

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)
на тему:

«Онлайн-платформа управління проєктами»

Виконав студент групи ПЗ-20-2
БОРДЮГ Роман Олегович

Керівник роботи:
ГОРДІЄНКО Ірина Валеріївна

Консультант роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
БАКАЛЮК Тетяна Анатоліївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«26» лютого 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БОРДЮГ Роман Олегович**

Керівник роботи: **ГОРДІЄНКО Ірина Валеріївна**

Тема роботи: **«Онлайн-платформа управління проєктами»**,

затверджена Наказом закладу вищої освіти від «23» лютого 2024 р., №74/с

Вихідні дані для роботи: **розробка онлайн- платформи управління проєктами зосереджена на забезпеченні ефективної організації роботи. Платформа має спрощувати процеси планування, відстеження та виконання проєктів.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Гордієнко І. В., Чижмотря О. Г.	27.02.2024р.	27.02.2024р.
2	Гордієнко І. В., Чижмотря О. Г.	27.02.2024р.	27.02.2024р.
3	Гордієнко І. В., Чижмотря О. Г.	27.02.2024р.	27.02.2024р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра включає онлайн-платформу управління проєктами та пояснювальну записку. Пояснювальна записка складається з тексту обсягом 49 сторінок друкованого тексту, 29 сторінок додатків, списку використаних джерел на 30 найменувань, що займає 3 сторінки. Робота містить 32 рисунки і 6 таблиць, загальний обсяг роботи становить 80 сторінок.

Мета роботи – створення онлайн-платформи управління проєктами з використанням сучасних технологій, що дозволяє скоротити час на планування та постановку задач.

У процесі роботи було проаналізовано аналогічні веб-додатки, їх функціонал та недоліки.

Результатом стала розробка веб-додатку, що покращує продуктивність, має зручний інтерфейс і дозволяє користувачам ефективно керувати проєктами та розподіляти задачі відповідно до потреб.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, ДОШКА, УПРАВЛІННЯ, БАЗА ДАНИХ, КЛІЄНТ, КОРИСТУВАЧ.

					ІПЗ.КР.Б-121-24-ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Онлайн-платформа управління проєктами	Літ.	Арк.	Аркушів
Розроб.		Р. О. Бордюг						
Керівник		І. В. Гордієнко					3	80
Рецензент		Т. А. Вакалюк				Житомирська політехніка, група ІПЗ-20-2		
Зав. каф.		Т. А. Вакалюк						

ABSTRACT

Bachelor's thesis includes a web application for project management and an explanatory note. The explanatory note consists of 49 pages of printed text, 29 pages of appendices, a list of 30 references occupying 3 pages. The work contains 32 figures and 6 tables, with a total volume of 80 pages.

The aim of the work is to create a platform for project management using modern technologies, which allows reducing the time for planning and task setting.

During the work, similar web applications, their functionality, and shortcomings were analyzed.

The result was the development of a web application that improves productivity, has a user-friendly interface, and allows users to effectively manage projects and distribute tasks according to their needs.

KEYWORDS: WEB-APPLICATION, BOARD, MANAGEMENT, DATABASE, CLIENT, USER.

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
Розділ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У РЕАЛІЗАЦІЇ ОНЛАЙН-ПЛАТФОРМИ УПРАВЛІННЯ ПРОЄКТАМИ	9
1.1 Постановка задачі	9
1.2 Аналіз аналогів програмного продукту	10
1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	15
Висновки до першого розділу	18
Розділ 2. ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ ПЛАТФОРМИ	20
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи	20
2.2 Розробка бази даних системи	25
2.3 Проєктування та реалізація алгоритмів роботи системи	27
2.4 Реалізація онлайн-платформи управління проєктами.....	29
Висновки до другого розділу.....	33
Розділ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОНЛАЙН- ПЛАТФОРМОЮ УПРАВЛІННЯ ПРОЄКТАМИ	34
3.1 Порядок встановлення та налаштування параметрів платформи.....	34
3.2 Структура інтерфейсу онлайн-платформи управління проєктами.....	34
3.3 Тестування платформи	44
Висновки до третього розділу	47
ВИСНОВКИ	48
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	49
ДОДАТКИ.....	52

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

JWT – JSON Web Token

PK – Primary Key

FK – Foreign Key

JSON – JavaScript Object Notation

CRUD – основні операції, що використовуються для управління даними
(Create , Read , Update , Delete)

SQL – Structure Query Language

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми відображає сучасна реальність: хвиля швидких технологічних як ніколи раніше вказала на проблеми планування та тайм-менеджменту. Ефективне управління проєктами є ключовим аспектом успішності бізнесу. Попит на зручні інструменти для планування, організації та контролю проєктів постійно зростає, а конкуренція на ринку програмного забезпечення стає все сильнішою, ставлячи перед розробниками завдання створити продукт, що відповідатиме сучасним вимогам. Тому розробка онлайн-платформи управління проєктами є дуже актуальною задачею, оскільки вона відповідає потребам сучасних бізнесів та відкриває широкі можливості для підвищення продуктивності та ефективності роботи.

Метою цього дослідження є розробка функціональної та ефективної онлайн-платформи управління проєктами з використанням сучасних технологій. Ця платформа повинна забезпечувати зручний інтерфейс [25] для планування, відстеження та координації завдань, а також сприяти підвищенню продуктивності. Також важливою метою є вивчення можливостей застосування передових технологій розробки у сфері управління проєктами та їх впливу на результативність роботи.

Для розробки онлайн-платформи управління проєктами потрібно:

1. Провести аналіз конкурентів, оцінити їх функціонал та виявити слабкі сторони. Це допоможе зрозуміти, які аспекти варто вдосконалити та які функції будуть найбільш корисними для користувачів.
2. Визначити, які технології будемо використовувати для розробки платформи. Необхідно обрати оптимальні інструменти для фронтенду та бекенду, а також бази даних, щоб забезпечити стабільну та ефективну роботу додатку.
3. Розробити онлайн-платформу з урахуванням знайденої інформації. Це включає створення зручного інтерфейсу, реалізацію основних функцій

для управління проєктами, забезпечення безпеки даних та оптимізацію роботи системи.

Об’єктом дослідження є процес управління проєктами в контексті сучасних веб-технологій. Вивчаються всі аспекти цього процесу, включаючи методи планування, організації, виконання та контролю завдань і ресурсів у межах проєктів. Також досліджується вплив веб-технологій, таких як React.js та Next.js на управління проєктами та їх продуктивність.

Предметом дослідження є розробка та впровадження онлайн-платформи управління проєктами, що базуватиметься на сучасних технологіях та відповідатиме поточним вимогам бізнесу. Досліджується функціональність даного додатку, його можливості та переваги порівняно з іншими інструментами управління проєктами.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У РЕАЛІЗАЦІЇ ОНЛАЙН-ПЛАТФОРМИ УПРАВЛІННЯ ПРОЄКТАМИ

1.1 Постановка задачі

Управління проєктами – це комплексний процес, що включає планування, розподіл завдань, відстеження прогресу та контроль за виконанням. Ігнорування принципів управління проєктами може призвести до затримок, невиконання поставлених цілей, а в деяких випадках – до провалу [20].

Звідси виникає потреба у зручних та ефективних інструментах для управління. Онлайн-платформа управління проєктами може стати таким інструментом, надаючи користувачам все необхідне для успішної реалізації своїх задумів.

Цільове призначення веб-додатку:

- Спростити процес управління проєктами.
- Надати програмні засоби для організації робочого процесу та сприяння ефективному виконанню поставлених завдань.
- Забезпечити планування, розподіл та моніторинг виконання завдань.
- Надати інструменти для відстеження та аналізу прогресу проєктів.

Основні етапи розробки веб-додатку:

Аналіз вимог:

- Визначити та проаналізувати існуюче програмне забезпечення для управління ІТ-проєктами.
- Зібрати та проаналізувати потреби користувачів систем управління ІТ-проєктами.
- Визначити функціональні та нефункціональні вимоги до системи.
- Розробити сценарії використання та визначити основні потреби користувачів.

Проектування інтерфейсу:

- Визначити структуру та навігацію системи

- Розробити макети та дизайн користувацького інтерфейсу.

Розробка основного функціоналу:

- Створення реєстрації та авторизації
- Створення бази даних та реалізація CRUD операцій
- Створення взаємодій з додатком

В результаті виконання вищезазначених завдань буде створена функціональна та ефективна система, яка надає користувачам все необхідне для роботи над проєктами: планування, виконання та відстеження.

1.2 Аналіз аналогів програмного продукту

Перед розробкою власного веб-застосунку управління проєктами варто провести ретельний аналіз конкурентів. Це дозволить ознайомитися з існуючими рішеннями на ринку та їхніми можливостями, визначити сильні та слабкі сторони конкурентів та розробити ефективну стратегію розробки та просування додатку.

Перший додаток який варто розглянути – це Trello (рис.1.1) [1]. Trello – це популярний інструмент для керування проєктами, який використовує дошки, списки і картки для організації завдань і командної роботи. Додаток відзначається своєю простотою та гнучкістю, що дозволяє адаптувати його до різних робочих процесів. Він має інтуїтивно зрозумілий інтерфейс, що сприяє підвищенню продуктивності та часової ефективності роботи. Підтримує додавання тегів, дедлайнів, вкладень та коментарів, що дозволяє ефективно відстежувати прогрес і співпрацювати з іншими користувачами в режимі реального часу.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

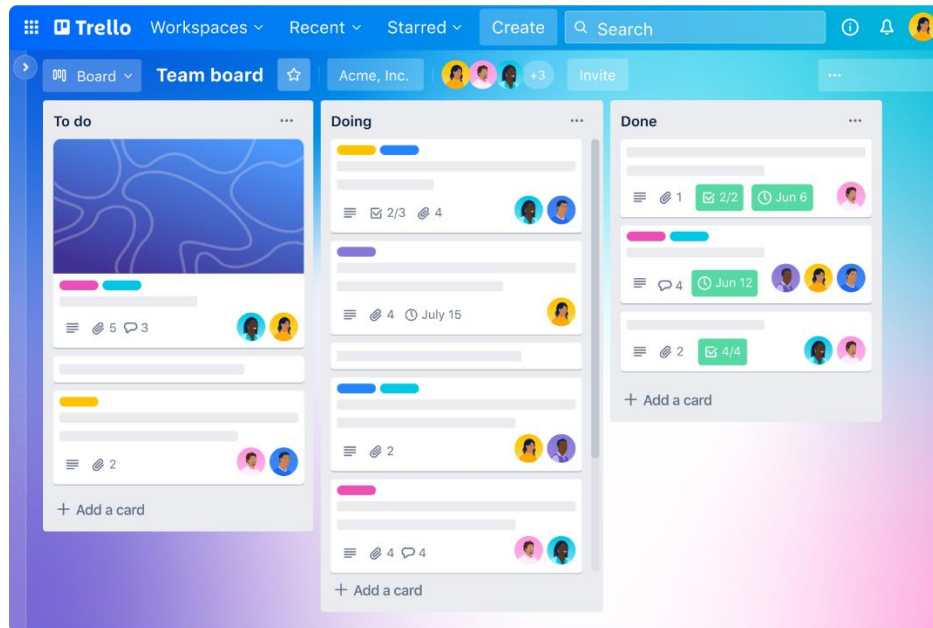


Рисунок 1.1 – Інтерфейс додаку Trello

Переваги Trello:

- Поріг входження: Інтуїтивно зрозумілий інтерфейс дозволяє швидко освоїтись і почати роботу без тривалого навчання.
- Адаптивність: Можливість адаптації під різні робочі процеси та проєкти завдяки використанню дошок, списків і карток.
- Співпраця в реальному часі: Користувачі можуть додавати коментарі, дедлайни, вкладення та теги, що полегшує командну роботу.

Недоліки Trello:

- Складність масштабування: При збільшенні кількості проєктів і задач інтерфейс може стати перевантаженим і менш зручним.
- Відсутність розширених функцій для управління ресурсами: Немає вбудованих інструментів для управління ресурсами та розподілу навантаження між членами команди.

Trello є зручним та гнучким інструментом для управління проєктами, що пропонує простий інтерфейс і ефективну організацію завдань. Він інтегрується з іншими сервісами, такі як Slack і Google Drive [26]. Основні обмеження включають відсутність аналітичних інструментів. Незважаючи на це, Trello залишається популярним завдяки своїй простоті та ефективності.

Наступний додаток, який було розглянуто – Monday.com (рис1.2) [3]. Це інструмент для управління проєктами, який надає широкі можливості налаштування робочих просторів для командної роботи. Завдяки своїй гнучкості і можливостям інтеграції з іншими сервісами, monday.com дозволяє ефективно організовувати робочі процеси та відстежувати прогрес виконання завдань.

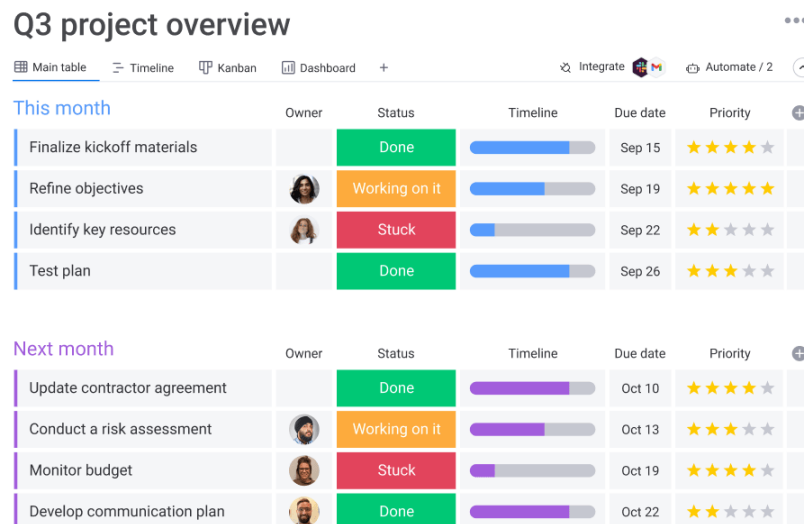


Рисунок 1.2 – Інтерфейс додатку Monday.com

Переваги:

- Широкі можливості налаштування: Платформа дозволяє створювати індивідуальні робочі простори, що дає змогу адаптувати інструмент під конкретні потреби команди або проєкту.
- Доступний збір звітності та аналітики: Вбудовані інструменти для створення звітів та аналізу даних допомагають командам слідкувати за прогресом і приймати обґрунтовані рішення.
- Координація роботи команд: Функції для спільної роботи, такі як коментування, призначення завдань і обмін файлами, сприяють ефективній комунікації та координації всередині команди.

Недоліки:

- Перенавантажений інтерфейс: Велику кількість функцій і можливостей може бути складно освоїти новачкам, що потребує часу на навчання.
- Перевантаження даними: Платформа може працювати повільніше при обробці великих обсягів даних або за великої кількості користувачів в мережі.
- Обмежені можливості мобільного додатку: Мобільний додаток не має всіх функцій веб-версії, що може обмежувати можливості користувачів, які працюють зі смартфонів.

Останній додаток який було розглянуто – це Jira (рис.1.3) [2]. Jira – потужний інструмент управління проектами, розроблений компанією Atlassian. Він надає широкі можливості для планування, відстеження та управління різними аспектами проекту, включаючи завдання, спринти, релізи та звіти. Jira є популярним вибором серед розробників програмного забезпечення та команд, що працюють в інших галузях, завдяки своїм потужним можливостям та гнучкості.

