розуміння потреб користувачів.

Під час аналізу об'єктів предметної області ви визначаєте основні сутності, що існують у системі. Це можуть бути реальні або концептуальні об'єкти, які мають певні властивості та поведінку. Наприклад, якщо ви розробляєте систему для управління проектами, основними об'єктами можуть бути "проект", "завдання", "команда", "користувач" тощо.

Крім того, аналізуючи об'єкти, ви встановлюєте взаємозв'язки між ними. Це описує, як об'єкти взаємодіють між собою та які залежності між ними існують. Наприклад, "проект" може мати багато "завдань", "команда" може працювати над кількома "проектами", "користувач" може бути призначений на "завдання" тощо. Встановлення взаємозв'язків допомагає зрозуміти логіку системи та визначити потрібні функціональні можливості.

Аналіз об'єктів предметної області і взаємозв'язків між ними допомагає побудувати ясну та структуровану модель системи. Цей аналіз є основою для подальшого проектування, розробки та впровадження системи. Він дозволяє уникнути недорозумінь, помилок та недоузгодженостей у процесі розробки, а також забезпечує адаптацію системи до потреб користувачів та вимог предметної області.

На ринку існує кілька популярних проектів з управління завданнями та процесами за допомогою Kanban-дошок. Ось опис декількох з них, разом з їх перевагами (+) та недоліками (-):

Trello:

- Інтуїтивний і простий у використанні інтерфейс.
- Багатофункціональність: можливість створювати списки завдань, додавати коментарі, прикріплювати файли тощо.
- Підтримка спільної роботи та колективного проектного управління.
- Обмежені можливості у безкоштовній версії.
- Відсутність деяких продвинутих функцій, які можуть бути необхідні для великих команд або складних проектів.

Jira Software:

- Розширені можливості для управління проектами та розробки ПЗ.
- Інтеграція з іншими популярними інструментами розробки ПЗ.
- Потужна система керування задачами та планування.
- Висока складність в освоєнні та конфігурації.
- Вартість ліцензії та підтримки може бути високою для невеликих команд або проектів.

Monday.com:

- Гнучка конфігурація та налаштування дошок для власних потреб.
- Велика кількість інтеграцій з іншими інструментами та сервісами.
- Зручність в візуалізації процесів та статусів завдань.
- Вартість підписки може бути значною, особливо для великих команд або підприємств.

Asana:

- Можливість планування та приоритезації завдань.
- Система сповіщень та нагадувань.
- Зручність в управлінні проектами та розподілі завдань між командами.
- Обмежені можливості у безкоштовній версії.
- Відсутність деяких розширених функцій для більш складних проектів.

Кожен з цих проектів має свої переваги та недоліки, і вибір залежить від конкретних потреб та уподобань команди чи організації.

Аналіз різних ринків свідчить, що ефективна організація часу - це тема, яка стосується всіх людей у всьому світі. Однак жоден з перелічених

вище веб-застосунків не пропонує повноцінного рішення для ефективного управління часом. Існує реальна потреба у створенні інформаційних ресурсів для ефективного планування та організації робочих завдань та проектів.

2.3. Формальна модель задачі

Формальна модель задачі для розробки Kanban board може бути наступною:

1. Множина об'єктів:
    - Дошка (Board): Представляє собою основний контейнер для стадій роботи та завдань.
    - Стадія (Stage): Представляє конкретну фазу роботи, наприклад, "Не розпочато", "В процесі", "Завершено".
    - Завдання (Task): Представляє окреме завдання, яке потрібно виконати. Містить інформацію про заголовок, опис, пріоритет, термін виконання та ін.
2. Взаємозв'язки між об'єктами:
    - Кожна дошка (Board) має декілька стадій (Stage), які визначають послідовність робочого процесу.
    - Кожна стадія (Stage) містить декілька завдань (Task), які перебувають на цій стадії.
    - Завдання (Task) можуть переміщатись між стадіями (Stage) залежно від їх прогресу та поточного стану.
3. Операції над об'єктами:
    - Створення нової дошки (Board).
    - Додавання нової стадії (Stage) до дошки (Board).
    - Додавання нового завдання (Task) до стадії (Stage).
    - Видалення дошки (Board) разом зі всіма стадіями (Stage) та завданнями (Task).

- Переміщення завдання (Task) з однієї стадії (Stage) на іншу.
- Зміна властивостей завдання (Task), таких як заголовок, опис, пріоритет, термін виконання тощо.

Ця формальна модель задачі визначає основні об'єкти, їх взаємозв'язки та операції, які можуть бути виконані над ними. Вона допомагає уявити структуру та функціональність Kanban board і є основою для подальшої розробки та реалізації веб-застосунку.

2.4. Інформаційна модель

Інформаційна модель для Kanban board може бути наступною:

1. Об'єкти і атрибути:
    - Дошка (Board):
        - board_id: унікальний ідентифікатор дошки.
        - назва: назва дошки.
        - стадії: список стадій, пов'язаних з даною дошкою.

    - Стадія (Stage):
        - stage_id: унікальний ідентифікатор стадії.
        - назва: назва стадії.
        - дошка: ідентифікатор дошки, до якої належить стадія.
        - завдання: список завдань, пов'язаних з даною стадією.

    - Завдання (Task):
        - task_id: унікальний ідентифікатор завдання.
        - заголовок: заголовок завдання.
        - опис: опис завдання.
        - пріоритет: пріоритет завдання.
        - термін: термін виконання завдання.

- стадія: ідентифікатор стадії, до якої належить завдання.

2. Взаємозв'язки між об'єктами:
    - Кожна дошка (Board) містить декілька стадій (Stage), пов'язаних з нею.
    - Кожна стадія (Stage) належить до певної дошки (Board).
    - Кожне завдання (Task) належить до певної стадії (Stage).

3. Операції та запити:
    - Створення нової дошки (Board) з вказаною назвою.
    - Отримання списку всіх дошок (Board) з їх атрибутами.
    - Отримання інформації про конкретну дошку (Board) за її ідентифікатором.
    - Додавання нової стадії (Stage) до дошки (Board) з вказаною назвою.
    - Отримання списку стадій (Stage) для певної дошки (Board) за її ідентифікатором.
    - Додавання нового завдання (Task) до стадії (Stage) з вказаними атрибутами.
    - Отримання списку завдань (Task) для певної стадії (Stage) за її ідентифікатором.
    - Переміщення завдання (Task) з однієї стадії (Stage) в іншу.
    - Оновлення атрибутів завдання (Task), таких як заголовок, опис, пріоритет, термін виконання.

Ця інформаційна модель визначає структуру та зв'язки між об'єктами, а також набір операцій та запитів, які можуть бути виконані над ними. Вона визначає, яку інформацію потрібно зберігати та як її організувати для ефективної роботи з Kanban board.

2.5. Функціональна модель завдання

Функціональна модель завдання для Kanban board може бути наступною:

1. Створення дошок:
    - Користувач може створити нову дошку, вказавши назву.
    - Система створює нову дошку з унікальним ідентифікатором і зберігає її в базі даних.
2. Додавання стадій до дошок:
    - Користувач може додати нову стадію до дошки, вказавши назву та ідентифікатор дошки.
    - Система перевіряє наявність дошки з вказаним ідентифікатором і додає нову стадію до цієї дошки.
3. Додавання завдань до стадій:
    - Користувач може додати нове завдання до стадії, вказавши заголовок, опис, пріоритет, термін виконання та ідентифікатор стадії.
    - Система перевіряє наявність стадії з вказаним ідентифікатором і додає нове завдання до цієї стадії.
4. Переміщення завдань між стадіями:
    - Користувач може перемістити завдання з однієї стадії до іншої.
    - Система перевіряє наявність початкової та цільової стадій і змінює приналежність завдання.
5. Оновлення атрибутів завдань:
    - Користувач може оновлювати атрибути завдань, такі як заголовок, опис, пріоритет, термін виконання.
    - Система зберігає оновлені дані про завдання.
6. Видалення дошок, стадій та завдань:

- Користувач може видалити дошку, стадію або завдання з системи.
- Система видаляє відповідний об'єкт з бази даних.

Ця функціональна модель описує основні операції, які можуть бути виконані користувачами системи Kanban board. Користувачі можуть створювати дошки, додавати стадії та завдання, переміщувати завдання між стадіями, оновлювати їх атрибути та видаляти їх з системи.

2.6. Вибір технологічних рішень завдання

При виборі технологічних рішень для реалізації веб-застосунку Kanban board(backend), можуть бути використані такі технології:

1. JavaScript (JS): JavaScript є основною мовою програмування для веб-розробки, і вона підходить для створення інтерактивних елементів і функціоналу веб-застосунку Kanban board. JS забезпечує можливість динамічної зміни сторінки, обробки подій, валідації даних та інші функції, необхідні для реалізації веб-застосунку.
2. Node.js: Node.js є платформою, яка дозволяє виконувати JavaScript на серверному боці. Використання Node.js дозволить створити серверну частину веб-застосунку, яка буде обробляти запити від клієнтів, зберігати дані і забезпечувати комунікацію з базою даних.
3. MongoDB: MongoDB є документ-орієнтованою базою даних, яка може бути використана для зберігання і управління даними веб-застосунку Kanban board. MongoDB дозволяє зберігати структуровані дані у вигляді документів, що спрощує роботу з ними та дозволяє швидше виконувати операції з базою даних.

