МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ВЕБЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ПРОЕКТАМИ ЗАСОБАМИ REACT»**

Виконав: студент 2 курсу, групи 8.1213-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
В.І. Скляр
(ініціали та прізвище)

Керівник доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Скляру Владиславу Ігоровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка вебзастосунку для управління проектами засобами React

керівник роботи Панасенко Євген Валерійович, к.ф.-м.н, доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » березня 2024 року № 385-с

2. Строк подання студентом роботи 25.11.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Дослідження предметної області.
2. Проєктування вебзастосунку.
3. Програмна реалізація вебзастосунку.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 07.03.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	18.03.2024	
2.	Збір вихідних даних.	22.04.2024	
3.	Обробка методичних та теоретичних	17.06.2024	
	джерел.		
4.	Розробка першого та другого розділу.	30.08.2024	
5.	Розробка третього розділу.	04.11.2024	
6.	Оформлення та нормоконтроль	20.11.2024	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	13.12.2024	

Студент _____ В.І. Скляр
(підпис) (ініціали та прізвище)

Керівник роботи _____ Є.В. Панасенко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка вебзастосунку для управління проектами засобами React»: 56 с., 25 рис., 11 табл., 23 джерела, 6 додатків.

ВЕБЗАСТОСУНОК, ІНТЕРФЕЙС, РЕАКТ, РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, УПРАВЛІННЯ ПРОЄКТАМИ, ФРОНТ-ЕНД.

Об’єкт дослідження – вебзастосунки для управління проєктами.

Мета роботи: розробка фронт-енд частини вебзастосунку для управління проєктами з використанням бібліотеки React.

Метод дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

В роботі проведений аналіз існуючих систем управління проєктами, обрано та спроєктовано архітектуру вебзастосунку управління проєктами. Реалізовано фронт-енд частину вебзастосунку.

В результаті роботи отримано програмне забезпечення для управління проєктами.

SUMMARY

Master’s qualifying paper “Development of a Web Application for Project Management Using React”: 56 pages, 25 figures, 11 tables, 23 references, 6 supplements.

WEB, APPLICATION, INTERFACE, REACT, SOFTWARE DEVELOPMENT, PROJECT MANAGEMENT, FRONT-END.

The object of the study is web applications for project management.

The aim of the study is development of the front-end part of a web application for project management using the React library.

The method of research are methods for collecting and analyzing software requirements, modeling methods, software design, construction, and testing.

The paper includes an analysis of existing project management systems, the selection and design of a web application architecture for project management, and the implementation of the front-end part of the web application.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 7
1 Дослідження предметної області 10
1.1 Системи управління проєктами 10
1.2 Аналіз існуючих рішень 14
2 Проєктування вебзастосунку 21
2.1 Визначення вимог до вебзастосунку 21
2.2 Архітектура вебзастосунку 24
2.3 Вибір технологій та інструментів реалізації вебзастосунку 30
3 Програмна реалізація вебзастосунку 33
3.1 Визначення основних компонентів React 33
3.2 Інтерфейс користувача 36
3.3 Робота користувача в застосунку (Керівництво користувача) 38
Висновки 43
Перелік посилань 45
Додаток А Компонент App.jsx 48
Додаток Б Компонент Body.jsx 49
Додаток В Компонент Board.jsx 50
Додаток Г Компонент AddBoard.jsx 51
Додаток Д Компонент List.jsx 53
Додаток Е Компонент Card.jsx 56

ВСТУП

Стрімкий розвиток інформаційно-комунікаційних технологій докорінно переформатовує сучасний бізнес-простір. Технології дозволяють розширювати географію ведення бізнесу і стирають обмеження в часі, змінюють логістику, маркетингову діяльність та менеджмент компаній. Дрібний бізнес і приватні підприємці, що своєчасно відчули можливості сучасних технологій для ведення бізнесу, змогли зайняти місця на ринках на рівні з великими корпораціями і колишніми монополістами. Водночас, надаючи інструменти для підвищення ефективності діяльності, технології підсилюють конкуренцію на ринку, чим змушують великий бізнес змінювати стратегію діяльності, а нових гравців – вибудовувати нову.

Реалізація стратегії суб'єкта підприємницької діяльності передбачає вирішення цілого ряду тактичних завдань – окремих проєктів. Таким чином, ефективне управління проєктами є складовою успішної реалізації стратегії, чого і вимагає сучасний світ бізнесу.

Разом з тим, успішна реалізація проєкту обумовлена не лише якісним плануванням і професійністю команди, а й від використання сучасних технологій і інструментів, що дозволяють оптимізувати процеси та підвищити продуктивність команди. Потреба в таких інструментах і стала поштовхом розвитку програмного забезпечення для управління проєктами, що забезпечують зручний доступ до даних проєкту, надають можливість співпраці між членами команди або між різними командами та дозволяють здійснювати моніторингу прогресу в режимі реального часу. Можливості інформаційно-комунікаційних технологій ще більш розширили можливості програмного забезпечення для управління проєктами. Вони дозволили працювати членам проєктної команди віддалено, формувати команди без прив'язки до розташування офісу і т. ін.

Розробники програмного забезпечення для управління проєктами, в умовах жорсткої конкуренції, постійно намагаються удосконалювати свої

продукти, розширюючи їх функціонал, часто надмірно, що викликає складнощі у початківців, робить інтерфейс заплутаним тощо. Разом з тим, використання такого програмного забезпечення вимагає від підприємців додаткових витрат на його придбання і підтримку.

Альтернативою для підприємців є розробка, в т.ч. і з залученням аутсорсингових ресурсів, та використання систем управління проєктами, що забезпечать виконання необхідних функцій. Окрім того, розробка такого продукту, може стати прибутковим SAAS-проєктом для продуктових IT-компаній.

Водночас, при розробці програмного забезпечення, важливим фактором є швидкість розробки. Іншим важливим аспектом розробки є інтерфейс програмного продукту, оскільки кінцевим споживачем є користувач, то і успіх пропонованого програмного рішення значною мірою залежить від якісного і зручного інтерфейсу, за лаштунками якого приховані складні алгоритми бек-енду.

У цьому контексті бібліотека React, зорієнтована на розробку інтерфейсів користувача, відкриває нові можливості для створення динамічних та інтерактивних вебзастосунків. Завдяки своїй компонентній архітектурі, React дозволяє розробникам швидко створювати гнучкі рішення, які легко адаптуються до потреб користувачів.

Таким чином, використання програмного забезпечення для управління проєктами є необхідним для сучасного бізнесу, а швидка розробка такого продукту, дозволить зекономити ресурси, що і обумовлює актуальність дослідження.

Об’єкт дослідження: вебзастосунки для управління проєктами. Це включає в себе аналіз їх функціональних можливостей, архітектурних рішень та впливу на ефективність управління проєктами.

Предмет дослідження: процеси розробки вебзастосунку для управління проєктами за допомогою React, а також методи і технології, що використовуються для забезпечення функціональності, юзабіліті та інтерактивності інтерфейсу.