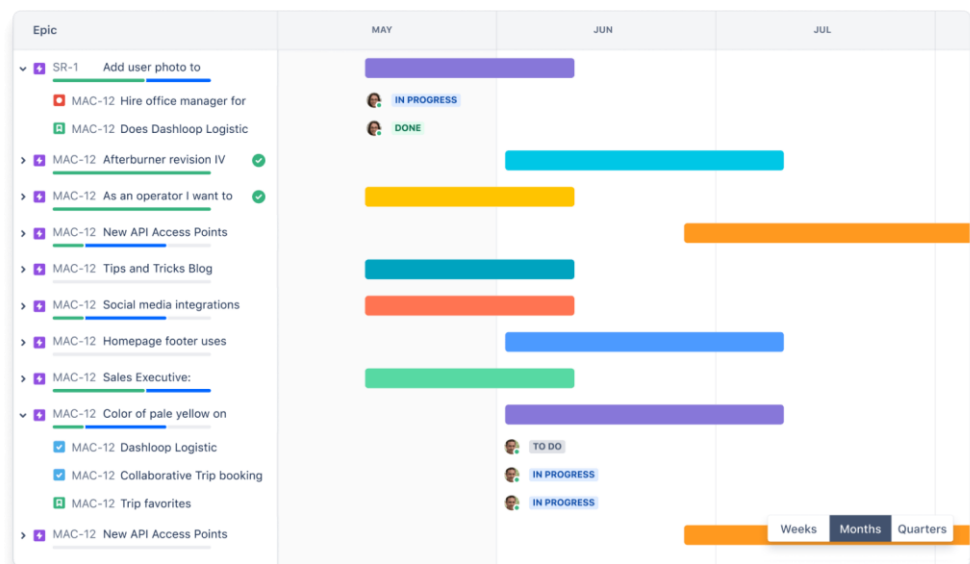


Рисунок 1.3 – Інтерфейс додатку Jira

Переваги:

- Широкий функціонал: Jira надає широкі можливості для управління проєктами, включаючи планування, відстеження прогресу, призначення завдань, створення звітів та багато іншого.
- Гнучкість: Платформа може бути налаштована під конкретні потреби команди або проєкту, дозволяючи створювати власні робочі процеси та використовувати різні методології, такі як Scrum, Kanban тощо.
- Інтеграція: Jira легко інтегрується з іншими популярними інструментами розробки програмного забезпечення, такими як GitHub, Bitbucket, Confluence тощо, що спрощує робочий процес і підвищує продуктивність розробників.

Недоліки:

- Складність: Для новачків Jira може виявитися складною у використанні через велику кількість функцій та налаштувань.
- Неоптимізований інтерфейс: Інтерфейс Jira не завжди є інтуїтивно зрозумілим, що може призвести до складнощів у використанні.
- Потреба у часі на навчання: Використання всіх можливостей Jira може вимагати часу на навчання та адаптацію.

Усі вищеописані проєкти націлені покращення планування та оптимізацію робочого процесу. Нижче наведено порівняльну таблицю (табл 1.1) цих трьох додатків

Таблиця 1.1

Порівняння інструментів додатків для управління проєктами

	Trello	Monday	Jira
Керування завданнями	+	+	+
Гнучкість	-	+	+
Інтеграція з іншими сервісами	-	+	+
Звітність та аналітика	-	+	+
Підтримка різних методологій	-	+	+
Інтерфейс	+	-	-

Отже Trello – простий та інтуїтивно зрозумілий інструмент для управління проєктами, ідеальний для невеликих команд або особистого використання. Monday.com надає гнучкість налаштування та багато можливостей для інтеграції, що робить його популярним серед найрізноманітніших команд. Jira вирізняється своєю потужністю та глибиною функціоналу, що ідеально підходить для великих команд та складних проєктів, але може вимагати більше часу на навчання та налаштування.

1.3 Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибір інструментальних засобів базується на їхній ефективності та здатності пришвидшувати розробку високоякісних веб-додатків. Використання React.js [5], Next.js [4] і Tailwind CSS [7] дозволяє швидко розробляти компоненти додатку та графічний інтерфейс, забезпечуючи високу продуктивність і зручність у роботі. MySQL [8] є реляційною базою даних з високим рівнем надійності, що є критичним для веб-додатків, оскільки вони часто потребують масштабування та потребують використання великих обсягів даних. Вимоги до апаратного забезпечення полягають у відповідному рівні продуктивності для роботи з великим обсягом даних та швидкого реагування на запити від користувачів.

React.js – це бібліотека JavaScript, яка використовується для створення користувацьких інтерфейсів та виконання певних операцій потрібних для роботи клієнтської частини додатку. Вона дозволяє розробникам ефективно створювати динамічні веб-додатки, які швидко реагують на зміни даних. За допомогою React.js [11], розробники можуть створювати компоненти, які можна використовувати багаторазово, що робить процес розробки більш ефективним та швидким.

Переваги React.js:

- Ефективний віртуальний DOM: React.js використовує віртуальний DOM для ефективного відображення змін у структурі дерева

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

компонентів. Він автоматично визначає, які елементи необхідно оновити в реальному DOM, що дозволяє зменшити кількість операцій зміни DOM та підвищити продуктивність додатку.

- Компонентна архітектура: React.js [10] базується на концепції компонентів, яка дозволяє розділити інтерфейс на невеликі, практично незалежні елементи. Це спрощує структуру додатку, дозволяє повторно використовувати код та робить розробку та підтримку додатку простішою.
- Масштабованість: додатки, побудовані з використанням React.js набагато простіше масштабувати завдяки широкій підтримці великої кількості сторонніх бібліотек та плагінів. Розробники можуть легко інтегрувати різноманітні функції та фреймворки для розширення функціональності своїх додатків.
- Широкий вибір інструментів: У React.js існує велика кількість інструментів для розробки, тестування та підтримки веб-додатків [29]. Від інтегрованих середовищ розробки до бібліотек для тестування компонентів – у React є все необхідне для ефективної розробки додатків.

Next.js – це фреймворк, в який інтегровано React.js для створення веб-додатків, що забезпечує високу продуктивність та покращену SEO-оптимізацію. Однією з ключових особливостей Next.js [24] є підтримка серверного рендерингу та статичного генерування, що дозволяє створювати динамічні та ефективні додатки, які швидко завантажуються. Фреймворк забезпечує зручність роботи з маршрутизацією, управлінням станом, автоматичною оптимізацією для зображень та стилів, і значно полегшує розробку та підтримку проєктів.

Переваги Next.js:

- Серверний та статичний рендеринг: Next.js підтримує як серверний, так і статичний рендеринг, що дозволяє створювати швидкі веб-додатки з високою продуктивністю.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

- Зручна маршрутизація: Інтегрована система маршрутизації дозволяє легко керувати шляхами та переходами між сторінками в додатку.
- Автоматична оптимізація: Next.js автоматично оптимізує додатки, зокрема керує попереднім завантаженням сторінок, кешуванням та мініфікацією ресурсів для забезпечення найкращої продуктивності.
- Розширені можливості CSS [23]: Підтримка різних методів стилізації, таких як CSS Modules, Styled Components або Tailwind CSS, дозволяє обрати підхід до розробки, який найбільше підійде для конкретного проєкту.
- Швидка розробка: Зручний інтерфейс, розширена функціональність та гарне документування дозволяють розробникам швидко створювати високоякісні веб-додатки.

Tailwind CSS [16] – це масштабована CSS бібліотека, яка дозволяє швидко створювати користувацький інтерфейс, використовуючи набір готових класів.

Основними перевагами Tailwind є:

- Атомарний підхід: Tailwind розбиває стилі на дрібні, незалежні класи, кожен з яких відповідає одній CSS-властивості. Це робить код зрозумілим, гнучким та простим у модифікації.
- Гнучкість та кастомізація: Tailwind пропонує широкий спектр готових класів, які можна налаштувати або додати власні. Це дає розробникам повний контроль над стилями та дозволяє створювати унікальний дизайн.
- Швидка розробка: Завдяки готовим класам та інтуїтивному синтаксису Tailwind, розробка інтерфейсів стає значно швидшою, порівняно з традиційним CSS.
- Масштабованість: Tailwind підходить для проєктів будь-якого розміру, від невеликих сайтів-лендінгів до складних веб-додатків. Він дозволяє створювати стильний та сучасний дизайн без зайвих зусиль.

MySQL – це потужна та надійна реляційна система керування базами даних, яка широко використовується для зберігання та обробки даних у веб-додатках та інших програмних продуктах. Вона відома своєю високою продуктивністю, надійністю та широким спектром функцій, які вона пропонує.

Основні переваги MySQL [30]:

- Надійність та стабільність: MySQL відома своєю надійністю та стабільністю. Вона широко використовується в різних проєктах, від простих веб-сайтів до великих корпоративних систем.
- Висока продуктивність: MySQL забезпечує швидкий доступ до даних та оптимальну роботу навіть при великому обсязі даних. Вона має оптимізовані алгоритми обробки запитів та управління даними.
- Широкий функціонал: MySQL має великий набір функцій і можливостей, включаючи підтримку різних типів даних, транзакцій, зовнішніх ключів, процедур та функцій.
- Відкритість та спільнота: MySQL є відкритим продуктом з великою активною спільнотою розробників. Це означає, що ви можете знайти безліч ресурсів, документації та підтримки в Інтернеті

Висновки до першого розділу

Проведено аналіз трьох інструментів управління проєктами: Trello, Monday.com та Jira. Кожен з них має свої переваги та недоліки, і вибір між ними залежить від конкретних потреб користувача та характеру проєкту. Trello відзначається простотою використання та гнучкістю, Monday.com надає широкий спектр функціональних можливостей для управління проєктами будь-якої складності, а Jira є потужним інструментом для команд, що працюють у великих організаціях та дотримуються методології Agile.

Крім того, було розглянуто інструментальні засоби та вимоги до апаратного забезпечення для розробки веб-додатків. React.js, Next.js і Tailwind CSS є ключовими технологіями, які дозволяють розробляти сучасні та ефективні

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

веб-додатки, а MySQL забезпечує потужну базу даних [17] для зберігання та обробки інформації про проекти.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ ПЛАТФОРМИ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Перш ніж розпочати розробку додатку, важливо чітко визначити його майбутній функціонал, враховуючи потреби користувачів та цілі бізнесу. Для цього потрібно ретельно проробити прецеденти використання, або сценарії, що описують взаємодію користувачів з додатком, та визначити вимоги до функціоналу на основі цих сценаріїв.

Прецеденти використання допомагають ідентифікувати потреби користувачів та основні функції, які повинен виконувати додаток. Після створення прецедентів можна переходити до розробки бази даних та її налаштування, оскільки це дасть можливість краще зрозуміти, які дані необхідно зберігати для підтримки функціоналу додатку.

Важливо, щоб структура бази даних відповідала потребам функціоналу та забезпечувала ефективне зберігання та доступ до інформації. Такий підхід дозволяє зменшити ймовірність неузгодженостей між функціоналом додатку та його базою даних, що в свою чергу сприяє зручності розвитку та підтримки системи в майбутньому.

Вимоги користувачів:

1. Можливість створювати власні проєкти (дошки)
2. Створення завдань та їх зручне відслідковування
3. Зручний інтерфейс
4. Керування профілем
5. Редагування та видалення завдань

Функціональні вимоги:

1. Реєстрація та авторизація
2. Створення нових дошок та їх редагування
3. Створення колонок та їх редагування
4. Створення завдань та їх редагування

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

5. Можливість призначити дедлайн
6. Можливість обрати пріоритет
7. Редагування профілю
8. Можливість відмічати завдання як виконане

Нефункціональні вимоги:

1. Безпека:
 - Шифрування даних
 - Захист від атак
2. Надійність:
 - Стійкість до витоку пам'яті.
 - Резервне копіювання
3. Продуктивність:
 - Час завантаження сторінок
 - Пропускна спроможність
 - Відгук від системи до користувача

Системні вимоги:

1. Вимоги до середовища виконання:
 - Оперативна пам'ять: мінімум 2 ГБ.
 - Процесор: тактова частота не менше 1.8 ГГц.
 - Підключення до мережі Інтернет.

Прецедент «Реєстрація користувача»

Опис: Цей прецедент описує процес реєстрації нового користувача в системі. Під час реєстрації користувач надає системі необхідну інформацію для створення облікового запису.

Основні кроки:

1. Вхід на сайт
2. Заповнення поля реєстрації
3. Підтвердження реєстрації кнопкою «Зареєструватися»
4. Валідація введених даних

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

5. Створення нового користувача
6. Користувач наразі має змогу увійти в систему

Виключні ситуації:

1. Якщо дана пошта або нікнейм зайняті іншим користувачем виводиться помилка
2. Якщо введені дані не відповідають валідації реєстрація неможлива

Прецедент «Створення дошки»

Опис: Цей прецедент описує процес створення нової дошки, введення усіх необхідних даних.

Основні кроки:

1. При першому входженні на сайт користувачу пропонується створити першу дошку
2. Відкриття форми для заповнення даних
3. Після заповнення даних очікується поки користувач натисне на кнопку «Створити дошку»
4. Запис нових даних у базу даних
5. Вивід створеного проєкту на екран

Виключні ситуації:

1. Користувач не ввів жодних даних та система не створила нову дошку, про що попередила користувача

Прецедент «Створення завдання»

Опис: Цей прецедент описує процес створення нового завдання, вибору дати дедлайну, пріоритетності, назначення підзавдань.

Основні кроки:

1. Користувач натискає на кнопку «Додати нове завдання»
2. Вивід форми створення завдання на екран
3. Заповнення даних
4. Додавання нових підзавдань
5. Вибір дати дедлайну та пріоритетності
6. Натискання на кнопку «Створити завдання»

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Прецедент «Редагування профілю»

Опис: Цей прецедент описує процес, за допомогою якого користувач може змінити інформацію свого профілю у системі. Користувач може оновити особисті дані, включаючи контактну інформацію, пароль.

Основні кроки:

1. Користувач переходить до вікна редагування профілю
2. Користувач обирає що саме він хоче замінити, контактну інформацію, пошту, пароль
3. Заповнення даних
4. Натискання кнопки «Зберегти зміни»
5. Перевірка коректності введених даних
6. Оновлення даних користувача

Виключні ситуації:

1. Якщо користувач ввів некоректні дані система попереджає користувача про некоректність введених даних
2. Якщо вказана пошта або нікнейм існує користувач буде попереджений про це

Тепер можна перейти до розробки базового макету бази даних (рис. 2.1), щоб визначити її структуру та основні компоненти. Зважаючи на вимоги до системи та очікувані функціональності, можна визначити необхідні таблиці, зв'язки між ними, атрибути для кожної таблиці та їх типи даних. Це дозволить зрозуміти, як дані будуть організовані та як система буде взаємодіяти з користувачами та іншими компонентами.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

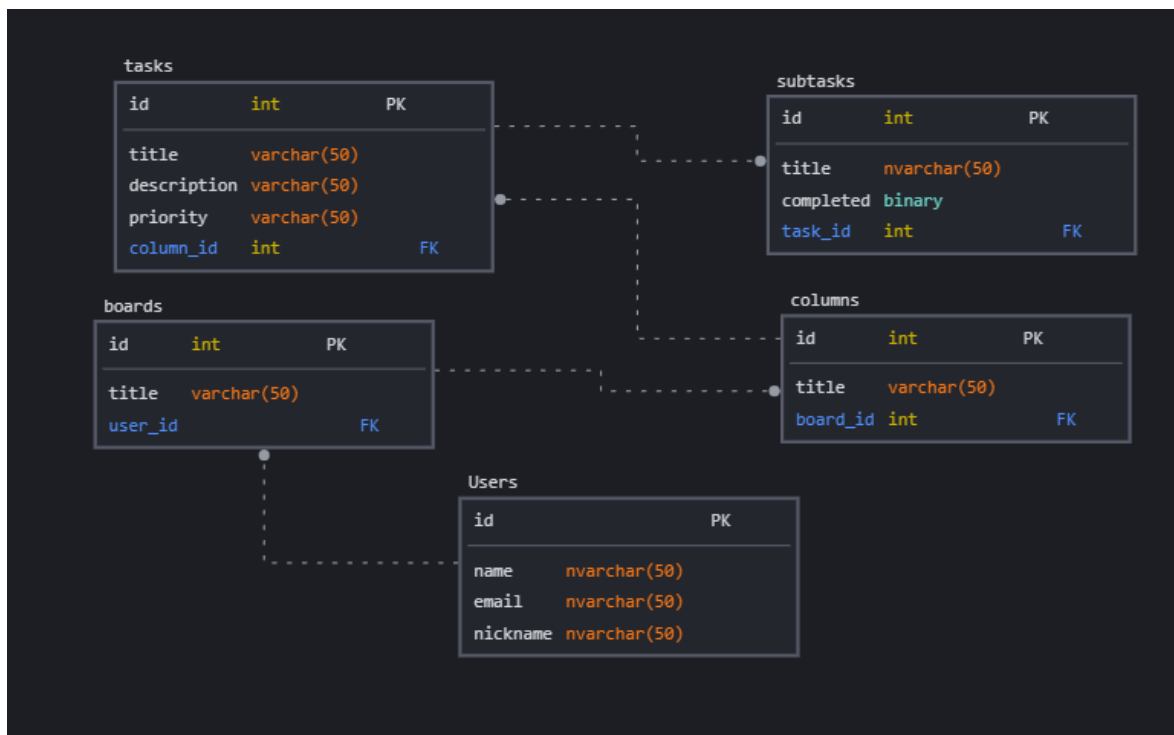


Рисунок 2.1 – Початкова діаграма

Базова діаграма бази даних складається з:

1. Users – ця таблиця міститиме інформацію про користувачів системи, таку як ім'я, електронну адресу, пароль тощо.
2. Boards – таблиця, що представляє робочі дошки або проєкти. Вона містить інформацію про назву дошки, опис, дату створення тощо.
3. Columns – таблиця, що описує різні колонки або розділи в межах дошки. Вона може містити дані про назву колонки, її порядковий номер, призначені завдання тощо.
4. Tasks – ця таблиця представляє конкретні завдання або задачі, розміщені в колонках дошки. Містить назву завдання, опис, дедлайн тощо.
5. Subtasks – таблиця, що відповідає за підзадачі або додаткові завдання, які пов'язані з основними завданнями. Може містити дані про назву підзадачі, статус виконання, призначений дедлайн тощо.