Такий стек технологій (JS, Node.js, MongoDB) є популярним та широко використовується для веб-розробки. Він забезпечує гнучкість, продуктивність та швидкість розробки, що може бути важливими факторами для успішної реалізації веб-застосунку Kanban board. Однак, вибір технологій також може залежати від досвіду розробника, потреб проекту та доступних ресурсів.

2.7. Висновки до розділу

Можна зробити наступні висновки:

1. Kanban board - це інструмент для організації робочого процесу та керування завданнями. Він дозволяє візуалізувати стадії роботи, відстежувати прогрес та зберігати необхідну інформацію про завдання.
2. При розробці веб-застосунку Kanban board необхідно провести аналіз предметної області, визначити основні об'єкти та їх взаємозв'язки. Це допоможе зрозуміти, які функції та можливості повинен мати застосунок.
3. Для розробки інформаційної моделі Kanban board можна використовувати поняття, такі як дошка, стадії роботи, завдання, користувачі тощо. Це допоможе визначити, які дані потрібно зберігати та як вони будуть взаємодіяти між собою.
4. Функціональна модель Kanban board повинна включати основні функції, такі як створення завдань, переміщення їх між стадіями, відстеження прогресу, нагадування, спільна робота користувачів тощо. Вона визначає, які можливості має надавати веб-застосунок.
5. При виборі технологічних рішень для розробки Kanban board можна використовувати JavaScript (JS) для фронтенду, Node.js для бекенду та MongoDB для зберігання даних. Цей стек технологій є

популярним та забезпечує продуктивність та гнучкість веб-застосунку.

Розробка веб-застосунку Kanban board є важливою для організації робочого процесу та ефективного керування завданнями. Використання аналізу предметної області, інформаційної та функціональної моделей, а також вибір правильних технологій є ключовими кроками у процесі розробки успішного Kanban board.

РОЗДІЛ 3. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Описання реалізації завдання

Основною метою бакалаврської кваліфікаційної роботи було створення вебзастосунку для організації власного часу.

Першим кроком потрібно вибрати “фундамент” проєкту, ним став - Node.js.

Створення клієнтської та серверної частини в Node.js має свою раціональність та визначену функціональність. Ось для чого потрібно створити кожну з цих частин(рисунок 3.1 та рисунок 3.2):

Клієнтська частина (Client):

1. Взаємодія з користувачем: Клієнтська частина відповідає за інтерфейс та взаємодію з користувачем. Вона забезпечує візуальне представлення Kanban board, його стадій, завдань та деталей. Клієнтська частина дозволяє користувачеві додавати, редагувати та видаляти завдання, переміщати їх між стадіями та здійснювати інші дії.
2. Локальне зберігання даних: Клієнтська частина може зберігати деяку інформацію локально на браузері користувача, щоб забезпечити швидку та незалежну роботу, навіть при відсутності з'єднання з сервером. Це може бути корисним для забезпечення плавної роботи та зменшення часу очікування.

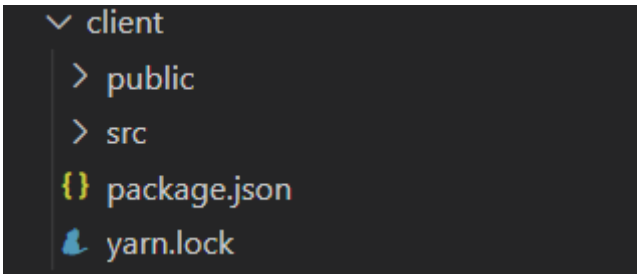

Рисунок 3.1 - Клієнт проєкту

Серверна частина (Server):

1. Обробка запитів: Серверна частина відповідає за обробку запитів, які надходять від клієнтів. Вона забезпечує логіку обробки та виконання дій, пов'язаних з Kanban board. Сервер приймає запити, перевіряє їх на коректність, взаємодіє з базою даних та відправляє відповідь клієнту.
2. Керування даними: Серверна частина відповідає за зберігання та управління даними, пов'язаними з Kanban board. Вона взаємодіє з базою даних (наприклад, MongoDB) для збереження та отримання інформації про стадії, завдання та інші деталі. Сервер забезпечує консистентність та безпеку даних.
3. Аутентифікація та авторизація: Серверна частина відповідає за аутентифікацію та авторизацію користувачів. Вона забезпечує захист доступу до Kanban board та його функціональності, забезпечуючи, що лише авторизовані користувачі можуть змінювати стани, завдання та інші аспекти.

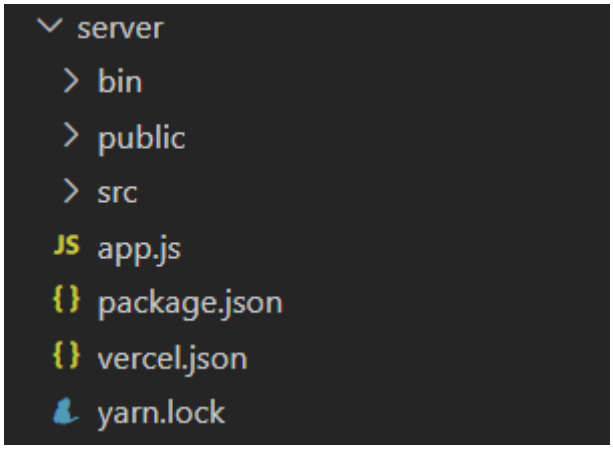

Рисунок 3.2 - Сервер проєкту

Створення клієнтської та серверної частини в Node.js дозволяє розділити функціональність та забезпечити ефективну комунікацію між ними. Клієнтська частина відповідає за взаємодію з користувачем та

локальне зберігання даних, тоді як серверна частина обробляє запити, керує даними та забезпечує безпеку та авторизацію. Такий підхід дозволяє створити потужний та функціональний веб-додаток Kanban board.

Наступним кроком прописується User API.

User API використовується для забезпечення взаємодії з користувачами веб-додатку. Його основна функціональність полягає в обробці запитів, пов'язаних з управлінням користувачами, їх аутентифікацією та авторизацією. Ось деякі конкретні завдання та функції, які можуть бути реалізовані за допомогою User API:

1. Реєстрація користувачів: User API дозволяє користувачам створювати нові облікові записи в системі. Це може включати збереження інформації про користувача, такої як ім'я, електронна пошта, пароль тощо.
2. Аутентифікація та авторизація: User API забезпечує механізми аутентифікації користувачів, перевірку їх ідентифікаційних даних та надання доступу до захищених ресурсів. Він використовує методи аутентифікації, такі як введення логіна та пароля або використання токенів, для перевірки прав користувача та надання доступу до відповідних функцій.
3. Управління профілем користувача: User API дозволяє користувачам змінювати свої профілі, оновлювати інформацію про себе, змінювати пароль, додавати чи видаляти зображення та інші пов'язані з профілем дії.
4. Відновлення паролю: User API може надавати можливість відновлення забутого паролю користувача. Це може включати відправку електронної пошти з посиланням на зміну паролю або використання інших методів підтвердження.

5. Управління правами доступу: User API дозволяє керувати рівнями доступу користувачів до різних функціональних частин додатку. Це включає надання прав адміністратора, модератора чи звичайного користувача залежно від ролі та привілеїв.

Загалом, User API є ключовим компонентом для управління користувачами веб-додатку. Він забезпечує безпеку, аутентифікацію та авторизацію, а також надає користувачам можливість взаємодіяти зі своїми профілями та виконувати різні дії, пов'язані з їх обліковими записами.

В контексті Node.js та веб-розробки, "контролер" (controller), "обробник" (handler), "модель" (model) та "маршрути" (routes) - це поняття, що використовуються для побудови структури та організації веб-додатків.

1. Контролер (Controller): Контролер в Node.js є складовою частиною архітектури MVC (Model-View-Controller) та використовується для обробки логіки додатку. Контролери відповідають за прийняття запитів від користувача та взаємодію з моделями та обробниками (handlers) для обробки запитів та формування відповідей (Рисунок 3.3).

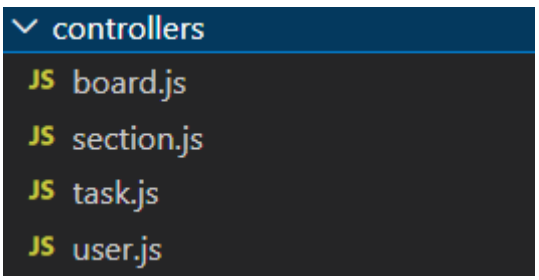

Рисунок 3.3 - Реалізація контролера

2. Обробник (Handler): Обробники (або також відомі як middleware) виконуються перед або після контролерів, і вони відповідають за обробку конкретних типів запитів або виконання певних дій. Вони можуть перевіряти та обробляти дані, здійснювати автентифікацію, валідацію або здійснювати інші необхідні перевірки перед передачею управління до контролера (Рисунок 3.4).