Мета дослідження: розробка фронт-енд частини вебзастосунку для управління проєктами з використанням бібліотеки React.

Для досягнення поставленої мети необхідно здійснити:

– аналіз джерел щодо теорії управління проєктами;
– аналіз існуючих рішень для визначення вимог до функціоналу застосунку та його архітектури;
– проєктування вебзастосунку для управління проєктами;
– програмна реалізація вебзастосунку з використанням React;
– тестування для перевірки функціональності та юзабіліті розробленого застосунку.

Таким чином, реалізація проєкту сприятиме розвитку навичок у сфері веброзробки і дозволить одержати практичне рішення для управління проєктами, що може бути актуальним як для малих команд, так і для великих організацій.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Системи управління проєктами

Ефективність діяльності суб’єкта підприємництва в значній степені зумовлена наявністю стратегічного управління. Водночас, задля реалізації стратегії формується пул тактичних задач, що, фактично, являють собою окремі проєкти [1].

З огляду на багатогранність семантики слова «проєкт», в рамках даної роботи під проєктом слід розуміти набір завдань, виконання яких спрямоване на досягнення певного набору цілей. Як правило, результатом є новий продукт або послуга (цінність). Проєкт потребує таких ресурсів, як час, робоча сила, матеріали та обладнання [1, 2].

Отже, проєкт має такі основні атрибути:

– ціль (цілі);
– терміни реалізації;
– завдання (набір завдань);
– ресурси (час, робоча сила, засоби реалізації, обладнання);
– обмеження ресурсів [1-3].

Управління проєктами є багатоетапним процесом і кількість етапів може різнитися в залежності від багатьох факторів. Загалом, виокремлюють п’ять основних етапів проєктного управління, розглянемо їх детальніше.

Ініціація – початкова стадія проєкту, стадія трансформації ідеї у контури майбутнього результату. Як правило, етап включає техніко-економічне обґрунтування, узгодження концепції проєкту з цілями організації та потребами стейкхолдерів, визначення мети проєкту, а також обсягу робіт і очікувані результати.

Планування – етап, що передбачає розбиття проєкту на окремі завдання, їх послідовність, визначення виконавців і термінів, а також оцінку ресурсів.

Безумовно, одним з найважливіших завдань цього етапу є розподіл ресурсів, що має гарантувати їх доступність протягом терміну реалізації проєкту. Також, у фазі планування здійснюється і управління ризиками – визначення потенційних загроз, насамперед, відсутності ресурсів в необхідній кількості та/або у потрібний час.

Виконання – на цьому етапі розподіляються завдання та налаштовуються комунікації між членами проєктної команди, впроваджуються функції контролю і забезпечення якості виконання завдань.

Моніторинг – цей етап передбачає відстеження відповідності плану процесу виконання проєкту, виявлення проблем і відхилень, своєчасне реагування та реалізація коригувальних заходів задля їх усунення.

Закриття проєкту – на цьому етапі завершуються всі роботи в проєкті як у разі успішної реалізації, так і у разі прийняття рішення про припинення його реалізації з різних причин. В будь-якому разі на цьому етапі проводиться аналіз результатів, звіт перед замовниками тощо. Маркерами закриття проєкту можуть виступати, наприклад, досягнення встановлених цілей; введення в дію і початок експлуатації та/або використання результатів виконання проєкту; припинення фінансування та переведення персоналу на іншу роботу; внесення змін у проєкт, що не були передбачені початковим задумом (модернізація) тощо [4].

Відповідно до класифікації проєктів, наведеній в [1, 5], за складністю їх поділяють на монопроєкти, мультипроєкти та мегапроєкти. В свою чергу, мультипроєкт може бути представлений, як пул монопроєктів, а мегапроєкт, відповідно, як пул монопроєктів і мультипроєктів (див. рис. 1) [5]. Таким чином, незалежно від складності проєкту, управління може бути зведене до управління монопроєктами та, за необхідності, їх синхронізацією та налаштуванням комунікації.

Для успішної реалізації проєкту важливе значення має його менеджмент [3]. Очевидно, що управління проєктами включає такі складові, як планування та організація наявних ресурсів задля досягнення мети проєкту.

Це може бути одноразовий проєкт або постійна діяльність, а складовими ресурсів є персонал, фінанси, технології та інтелектуальна власність.



Рисунок 1 – Класифікації проєктів за складністю

Управління проєктами є актуальним для будь-яких галузей економічної діяльності. Наприклад, менеджери проєктів, незалежно від галузі, як правило, визначають цілі та завдання проєкту, визначають терміни реалізації компонентів проєкту та здійснюють контроль їх таймлайну, здійснюють контроль якості завершених компонентів тощо [5]. Тому, не дивлячись на те, що такий розділ в теорії управління з’явився зовсім недавно, вже напрацьовані і широко використовуються в практичній діяльності стандартні і уніфіковані принципи, правила, умови та ін. [6], тобто, методологія.

Методологія являє собою набір принципів, правил, умов, дій, дотримання яких забезпечує якісну реалізацію проєкту. Методологія, як правило, визначає етапи робочих процесів, форму контролю виконання задач та комунікації між членами команди проєкту, формат прийняття рішень тощо. Вибір вірної методології має суттєвий вплив на ефективність роботи проєктної команди та досягнення поставлених цілей [7]. Застосування обраної методології в управлінні проєктами забезпечує:

– чіткій розподіл ролей і обов’язків, розуміння процесів всіма членами команди;
– чітке визначення обсягу ресурсів та прогнозування кінцевого результату ще на старті проєкту;
– підвищення продуктивності членів команди;

- швидке вирішення робочих питань;
- професійне зростання членів команди;
- покращення загальної атмосфери в команді та згуртування співробітників;
- відсутність невизначеності, оскільки кожен співробітник чітко знає зону відповідальності кожного члена команди;
- уникнення помилок, оскільки методології містять найкращі практики і досвід, що мають приводити до результату [6, 8].

Серед основних методологій, що використовуються у проєктному управлінні, можна виділити Водоспад (Waterfall), Agile, PRINCE2 [7, 9].

Водоспад передбачає поетапне виконання проєкту, коли кожен наступний етап виконується після завершення попереднього. Як правило, ця методологія застосовується для проєктів з чітко визначеними та стабільними вимогами.

Agile – гнучка ітеративна методологія в якій поставлені задачі виконуються короткими циклами. При цьому допускається паралельне виконання задач і доопрацювання вже виконаних задач в наступних циклах. Ця методологія застосовується в проєктах, які можуть змінюватись. Акцент в ній робиться на взаємодію між членами команди і зворотній зв’язок з замовниками. Часто в окремі методології виділяють Scrum і Kanban, які, фактично, являють собою Agile-фреймворки. Scrum є варіантом реалізації Agile з короткими циклами (спринтами), що мають визначений обсяг робіт та терміни виконання. Kanban зорієнтований на оптимізації потоку завдань та візуалізації робочого процесу.

PRINCE2 являє собою структурований підхід з чітким визначенням ролей і процесів. Методологія орієнтована на великі і складні проєкти.