2.2 Розробка бази даних системи

Для збереження файлів та інформації у додатку використовується MySQL. Це система управління базами даних, яка базується на реляційній моделі [14].

База даних системи складається з п'яти таблиць:

1. Users – містить інформацію про користувачів системи
2. Columns – описує різні колонки або розділи в межах дошки
3. Tasks – представляє конкретні завдання або задачі, розміщені в колонках дошки
4. Subtasks – відповідає за підзадачі або додаткові завдання, які пов'язані з основними задачами
5. Boards – представляє робочі дошки або проєкти

Опис кожної з таблиць бази даних наведений у таблицях 2.1 – 2.5.

Одна з основних таблиць це «Користувач». На нього призначено унікальний згенерований ідентифікатор UUID, шлях де зберігається фото профілю та інша інформація.

Таблиця 2.1

Опис таблиці Users

Поле	Тип даних	ПК	ЗК	Опис
id	varchar	+	-	Ідентифікатор
Name	varchar	-	-	Ім'я користувача
Surname	varchar	-	-	Прізвище користувача
Nickname	varchar	-	-	Нікнейм
Email	varchar	-	-	Пошта
Photo_path	Varchar	-	-	Шлях до фото
password	varchar	-	-	Пароль

Таблиця «Дошка». У даній таблиці знаходиться зовнішній ключ користувача. Це зроблено для того, щоб у користувача відображались тільки ті дошки, власником яких він є.

Таблиця 2.2

Опис таблиці Boards

Поле	Тип даних	ПК	ЗК	Опис
id	int	+	-	Ідентифікатор
Title	varchar	-	-	Назва дошки
Id_user	varchar	-	+	Зовнішній ключ користувача

Таблиця «Колонки» також пов'язана з таблицею «Дошки» для перевірки чи належить певна колонка до дошки.

Таблиця 2.3

Опис таблиці Columns

Поле	Тип даних	ПК	ЗК	Опис
id	int	+	-	Ідентифікатор
name	varchar	-	-	Назва колонки
Id_boards	int	-	+	Зовнішній ключ дошки

У таблиці «Завдання» зберігається зовнішній ключ колонки. Це допомагає при переміщенні завдань визначати до якої колонки належить завдання.

Таблиця 2.4

Опис таблиці tasks

Поле	Тип даних	ПК	ЗК	Опис
id	int	+	-	Ідентифікатор
Title	varchar	-	-	Назва завдання
description	text	-	-	Опис завдання
Id_columns	Int	-	+	Зовнішній ключ колонки
priority	Varchar	-	-	Пріоритет
deadline	Date	-	-	Дедлайн
Index_task	int	-	-	Індекс розташування у колонці

Таблиця «Підзавдання». Сюди зберігаються додаткові задачі, які користувач створює самостійно.

Опис таблиці subtasks

Поле	Тип даних	ПК	ЗК	Опис
id	int	+	-	Ідентифікатор
Title	varchar	-	-	Назва підзавдання
Is_completed	tinyint	-	-	Статус чи завдання виконано
Id_task	Int	-	+	Зовнішній ключ завдання

Отже, кожна таблиця у нашій базі даних має своє призначення та спрямована на зберігання конкретної категорії даних. Кожна таблиця відповідає за певний аспект функціонування системи, що сприяє організації даних у логічні групи та забезпечує зручний доступ до них. Такий підхід сприяє ефективному управлінню та розвитку системи в майбутньому, забезпечуючи необхідну гнучкість та розширюваність для впровадження нових функціональних можливостей (рис.2.2) [27].

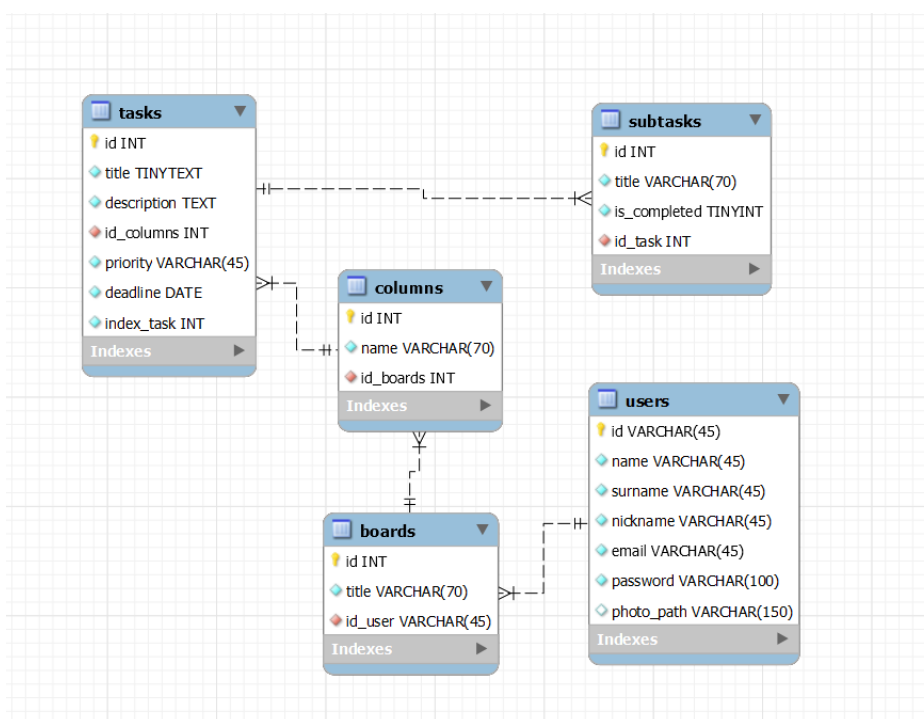


Рисунок 2.2 – Діаграма бази даних

2.3 Проектування та реалізація алгоритмів роботи системи

Під час входу користувача у систему, якщо він ще не створив жодної дошки, він буде перенаправлений на окрему сторінку для її створення. На цій

сторінці користувач матиме можливість створити нову дошку, визначити її назву та інші необхідні параметри. Після створення дошки користувач матиме можливість створювати необмежену кількість завдань та колонок, редагувати їх опис та інші атрибути. Крім того, користувач може легко переміщувати завдання між колонками, змінювати їх порядок та редагувати будь-яку іншу інформацію, пов'язану з завданнями та дошками. Це дозволяє користувачам ефективно організовувати та керувати своїми завданнями та розуміти процес роботи додатку інтуїтивно, забезпечуючи зручний і простий спосіб управління їхніми проєктами.

У користувача є можливість редагувати свій профіль, змінювати пароль, змінювати фото профілю та налаштовувати тему інтерфейсу, перемикаючи її зі світлої на темну або навпаки. Це забезпечує додаткову персоналізацію та зручність використання системи, дозволяючи кожному користувачу налаштувати її під свої потреби та вподобання.

Нижче на діаграмі описано активності користувача:

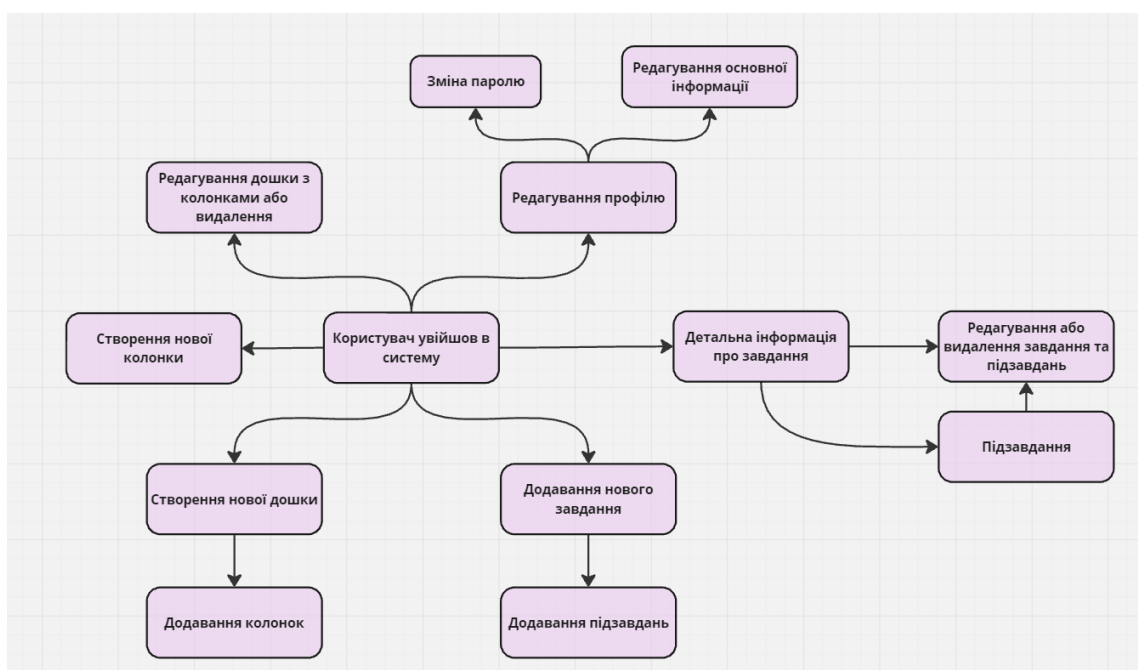


Рисунок 2.3 – Діаграма активності користувача

2.4 Реалізація онлайн-платформи управління проєктами

Проєкт складається з двох основних частин, завдяки фреймворку Next.js, який дозволяє легко відділити бекенд-логіку від фронтенду [13]. Перша частина – це фронтенд, де використовується React.js для генерації всіх компонентів інтерфейсу. Завдяки React.js, користувачі отримують швидкий та інтерактивний досвід взаємодії з додатком. Друга частина – це бекенд, де реалізована вся основна логіка додатку [19]. Там обробляються запити від користувачів, відбувається взаємодія з базою даних, виконується та забезпечується безпечність та ефективність роботи системи. Такий підхід з використанням Next.js дозволяє створити надійну і функціональну платформу для управління проєктами.

Можна розглянути основні етапи роботи та функціональні можливості додатку.

Процес реєстрації відбувається таким чином: під час реєстрації генерується унікальний ідентифікатор для кожного користувача, а пароль шифрується за допомогою бібліотеки bcrypt. Це забезпечує високий рівень безпеки для користувачів. Крім того, система перевіряє, чи вже існує в базі даних вказаний email або нікнейм, що дозволяє уникнути дублювання облікових записів.

Функція шифрування паролю:

```
async function hashPassword(password) {
  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    return hashedPassword;
  } catch (error) {
    throw new Error('Error hashing password');
  }
}
```

Під час авторизації користувача відбувається генерація jwt [9], та його перевірка при вході на сайт завдяки функції:

```
import jwt from "jsonwebtoken";

export const verifyJwt = (req, next) => {
  const token = req.headers["access-token"];
  if(!token) {
    return new Response(
```

```

        JSON.stringify({ success: false, data: "Don't have web-token" })),
        {
            status: 404,
        }
    );
} else {
    jwt.verify(token, process.env.SECRET_KEY, (err, decoded) => {
        if(err) {
            return new Response(
                JSON.stringify({ success: false, data: "Not authenticated" })),
                {
                    status: 404,
                }
            );
        } else {
            console.log(decoded)
            next();
        }
    })
}
}
}

```

Функція `verifyJwt` [12] використовується для автентифікації запитів на сервері, забезпечуючи безпеку даних користувачів. Вона перевіряє наявність та дійсність JWT (JSON Web Token) у заголовку запиту під назвою "access-token".

Коли запит надходить на сервер, функція спочатку перевіряє, чи присутній токен у заголовку. Якщо токен відсутній, функція повертає відповідь зі статусом 404. Якщо токен присутній, функція використовує метод `jwt.verify` для перевірки його дійсності за допомогою секретного ключа. Якщо токен недійсний або застарілий, функція повертає відповідь зі статусом 404 і повідомленням "Not authenticated".

Якщо токен успішно пройшов перевірку, функція декодує його, виводить інформацію у консоль і викликає функцію `next()`, яка передає управління наступному обробнику запиту. Це дозволяє серверу продовжити обробку запиту і надавати доступ до захищених ресурсів. Таким чином, `verifyJwt` забезпечує, що тільки автентифіковані користувачі можуть отримати доступ до певних частин системи, підвищуючи рівень безпеки додатку.

Був створений власний хук useDark [21], який дозволяє контролювати тему, яку використовує користувач – темну чи світлу. Цей хук забезпечує зручний спосіб взаємодії з темою в інтерфейсі додатку. Шляхом його використання можна легко змінювати вигляд додатку в залежності від уподобань користувача або налаштувань, що надаються. Він може використовуватися для встановлення стилів, компонентів і різних аспектів дизайну, які пов'язані з вибором теми. Такий підхід до керування темами дозволяє забезпечити користувачам комфортне використання додатку відповідно до їхніх вподобань та потреб.

```
import { useState, useEffect } from 'react'

export default function useDark() {
  const [theme, setTheme] = useState(localStorage.theme)
  const themeColor = theme === 'dark' ? 'light' : 'dark'

  useEffect(() => {
    const root = window.document.documentElement;

    root.classList.remove(themeColor)
    root.classList.add(theme)

    localStorage.setItem('theme', theme)
  }, [theme, themeColor])

  return[themeColor, setTheme]
}
```

Для зручного переміщення завдань у додатку було впроваджено функцію перетягування. Для цього фіксується поточний індекс завдання в колонці. Цей індекс є важливим для того, щоб визначити конкретне місце, куди користувач хоче перемістити завдання всередині колонки. Після того, як користувач відпускає завдання у новій колонці, спрацьовує функція handleOnDrop, яка відповідає за зміну зовнішнього ключа колонки у завданні на новий. Цей механізм дозволяє ефективно та інтуїтивно переміщувати завдання між різними стовпцями у додатку, підвищуючи його зручність та продуктивність [18].