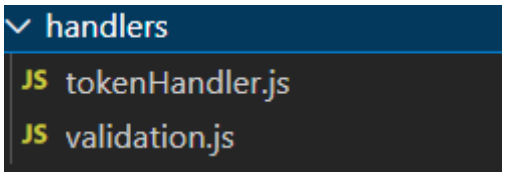

Рисунок 3.4 - Реалізація обробника

3. Модель (Model): Моделі в Node.js представляють собою структури даних або об'єкти, які використовуються для збереження та маніпулювання даними. Вони представляють сутності додатку, такі як користувачі, продукти, замовлення тощо. Моделі включають методи для отримання, збереження, оновлення та видалення даних в базі даних або іншому джерелі даних (Рисунок 3.5).

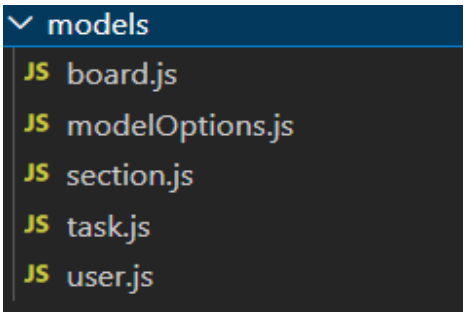

Рисунок 3.5 - Реалізація моделі

4. Маршрути (Routes): Маршрути визначають, які URL-шляхи (роути) відповідають на які запити, тобто який контролер та

обробник повинні бути виконані для кожного конкретного запиту. Вони декларуються у вигляді шаблонів URL та асоціюються з відповідними контролерами та обробниками. Маршрути дозволяють налаштовувати веб-додаток на обробку запитів і перенаправляти їх на відповідні обробники та контролери (Рисунок 3.6).

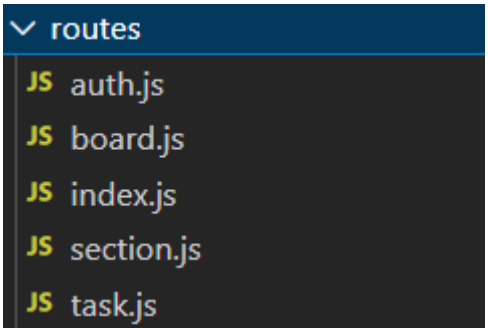

Рисунок 3.6 - Реалізація маршрутів

Ці компоненти допомагають організувати та розділити різні аспекти логіки та функціональності веб-додатку в Node.js. Контролери взаємодіють з моделями та обробниками, обробники виконують проміжні дії та валідацію, моделі здійснюють роботу з даними, а маршрути визначають, які запити співставляються з якими контролерами та обробниками. Ця організація сприяє покращенню структури, розширюваності та підтримки веб-додатку.

Прописуємо Boards API:

Boards API (Application Programming Interface) є частиною веб-додатка, яка надає можливість взаємодії з даними дошок (boards). Вона дозволяє клієнтським додаткам створювати, зчитувати, оновлювати та видаляти дані дошок з веб-додатку.

Основні операції, які можна виконати за допомогою Boards API, включають:

1. Створення дошки: API надає можливість створювати нові дошки, вказуючи необхідну інформацію, таку як назва дошки, опис, дата створення тощо. Після створення дошки їй буде присвоєно унікальний ідентифікатор, щоб забезпечити можливість подальшої ідентифікації та взаємодії з нею.
2. Отримання даних дошки: За допомогою API можна отримати інформацію про конкретну дошку за її ідентифікатором. Це може включати назву, опис, списки завдань (tasks), дедлайни, учасників та інші деталі.
3. Оновлення дошки: З API можна оновлювати дані дошки, наприклад, змінювати назву, опис, додавати або видаляти завдання, змінювати статуси завдань, оновлювати дедлайни тощо. Це дозволяє забезпечити актуальність та коректність даних дошки.
4. Видалення дошки: API дозволяє видаляти дошку з системи за її ідентифікатором. Ця операція повністю видаляє всі дані дошки із системи, і вона не може бути відновлена.

Додатково до основних операцій, Boards API може надавати інші функції, такі як перегляд списків завдань, робота з користувачами та їх доступами до дошки, фільтрація та сортування даних тощо. API зазвичай використовує стандартні HTTP-методи (GET, POST, PUT, DELETE) для взаємодії з ресурсами та повертає дані у форматі JSON або іншому форматі, який легко обробляти клієнтськими додатками.

Завдяки Boards API клієнтські додатки можуть інтегруватись з веб-додатком і отримувати доступ до функцій та даних дошок, що дозволяє зручно та ефективно управляти задачами, спільно працювати над проектами та забезпечувати синхронізацію між різними пристроями та користувачами.

В вебзастосунку реалізація Boards API виглядає так:

Controllers/boards.js(рисунок 3.7.1, 3.7.2, 3.7.3):

server > src > v1 > controllers > JS board.js > ...

```
const Board = require('../models/board')
const Section = require('../models/section')
const Task = require('../models/task')

exports.create = async (req, res) => {
  try {
    const boardsCount = await Board.find().count()
    const board = await Board.create({
      user: req.user._id,
      position: boardsCount > 0 ? boardsCount : 0
    })
    res.status(201).json(board)
  } catch (err) {
    res.status(500).json(err)
  }
}

exports.getAll = async (req, res) => {
  try {
    const boards = await Board.find({ user: req.user._id }).sort('-position')
    res.status(200).json(boards)
  } catch (err) {
    res.status(500).json(err)
  }
}

exports.updatePosition = async (req, res) => {
  const { boards } = req.body
  try {
    for (const key in boards.reverse()) {
      const board = boards[key]
      await Board.findByIdAndUpdate(
        board.id,
        { $set: { position: key } }
      )
    }
    res.status(200).json('updated')
  } catch (err) {
    res.status(500).json(err)
  }
}

exports.getOne = async (req, res) => {
  const { boardId } = req.params
  try {
    const board = await Board.findOne({ user: req.user._id, _id: boardId })
    if (!board) return res.status(404).json('Board not found')
    const sections = await Section.find({ board: boardId })
    for (const section of sections) {
      const tasks = await Task.find({ section: section.id }).populate('section').sort('-position')
      section._doc.tasks = tasks
    }
    board._doc.sections = sections
    res.status(200).json(board)
  } catch (err) {
    res.status(500).json(err)
  }
}

exports.update = async (req, res) => {
  const { boardId } = req.params
  const { title, description, favourite } = req.body

```

Рисунок 3.7.1 - Реалізація Controllers/boards.js

server > src > v1 > controllers > JS board.js > ...

```
    try {
      if (title === '') req.body.title = 'Untitled'
      if (description === '') req.body.description = 'Add description here'
      const currentBoard = await Board.findById(boardId)
      if (!currentBoard) return res.status(404).json('Board not found')

      if (favourite !== undefined && currentBoard.favourite !== favourite) {
        const favourites = await Board.find({
          user: currentBoard.user,
          favourite: true,
          _id: { $ne: boardId }
        }).sort('favouritePosition')
        if (favourite) {
          req.body.favouritePosition = favourites.length > 0 ? favourites.length : 0
        } else {
          for (const key in favourites) {
            const element = favourites[key]
            await Board.findByIdAndUpdate(
              element.id,
              { $set: { favouritePosition: key } }
            )
          }
        }
      }

      const board = await Board.findByIdAndUpdate(
        boardId,
        { $set: req.body }
      )
      res.status(200).json(board)
    } catch (err) {
      res.status(500).json(err)
    }
  }

  exports.getFavourites = async (req, res) => {
    try {
      const favourites = await Board.find({
        user: req.user._id,
        favourite: true
      }).sort('-favouritePosition')
      res.status(200).json(favourites)
    } catch (err) {
      res.status(500).json(err)
    }
  }

  exports.updateFavouritePosition = async (req, res) => {
    const { boards } = req.body
    try {
      for (const key in boards.reverse()) {
        const board = boards[key]
        await Board.findByIdAndUpdate(
          board.id,
          { $set: { favouritePosition: key } }
        )
      }
      res.status(200).json('updated')
    } catch (err) {
      res.status(500).json(err)
    }
  }

```

Рисунок 3.7.2 - Реалізація Controllers/boards.js

Рисунок 3.7.3 - Реалізація Controllers/boards.js

Models/boards.js(рисунок 3.8):