Також можуть використовуватись і гібридні підходи, що передбачають поєднання різних методологій, як правило, для окремих етапів проєкту, наприклад, використання Agile для розробки програмного продукту, а Waterfall для його тестування [7, 9].

Вибір методології залежить від специфіки проєкту, вимог замовника та культури організації. Важливо адаптувати підходи відповідно до змін у процесі реалізації проєкту для досягнення найкращих результатів.

1.2 Аналіз існуючих рішень

Особливе місце в сфері управління проєктами посідають інструменти, що дозволяють автоматизувати процеси управління, насамперед, загальне планування та розподіл задач, планування і контроль використання ресурсів, формування звітності, забезпечення командної комунікації, управління ризиками та контроль якості тощо. Окрім того, такі інструменти мають забезпечувати багатокористувацький інтерфейс і рольовий доступ до системи задля розмежування прав доступу членів проєктної команди. Наприклад, керівник проєкту повинен мати повний доступ до компонента планування, можливість редагування даних, ризиків тощо, натомість, інші члени команди мають бачити лише частини плану, що стосуються виключно їх обов’язків.

На сьогоднішній день ринок програмного забезпечення пропонує широкий спектр інструментарію управління проєктами, через що прийняття правильного рішення при виборі програмного забезпечення є доволі складним, оскільки пошук оптимального інструменту для потреб проєктної команди може забрати багато часу, а результат вибору може розчарувати після початку його використання.

На підставі дослідження понад 2000 інструментів керування проєктами з 2012 року, ресурсом dpm – The Digital Project Manager – сформований список 15 найпопулярніших програмних засобів управління проєктами [10]. Водночас, на ресурсі Forbes Advisor представлено 10 кращих, за їх версією, програмних засобів управління проєктами [11]. Зазначені списки наведені в таблиці 1.

Таблиця 1 – Перелік популярних інструментів керування проєктами за версіями The Digital Project Manager та Forbes Advisor

The Digital Project Manager	**Forbes Advisor**
1. monday.com	1. monday.com
2. Celoxis	2. ClickUp
3. Productive	3. Asana
4. Wrike	4. Zoho Projects
5. Hub Planner	5. Smartsheet
6. ClickUp	6. Wrike
7. Zoho Projects	7. Teamwork.com
8. Visor	8. Notion
9. Quickbase	9. Jira
10. Bonsai Agency Software	10. Trello
11. Workzone	
12. ReptonPOD	
13. Teamhood	
14. Quire	
15. Bubble PPM	

Як можна бачити, в обох списках присутні monday.com, Wrike, ClickUp, Zoho Projects.

В середовищі українських проєктних менеджерів серед популярних інструментів згадуються monday.com, ClickUp, Asana, Jira, Trello, які присутні, принаймні, в одному зі списків вище [2, 13]. Згідно даних сервісу Google Trends по кількості пошукових запитів в Україні за останні 5 років, беззаперечним лідером серед користувачів систем для управління проєктами є Jira, не дивлячись на те, що в одному списку вона посіла передостаннє місце, а в іншому взагалі відсутня. Водночас, monday.com згадується найменше (див. рис. 2) [14].

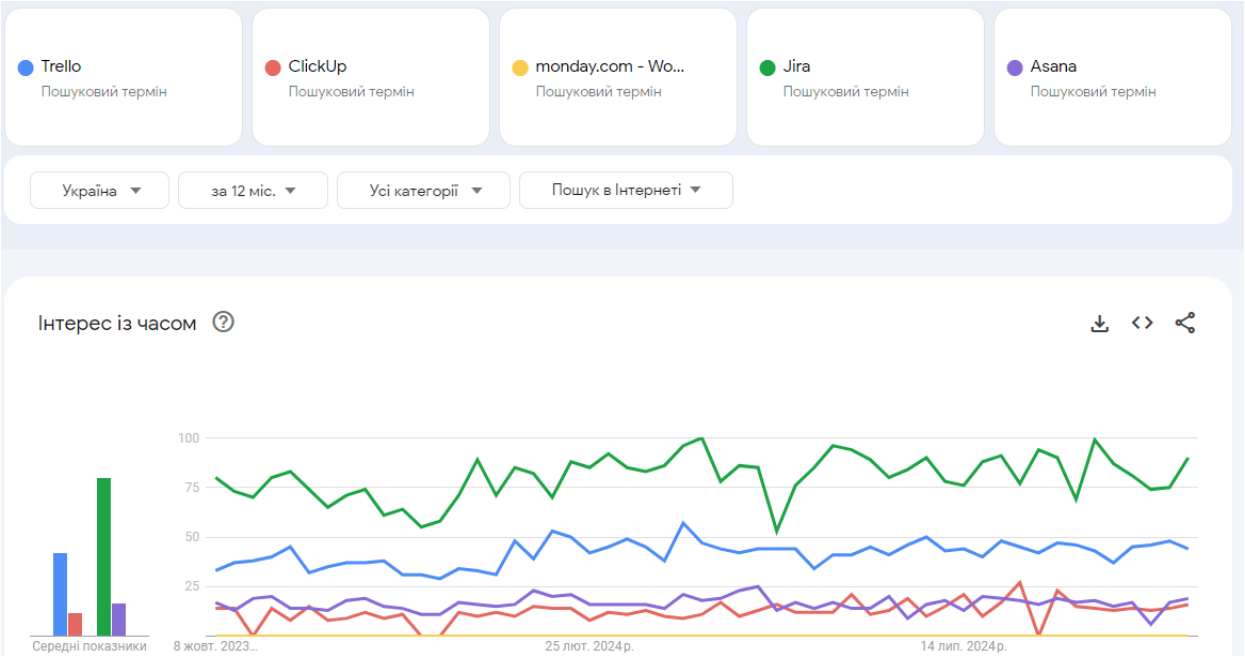


Рисунок 2 – Дані сервісу Google Trends по кількості пошукових запитів в Україні за останні 5 років

Порівняльна характеристика, згаданих вище популярних інструментів управління проєктами, з зазначенням їх вартості, переваг і недоліків, наведена в таблиці 2. Вартість, як правило, зазначають з розрахунку 1 користувача на місяць, хоча існують і інші тарифні плани, але такий підхід дозволяє оцінити проєктному менеджеру перспективні витрати.