В даній функції ми відправляємо дані коли починаємо тягнути:

```
function handleDragTask(event) {
  event.dataTransfer.setData(
```

					ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "text",
        JSON.stringify({ taskId: task.id, prevColIndex: column.id, index })
    );
    onUpdateTask();
}

```

А тут відбувається отримання даних та їх перезапис у базі даних:

```

async function handleOnDrop(event, dropIndex, dropTaskId) {
    const { prevColIndex, taskId, index } = JSON.parse(
        event.dataTransfer.getData("text")
    );
    const updateTaskResponse = await fetch(`/api/tasks`, {
        method: "PUT",
        headers: {
            "Content-Type": "application/json",
        },
        body: JSON.stringify(updateData),
    });
    const updateTaskResult = await updateTaskResponse.json();
    if (!updateTaskResult.success) {
        console.error(updateTaskResult.data);
        return;
    }
}

```

Компоненти Center, Column та Task є ключовими для відображення вмісту поточної дошки у додатку. Коли користувач клікає на будь-яку доступну дошку, відбувається отримання інформації про неї. Ця інформація потім записується у глобальний стан за допомогою Redux [6], що дозволяє зручно та ефективно керувати станом додатку. Використання Redux було обрано з урахуванням його переваг у роботі зі станом [15]: це робить можливим отримання актуальної інформації про дошку з будь-якого компонента чи модуля додатку. Такий підхід спрощує розробку та покращує її масштабованість, дозволяючи зосередитися на логіці та функціоналі, не втрачаючи при цьому ефективності управління станом.