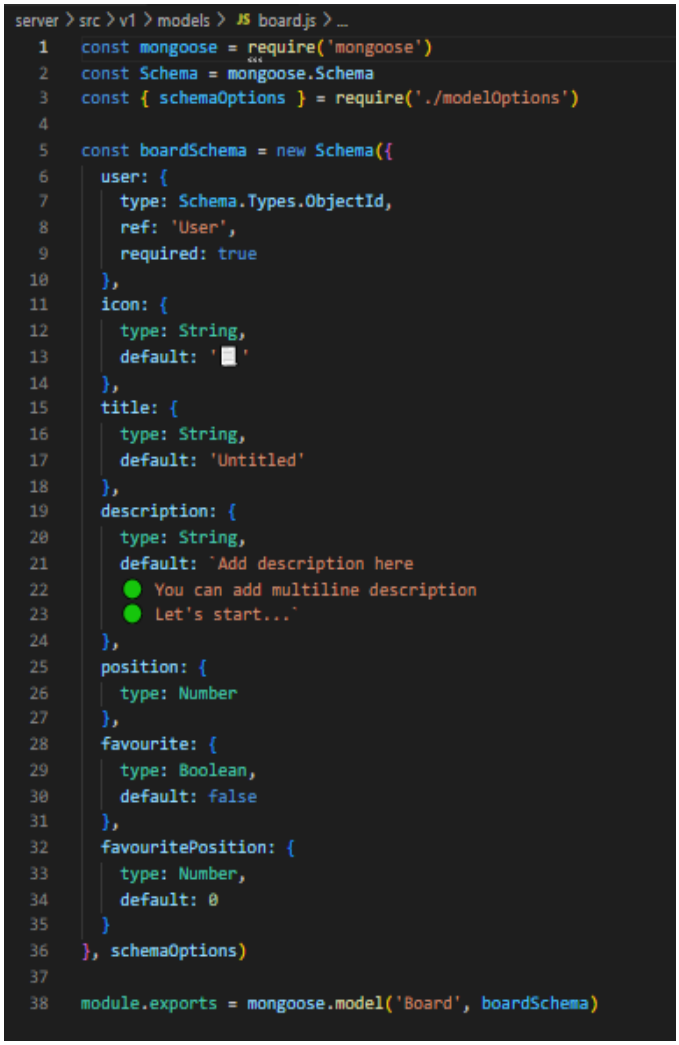

Рисунок 3.8 - Реалізація Models/boards.js

Models/section.js(рисунок 3.9):

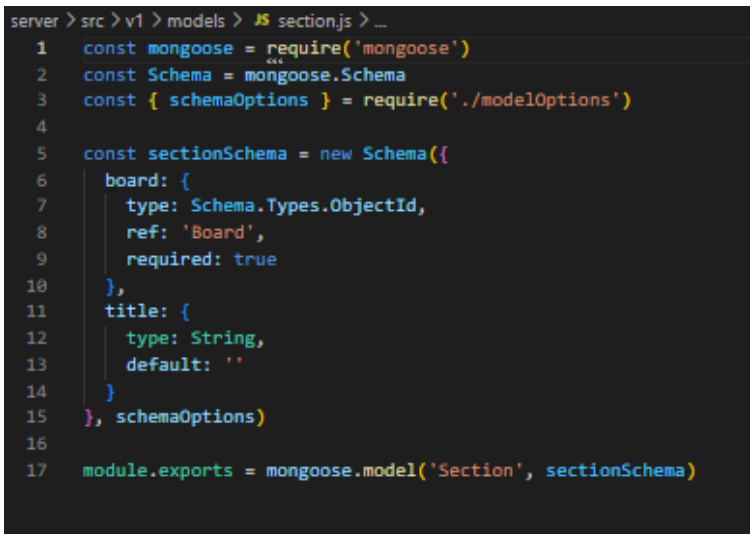

Рисунок 3.9 - Реалізація Models/section.js

Models/task.js(рисунок 3.10):

Рисунок 3.10 - Реалізація Models/task.js

Routes/board.js(рисунок 3.11):

server > src > v1 > routes > JS board.js > ...

```
const router = require('express').Router()
const { param } = require('express-validator')
const validation = require('../handlers/validation')
const tokenHandler = require('../handlers/tokenHandler')
const boardController = require('../controllers/board')

router.post(
  '/',
  tokenHandler.verifyToken,
  boardController.create
)

router.get(
  '/',
  tokenHandler.verifyToken,
  boardController.getAll
)

router.put(
  '/',
  tokenHandler.verifyToken,
  boardController.updatePosition
)

router.get(
  '/favourites',
  tokenHandler.verifyToken,
  boardController.getFavourites
)

router.put(
  '/favourites',
  tokenHandler.verifyToken,
  boardController.updateFavouritePosition
)

router.get(
  '/:boardId',
  param('boardId').custom(value => {
    if (!validation.isObjectId(value)) {
      return Promise.reject('invalid id')
    } else return Promise.resolve()
  }),
  validation.validate,
  tokenHandler.verifyToken,
  boardController.getOne
)

router.put(
  '/:boardId',
  param('boardId').custom(value => {
    if (!validation.isObjectId(value)) {
      return Promise.reject('invalid id')
    } else return Promise.resolve()
  }),
  validation.validate,
  tokenHandler.verifyToken,
  boardController.update
)

router.delete(
  '/:boardId',
  param('boardId').custom(value => {
    if (!validation.isObjectId(value)) {
      return Promise.reject('invalid id')
    } else return Promise.resolve()
  }),
  validation.validate,
  tokenHandler.verifyToken,
  boardController.delete
)


module.exports = router
```

Рисунок 3.11 - Реалізація Routes/boards.js

Routes/auth.js(рисунок 3.12):

Рисунок 3.12 - Реалізація Routes/auth.js

Прописуємо Section API:

Section API (Application Programming Interface) є частиною веб-додатка, яка надає можливість взаємодії з розділами (sections). Вона дозволяє клієнтським додаткам створювати, зчитувати, оновлювати та видаляти дані розділів з веб-додатку.

Основні операції, які можна виконати за допомогою Section API, включають:

1. Створення розділу: API надає можливість створювати нові розділи, вказуючи необхідну інформацію, таку як назва розділу, опис, тип тощо. Після створення розділу йому буде присвоєно унікальний ідентифікатор, щоб забезпечити можливість подальшої ідентифікації та взаємодії з ним.
2. Отримання даних розділу: За допомогою API можна отримати інформацію про конкретний розділ за його ідентифікатором. Це може включати назву, опис, тип, списки завдань (tasks) та інші деталі, які пов'язані з розділом.
3. Оновлення розділу: З API можна оновлювати дані розділу, наприклад, змінювати назву, опис, тип, додавати або видаляти завдання, змінювати їх статуси тощо. Це дозволяє забезпечити актуальність та коректність даних розділу.
4. Видалення розділу: API дозволяє видаляти розділ з системи за його ідентифікатором. Ця операція повністю видаляє всі дані розділу із системи, і вони не можуть бути відновлені.

Додатково до основних операцій, Section API може надавати інші функції, такі як перегляд завдань у конкретному розділі, фільтрація та сортування даних, робота з учасниками розділу та їх доступами, а також багато іншого, що специфічно для конкретного додатка або потреб користувачів.

Завдяки Section API, клієнтські додатки можуть ефективно взаємодіяти з розділами веб-додатка, забезпечуючи гнучкість, зручність та можливість швидкої роботи з даними розділів.

В вебзастосунку реалізація Section API виглядає так:

Controllers/section.js(рисунок 3.13):

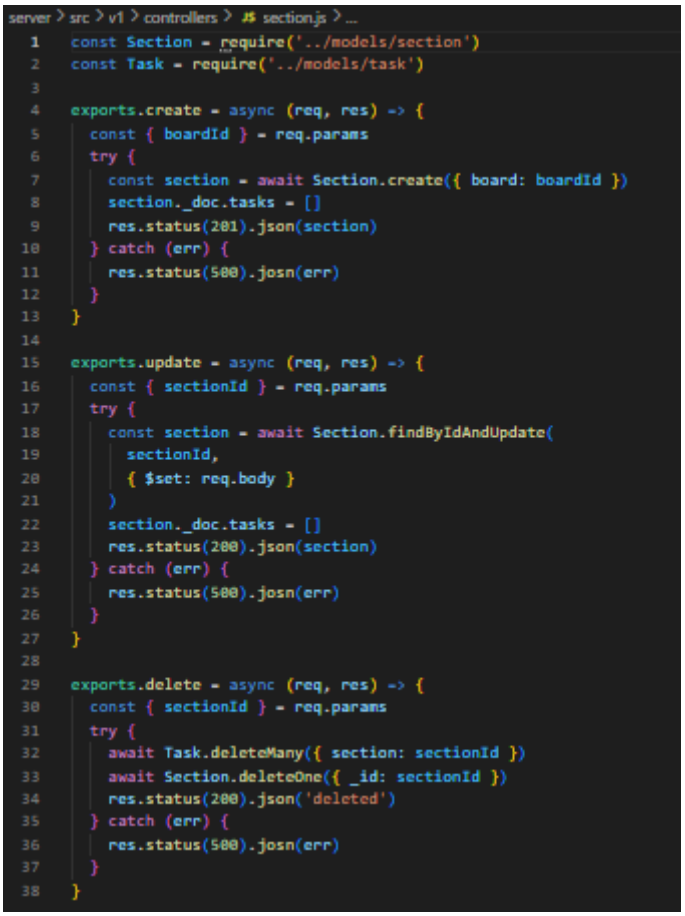

Рисунок 3.13 - Реалізація Controllers/section.js

Routes/section.js(рисунок 3.14):

Реалізація 3.14 - Реалізація Routes/section.js

Прописуємо Task API:

Task API (Application Programming Interface) є частиною веб-додатка, яка надає можливість взаємодії з завданнями (tasks). Вона дозволяє клієнтським додаткам створювати, зчитувати, оновлювати та видаляти дані завдань з веб-додатка.