Таблиця 2 – Порівняльна характеристика популярних систем управління проєктами

Система	**Безкошт. версія**	**Ціна, $/ користувач/ місяць**	**Переваги**	**Недоліки**
monday.com	пробна версія	8	Легке підключення до інших інструментів через бібліотеку інтеграції. Автоматичні засоби легко створювати та не потребують кодування.	Діаграму Ганта не можна використовувати для створення проєкту з нуля. Деякі перегляди мають проблеми з автоматичним оновленням. Більшість основних

Продовження табл. 2

Система	Безкошт. версія	Ціна, $/ користувач/ місяць	Переваги	Недоліки
monday.com			Можна налаштувати відповідно до будь-якого робочого процесу. Величезний акцент на співпраці.	функцій керування проєктами є тільки в плані Pro.
ClickUp	пробна версія	7	Покращена продуктивність після останнього оновлення до ClickUp 3.0 Низька ціна порівняно з кількістю функцій, які він пропонує. Велика база знань із корисними статтями та посібниками.	AI доступний за додаткову плату, незалежно від тарифного плану. Вимагає значних витрат часу для налаштування платформи. Безкоштовна версія стає доступною лише для читання після досягнення обмежень.
Asana	до 15 користувачів	10,99	Доступний безкоштовний план Вбудовані засоби обміну повідомленнями Управління завданнями міжгрупової співпраці.	Дорожче, ніж у середньому по галузі Може бути занадто складним на початку для користувачів.
Jira	пробна версія	7,50	Доступний безкоштовний план Більшість функцій доступні для всіх рівнів плану Інтегрується з популярними інструментами розробників, такими як GitHub і Figma Керування	Обмежена підтримка планів нижчого рівня Складний гостьовий доступ.

Кінець табл. 2

Система	Безкошт. версія	Ціна, $/ користувач/ місяць	Переваги	Недоліки
Jira			ресурсами за допомогою планування потужності (вищі рівні).	
Trello	до 10 канбан-дошок	5	Зручний і інтуїтивно зрозумілий дизайн Використання перетягування Спрощує керування завданнями та проєктами. Інтегрується з багатьма іншими програмами.	Обмежена безкоштовна версія Масштабованість коштує дорого.

Всі перераховані інструменти мають безкоштовні версії з обмеженнями, що дозволять їх використовувати виключно для ознайомлення або для реалізації невеликих проєктів. Зазначені програмні засоби мають значний обсяг функціоналу, що може задовольнити будь-якого користувача. Натомість, на думку експертів, саме через значний обсяг можливостей, інтерфейс стає заплутаним, що викликає великі проблеми у початківців. Як наслідок, ці системи мають і незручні мобільні версії. До того ж, більшість користувачів взагалі не використовують весь доступний потенціал функціональних можливостей, зосередившись виключно на плануванні, розподілі завдань і встановленні дедлайнів [15].

В таблиці 3 наведені основні можливості систем управління проєктами, а в таблиці 4 – засоби візуалізації робочих процесів.

Зазначені системи мають схожий інтерфейс, що забезпечує створення завдань, встановлення термінів їх виконання, додавання членів команди, тощо. Для порівняння наведені інтерфейси Jira, Trello, monday.com на рисунках 3 – 5 відповідно.

Таблиця 3 – Основні можливості систем управління проєктами

monday.com	ClickUp	Asana	Jira	Trello
Автоматизація робочого процесу; Інтеграція з електронною поштою; Управління грошовими та часовими витратами; Діаграма Ганта; Управління ресурсами; Робота з документами.	Автоматизація робочих процесів; Інтеграція з електронною поштою; Керування фінансовими та часовими ресурсами; Діаграми Ганта для візуалізації проєктів; Управління ресурсами та робота з документами.	Конструктор робочого процесу; Діаграма Ганта; Канбан-дошки; Календар; Інтеграція додатків; Візуалізація етапів роботи.	Планування проєкту; Встановлення етапів; Налаштування робочого процесу; Робота з Канбан-дошками; Управління доступом та правами для користувачів; Аналіз проєкту та звітність.	Повідомлення та сповіщення про дедлайни; Проста організація міток, тегів та коментарів; Контрольний список прогресу; Мобільна оптимізація; Канбан-дошки; Зручний інтерфейс.

Таблиця 4 – Засоби візуалізації робочих процесів

	monday.com	ClickUp	Asana	Jira	Trello
Дашборд	+			+	
Канбан-дошки	+		+	+	+
Діаграма Ганта	+	+	+		
Timeline/Календар	+	+	+	+	
Інтеграція з email	+	+		+	+