```

async function handleBoardClick(boardId) {
    const response = await fetch(`/api/boards?id=${boardId}`);
    const result = await response.json();
    dispatch(activeBoardActions.setActiveBoard(result.data));
}

```

Коли відбувається клік по дошці, отримується інформація про цю дошку з бази даних та одразу записується у глобальний стан.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

Висновки до другого розділу

Розробка онлайн-платформи управління проєктами налічувала кілька ключових етапів. Перш ніж приступити до розробки, було важливо чітко визначити вимоги до функціоналу, зокрема шляхом розробки прецедентів використання та визначення функціональних, нефункціональних та системних вимог. На основі цих вимог була спроектована база даних, що дозволила організувати інформацію про користувачів, дошки, завдання тощо у логічні групи.

Після цього було реалізовано веб-додаток з використанням Next.js та React.js. Безпека була важливим аспектом, тому під час реєстрації та авторизації користувачів використовувалися шифрування паролів та механізми JWT для забезпечення безпеки даних. Такий підхід дозволив створити надійну та функціональну платформу для ефективного управління проєктами.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОНЛАЙН-ПЛАТФОРМОЮ УПРАВЛІННЯ ПРОЄКТАМИ

3.1 Порядок встановлення та налаштування параметрів платформи

Для успішного налаштування потрібно встановити необхідне програмне забезпечення та налаштувати параметри підключення до бази даних.

Потрібно встановити:

- Node.js v22.3
- MySQL v8.0

Також потрібно налаштувати відповідні параметри при підключенні до бази даних:

- Host: Вказати IP-адресу або доменне ім'я сервера, на якому розташована база даних.
- Port: Вказати порт, через який здійснюється з'єднання з MySQL сервером. Зазвичай це стандартний порт 3306.
- User: Ім'я користувача, який має дозвіл на доступ до бази даних.
- Password: Пароль користувача, який використовується для автентифікації при підключенні до бази даних.
- Database: Назва бази даних.

3.2 Структура інтерфейсу онлайн-платформи управління проєктами

Перед тим як відвідати сайт користувачу потрібно зареєструватись, або увійти, якщо у нього вже є аккаунт. На екрані присутня форма реєстрації та можливість увійти в акаунт, якщо користувач вже зареєстрований. Натискання на напис під кнопкою реєстрації відкриє нову форму у якій потрібно буде пройти автентифікацію.

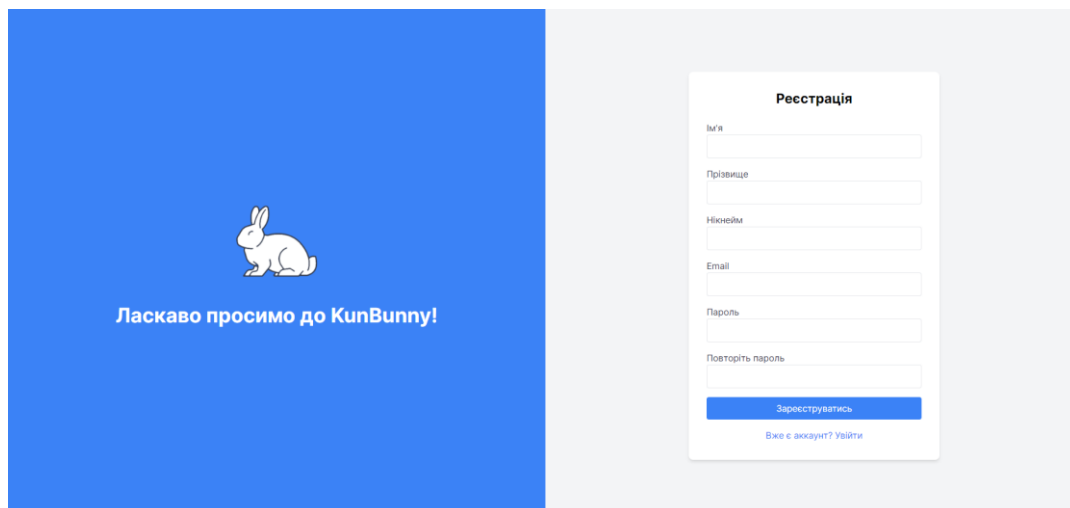


Рисунок 3.1 – Привітний екран та форма реєстрації

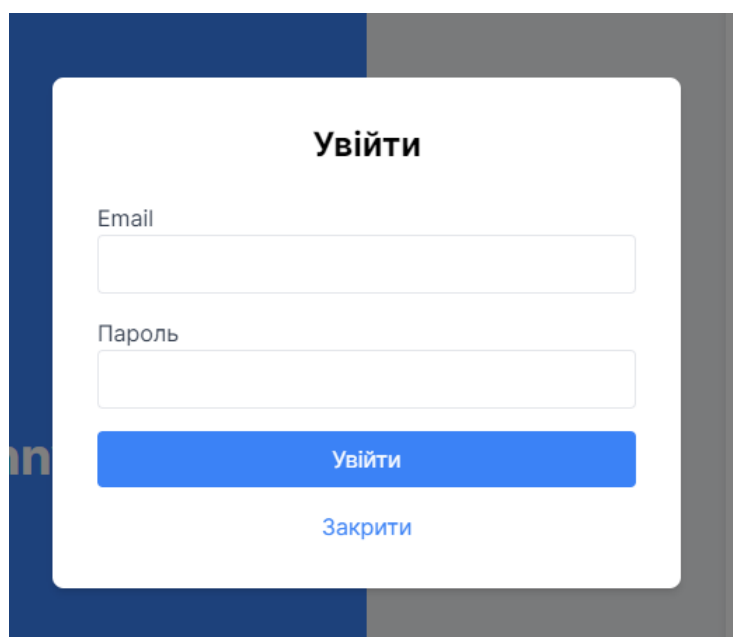


Рисунок 3.2 – Форма для автентифікації

Після автентифікації на сайті користувачеві відразу пропонується почати роботу, створивши нову дошку. Ця функція надає можливість відразу почати організацію завдань, проєктів або будь-яких інших справ. Створення нової дошки допомагає швидко розпочати роботу без зайвих перешкод.

При створенні нової дошки з'являється модальне вікно, де користувачеві потрібно ввести інформацію для налаштування дошки. Він має можливість вказати назву дошки, а також назви колонок, які бажає мати на дошці. Після цього він може додати будь-яку кількість колонок, щоб створити належну структуру для своїх завдань або проєктів. Ця функція дозволяє користувачам

індивідуалізувати свої дошки відповідно до їхніх потреб і уподобань, що полегшує управління завданнями та планування проєктів.

Рисунок 3.3 – Створення нової дошки

Після створення дошки відбувається автоматичний перехід на головний екран, де користувачеві надається доступ до широкого спектру функцій. Тут він може створювати додаткові дошки для різних проєктів або завдань, редагувати наявні дошки, додавати нові колонки або редагувати існуючі, а також створювати та редагувати завдання в межах кожної дошки.

Покрокова навігація та доступність функцій дозволяють користувачам швидко та ефективно розпочати роботу над своїми проєктами. Крім того, вони можуть перейти в свій профіль для зміни або перегляду особистої інформації, вибору світлої або темної теми відповідно до власних уподобань, а також приховати бічну панель для збільшення робочого простору. Це забезпечує комфортний та продуктивний користувацький досвід, допомагаючи користувачам зосередитися на важливих завданнях та досягненні своїх цілей.

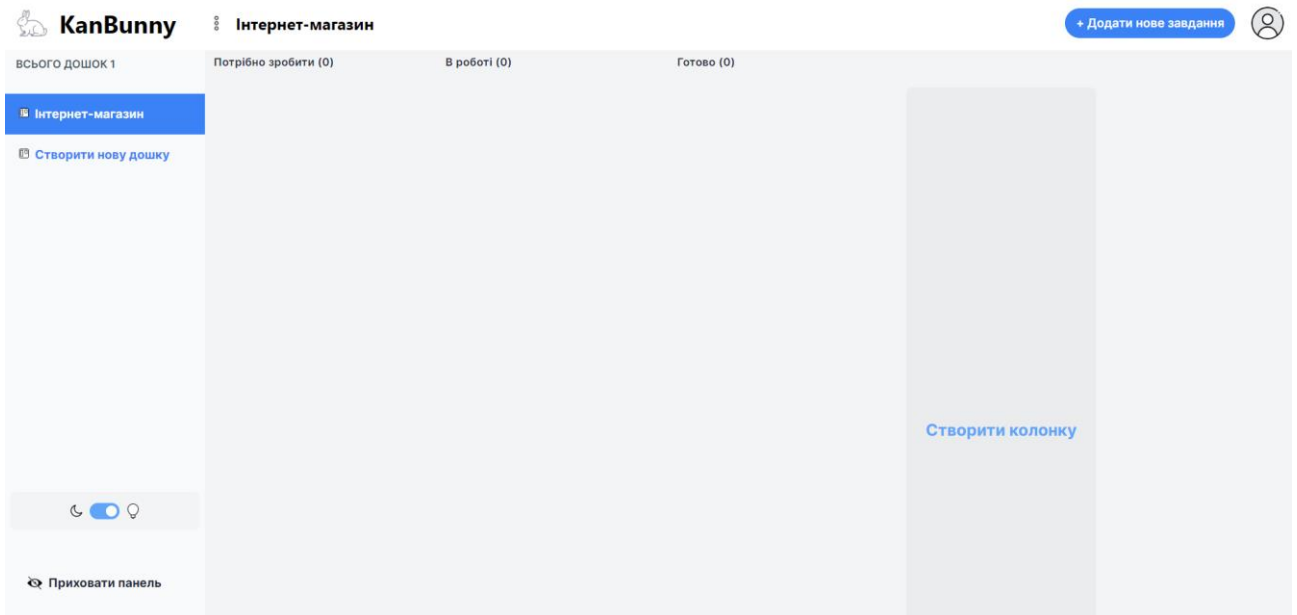


Рисунок 3.4 – Головний екран

При додаванні нового завдання користувачу пропонується заповнити лише необхідну інформацію для швидкої та зручної організації завдань. Він може встановити пріоритет, вказати дедлайн та обрати колонку, до якої потрібно додати завдання. Крім того, є можливість додати підзавдання для розбиття основного завдання на частини.

Рисунок 3.5 – Додавання завдання

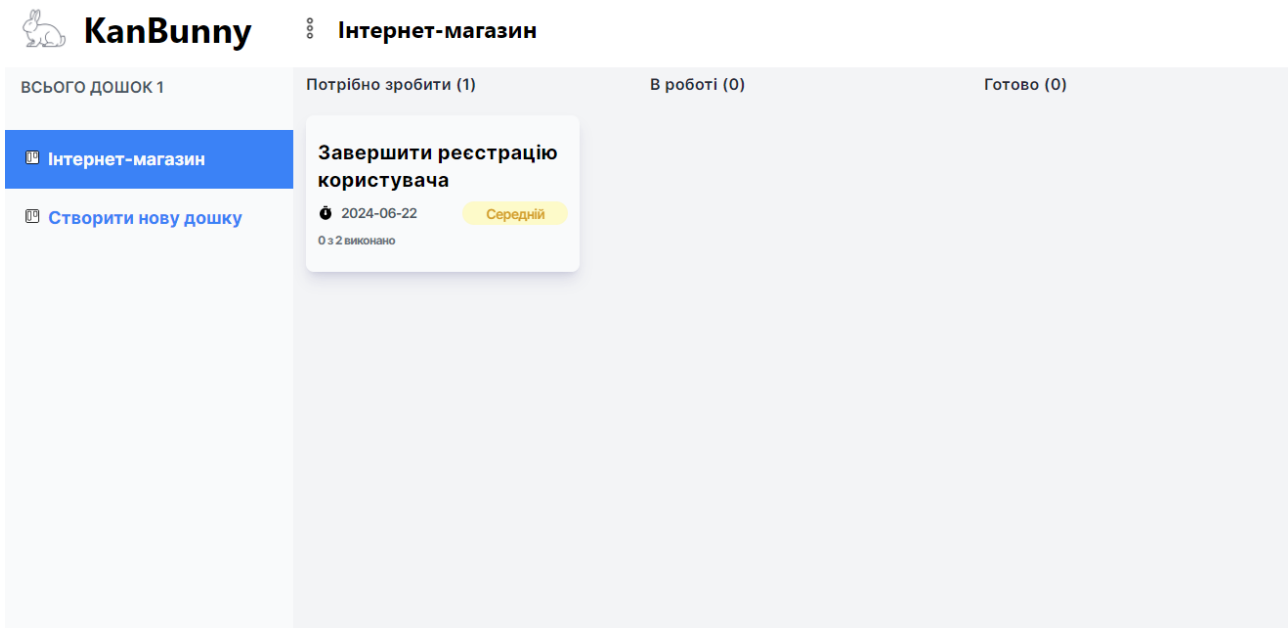


Рисунок 3.6 – Додано нове завдання

При переході на окреме завдання користувач має можливість докладніше розглянути його, перевірити всі деталі, такі як пріоритет, дедлайн та підзавдання. Крім того, він може відмітити підзавдання як виконані, оновити інформацію про завдання та змінити його статус. Також користувач може зручно переміщувати завдання між колонками, просто перетягуючи їх відповідно до потреб та пріоритетів проєкту.

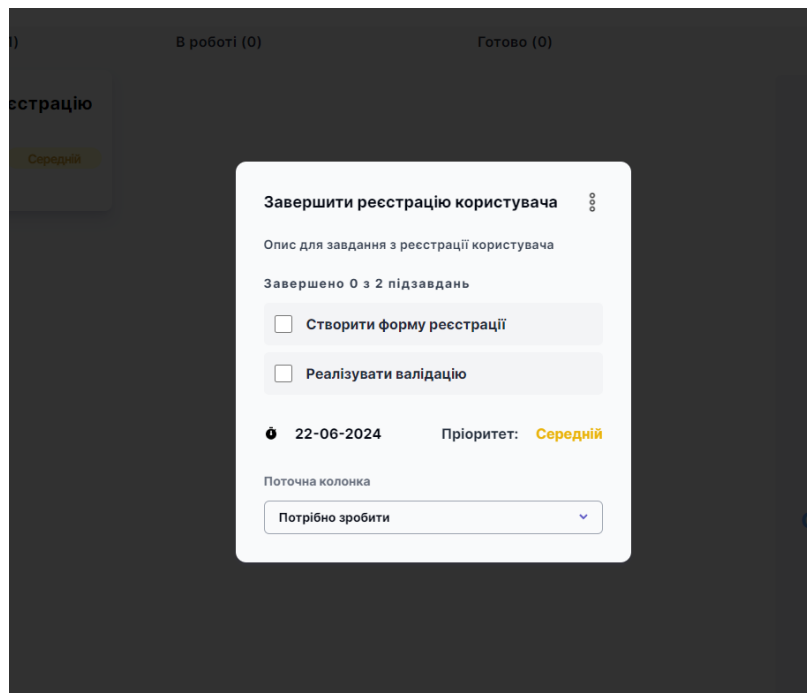


Рисунок 3.7 – Детальна інформація про завдання

Натиснувши на меню, у користувача є можливість редагування завдання, або ж його видалення.

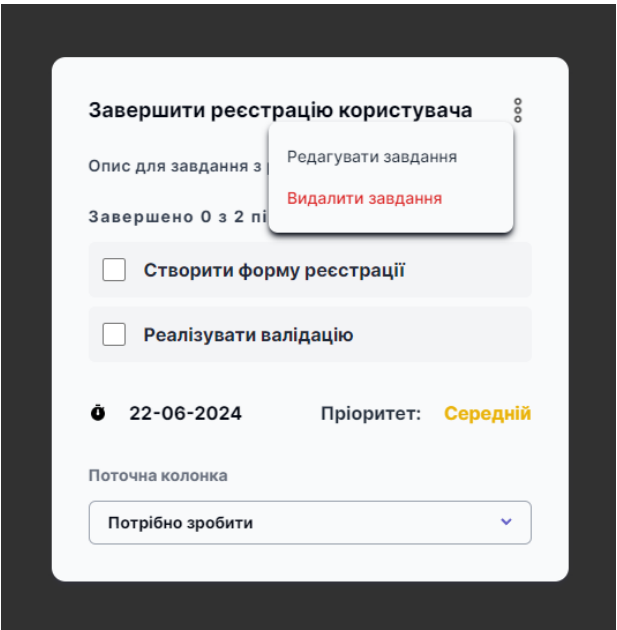


Рисунок 3.8 – Меню завдання

При редагуванні відкривається модальне вікно, у якому можна відредагувати всі дані, додати нові підзавдання, змінити пріоритет або дедлайн.

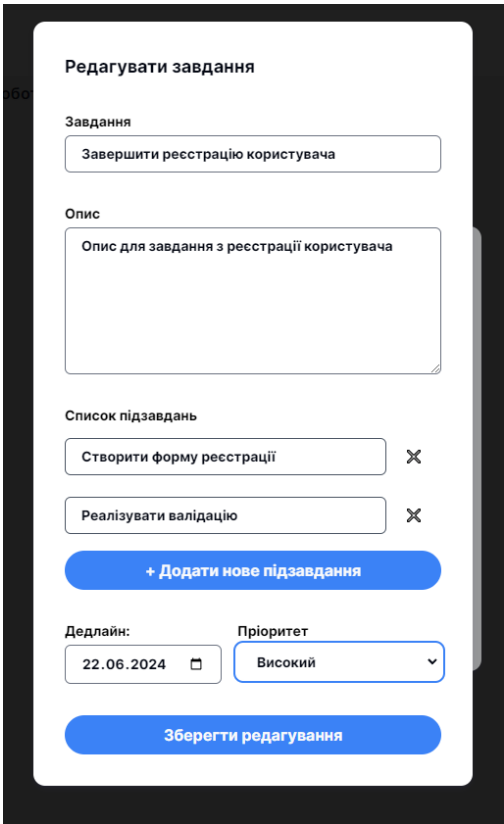


Рисунок 3.9 – Редагування завдання

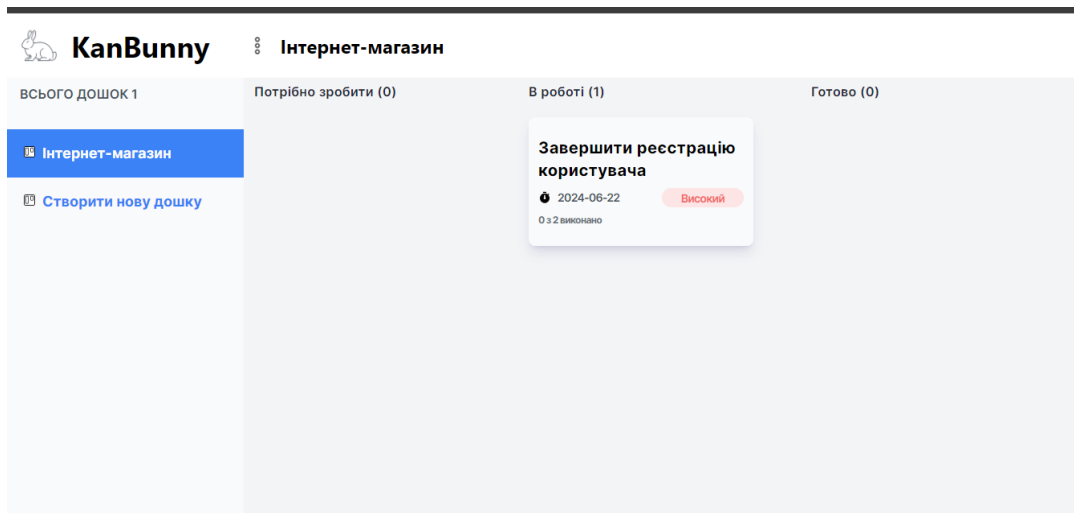


Рисунок 3.10 – Переміщення завдання

Якщо всі підзавдання у завданні будуть відзначені як виконані, то саме завдання автоматично отримає статус "виконано". Це відобразатиметься в закресленому вигляді, щоб користувач мав чітке уявлення про його стан.

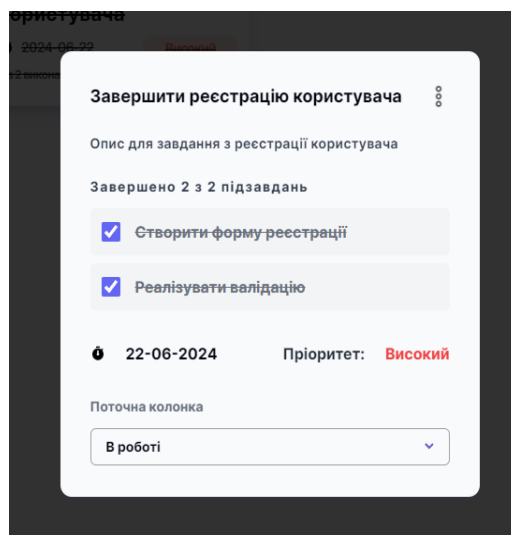


Рисунок 3.11 – Виконання усіх підзавдань

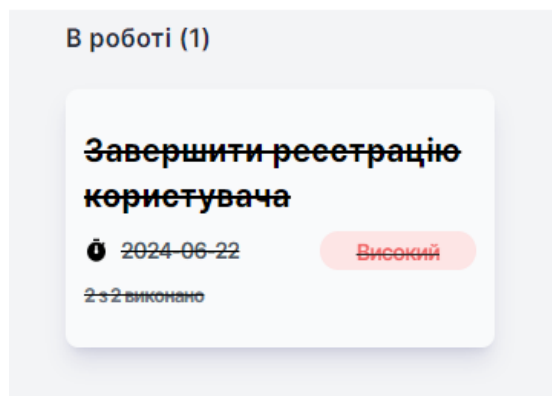


Рисунок 3.12 – Закреслений вигляд виконаного завдання

У дошки також є можливість редагування та видалення

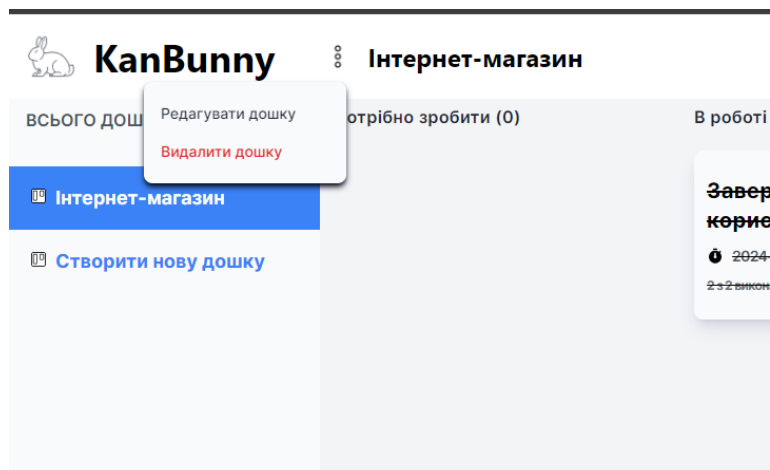


Рисунок 3.13 – Меню дошки

Під час редагування дошки користувач може внести зміни, щоб дошка краще відповідала поточним потребам та умовам проєкту. Він може змінити назву самої дошки, щоб краще відображати її зміст або контекст. Також він має можливість змінювати назви колонок, їх кількість та видаляти непотрібні колонки, щоб точніше відобразити етапи робочого процесу або категорії завдань.

Рисунок 3.14 – Редагування дошки

При натисканні на ім'я користувача відображається інформація про його профіль, де можна переглянути та відредагувати основні дані, такі як ім'я,

електронна пошта тощо, та змінити фотографію. Також користувач має можливість вийти з профілю для забезпечення безпеки своїх даних.

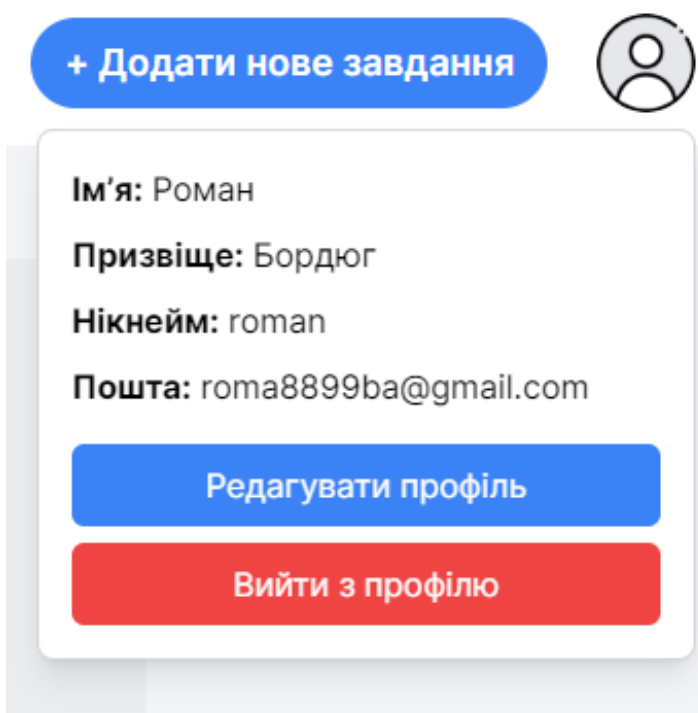


Рисунок 3.15 – Меню профілю

Редагувати профіль

Ім'я

Роман

Прізвище

Бордюг

Нікнейм

roman

Пошта

roma8899ba@gmail.com

Фотографія профілю

Завантажити фото

Змінити пароль

Зберегти

Рисунок 3.16 - Редагування профілю

Завантаження фото свого профілю

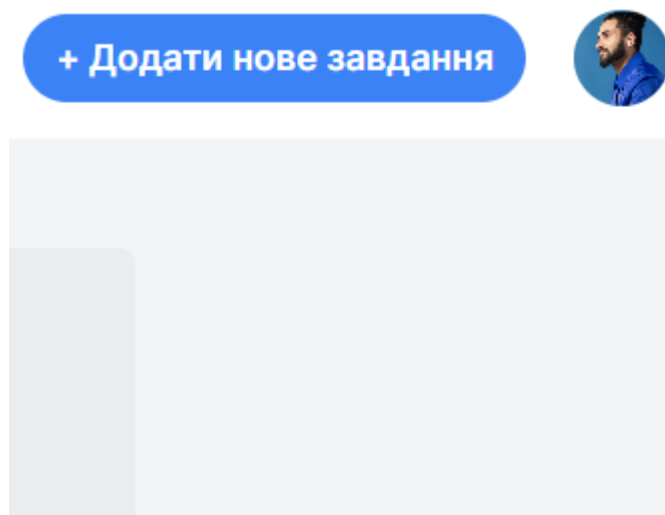


Рисунок 3.17 – Фото профілю

Крім того, в додатку існує можливість змінити дизайн на темну тему, щоб забезпечити зручність користування.

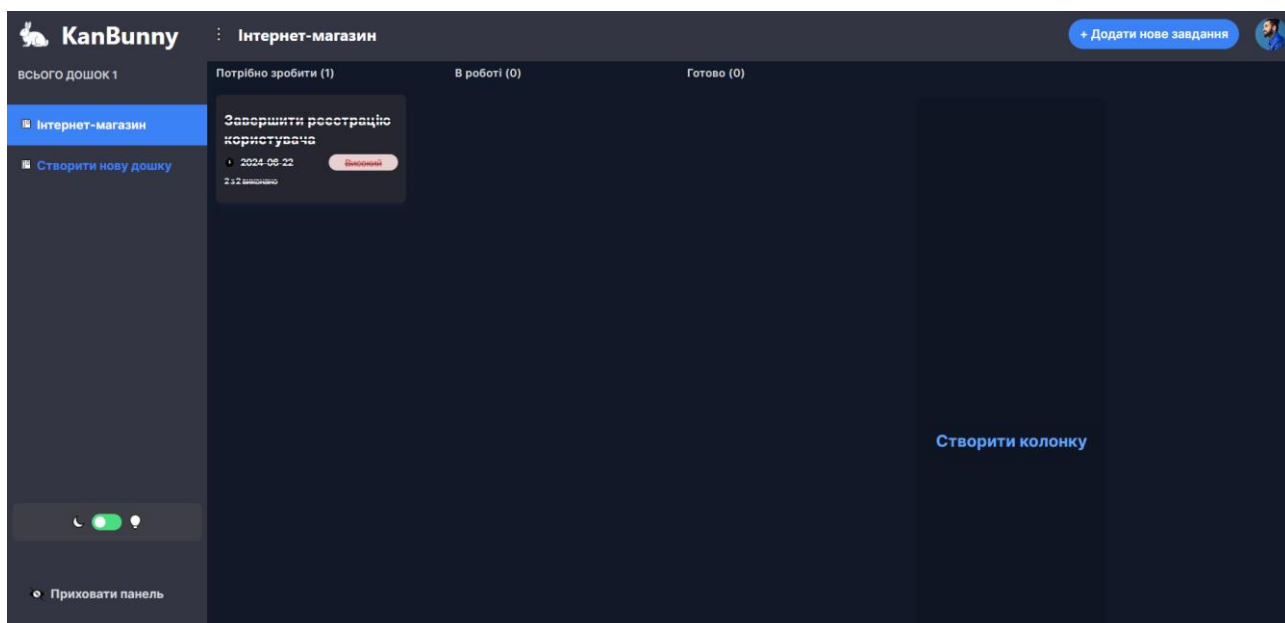


Рисунок 3.18 – Темна тема інтерфейсу

За допомогою адаптації під мобільні пристрої додаток стає доступним для використання на різних пристроях, забезпечуючи зручність користування як на комп'ютерах, так і на смартфонах чи планшетах.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

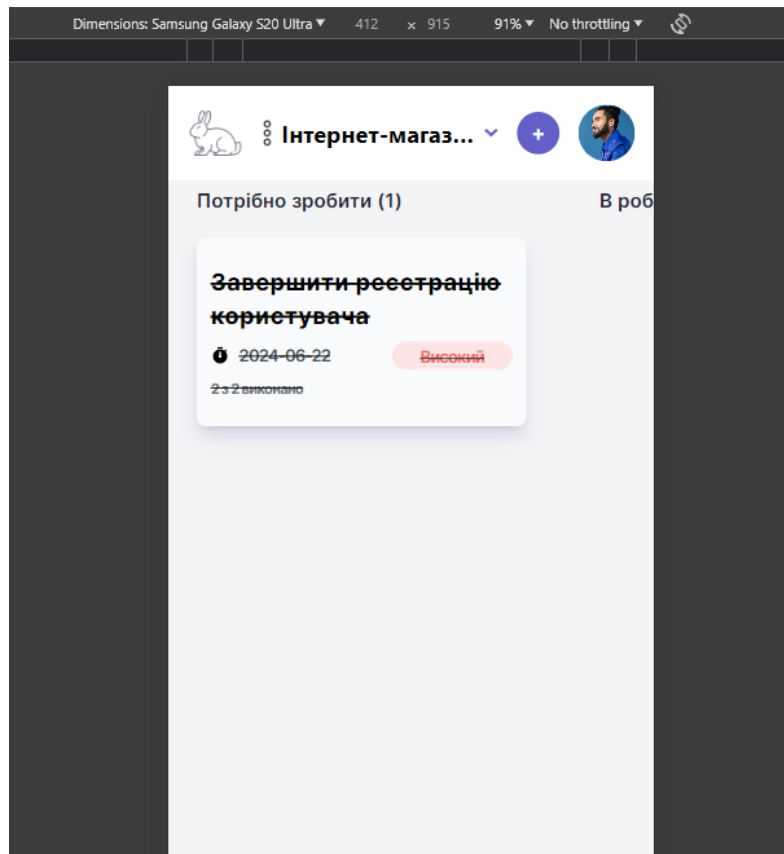


Рисунок 3.19 – Мобільна адаптація додатку

3.3 Тестування платформи

Тестування може відбуватись як вручну, так і за допомогою сервісів. Я буду використовувати ручне тестування, тому що воно надає більшу гнучкість та адаптивність [22]. Ручне тестування дозволяє оперативно вносити зміни в процес тестування, враховуючи реальні користувацькі сценарії, що можуть бути непрогнозованими або занадто складними для автоматизованих скриптів.

Для початку перевіримо форми реєстрації та авторизації. Спочатку, якщо користувач вводить некоректні або неповні дані під час реєстрації, додаток повинен відобразити відповідне попередження або повідомлення про помилку, що вказує на необхідність виправлення помилок. Наприклад, якщо користувач не заповнив обов'язкові поля, такі як електронна пошта або пароль, або якщо введені дані не відповідають вимогам безпеки, додаток повинен сповіщати про це. Аналогічно, під час авторизації, якщо введені дані для входу, такі як електронна пошта або пароль, не співпадають з тими, що вже зареєстровані в

системі, користувач повинен отримати чітке повідомлення про невідповідність даних і необхідність повторного введення. Таким чином, ці перевірки забезпечують коректність введених даних.

Рисунок 3.20 – Перевірка на коректність форми автентифікації

Рисунок 3.21 – Введення помилкових даних

Рисунок 3.22 - Введення помилкових даних

Далі можна провести перевірку додавання та видалення завдання. При додаванні нового завдання спочатку варто переконатись, що інтерфейс дозволяє користувачеві ввести всі необхідні деталі, такі як назва завдання, дедлайн, пріоритет та відповідна колонка. Після заповнення форми та підтвердження додавання, завдання має з'явитися в обраній колонці з усіма введеними даними. Перевіримо функцію видалення завдання. Оберемо завдання, яке потрібно видалити, та підтвердимо видалення.

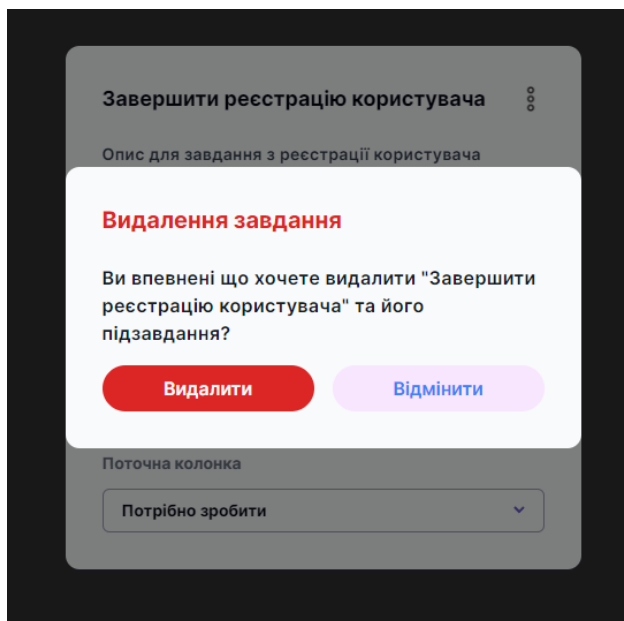


Рисунок 3.23 – Видалення завдання

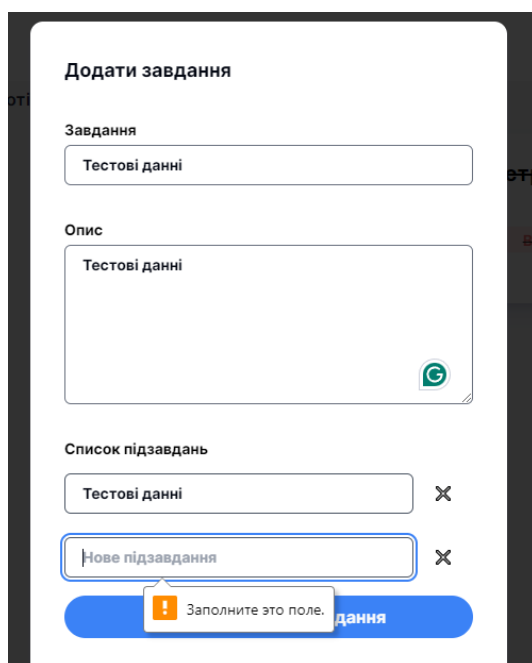


Рисунок 3.24 – Додавання нового завдання

Далі перевіримо редагування даних користувача, зміну пошти, пароллю.

При спробі змінити пошту на існуючу у користувача з'явиться відповідне попередження про неможливість зміни пошти. Якщо користувач хоче змінити пароль, але вводить некоректні дані старого паролю, зміна паролю буде неможлива.

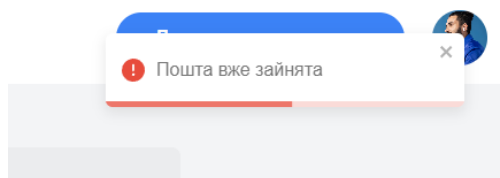


Рисунок 3.25 – Зміна пошти користувача

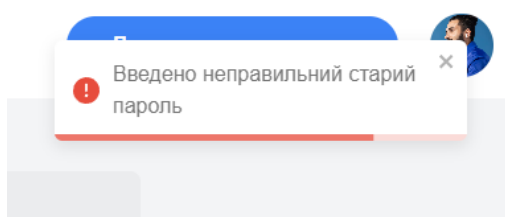


Рисунок 3.26 – Зміна пароля користувача

Висновки до третього розділу

У даному розділі були розглянуті можливості онлайн-платформи управління проєктами, який надає користувачам зручний інструмент для організації їхніх завдань та проєктів. Додаток включає в себе функції реєстрації та авторизації, додавання, редагування та видалення завдань, а також можливість редагування особистих даних користувача.

Під час тестування додатку було виявлено, що інтерфейс дозволяє зручно виконувати усі необхідні операції, а також забезпечує коректну реакцію на некоректно введені дані та виключні ситуації. Тестування підтвердило ефективність та надійність роботи додатку управління проєктами.

ВИСНОВКИ

У результаті аналізу інструментів управління проєктами, а також вимог до апаратного та програмного забезпечення для розробки веб-додатків, стало очевидним, що успішна розробка ефективного інструменту для управління проєктами базується на ретельному плануванні та виборі оптимальних технологій.

Процес розробки веб-додатку включав різні етапи, починаючи з дослідження потреб користувачів та визначення вимог, і завершуючи реалізацією програмного продукту з урахуванням вимог безпеки та функціональності.

Під час розробки онлайн-платформи управління проєктами, було не лише важливо, але й необхідно перевірити його функціональність та надійність шляхом тестування. Цей процес дозволив підтвердити ефективність та коректність роботи додатку в реальних умовах.

Отже, на основі вивчення інструментів управління проєктами, вимог до розробки веб-додатків та результатів тестування, було розроблено високоефективний та надійний веб-додаток, який надає користувачам зручний інструмент для ефективного керування їхніми проєктами.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Trello [Електронний ресурс] – Режим доступу до ресурсу: <https://trello.com/uk>
2. Jira [Електронний ресурс] – Режим доступу до ресурсу: <https://www.atlassian.com/ru/software/jira>
3. Monday [Електронний ресурс] – Режим доступу до ресурсу: <https://monday.com/>.
4. NextJs [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs>
5. React [Електронний ресурс] – Режим доступу до ресурсу: <https://react.dev/>
6. Redux [Електронний ресурс] - Режим доступу до ресурсу: <https://redux.js.org/>
7. TailWind [Електронний ресурс] - Режим доступу до ресурсу: <https://v2.tailwindcss.com/docs>
8. MySQL [Електронний ресурс] - Режим доступу до ресурсу: <https://dev.mysql.com/doc/>
9. JWT [Електронний ресурс] - Режим доступу до ресурсу: <https://jwt.io/introduction>
10. Medium [Електронний ресурс] - Режим доступу до ресурсу: <https://medium.com/>
11. Udemy [Електронний ресурс] - Режим доступу до ресурсу: <https://ua.udemy.com/organization/home/>
12. FreeCodeCamp [Електронний ресурс] - Режим доступу до ресурсу: <https://www.freecodecamp.org/>
13. MDN [Електронний ресурс] - Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
14. Stack Overflow [Електронний ресурс] - Режим доступу до ресурсу: <https://stackoverflow.com/>

15. W3School [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.w3schools.com/>
16. CSS Tricks [Електронний ресурс] - Режим доступу до ресурсу:
<https://css-tricks.com/>
17. Dev To [Електронний ресурс] - Режим доступу до ресурсу:
<https://dev.to/>
18. CodePen [Електронний ресурс] - Режим доступу до ресурсу:
<https://codepen.io/>
19. CodeSandBox [Електронний ресурс] - Режим доступу до ресурсу:
<https://codesandbox.io/>
20. Smashing Magazine [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.smashingmagazine.com/>
21. useHooks [Електронний ресурс] - Режим доступу до ресурсу:
<https://usehooks.com/>
22. CodeAcademy [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.codecademy.com/>
23. CSSGrid [Електронний ресурс] - Режим доступу до ресурсу:
<https://cssgrid.io/>
24. Javascript Info [Електронний ресурс] - Режим доступу до ресурсу:
<https://javascript.info/>
25. FrontEndMasters [Електронний ресурс] - Режим доступу до ресурсу:
<https://frontendmasters.com/>
26. GeeksForGeeks [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.geeksforgeeks.org/>
27. UdaCity [Електронний ресурс] - Режим доступу до ресурсу:
<https://www.udacity.com/>
28. Мартін Фаулер. "Рефакторинг: Покращення дизайну існуючого коду". Addison-Wesley Professional, 2018. — 464 с.
29. Майкл Хартл. "React.js Essentials". Packt Publishing, 2016. — 274 с.

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

30. Кейт Стайлтон, Метт Бартерс. "Бази даних MySQL: Підручник з прикладами". Wiley, 2018. — 864 с.

					ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг Center.jsx