Основні операції, які можна виконати за допомогою Task API, включають:

1. Створення завдання: API надає можливість створювати нові завдання в межах розділів або підзадач в межах існуючих завдань. Під час створення можна вказати різноманітну інформацію, таку як назва завдання, опис, призначення, терміни виконання тощо. Завданню також можна присвоїти певний статус, наприклад, "В процесі", "Завершено" або "Відкладено".
2. Отримання даних завдання: За допомогою API можна отримати інформацію про конкретне завдання за його ідентифікатором. Це може включати назву, опис, призначення, терміни виконання, статус завдання та інші важливі деталі.
3. Оновлення завдання: З API можна оновлювати дані завдання, змінювати їх статуси, терміни виконання, призначення, опис та інші атрибути. Це дозволяє забезпечити актуальність та коректність даних завдання в системі.
4. Видалення завдання: API дозволяє видаляти завдання з системи за їх ідентифікаторами. Ця операція повністю видаляє дані завдання із системи, і вони не можуть бути відновлені.

Додатково до основних операцій, Task API може надавати інші функції, такі як прикріплення файлів до завдань, коментування завдань, робота з підзавданнями, зміна пріоритетів тощо. Всі ці можливості допомагають клієнтським додаткам ефективно керувати завданнями в системі та сприяють кращій організації та співпраці в команді.

В вебзастосунку реалізація Task API виглядає так:

Controllers/task.js(рисунок 3.15.1 та 3.15.2):

server > src > v1 > controllers > JS task.js > ...

```
const Task = require('../models/task')
const Section = require('../models/section')

exports.create = async (req, res) => {
  const { sectionId } = req.body
  try {
    const section = await Section.findById(sectionId)
    const tasksCount = await Task.find({ section: sectionId }).count()
    const task = await Task.create({
      section: sectionId,
      position: tasksCount > 0 ? tasksCount : 0
    })
    task._doc.section = section
    res.status(201).json(task)
  } catch (err) {
    res.status(500).json(err)
  }
}

exports.update = async (req, res) => {
  const { taskId } = req.params
  try {
    const task = await Task.findByIdAndUpdate(
      taskId,
      { $set: req.body }
    )
    res.status(200).json(task)
  } catch (err) {
    res.status(500).json(err)
  }
}

exports.delete = async (req, res) => {
  const { taskId } = req.params
  try {
    const currentTask = await Task.findById(taskId)
    await Task.deleteOne({ _id: taskId })
    const tasks = await Task.find({ section: currentTask.section }).sort('postition')
    for (const key in tasks) {
      await Task.findByIdAndUpdate(
        tasks[key].id,
        { $set: { position: key } }
      )
    }
    res.status(200).json('deleted')
  } catch (err) {
    res.status(500).json(err)
  }
}

```

Рисунок 3.15.1 - Реалізація Controllers/task.js

Рисунок 3.15.2 - Реалізація Controllers/task.js

Routes/task.js(рисунок 3.16):

Рисунок 3.16 - Реалізація Routes/task.js

Загалом таким чином виглядає реалізація вебзастосунку для організації власного часу (backend).

3.2. Аналіз отриманих результатів

Для запусу вебзастосунку для організації власного часу потрібно запустити власний локальний сервер, після чого запускаємо клієнт та отримуємо такий результат(Рисунок 3.17):

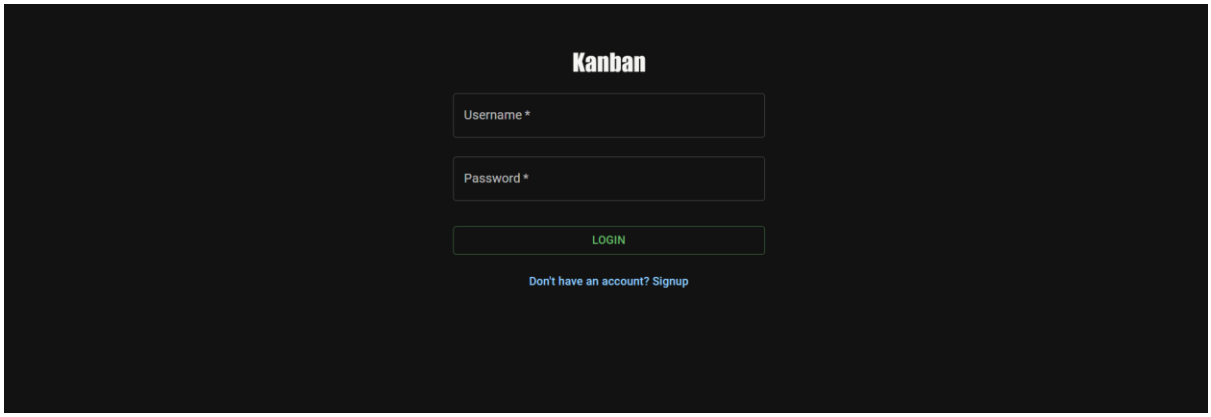

Рисунок 3.17 - Головна сторінка авторизації

На даній сторінці знаходяться поля для вводу логіну та паролю, кнопка авторизації та кнопка реєстрації, за якою можна перейти на сторінку реєстрації(Рисунок 3.18).

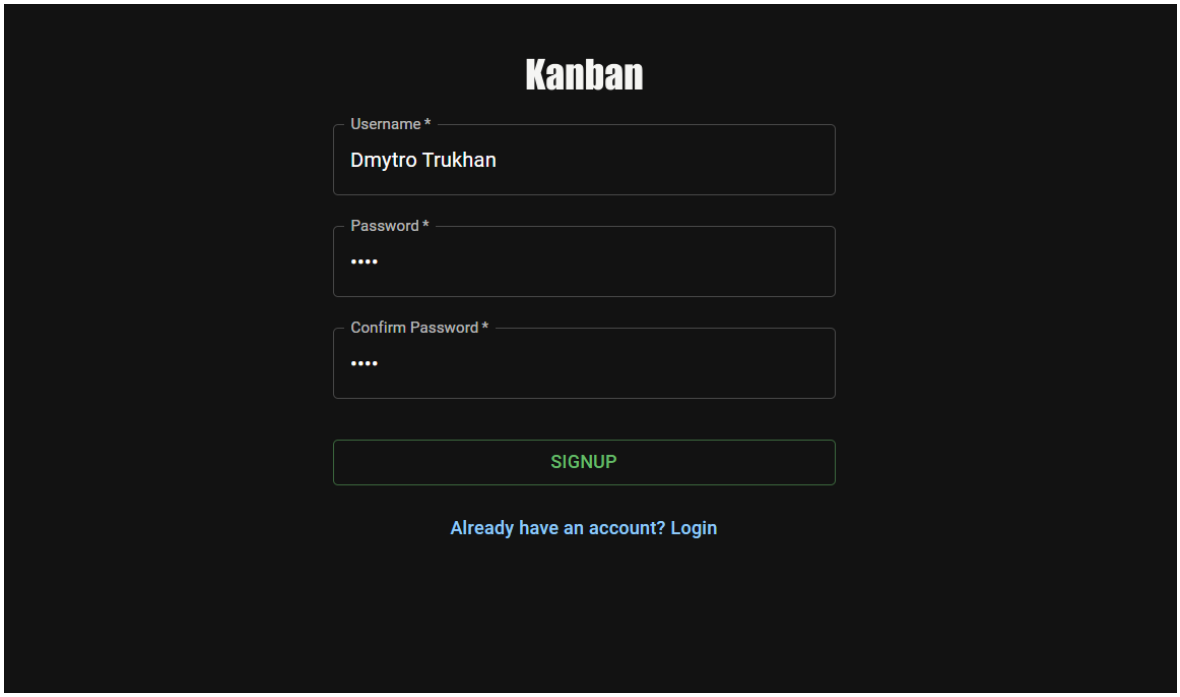

Рисунок 3.18 - Сторінка реєстрації

При введені некоректних даних користувач одразу буде повідомлений (Рисунок 3.19).