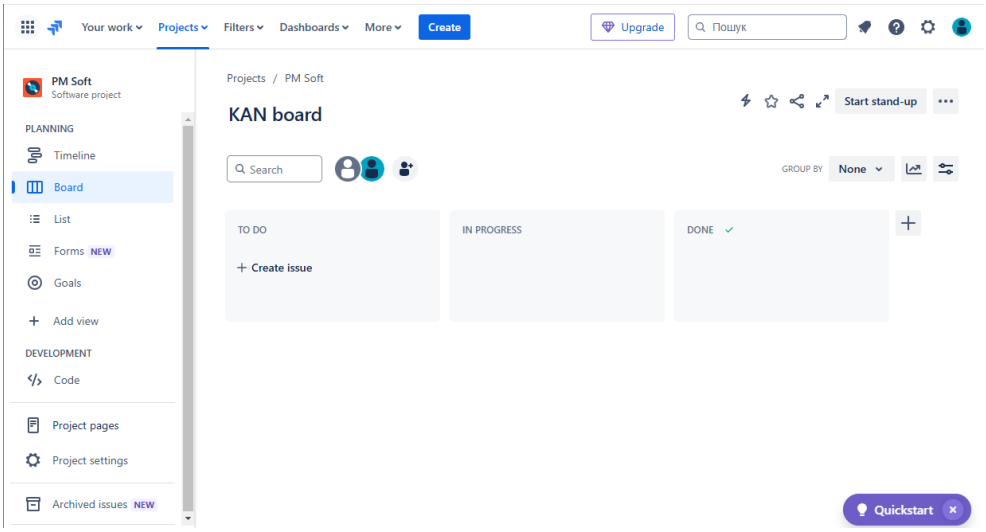


Рисунок 3 – Інтерфейс Jira

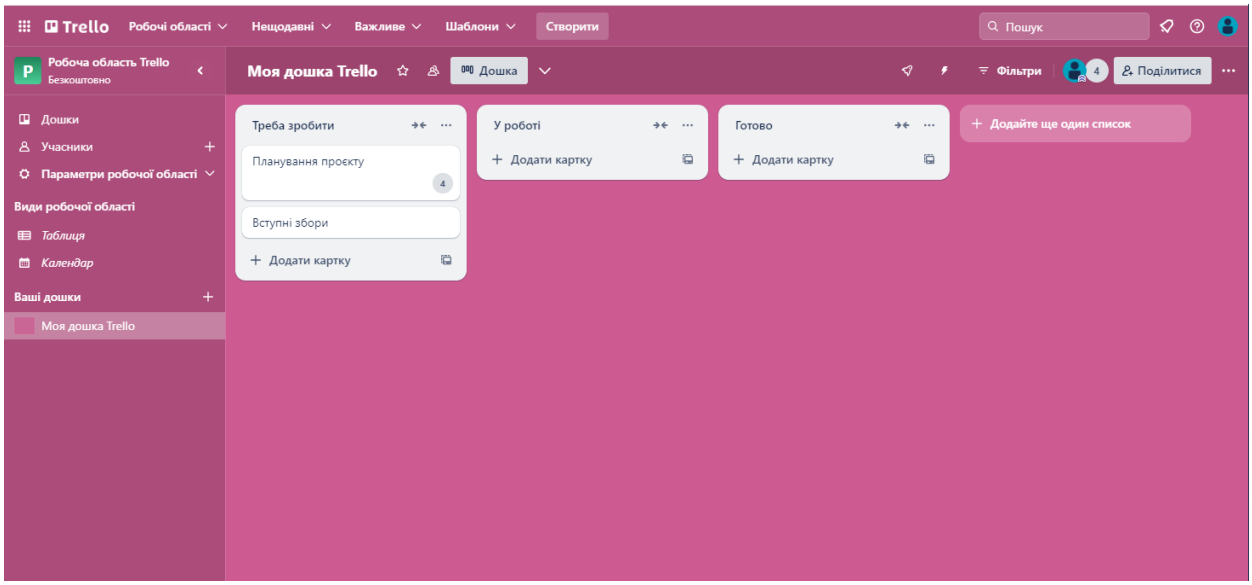


Рисунок 4 – Інтерфейс Trello

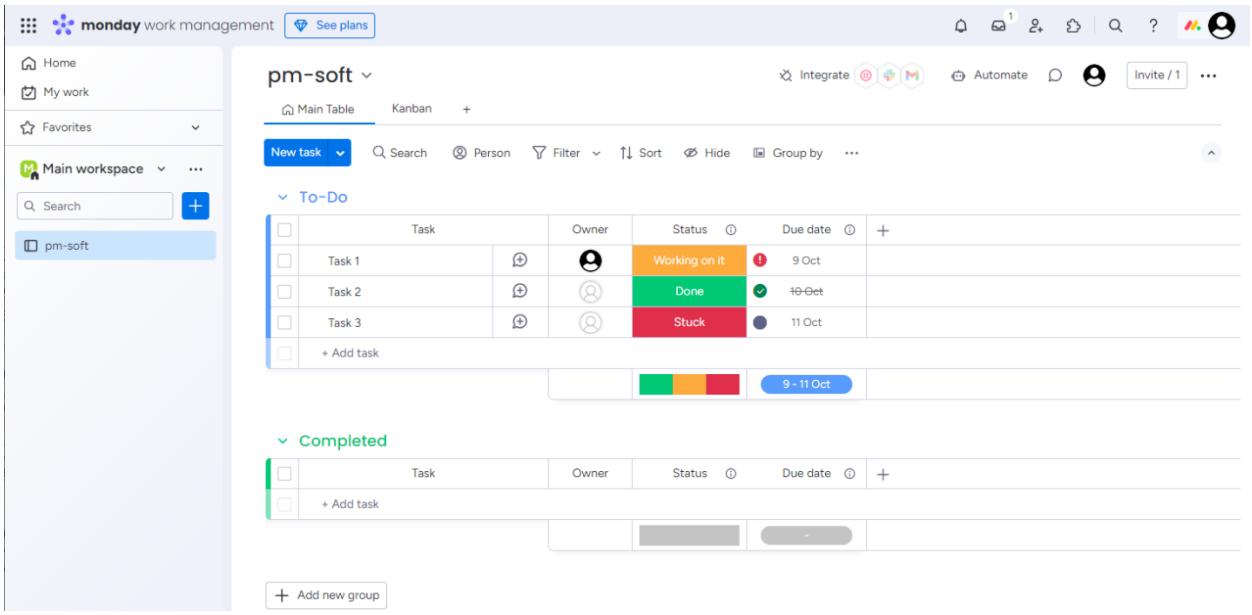


Рисунок 5 – Інтерфейс monday.com

Очевидно, що інтерфейс систем побудований за одним шаблоном: хедер, бокова панель з меню і робоча область (область контенту), що свідчить про оптимальність саме такої структури інтерфейсу для систем управління проєктами.

2 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ

2.1 Визначення вимог до вебзастосунку

Огляд існуючих рішень для управління проєктами дозволяє зробити висновок про мінімально необхідний набір функціоналу для реалізації аналогічного рішення. Так, наприклад, Trello має найменший функціонал, натомість, це не заважає йому впевнено тримати другу позицію за популярністю серед вітчизняних проєктних менеджерів. Водночас, як зазначалося в попередньому розділі, існуючі рішення переобтяжені функціоналом, скоріше за все в результаті жорсткої конкуренції, що призводить до незручності у використанні таких систем, особливо новачками. До того ж, така переобтяженість функціоналом робить мобільні застосунки вкрай незручними.

Таким чином, система управління проєктами має забезпечувати:

– створення і редагування дошок;
– створення і редагування списків;
– створення і редагування карток;
– створення і редагування списку учасників;
– авторизацію користувачів.

Також, слід зауважити, що інший функціонал, який виходить за межі перерахованого, як правило доступний в платних версіях програми. Окрім того, обмеження, перерахованих вище систем, накладаються і на кількість доступних дошок, кількість команд та/або кількість учасників проєкту та/або системи.

Дошка, фактично, являє собою деталізацію проєкту в розрізі задач. Вона дозволяє відстежувати інформацію щодо робочих процесів або команд чи виконавців. За допомогою дошки можна впорядкувати задачі та дрібні деталі проєкту. Основним атрибутом дошки є її назва, що може відповідати назві проєкту, а також менеджер проєкту [16].

На дошці розміщуються списки, що містять картки або завдання різних стадій реалізації проєкту. Здебільшого дошка має три списки:

– зробити (ToDo, To Do);
– в роботі (Doing, InProgress);
– виконано (Done).

На рисунку 6 наведено дошку системи управління проєктами Jira з трьома списками TO DO, IN PROGRESS, DONE. Кнопка «+» дозволяє створити новий список.

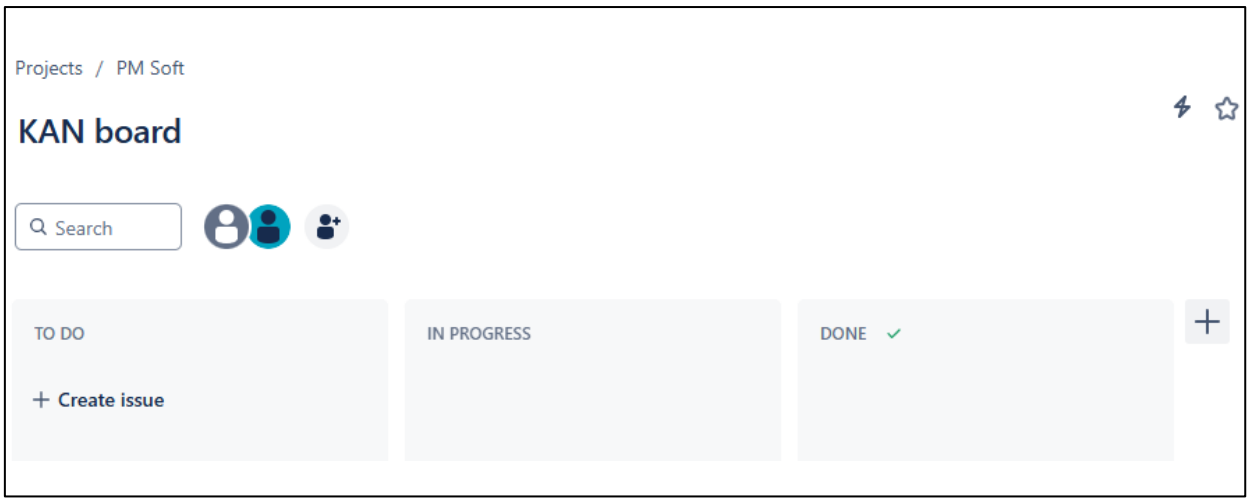


Рисунок 6 – Дошка Jira з трьома списками TO DO, IN PROGRESS, DONE

Водночас, відсутні обмеження на створення нових списків і, за необхідності, можна створювати скільки завгодно нових списків та називати їх як заманеться. Основним атрибутом списку є його назва [16].