```

import { useEffect, useState } from "react";
import { useSelector } from "react-redux";
import Column from "../Column";
import Sidebar from "../Sidebar";
import FirstTimeBoard from "../FirstTimeBoard";
import AddBoardModal from "@modals/AddBoardModal";

export default function Center({ boardModal, setBoardModal }) {

  const token = localStorage.getItem("token");

  const [updateCenter, setUpdateCenter] = useState(false);

  function onUpdateCenter() {
    setUpdateCenter((prev) => !prev);
  }

  const [sidebarStatus, setSidebarStatus] = useState(true);

  const [pageSize, setPageSize] = useState([
    window.innerWidth,
    window.innerHeight,
  ]);

  useEffect(() => {
    function pageResize() {
      setPageSize([window.innerWidth, window.innerHeight]);
    }

    window.addEventListener("resize", pageResize);

    return () => {
      window.removeEventListener("resize", pageResize);
    };
  }, [updateCenter]);

  // Получаем доски
  const activeBoard = useSelector((state) => state.activeBoard.board);
  const [boards, setBoards] = useState([]);

  const fetchBoards = async () => {
    const response = await fetch(`/api/boards?token=${token}`);
    const data = await response.json();
    if (data.success) {
      setBoards(data.data);
    }
  }
}

```

					ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    });

    // Получаем колонки
    const [columns, setColumns] = useState([]);

    const fetchColumns = async () => {
        const response = await fetch(`/api/columns?board_id=${activeBoard.id}`);
        const data = await response.json();
        if (data.success) {
            setColumns(data.data);
        }
    };

    useEffect(() => {
        fetchBoards();
        fetchColumns();
    }, [updateCenter, activeBoard]);

    return (
        <div
            className={
                pageSize[0] >= 768 && sidebarStatus
                    ? "bg-gray-100 h-screen flex scrollbar-hide dark:bg-gray-900
overflow-x-scroll gap-6 ml-[260px]"
                    : "bg-gray-100 h-screen flex scrollbar-hide dark:bg-gray-900
overflow-x-scroll gap-6"
            }>
            {pageSize[0] >= 768 && (
                <Sidebar
                    sidebarStatus={sidebarStatus}
                    setSidebarStatus={setSidebarStatus}
                    boards={boards}
                    updateCenter={onUpdateCenter}
                />
            )}

            {columns.length > 0 ? (
                <>
                    {columns.map((column) => (
                        <Column
                            key={column.id}
                            column={column}
                            columns={columns}
                            updateCenter={onUpdateCenter}
                            activeBoard={activeBoard}
                        />
                    ))}
                </>
            ) : (
                <div
                    onClick={() => {

```

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        setBoardModal(true);
    }}
    className="h-screen dark:bg-[#00000020] flex justify-center
items-center font-bold text-2xl hover:text-blue-600 transition duration-300 cursor-
pointer bg-[#00000008] scrollbar-hide mb-2 mx-5 pt-[90px] min-w-[280px] text-blue-
400 rounded-lg mt-[135px]">
        Створити колонку
    </div>
</>
) : (
    <FirstTimeBoard type="edit" fetchBoards={fetchBoards} />
)

{boardModal && (
    <AddBoardModal
        setBoardModal={setBoardModal}
        type="edit"
        onAdd={onUpdateCenter}
    />
)}
</div>
);
}

```

Лістинг Task.jsx

```

import TaskSubtaskModal from "@/modals/TaskSubtaskModal";
import { useEffect, useState } from "react";

const deadlineIcon = "assets/deadlineIcon.png";

export default function Task({
    task,
    tasks,
    columns,
    updateCenter,
    activeBoard,
    updater,
    column,
    index
}) {
    const [updateTask, setUpdateTask] = useState(false);

    function onUpdateTask() {
        setUpdateTask((prev) => !prev);
        updateCenter();
    }
}

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const [taskModal, setTaskModal] = useState(false);

//Отримуємо підзавдання
const [subtasks, setSubtasks] = useState([]);