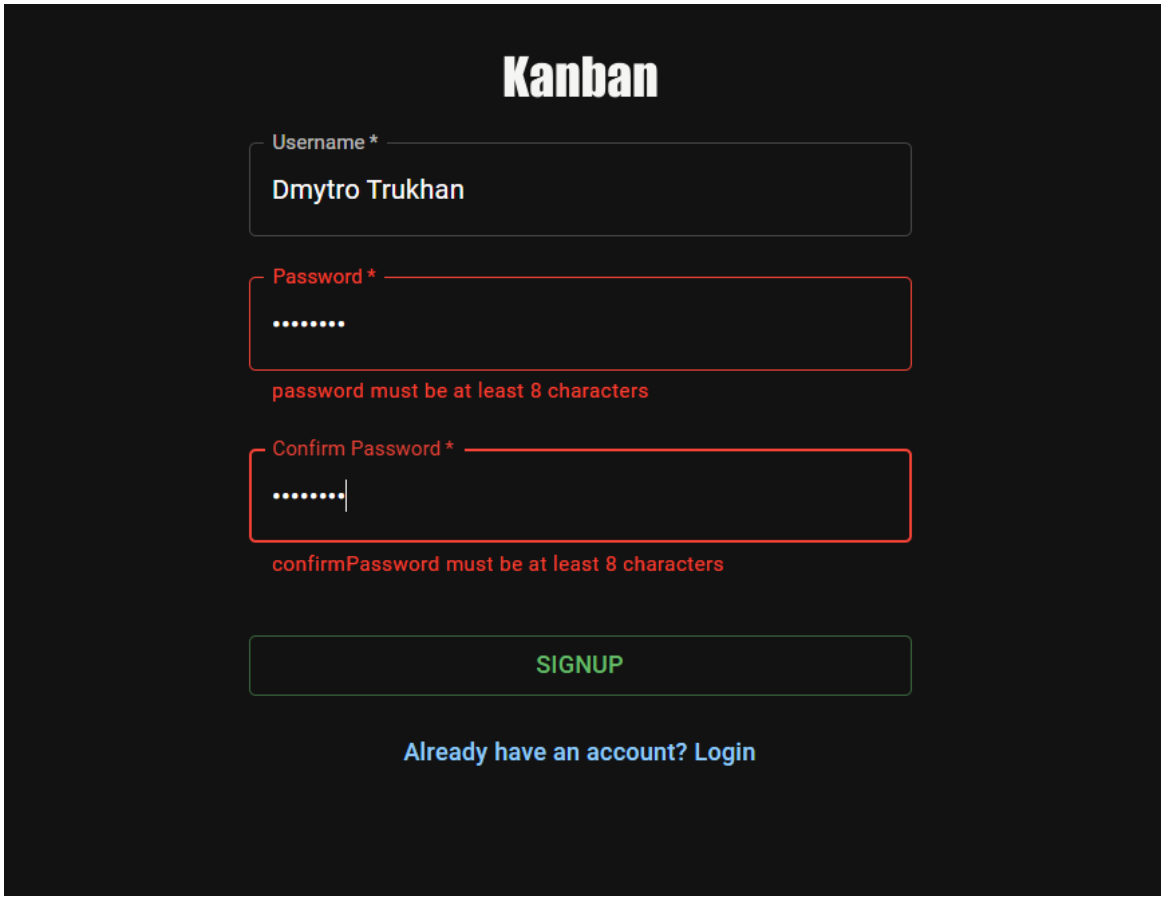

Рисунок 3.19 - Повідомлення про некоректні дані

Після успішної реєстрації та авторизації переходимо на головну сторінку вебзастосунку(Рисунок 3.20).

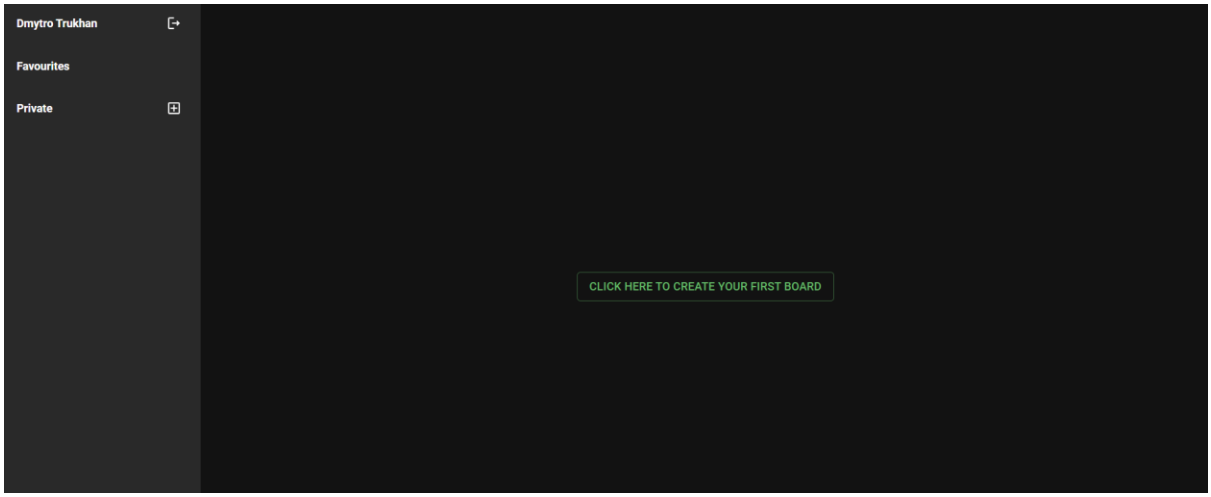

Рисунок 3.20 - Головна сторінка вебзастосунку

По центру екрану знаходиться кнопка для створення першої дошки. Загалом на екрані також доступні такі кнопки як “вихід” та додати дошку.

Після створення першої дошки з’являється більше функцій(Рисунок 3.21).

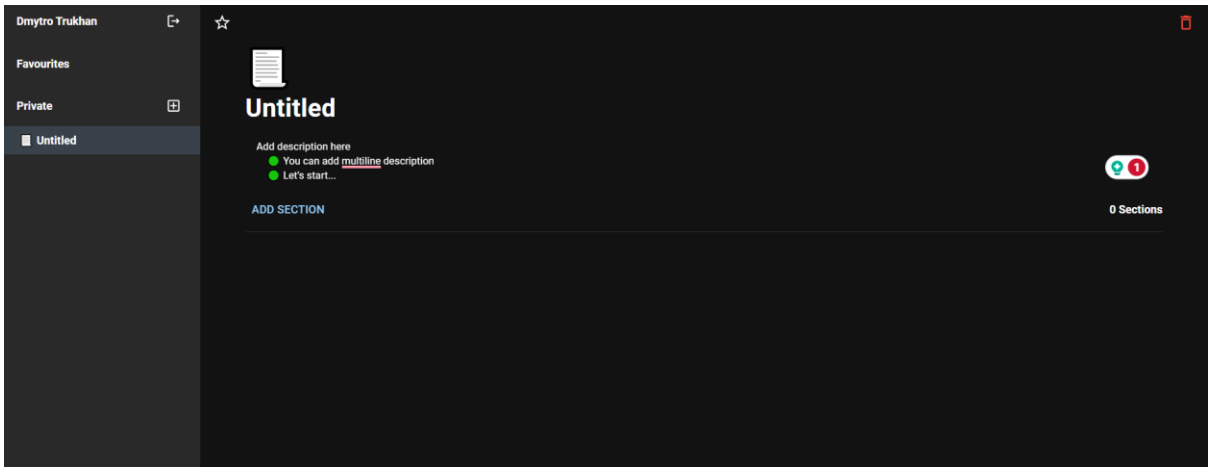

Рисунок 3.21 - Сторінка з більшим функціоналом

Створену дошку можна редагувати, змінюючи її логотип, вибравши емодзі(Рисунок 3.22).