Картка, в свою чергу, є найбільш деталізованою одиницею інформації на дошці, що відображає завдання, які необхідно виконати [16].

Основним атрибутом картки є зміст завдання. Оскільки важливим для реалізації проєкту є дотримання дедлайнів та персоналізація відповідальності виконавців, то багато систем дозволяють додавати до карток дедлайни і виконавців, що полегшує менеджеру контроль за перебігом реалізації проєкту. Як правило, картки переміщуються з одного списку в інший простим перетягуванням.

На рисунку 7 наведено картки «Планування проєкту» та «Вступні збори» у списках «Треба зробити» і «У роботі» відповідно на дошці системи управління проєктами Trello.

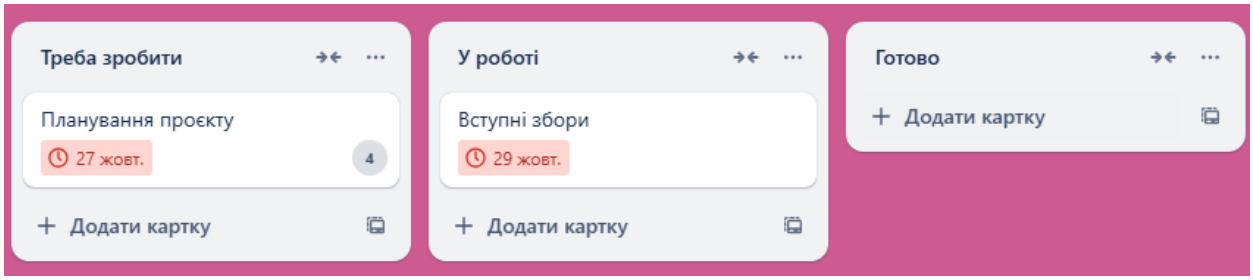


Рисунок 7 – Картки «Планування проєкту» та «Вступні збори» на дошці системи управління проєктами Trello

Визначимо основні вимоги до проєктованої системи.

Створення нової дошки. Одна дошка відповідає одному проєкту. При створенні дошки необхідно задати її назву і менеджера проєкту.

Створення і редагування списків на кожній дошці. Для створення списку достатньо задати його назву.

Створення і редагування карток. Одна картка відповідає одному завданню проєкту. При створенні картки необхідно задати її назву або зміст завдання, відповідального працівника зі списку доданих до системи і дедлайн завершення виконання завдання. Картки поміщаються у відповідних списках і можуть переміщуватись між ними. При наближенні дедлайну, наприклад, за два дні до нього, картка повинна мати окрему позначку.

Створення і редагування списку учасників. Основними атрибутами учасників, що необхідні для забезпечення базового функціоналу застосунку, є ім’я, електронна пошта і пароль. Логічним є розмежування доступу користувачів, у відповідності до їх ролей, до функціоналу застосунку, наприклад, додавати нових користувачів має адміністратор системи. Існують технології, що вже, фактично, набули статусу стандарту в клієнт-серверних додатках, наприклад, RBAC (Role-Based Access Control) або ACF (Access Control Filter), але їх реалізація, з міркувань безпеки і контролю над ресурсами,

здійснюється на серверній частині застосунку, а на клієнтську частину може бути покладена, і то часткова, валідація даних, наприклад, відповідність даних формату електронної пошти, введених користувачем у поле “email” форми.

Таким чином, в роботі буде реалізовано фронт-енд частину додатку, що забезпечує перераховані вище функції та не накладає обмеження на кількість проєктів та кількість учасників.

2.2 Архітектура вебзастосунку

Під архітектурою застосунку, в т.ч. і вебзастосунку, розуміють його структуру та організацію, а саме, з яких компонентів складається застосунок та як вони взаємодіють між собою. Від архітектури залежить як продуктивність, так і зручність обслуговування та можливість масштабування продукту.

До основних компонентів архітектури відносять:

– інтерфейс користувача (UI) – це набір засобів, що забезпечує взаємодію користувача з програмним застосунком;
– бізнес-логіка – реалізація предметної області в застосунку;
– модель даних – опис структури даних, що використовуються в застосунку, в т.ч. сутності та їхні атрибути;
– інфраструктура – включає в себе ресурси, необхідні для роботи програми, наприклад, сервери, бази даних тощо [17 – 19].

На сьогоднішній день виокремлюються різні види архітектур, давайте розглянемо.

Монолітна архітектура – функціонал програмного продукту локалізований в єдиному кодовому сховищі, що суттєво спрощує розгортання, натомість, ускладнює масштабування.

Мікросервісна архітектура – на противагу монолітній архітектурі, розподіляє функціонал на окремі та незалежні сервіси, кожен з яких виконує

конкретну функцію, що дозволяє розгортати їх незалежно один від одного. Як наслідок, просте масштабування та оновлення частин системи.

Серверлесс-архітектура – реалізація функціоналу застосунку здійснюється хмарним програмним забезпеченням [17, 18].

Виходячи з поставлених в роботі завдань, реалізується інтерфейсна частина додатку на мікросервісній архітектурі, що забезпечує взаємодію користувача з застосунком.

Інтерфейс будь-якого застосунку є надзвичайно важливим елементом, оскільки, окрім забезпечення взаємодії користувача з додатком, безпосередньо впливає на його сприйняття і задоволеність від програмного продукту, залишаючи поза увагою складні алгоритми бек-енду та організацію баз даних. Чіткий, інтуїтивно зрозумілий, привабливий та функціональний інтерфейс здатен забезпечити комфортні умови для користувача та швидке вирішення поставлених завдань, і, як наслідок, підвищити продуктивності його роботи та знизити ймовірність помилок [17 – 19].

Окрім того, інтерфейс застосунку асоціюється і з брендом розробника, що, в свою чергу, впливає на довіру та лояльність клієнтів [19, 20].

Також інтерфейс має враховувати такі питання безпеки, як автентифікація та авторизація, задля забезпечення захисту особистої інформації користувача [19, 20].

Відповідно з вимогами до застосунку, сформульованими у попередньому пункті роботи, інтерфейс проєктованого застосунку має забезпечувати реалізацію наступних дій користувача:

– авторизація користувача;
– вихід з облікового запису користувача;
– створення нових дошок;
– перегляд існуючих дошок;
– створення нових списків;
– створення нових карток;
– редагування карток;
– переміщення карток між списками;

– створення списку учасників;

– редагування списку учасників.