// console.log("TASKID SUBTASKS", task.title, task.id, subtasks)

useEffect(() => {
    fetchSubtasks();
}, [updateTask, activeBoard, updater]);

const fetchSubtasks = async () => {
    const resp = await fetch(`/api/subtasks?id_task=${task.id}`);
    const res = await resp.json();
    if (res.success) {
        setSubtasks(res.data);
    }
};

const countCompletedSubtasks = subtasks.filter((el) => el.is_completed);

function handleDragTask(event) {
    event.dataTransfer.setData(
        "text",
        JSON.stringify({ taskId: task.id, prevColIndex: column.id, index })
    );
    onUpdateTask();
}

const getPriorityColor = (priority) => {
    switch (priority) {
        case "Низький":
            return "text-green-600 bg-green-100 opacity-90";
        case "Середній":
            return "text-yellow-600 bg-yellow-100 opacity-90";
        case "Високий":
            return "text-red-500 bg-red-100 opacity-90";
        default:
            return { text: "", bg: "" };
    }
};

return (
    <div
        className={`${
            countCompletedSubtasks.length === subtasks.length &&
            "line-through"
        }`} >
        <div
            onDragStart={handleDragTask}
            draggable

```

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

```

onClick={() => {
  setTaskModal(true);
}}
className="w-[280px] first:my-5 rounded-lg bg-gray-50 dark:bg-
[#252630]
shadow-[#38367e29] py-6 px-3 shadow-lg hover:text-blue-500
dark:text-gray-50
dark: hover:text-blue-500 cursor-pointer">
<p className="font-bold text-xl tracking-wide">{task.title}</p>

<div className="flex items-center justify-center mt-2">
  <img
    src={deadlineIcon}
    alt="Deadline Icon"
    className="w-4 h-4 mr-2"
  />
  <p
    className={`font-semibold text-sm dark:text-gray-50
tracking-tighter ${
      new Date().getDate() >= new
Date(task.deadline).getDate()
      ? "text-red-500"
      : "text-gray-600"
    }`}>
    {new Date(
      new Date(task.deadline).setDate(
        new Date(task.deadline).getDate() + 1
      )
    )
      .toISOString()
      .slice(0, 10)}
  </p>

  <div
    className={`ml-auto rounded-full ${getPriorityColor(
      task.priority
    )}`}>
    <span
      className={`mx-6 font-semibold text-sm tracking-
tighter ${getPriorityColor(
        task.priority
      )}`}>
      {task.priority}
    </span>
  </div>
</div>

<p className="font-bold text-xs tracking-tighter mt-2 text-gray-500
dark:text-gray-200">
  {countCompletedSubtasks.length} з {subtasks.length} виконано

```

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </p>
    </div>

    {taskModal && (
        <TaskSubtaskModal
            setTaskModal={setTaskModal}
            columns={columns}
            task={task}
            tasks={tasks}
            subtasks={subtasks}
            completedSubtasks={countCompletedSubtasks}
            updateTask={onUpdateTask}
        />
    )}
</div>
);
}

```

Лістинг Header.jsx

```

"use client";

import { useEffect, useState } from "react";
import HeaderDropDown from "../HeaderDropDown";
import AddBoardModal from "@modals/AddBoardModal";
import { useSelector } from "react-redux";
import TaskModal from "@modals/TaskModal";
import BurgerMenu from "../BurgerMenu";
import DeleteSomeModal from "@modals/DeleteSomeModal";
import UserIcon from "../UserIcon";

const headerDown = "/assets/headerDown.svg";
const headerUp = "/assets/headerUp.svg";
const verticalMenu = "/assets/verticalMenu.png";
const logo = "assets/logo.png"

export default function Header({ boardModal, setBoardModal, allUpdate }) {
    const updater = useSelector((state) => state.updater.updater);
    const token = localStorage.getItem("token");

    // Modal windows
    const [dropDown, setDropDown] = useState(false);
    const [openTaskModal, setOpenTaskModal] = useState(false);
    const [burgerMenuOpen, setBurgerMenuOpen] = useState(false);
    const [deleteModal, setDeleteModal] = useState(false);

    const [functionType, setFunctionType] = useState("create");

    function headerUpdate() {
        allUpdate();
    }
}

```

					ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    function isEditModalOpen() {
        setBoardModal((prev) => !prev);
        setBurgerMenuOpen((prev) => !prev);
    }

    function setDeleteModalOpen() {
        setDeleteModal((prev) => !prev);
        setBurgerMenuOpen((prev) => !prev);
    }

    function onDelete() {
        setDeleteModal((prev) => !prev);
        allUpdate();
        fetchBoards();
    }

    const activeBoard = useSelector((state) => state.activeBoard.board);

    // Fetch boards
    const [boards, setBoards] = useState([]);
    const [userData, setUserData] = useState([]);

    function handleAddBoard() {
        fetchBoards();
    }

    useEffect(() => {
        fetchBoards();
        fetchUser();
    }, [updater]);

    const fetchBoards = async () => {
        const resp = await fetch(`/api/boards?token=${token}`);
        const res = await resp.json();
        if (res.success) {
            setBoards(res.data);
        }
    };

    const fetchUser = async () => {
        const resp = await fetch(`/api/users?token=${token}`);
        const res = await resp.json();
        if (res.success) {
            setUserData(res.data);
        }
    };
};

```

```

return (
  <div className="p-4 fixed left-0 bg-white dark:bg-[#343542] z-50 right-0">
    <header className="flex justify-between dark:text-white items-center">
      {/* HeaderLeftSide */}
      <div className="flex items-center space-x-2 md:space-x-4">
        <img src={logo} alt="logo" className="h-12 w-12"></img>
        {/* ProjectName */}
        <h3 className="hidden md:inline-block font-bold font-sans
md:text-4xl">
          KanBunny
        </h3>

        <div className="flex items-center">
          <img
            onClick={() => {
              setFunctionType("edit");
              setBurgerMenuOpen(true);
              setDropDown(false);
              setOpenTaskModal(false);
            }}
            src={verticalMenu}
            alt="verticalMobile"
            className="cursor-pointer h-6 md:ml-8"
          />
          {burgerMenuOpen && (
            <BurgerMenu
              type={"дошки"}
              setEditModal={isEditModalOpen}
              setDeleteModal={setDeleteModalOpen}
              setBurgerMenuOpen={setBurgerMenuOpen}
            />
          )}
          <h3 className="truncate max-w-[300px] md:text-2xl text-xl
md:ml-4 font-bold font-sans">
            {activeBoard.title}
          </h3>
          <img
            onClick={() => {
              setFunctionType("create");
              setDropDown((state) => !state);
            }}
            src={dropDown ? headerUp : headerDown}
            alt="dropdown"
            className="w-3 ml-2 md:hidden cursor-pointer"
          />
        </div>
      </div>
      {/* HeaderRightSide */}

      <div className="flex space-x-4 items-center md:space-x-6">

```

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <button
            className="button hidden md:block bg-blue-500"
            onClick={() => {
                setOpenTaskModal((prev) => !prev);
            }}>
            + Додати нове завдання
        </button>
        <button
            className="button py-1 px-3 md:hidden"
            onClick={() => {
                setOpenTaskModal((prev) => !prev);
            }}>
            +
        </button>
        <UserIcon user={userData} />

    </div>
</header>
{dropDown && (
    <HeaderDropDown
        setBoardModal={setBoardModal}
        setDropDown={setDropDown}
        boards={boards}
        headerUpdate={headerUpdate}
    />
)}
{boardModal && (
    <AddBoardModal
        type={functionType}
        setBoardModal={setBoardModal}
        onAdd={handleAddBoard}
    />
)}
{openTaskModal && (
    <TaskModal
        type="create"
        device="mobile"
        setOpenTaskModal={setOpenTaskModal}
        allUpdate={allUpdate}
    />
)}
{deleteModal && (
    <DeleteSomeModal
        type="board"
        title={activeBoard.title}
        setDeleteOpen={setDeleteModal}
        onDelete={onDelete}
    />
)}
</div>

```

					ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    );
}

```

Лістинг Column.jsx

```

import { useEffect, useState } from "react";
import Task from "../Task";
import { useDispatch, useSelector } from "react-redux";
import { updaterActions } from "@store/updater";

export default function Column({ column, columns, updateCenter, activeBoard, index
}) {
    const updater = useSelector((state) => state.updater.updater);
    const dispatch = useDispatch();

    const [tasks, setTasks] = useState([]);

    useEffect(() => {
        fetchTasks();
    }, [activeBoard, column, updater]);

    const fetchTasks = async () => {
        const response = await fetch(`/api/tasks?id_columns=${column.id}`);
        const data = await response.json();
        if (data.success) {
            setTasks(data.data.sort((a, b) => a.index_task - b.index_task));
        }
    };

    function handleDragTask(event) {
        event.preventDefault();
    }

    async function handleOnDrop(event, dropIndex, dropTaskId) {
        const { prevColIndex, taskId, index } = JSON.parse(
            event.dataTransfer.getData("text")
        );

        if (column.id !== prevColIndex) {
            let max = 0;

            for(const task of tasks) {
                task.index_task > max && (max = task.index_task);
            }

            const updateData = { columnId: column.id, taskId: taskId, index_task:
max + 1};

```

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const updateTaskResponse = await fetch(`/api/tasks`, {
  method: "PUT",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify(updateData),
});
const updateTaskResult = await updateTaskResponse.json();
if (!updateTaskResult.success) {
  console.error(updateTaskResult.data);
  return;
}
//dispatch(updaterActions.setUpdater());
updateCenter();
} else if (dropTaskId) {

  const updateFirstData = {taskId, dropIndex};

  const updateFirstTaskResponse = await fetch(`/api/tasks`, {
    method: "PUT",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(updateFirstData),
  });
  const updateFirstTaskResult = await updateFirstTaskResponse.json();
  if (!updateFirstTaskResult.success) {
    console.error(updateFirstTaskResult.data);
    return;
  }

  const updateSecondData = {dropTaskId, index};

  const updateSecondTaskResponse = await fetch(`/api/tasks`, {
    method: "PUT",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify(updateSecondData),
  });
  const updateSecondTaskResult = await updateSecondTaskResponse.json();
  if (!updateSecondTaskResult.success) {
    console.error(updateSecondTaskResult.data);
    return;
  }
  updateCenter();
}
}

```

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    return (
      <div
        onDrop={handleOnDrop}
        onDragOver={handleDragTask}
        className="scrollbar-hide mx-6 pt-[85px] min-w-[270px]">
        <p className="font-semibold flex items-center gap-2 tracking-wide
md:tracking-wide text-gray-800 dark:text-gray-300">
          {column.name} ({tasks.length})
        </p>
        {tasks.length > 0 &&
          tasks.map((task) => (
            <div
              key={task.id}
              onDrop={(e) => handleOnDrop(e, task.index_task, task.id)}>
              <Task
                key={task.id}
                task={task}
                tasks={tasks}
                columns={columns}
                updateCenter={updateCenter}
                activeBoard={activeBoard}
                updater={updater}
                column={column}
                index={task.index_task}
              />
            </div>
          )))
      </div>
    );
  }

```

Лістинг AddBoardModal.jsx

```

import { useEffect, useState } from "react";
import { useDispatch, useSelector } from "react-redux";
import { activeBoardActions } from "@store/activeBoard";
import { v4 } from "uuid";
import { DragDropContext, Droppable, Draggable } from 'react-beautiful-dnd';

const iconCancel = "assets/cancel.png";

export default function NewBoardModal({ setBoardModal, type, onAdd }) {
  const token = localStorage.getItem("token");
  const currentBoard = useSelector((state) => state.activeBoard.board);
  const dispatch = useDispatch();

  const [editedColumns, setEditedColumns] = useState([]);

```

```

const [brandNewColumns, setBrandNewColumns] = useState([
  { name: "", id: v4() },
  { name: "", id: v4() },
]);

function handleDelete(id) {
  const filteredColumns = brandNewColumns.filter(
    (element) => element.id !== id
  );
  setBrandNewColumns(filteredColumns);
}

if (type === "edit") {
  async function fetchColumns() {
    const response = await
fetch(`/api/columns?board_id=${currentBoard.id}`);
    const result = await response.json();
    if (!result.success) {
      console.error(result.data);
    }
    setBrandNewColumns(result.data);
    setEditedColumns(result.data);
  }

  useEffect(() => {
    fetchColumns();
  }, []);
}

async function handleSubmit(e) {
  e.preventDefault();
  setBoardModal(false);

  const editedFiltered = brandNewColumns
    .filter((item1) => editedColumns.some((item2) => item2.id ===
item1.id))
    .map((item) => item);

  const removedIds = editedColumns
    .map((item) => item.id)
    .filter((id) => !editedFiltered.map((item) => item.id).includes(id));

  const newColumnsFiltered = brandNewColumns
    .filter((item2) => !editedColumns.some((item1) => item1.id ===
item2.id))
    .map((item) => item.name);

  const formData = new FormData(e.target);

  const data = { boardName: null, columns: [] };

```

```

for (let [key, value] of formData.entries()) {
  if (key === "boardName") {
    data.boardName = value;
  } else {
    data.columns.push(value);
  }
}

if (type === "create") {
  const createBoardResponse = await fetch(`/api/boards`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({ boardName: data.boardName, token }),
  });
  const createBoardResult = await createBoardResponse.json();
  if (!createBoardResult.success) {
    console.error(createBoardResult.data);
  }

  const createColumnsResponse = await fetch(`/api/columns`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      boardId: createBoardResult.data,
      columns: data.columns,
    }),
  });
  const createColumnsResult = await createColumnsResponse.json();
  if (!createColumnsResult.success) {
    console.error(createColumnsResult.data);
  }

  onAdd();
}

if (type === "edit") {
  const updateBoardResponse = await fetch(`/api/boards`, {
    method: "PUT",
    headers: {
      "Content-Type": "application/json",
    },
    body: JSON.stringify({
      boardName: data.boardName,
      boardId: currentBoard.id,
    }),
  });

```

```

});
const updateBoardResult = await updateBoardResponse.json();
if (!updateBoardResult.success) {
    console.error(updateBoardResult.data);
}

dispatch(
    activeBoardActions.setActiveBoard({
        title: data.boardName,
        id: currentBoard.id,
    })
);

const updateColumnsResponse = await fetch(`/api/columns`, {
    method: "PUT",
    headers: {
        "Content-Type": "application/json",
    },
    body: JSON.stringify({
        columns: editedFiltered,
    }),
});
const updateColumnsResult = await updateColumnsResponse.json();
if (!updateColumnsResult.success) {
    console.error(updateColumnsResult.data);
}

if (newColumnsFiltered.length > 0) {
    const createColumnsResponse = await fetch(`/api/columns`, {
        method: "POST",
        headers: {
            "Content-Type": "application/json",
        },
        body: JSON.stringify({
            boardId: currentBoard.id,
            columns: newColumnsFiltered,
        }),
    });
    const createColumnsResult = await createColumnsResponse.json();
    if (!createColumnsResult.success) {
        console.error(createColumnsResult.data);
    }
}

if (removedIds.length > 0) {
    for (const removedId of removedIds) {
        const response = await fetch(`/api/columns?id=${removedId}`, {
            method: "DELETE",
        });
        const result = await response.json();
    }
}

```

					ІІЗ.КР.Б – 121 – 24 – ІІЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (!result.success) {
            console.error(result.data);
        }
    }
}

onAdd();
}

function handleColumnChange(index, newValue) {
    const updatedColumns = [...brandNewColumns];
    updatedColumns[index].name = newValue;
    setBrandNewColumns(updatedColumns);
}

return (
    <div
        onClick={
            (event) => {
                if (event.target === event.currentTarget) {
                    setBoardModal(false);
                }
            }
        }
        className={
            `fixed right-0 left-0 top-0 bottom-0 px-2 scrollbar-hide py-4 overflow-scroll
            z-50 justify-center items-center flex bg-black ${
                type === "edit" ? "bg-opacity-60" : "bg-opacity-80"
            }`
        }
        <form
            onSubmit={handleSubmit}
            className="scrollbar-hide overflow-y-scroll max-h-[95vh] bg-white
            dark:bg-gray-900
            text-black dark:text-gray-50 font-bold shadow-md shadow-[#364e7e1a]
            outline-none max-w-md mx-auto w-full px-8 py-8 rounded-xl">
                <h3 className="text-lg">
                    {type === "edit" ? "Редагувати дошку" : "Створити дошку"}
                </h3>

                <div className="mt-8 flex flex-col space-y-3">
                    <label className="text-sm dark:text-gray-50 text-gray-900">
                        Назва дошки
                    </label>
                    <input
                        required
                        name="boardName"
                        className="bg-transparent px-4 py-2 rounded-md text-sm
                        border
                        border-gray-600 focus:outline-blue-500 outline-1 ring-0"
                        placeholder="Курсова робота"
                        id="board-name-input"