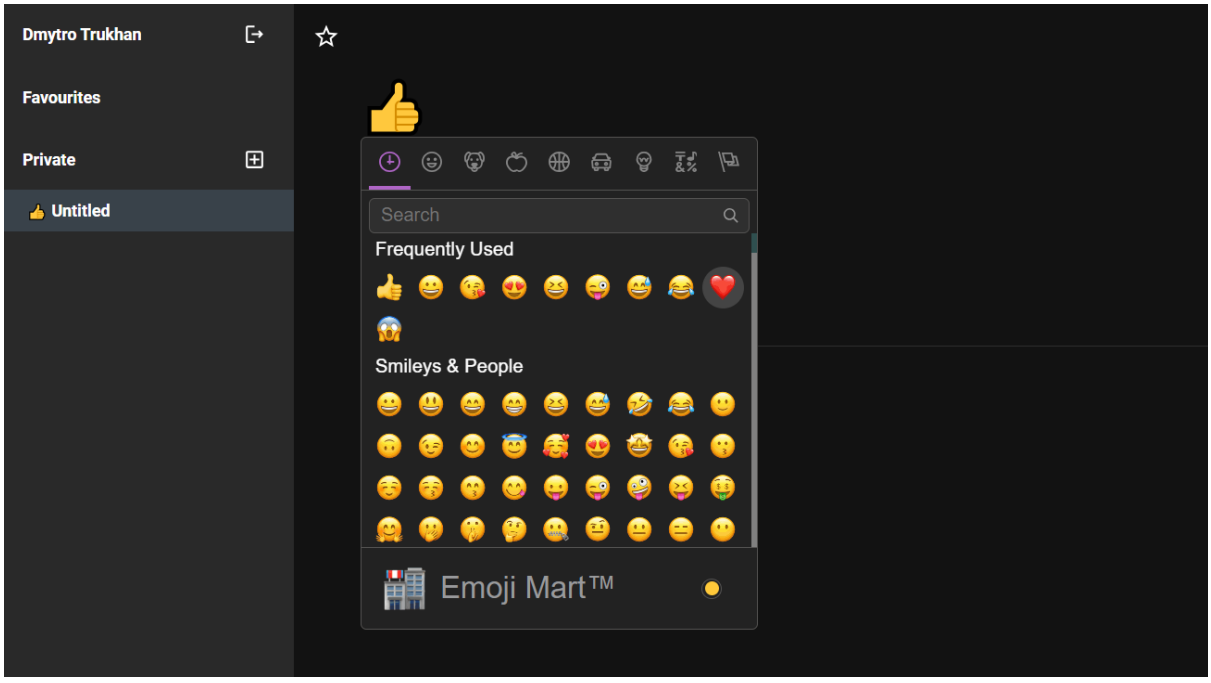

Рисунок 3.22 - Емодзі

Для кращого розуміння для чого створена дошка, реалізована функція додавання опису(Рисунок 3.23).

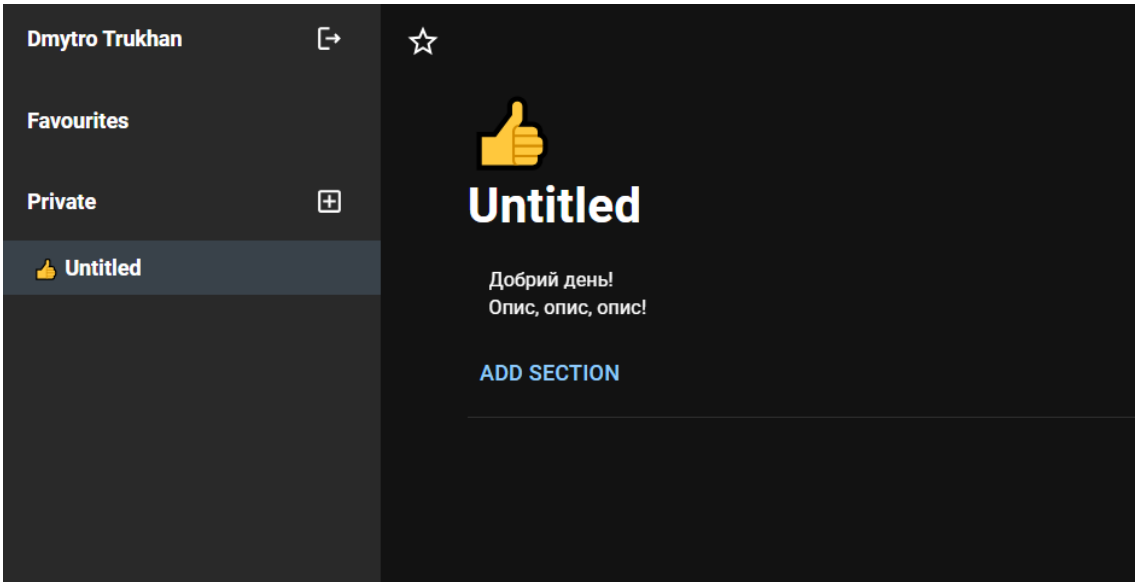

Рисунок 3.23 - Опис дошки

В дошку можна додавати секції та підсекції(Рисунок 3.24).

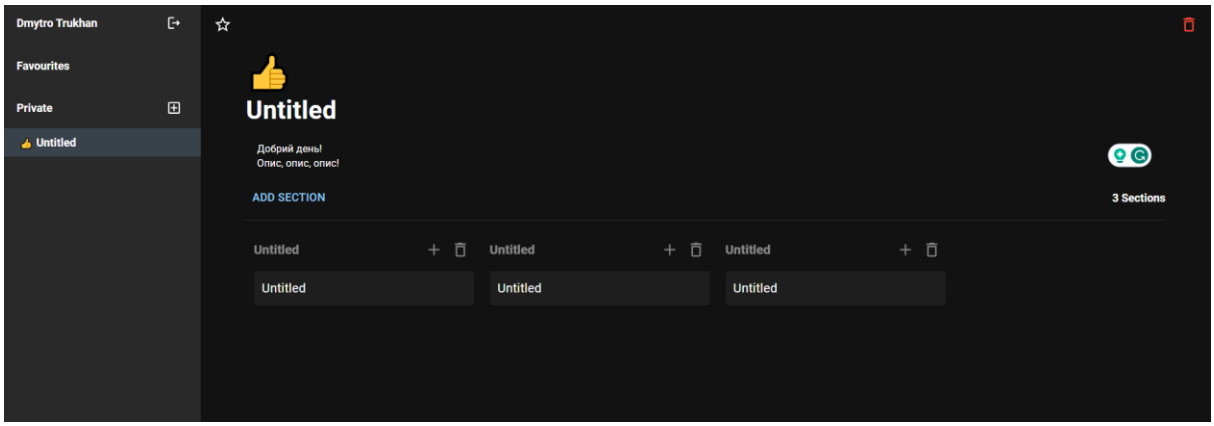

Рисунок 3.24 - Секції та підсекції

Для кожної підсекції(таску) є можливість детального редагування(Рисунок 3.25).

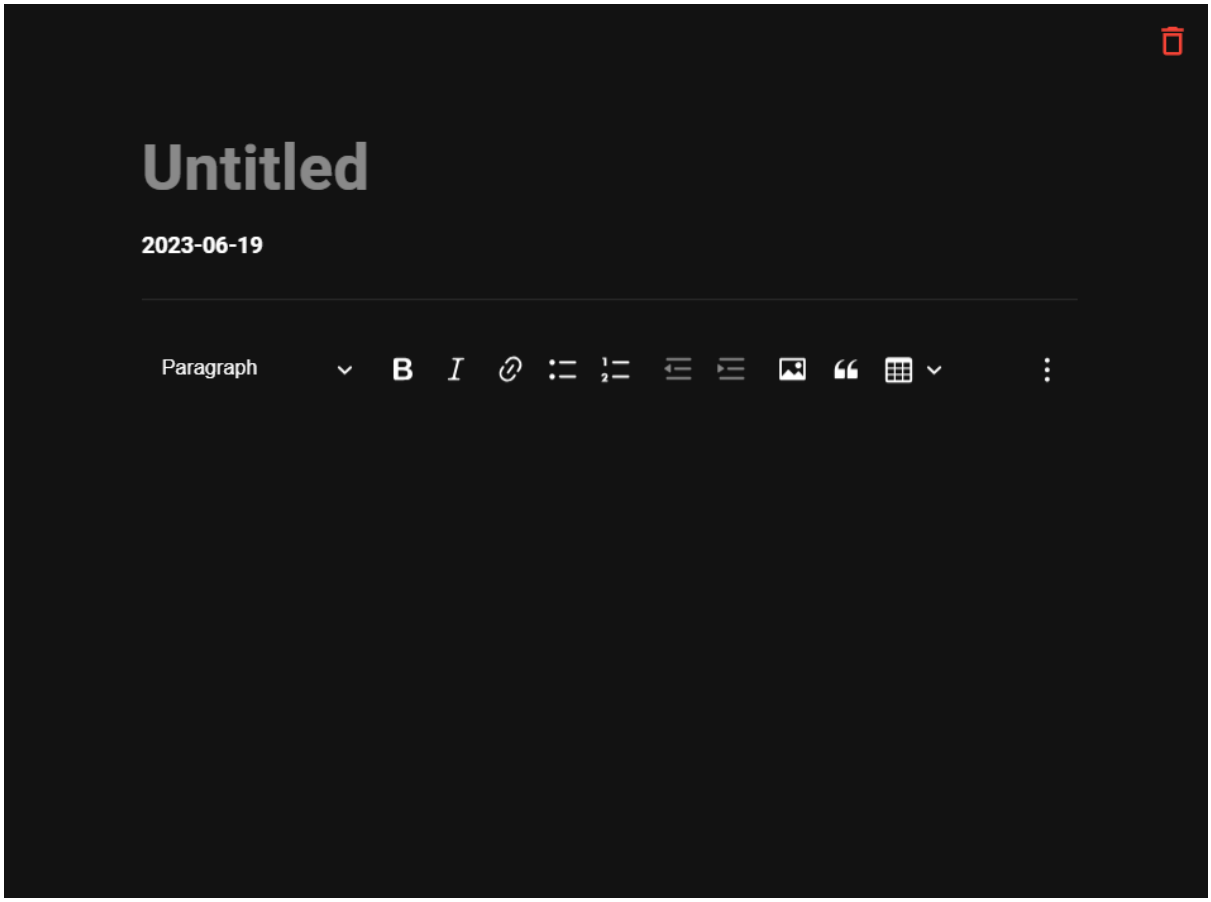

Рисунок 3.25 - Редагування підсекції(таску)

Кожну з дошок можна видалити нажавши на червону кнопку смітника справа зверху (Рисунок 3.26).