Структурна схема інтерфейсу додатку наведена на рисунку 8.



Рисунок 8 – Структура схема інтерфейсу додатку

Структурні елементи інтерфейсу:

– хедер – містить логотип і кнопку авторизації;

– бокова панель – містить меню (4);

– область контенту – містить списки з картками обраної дошки або список учасників;

– меню – містить посилання для переходу до дошки або списку учасників.

Іншою важливою складовою застосунку є модель даних, що являє собою абстракцію об’єктів предметної області та взаємозв’язків між ними.

Взаємодія інтерфейсу і моделі даних – важлива складова застосунку, що забезпечує його функціональність та зручність використання.

Інтерфейс у цій зв’язці має забезпечити виконання наступних функцій.

Динамічне оновлення даних – відображення у реальному часі змін в моделі даних, тобто, будь-які зміни, внесені користувачем, мають бути миттєво відображені у вікні застосунку. Це робить використання застосунку зручним та забезпечує користувача актуальними даними.

Валідація даних. Розповсюдженою практикою є валідація даних, як правило часткова, на стороні клієнта перед їх відправкою до моделі.

Наприклад, до полів форми можна додати правила валідації, що попереджають користувача про неправильний формат введення даних або обов’язковість заповнення полів. Валідація на стороні клієнта дозволяє уникати помилок та забезпечувати цілісність даних у базі, зменшує навантаження на сервер.

Зворотний зв’язок. Через інтерфейс користувач має одержувати чіткий зворотний зв’язок про стан виконання операцій з даними моделі – виникнення помилки, успішна або неуспішна операція тощо.

Як правило, зберігання даних об’єктів моделі реалізується у базі даних.

Основні сутності предметної області – це дошка, список, картка та учасник. Відповідно, задля забезпечення реалізації визначеного кола функціоналу, база даних застосунку має таку структуру:

– таблиця user – містить дані про учасників системи та їх облікові дані для авторизації в системі (див. табл. 5);
– таблиця board – містить дані про дошки (див. табл. 6);
– таблиця list – містить дані про списки (див. табл. 7);
– таблиця card – містить дані про картки (див. табл. 8);

Таблиця 5 – Структура таблиці user

Назва поля	**Призначення**
id	id користувача (ПК – первинний ключ)
username	Логін користувача
email	Email користувача
name	Ім’я користувача
password_hash	Хеш пароля

Таблиця 6 – Структура таблиці board

Назва поля	**Призначення**
id	id дошки (ПК)
user_id	id користувача, що є менеджером проєкту (ЗК – зовнішній ключ)

Продовження табл. 6

Назва поля	**Призначення**
title	Назва дошки
created_at	Час створення запису в таблиці БД

Таблиця 7 – Структура таблиці list

Назва поля	**Призначення**
id	id списку (ПК)
board_id	id дошки, на якій розміщений список (ЗК)
title	Назва списку
position	Положення списку на дошці

Таблиця 8 – Структура таблиці card

Назва поля	**Призначення**
id	id списку (ПК)
list_id	id дошки, на якій розміщений список (ЗК)
title	Назва картки/зміст завдання
user_id	id користувача, що є відповідальним за виконання завдання, розміщеного на картці (ЗК)
deadline	Дата завершення завдання

Зв’язки між таблицями наведено на ER-діаграмі бази даних на рисунку 9. Як правило, фронт-енд одержує дані з бази даних через API. Оскільки в роботі розробляється клієнтська частина додатку, то замість реальної бази даних використовується sessionStorage – об’єкт вебсховища, що дозволяє зберігати дані у браузері для поточної сесії. Дані, збережені в sessionStorage, доступні лише на вкладці браузера, де вони були створені, і видаляються автоматично при закритті цієї вкладки або браузера. Тестовий набір даних було підготовлено в файлі data.json, а завантаження відбувається на початку роботи застосунку. Фрагменти даних з файлу data.json, що

демонструють структуру даних дошок і користувачів, наведені на рисунках 10 і 11 відповідно.

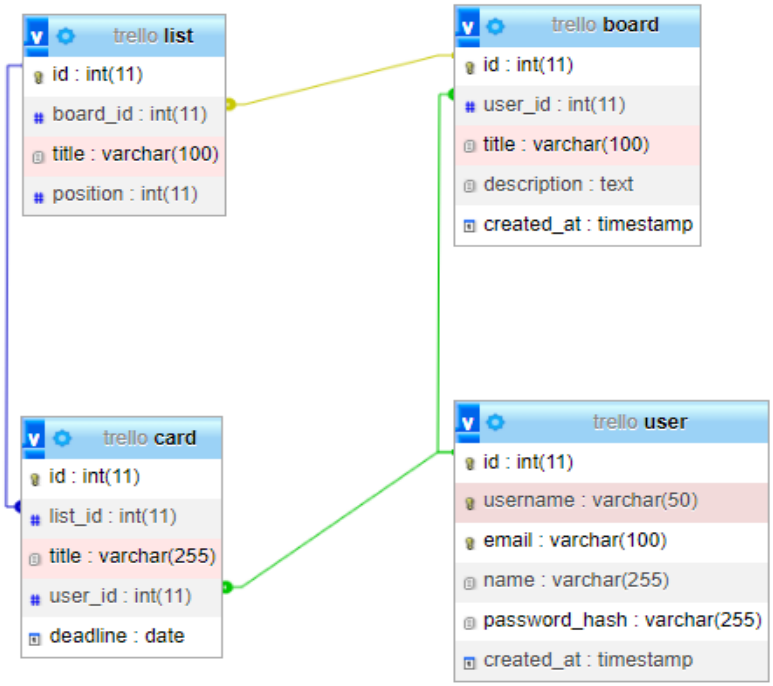


Рисунок 9 – ER-діаграма бази даних застосунку


```
{
  "active": 0,
  "boards": [
    {
      "name": "Маркетинговий план",
      "bgcolor": "#069",
      "manager": {
        "id": "1",
        "name": "Петренко Петро"

      },
      "list": [
        {
          "id": "1",
          "title": "To Do",
          "items": [
            {
              "id": "11",
              "title": "Повідомити клієнтів",
              "deadline": "2024-12-01",
              "assignee": {
                "id": "2",
                "name": "Жук Марина"
              }
            },
            {
              "id": "12",
              "title": "Чернетка сторінки продукту",
              "deadline": "2024-12-01",
              "assignee": {
                "id": "4",
                "name": "Іненко Інна"
```


Рисунок 10 – Фрагмент тестових даних з файлу data.json, що демонструють структуру даних дошок



Рисунок 11 – Фрагмент тестових даних з файлу data.json, що демонструють структуру даних користувачів

2.3 Вибір технологій та інструментів реалізації вебзастосунку

Широке розповсюдження вебдодатків зумовлене низкою їх переваг у порівнянні з використанням інших видів додатків, насамперед, десктопних. Розглянемо основні переваги.