```

```

        defaultValue={type === "edit" ? currentBoard.title : null}
      />
    </div>

    <div className="mt-8 flex flex-col space-y-3">
      <label className="text-sm dark:text-gray-50 text-gray-900">
        Колонки дошки
      </label>

      {brandNewColumns.map((column, index) => (
        <div key={column.id} className="flex items-center w-full">
          <input
            placeholder="Нова колонка"
            required
            name={`columnName${index}`}
            className="bg-transparent flex-grow px-4 py-2
rounded-md text-sm border
border-gray-600 outline-none focus:outline-
blue-500"
            type="text"
            value={column.name}
            onChange={(e) => handleColumnChange(index,
e.target.value)}
          />
          <img
            src={iconCancel}
            className=" cursor-pointer m-4 h-6 w-6"
            onClick={() => {
              handleDelete(column.id);
            }}
          />
        </div>
      )))}
    </div>

    <div>
      <button
        type="button"
        onClick={() => {
          const newColumn = { name: "", id: v4() };
          setBrandNewColumns((prev) => [...prev, newColumn]);
        }}
        className=" w-full items-center hover:opacity-80 dark:text-
blue-500
dark:bg-gray-50 text-gray-50 bg-blue-500 py-2 rounded-full mt-
2">
        + Додати нову колонку
      </button>
    </div>

```

```

        <button
            className=" w-full items-center hover:opacity-70 dark:text-
gray-50
            dark:bg-blue-500 mt-8 relative text-gray-50 bg-blue-500 py-2
rounded-full "
            type="submit">
            {type === "create" ? "Створити дошку" : "Зберегти зміни"}
        </button>
    </div>
</form>
</div>
);
}

```

Лістинг TaskModal.jsx

```

import { updaterActions } from "@store/updater";
import { useEffect, useState } from "react";
import { useDispatch, useSelector } from "react-redux";
import { v4 } from "uuid";

const iconCancel = "assets/cancel.png";

export default function TaskModal({
    type,
    device,
    setOpenTaskModal,
    task,
    subtasksEdit,
    allUpdate,
}) {
    const [taskName, setTaskName] = useState(
        type === "create" ? null : task.title
    );
    const [description, setDescription] = useState(
        type === "create" ? null : task.description
    );
    const [columns, setColumns] = useState([]);

    const activeBoard = useSelector((state) => state.activeBoard.board);
    const dispatch = useDispatch();
    // console.log('activeBoard', activeBoard)

    const [subtasks, setSubtasks] = useState(
        type === "create"
        ? [
            { title: "", isCompleted: false, id: v4() },
            { title: "", isCompleted: false, id: v4() },

```

```

        ]
        : subtasksEdit
    );

function handleTaskChange(index, newValue) {
    const updatedTask = [...subtasks];
    updatedTask[index].title = newValue;
    setSubtasks(updatedTask);
}

function handleDelete(id) {
    const filtered = subtasks.filter((element) => element.id !== id);
    setSubtasks(filtered);
}

async function handleSubmit(event) {
    event.preventDefault();
    const formData = new FormData(event.target);

    const data = {
        taskId: null,
        taskTitle: null,
        taskDescription: null,
        columnId: null,
        subtasks: [],
    };

    if (type === "create") {
        for (let [key, value] of formData.entries()) {
            if (key === "columnId") {
                data.columnId = +value;
            } else if (key === "taskTitle") {
                data.taskTitle = value;
            } else if (key === "taskDescription") {
                data.taskDescription = value;
            } else if (key === "priority") {
                data.priority = value;
            } else if (key === "deadline") {
                data.deadline = value;
            } else {
                data.subtasks.push(value);
            }
        }
        console.log(data)
        //Створення задачі
        const createTasksResponse = await fetch(`/api/tasks`, {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({
                columnId: data.columnId,

```

```

        taskTitle: data.taskTitle,
        taskDescription: data.taskDescription,
        priority: data.priority,
        deadline: data.deadline
    })),
    });
const createTaskResult = await createTasksResponse.json();
if (!createTaskResult.success) {
    console.log(createTaskResult.data);
    return;
}

//Створення підзавдання
const createSubtasksResponse = await fetch(`/api/subtasks`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({
        taskId: createTaskResult.data,
        subtasks: data.subtasks,
    }),
});
const createSubtaskResult = await createSubtasksResponse.json();
if (!createSubtaskResult.success) {
    console.log(createSubtaskResult.data);
    return;
}
dispatch(updaterActions.setUpdater());
setOpenTaskModal();
}

if (type === "edit") {
    //Фільтрований масив для редагування підзавдань
    const editFiltered = subtasks
        .filter((item1) => subtasksEdit.some((item2) => item2.id ===
item1.id))
        .map((item) => item);

    //Фільтрований масив для видалення підзавдань
    const removedIds = subtasksEdit
        .map((item) => item.id)
        .filter((id) => !editFiltered.map((item) => item.id).includes(id));

    //Фільтрований масив для додавання підзавдання
    const newSubtasksFiltered = subtasks
        .filter((item2) => !subtasksEdit.some((item1) => item1.id ===
item2.id))
        .map((item) => item.title);

    data.taskId = task.id;
    for (let [key, value] of formData.entries()) {

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (key === "taskTitle") {
            data.taskTitle = value;
        } else if (key === "taskDescription") {
            data.taskDescription = value;
        } else if (key === "priority") {
            data.priority = value;
        } else if (key === "deadline") {
            data.deadline = value;
        } else {
            data.subtasks.push(value);
        }
    }

    //Оновлення опису та назви завдання
    const editTaskResponse = await fetch(`/api/tasks`, {
        method: "PUT",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify({
            taskId: data.taskId,
            taskTitle: data.taskTitle,
            taskDescription: data.taskDescription,
            priority: data.priority,
            deadline: data.deadline
        })),
    });
    const editTaskResult = await editTaskResponse.json();
    if (!editTaskResult.success) {
        console.log(editTaskResult.data);
    }

    const updateSubtasksResponse = await fetch(`/api/subtasks`, {
        method: "PUT",
        headers: {
            "Content-Type": "application/json",
        },
        body: JSON.stringify({
            subtasks: editFiltered,
        })),
    });
    const updateSubtasksResult = await updateSubtasksResponse.json();
    if (!updateSubtasksResult.success) {
        console.error(updateSubtasksResult.data);
    }

    //Якщо було створено нові підзавдання
    if (newSubtasksFiltered.length > 0) {
        const createSubtasksResponse = await fetch(`/api/subtasks`, {
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        taskId: task.id,
        subtasks: newSubtasksFiltered,
    })),
    });
    const createSubtaskResult = await createSubtasksResponse.json();
    if (!createSubtaskResult.success) {
        console.log(createSubtaskResult.data);
    }
}

//Видалення підзавдань
if (removedIds.length > 0) {
    for (const removedId of removedIds) {
        const response = await fetch(`/api/subtasks?id=${removedId}`, {
            method: "DELETE",
        });
        const result = await response.json();
        if (!result.success) {
            console.error(result.data);
        }
    }
}

dispatch(updaterActions.setUpdater());
setOpenTaskModal();
}
}

async function fetchColumns() {
    const response = await fetch(`/api/columns?board_id=${activeBoard.id}`);
    const result = await response.json();
    if (!result.success) {
        console.error(result.data);
    }
    setColumns(result.data);
}

useEffect(() => {
    fetchColumns();
}, []);

return (
    <form
        onSubmit={handleSubmit}
        onClick={event => {
            if (event.target === event.currentTarget) {
                setOpenTaskModal(false);
            }
        }}
        className={

```

```

        device === "mobile"
        ? `py-6 px-6 pb-40 absolute overflow-y-scroll left-0 flex
right-0 bottom-[-100vh] top-0
        bg-black ${type === "edit" ? "bg-opacity-40" : "bg-opacity-
80"}`
        : `py-6 px-6 pb-40 absolute overflow-y-scroll left-0 flex
right-0 bottom-0 top-0
        bg-black ${type === "edit" ? "bg-opacity-40" : "bg-opacity-
80"}`
    }>
    <div
        className=" scrollbar-hide overflow-y-scroll max-h-[95vh] my-auto
bg-white
        dark:bg-[#2b2c37] text-gray-950 dark:text-gray-50 font-bold shadow-md
shadow-[rgba(54,61,126,0.12)]
        max-w-md mx-auto w-full px-8 py-8 rounded-xl">
        <h3 className=" text-lg">
            {type === "edit" ? "Редагувати завдання" : "Додати завдання"}
        </h3>

        <div className=" mt-8 flex flex-col space-y-1">
            <label className=" text-sm dark:text-gray-50 text-black">
                Завдання
            </label>
            <input
                name="taskTitle"
                required
                value={taskName ? taskName : undefined}
                onChange={(event) => setTaskName(event.target.value)}
                type="text"
                className=" bg-transparent px-4 py-2 outline-none
focus:border-0 rounded-md text-sm border
                border-gray-500 focus:outline-blue-500 ring-0"
                placeholder="Завершити реєстрацію користувача"
            />
        </div>

        <div className=" mt-8 flex flex-col space-y-1">
            <label className=" text-sm dark:text-gray-50 text-
black">Опис</label>
            <textarea
                name="taskDescription"
                required
                value={description ? description : undefined}
                onChange={(event) => setDescription(event.target.value)}
                type="text"
                className=" min-h-[150px] bg-transparent px-4 py-2 outline-
none focus:border-0 rounded-md text-sm border
                border-gray-500 focus:outline-blue-500 ring-0"

```

					ІПЗ.КР.Б – 121 – 24 – ІЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

```
placeholder="Завершити реєстрацію користувача до
19.06.2024"
/>
</div>

<div className=" mt-8 flex flex-col space-y-1">
  <label className=" text-sm dark:text-gray-50 text-black">
    Список підзавдань
  </label>

  {subtasks.map((subtask, index) => (
    <div key={subtask.id} className="flex items-center w-full">
      <input
        required
        name={`subtaskName${index}`}
        className="bg-transparent flex-grow px-4 py-2
rounded-md text-sm border
border-gray-600 outline-none focus:outline-
blue-500"
        type="text"
        value={subtask.title ? subtask.title : undefined}
        onChange={(e) => handleTaskChange(index,
e.target.value)}
        placeholder="Нове підзавдання"
      />
      <img
        src={iconCancel}
        className=" cursor-pointer m-4 h-6 w-6"
        onClick={() => {
          handleDelete(subtask.id);
        }}
      />
    </div>
  )))}
</div>
<div>
  <button
    type="button"
    onClick={() => {
      const newTask = { title: "", isCompleted: false, id:
v4() };

      setSubtasks((prev) => [...prev, newTask]);
    }}
    className=" w-full items-center hover:opacity-80 dark:text-
blue-500
dark:bg-gray-50 text-gray-50 bg-blue-500 py-2 rounded-full mt-
2">
    + Додати нове підзавдання
  </button>
</div>
```

```

<div className=" mt-8 flex flex-col md:flex-row">
  <div className="flex flex-col space-y-1">
    <label
      className="text-sm dark:text-gray-50 text-black">
      Дедлайн:
    </label>
    <input
      required
      defaultValue={
        type === 'edit' && new Date(new
Date(task.deadline).setDate(new Date(task.deadline).getDate() +
1)).toISOString().slice(0, 10)
      }
      name="deadline"
      type="date"
      className="bg-transparent px-4 py-2 outline-none
focus:border-0 rounded-md text-sm border border-gray-500 focus:outline-blue-500
ring-0"
    />
  </div>

  <div className="ml-4 flex-grow">
    <div className="flex flex-col space-y-1">
      <label
        className="text-sm dark:text-gray-50 text-black">
        Пріоритет
      </label>
      <select
        name="priority"
        defaultValue={type === 'edit' && task.priority}
        className="bg-transparent px-4 py-2 outline-none
focus:border-0 rounded-md text-sm border border-gray-500 focus:outline-blue-500
ring-0">

        <option value="Низький">Низький</option>
        <option value="Середній">Середній</option>
        <option value="Високий">Високий</option>
      </select>
    </div>
  </div>
</div>

<div className=" mt-8 flex flex-col space-y-3">
  {type === "create" && (
    <>
      <label className=" text-sm dark:text-gray-50 textgray-
500">

        Поточна колонка
      </label>
      <select

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        name="columnId"
        className=" select-column flex flex-grow px-4 py-2
rounded-md text-sm bg-transparent
        focus:border-0 border border-gray-400 focus:outline-blue-500
outline-none">
        {columns.map((col, index) => (
            <option key={col.id} value={col.id}>
                {col.name}
            </option>
        ))}
    </select>
</>
    })

    <button
        type="submit"
        className=" w-full items-center text-gray-50 bg-blue-500
py-2 rounded-full">
        {type === "edit" ? "Зберегти редагування" : "Створити
завдання"}
    </button>
</div>
</div>
</form>
    );
}

```

Лістинг page.js

```

"use client";

import Center from "@components/Center";
import FirstTimeBoard from "@components/FirstTimeBoard";
import Header from "@components/Header";
import useDark from "@hooks/useDark";
import { verifyJwt } from "@lib/jwt/verifyJwt";
import { activeBoardActions } from "@store/activeBoard";
import { Fragment, useEffect, useState } from "react";
import { useDispatch, useSelector } from "react-redux";

const token = localStorage.getItem("token");

if (typeof window !== "undefined") {
    if (token) {
        verifyJwt(
            { headers: { "access-token": token } },
            () => {
                console.log("LOGGED IN");
            },
            () => {

```

					ІПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        window.location.href = "/register";
    }
    );
} else {
    window.location.href = "/register";
}
}

export default function Home() {
    //useDark();
    //const token = localStorage.getItem("token");

    const [allUpdate, setAllUpdate] = useState(false);

    function onUpdate() {
        setAllUpdate((prev) => !prev);
    }

    const dispatch = useDispatch();
    const [boardModal, setBoardModal] = useState(false);

    const [boards, setBoards] = useState([]);
    const [success, setSuccess] = useState(false);

    const activeBoard = useSelector((state) => state.activeBoard.board);

    useEffect(() => {
        fetchBoards();
    }, [allUpdate]);

    const fetchBoards = async () => {
        const resp = await fetch(`/api/boards?token=${token}`);
        const res = await resp.json();
        if (res.success) {
            setBoards(res.data);
            setSuccess(true);
            if (res.data.length > 0 && !activeBoard.id) {
                dispatch(activeBoardActions.setActiveBoard(res.data[0]));
            }
            return;
        }
        setSuccess(false);
        return;
    };

    return (
        <>
        <div className=" overflow-hidden overflow-x-scroll">
            <div>
                <h1></h1>

```

					ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </div>
        {boards.length > 0 ? (
            <Fragment>
                <Header
                    boardModal={boardModal}
                    setBoardModal={setBoardModal}
                    allUpdate={onUpdate}
                />
                <Center boardModal={boardModal}
                    setBoardModal={setBoardModal} />
            </Fragment>
        ) : (
            <FirstTimeBoard type="create" fetchBoards={fetchBoards} />
        )}
    </div>
</>
);
}

```

					ПЗ.КР.Б – 121 – 24 – ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		