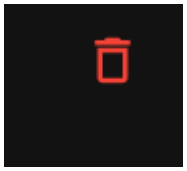

Рисунок 3.26 - Видалення дошки

Також кожну дошку можна додати в розділ “Улюблене”(Рисунок 3.27).

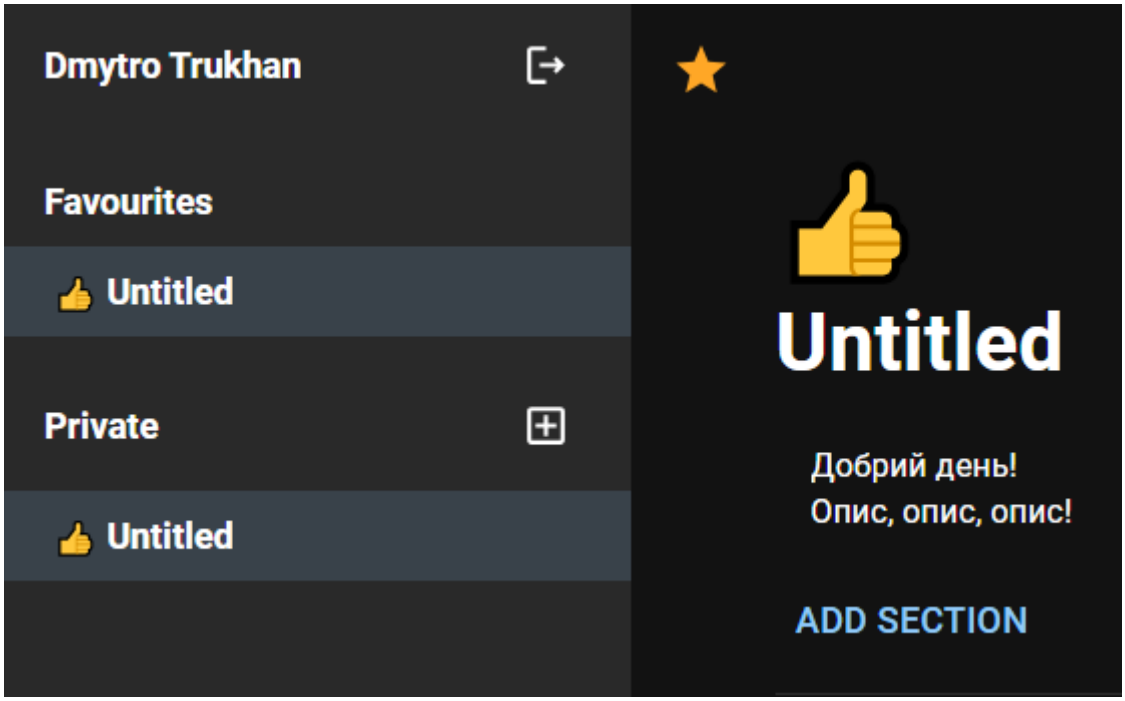

Рисунок 3.27 - Функція та розлід “Улюблене”

Загалом вебзастосунок покриває всі базові потреби користувачів для того, щоб мати змогу грамотно організувати свій власний час та робити це з комфортом.

3.3. Висновки до розділу

У розділі детально описано реалізацію проєкту «Kanban Board», включаючи вибір необхідних технологій та створення клієнтської та серверної частини додатку. Також проаналізовано результати роботи додатку «Kanban Board» та його взаємодію з користувачами.

Розділ починається з огляду доступних технологій та обґрунтування вибору JavaScript (JS), Node.js та MongoDB для створення нашого додатку «Kanban Board». JS є мовою програмування, яка підтримується браузерами і використовується для розробки функціональності клієнтської сторони додатку. Node.js є середовищем виконання JavaScript, яке дозволяє нам виконувати код JavaScript на серверній стороні. MongoDB обрано як базу

даних для збереження і керування даними про дошку, секції та задачі в нашому додатку.

Далі описана архітектура додатку «Kanban Board», включаючи розділення на клієнтську та серверну частини. Клієнтська частина використовує фреймворк React.js для побудови інтерфейсу користувача та взаємодії з сервером. Серверна частина реалізована з використанням Express.js, який дозволяє обробляти HTTP-запити, встановлювати маршрути та взаємодіяти з базою даних MongoDB.

Для організації логіки додатку використовуються такі компоненти, як контролери (controllers), обробники (handlers), моделі (models) та маршрути (routes). Контролери відповідають за обробку запитів, обробники здійснюють конкретні дії над даними, моделі представляють структуру даних і здійснюють взаємодію з базою даних, а маршрути визначають шляхи доступу до різних ресурсів додатку.

Таким чином, розділ розкриває всі аспекти реалізації проєкту «Kanban Board», включаючи вибір технологій, опис архітектури та розбивку на компоненти. Результати роботи додатку досліджуються та аналізуються з метою забезпечення ефективної та задовільної взаємодії з користувачами.

ВИСНОВКИ

Дипломний проект "Kanban Board" є комплексним інформаційним рішенням для організації та керування задачами і проектами. У ході розробки проекту були розглянуті різні аспекти, починаючи від аналізу предметної області і визначення вимог до системи, і закінчуючи вибором технологій та реалізацією відповідних компонентів.

Розділ аналізу предметної області дозволив з'ясувати основні потреби та вимоги користувачів, а також визначити ключові об'єкти та взаємозв'язки між ними. Це стало основою для подальшого проектування та розробки системи.

У формальній моделі були визначені основні елементи та параметри задачі, що дозволило уточнити суть проекту та його функціональні можливості. Інформаційна модель вказує на те, як дані будуть зберігатися та оброблятися в системі, зокрема за допомогою бази даних MongoDB.

Функціональна модель описує, які функції та операції будуть доступні користувачам у системі. Це включає створення та управління дошками, секціями та задачами, а також можливість зміни статусів та присвоєння відповідальних осіб.

Вибір технологічних рішень уповільнюється на JavaScript, Node.js та MongoDB. Ці технології є потужними та широко використовуються в розробці веб-додатків, забезпечуючи гнучкість, продуктивність та масштабованість.

Висновки підкреслюють важливість розробки системи керування задачами і проектами, так як вони допомагають організувати робочий процес, збільшити продуктивність та забезпечити ефективне спілкування між учасниками проекту.

У цілому, дипломний проект "Kanban Board" пропонує комплексний підхід до організації та керування задачами, розроблений на основі

детального аналізу потреб користувачів та використання сучасних технологій. Цей проект може бути корисним для різних команд і організацій, які прагнуть досягти більшої ефективності та результативності у своїх проектах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація Node.js. – [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/docs (дата звернення: 05.06.2023).
2. Документація Express.js. – [Електронний ресурс]. – Режим доступу: https://expressjs.com/ (дата звернення: 05.06.2023).
3. Документація MongoDB. – [Електронний ресурс]. – Режим доступу: https://docs.mongodb.com/ (дата звернення: 05.06.2023).
4. Документація Mongoose. – [Електронний ресурс]. – Режим доступу: https://mongoosejs.com/ (дата звернення: 05.06.2023).
5. Stack Overflow. – [Електронний ресурс]. – Режим доступу: https://stackoverflow.com/ (дата звернення: 05.06.2023).
6. Medium. – [Електронний ресурс]. – Режим доступу: https://medium.com/ (дата звернення: 05.06.2023).
7. Dev.to. – [Електронний ресурс]. – Режим доступу: https://dev.to/
8. GitHub. – [Електронний ресурс]. – Режим доступу: https://github.com/ (дата звернення: 05.06.2023).
9. IEEE Xplore. – [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/ (дата звернення: 05.06.2023).
10. Google Scholar. – [Електронний ресурс]. – Режим доступу: https://scholar.google.com/ (дата звернення: 05.06.2023).
11. Lutz M. Python Crash Course: A Hands-On, Project-Based Introduction to Programming. – No Starch Press, 2019.
12. Sweigart A. Automate the Boring Stuff with Python: Practical Programming for Total Beginners. – No Starch Press, 2019.
13. McKinney W. Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython. – O'Reilly Media, 2019.

14. Grus J. Data Science from Scratch: First Principles with Python. – O’Reilly Media, 2019.
15. Hughes B. Fluent Python: Clear, Concise, and Effective Programming. – O’Reilly Media, 2022.