Доступ через мережу. Фактично, для використання вебдодатку, користувачу необхідно мати доступ до мережі, а з програмного забезпечення – браузер. Це робить можливим віддалену роботу незалежно від часу і місця перебування користувача, що особливо актуальним стало в останні роки – період пандемії коронавірусної хвороби і повномасштабного вторгнення військ РФ на територію України.

Відсутність необхідності інсталяції і оновлення програмного забезпечення додатку. Оскільки переважна більшість програмного забезпечення вебзастосунку розміщена на сервері, то і його встановлення та оновлення покладається на розробників або власників ресурсу.

Відсутність серйозних вимог до апаратного забезпечення пристроїв користувачів. Виконання всіх, або, принаймні, переважної більшості, скриптів вебзастосунку здійснюється на сервері, надсилаючи на пристрій користувача лише результати обробки даних для відображення у браузері.

Саме через ці переваги компанії зосереджують увагу на вебрішеннях, щоб підвищити зручність для користувачів та швидкість їх адаптації до швидкозмінюваного бізнес-простору [21].

Окрім того, розробникам програмного забезпечення і продуктовим компаніям слід враховувати, що користувач оцінює додатки за інтерфейсом, насамперед, його зручністю і зрозумілістю [16, 17]. Натомість, основна і найважливіша частина додатку – бекенд – прихована від нього саме за інтерфейсом. Відповідно, розробники мають приділяти значну увагу і розробці засобів взаємодії користувача з застосунком.

Очевидно, що стрімкий розвиток вебтехнологій викликав появу значної кількості фреймворків і бібліотек, що мають на меті створення програмних рішень у відповідності з сучасними стандартами та задля максимального задоволення потреб користувачів [21].

Одним з найпопулярніших на сьогоднішній день інструментів є React. React – це JavaScript-бібліотека, розроблена компанією Facebook, і орієнтована на розробку інтерфейсів користувача [22].

React дозволяє спростити створення інтерфейсу користувача, розбиваючи його на невеликі компоненти, які, до того ж, можуть повторно використовуватись. Можливість опису вигляду різних компонентів у кожному стані додатку робить React гнучким інструментом в цій області. При зміні даних React оновить та відрендерить лише потрібні компоненти. До того ж, декларативні інтерфейси роблять код більш передбачуваним, що, в свою чергу, робить його налагодження більш зручним і простішим.

Розглянемо основні **переваги React**.

Використання концепції Virtual DOM. Використання цієї концепції в React дозволяє зменшити кількість маніпуляцій з реальним DOM і прискорити рендеринг, що суттєво впливає на загальну продуктивність вебзастосунку.

Модульна архітектура. Підтримка модульної архітектури в React надає розробникам можливість створювати незалежні компоненти з можливістю повторного використання, що суттєво спрощує керування кодом.

Використання JSX-синтаксису. JSX-синтаксис поєднує JavaScript та HTML, що робить код легшим для сприйняття і більш зрозумілим.

Велика спільнота. React – проєкт з великою спільнотою, яка надає розробникам доступ до численних ресурсів, бібліотек та інструментів, як наслідок, процес розробки стає більш ефективним.

Простота інтеграції. Інтерфейси, розроблені на React, легко налаштовуються на взаємодію з будь-яким бек-ендом [22, 23].

Вибір React для розробки клієнтської частини вебзастосунку ґрунтується на його продуктивності, модульності та гнучкості, що робить його ідеальним інструментом розробки сучасних вебсервісів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

3.1 Визначення основних компонентів React

Аналіз, проведений в попередніх розділах дозволяє визначити основні компоненти, що дозволять реалізувати функціонал додатку. Перелік компонентів і їх призначення наведені в таблиці 9.

Таблиця 9 – Компоненти застосунку

Компонент	Призначення компонента
Board	Відображення дошки; реалізує перетягування елементів (бібліотека react-beautiful-dnd); реалізує організацію списків в межах дошки (див. додаток В).
AddBoard	Створення нової дошки, дозволяючи користувачеві вводити назву дошки та вибирати її колір фону (див. додаток Г).
Card	Відображення картки завдання (інформація про зміст завдання, термін виконання та виконавця); використовує контекст FunctionsContext для перевірки дедлайну (змінює вигляд картки при наближенні до нього) (див. додаток Е).
AddCard	Створення нових карток (введення інформації про зміст завдання, термін виконання та виконавця); показ/приховання форми додавання картки; використовує хук useEffect для завантаження даних з sessionStorage.
Header	Містить логотип, що забезпечує доступ до стартової сторінки, та кнопку виходу.
Body	Налаштування маршрутизації за допомогою бібліотеки react-router-dom; визначає маршрути для компонентів: Login, Browse та Participants (див. додаток Б).
List	Виведення списків карток (див. додаток Д).
Browse	Відображення основного інтерфейсу для перегляду дошок, використовуючи дані з контексту BoardContext. Він включає в себе компоненти Header, Sidebar і Main, забезпечуючи структуру сторінки, а також обробляє стан завантаження, відображаючи повідомлення “Loading...” до моменту отримання даних про дошки.
SideBar	Забезпечує навігацію для управління дошками та учасниками, дозволяє користувачам переглядати існуючі дошки, додавати нові дошки, а також переходити до сторінки учасників.

Кінець таблиці 9

Компонент	Призначення компонента
Login	Реалізація форми авторизації, що дозволяє користувачам вводити електронну пошту та пароль для входу в систему; використовує хук useState для управління станом видимості помилок; використовує хук useRef для доступу до полів вводу.
Main	Відображення основного контенту дошки; використовує контекст BoardContext для отримання та оновлення даних про дошки.
Participants	Управління списком учасників, дозволяє додавати, редагувати та видаляти учасників системи.
Validate	Валідація електронної пошти та пароля на відповідність заданим критеріям, використовує регулярні вирази для перевірки формату електронної пошти та складності пароля.

Структура файлів і папок React-проєкту (див. табл. 10) дозволяє легко масштабувати застосунок та змінювати його компоненти по мірі розширення функціоналу та зміні вимог, що забезпечує гнучкість і простоту розробки.

Таблиця 10 – Структура файлів і папок React-проєкту

Папка/файл	Призначення/вміст
node_modules	Папка містить встановлені залежності проєкту.
public	Папка для статичних файлів, які будуть доступні за їх оригінальними іменами без змін у процесі зборки. Наприклад, тут розміщуються зображення та шрифти для прямого доступу через URL.
src	Основна папка коду застосунку, містить всі компоненти.
App.jsx	Головний компонент застосунку, в ньому починається створення інтерфейсу користувача.
main.jsx	Точка входу для React-застосунку. Компонент відповідає за рендеринг кореневого компонента та його прив’язку до елемента в index.html.
App.css	Стилі для компонента App.jsx.
index.html	Точка входу для застосунку. Vite (якщо використовується цей інструмент) обслуговує цей файл безпосередньо під час розробки, впроваджуючи необхідні скрипти для завантаження застосунку.
package.json	Файл конфігурації пакетного менеджера, містить інформацію про проєкт та його залежності.

Кінець таблиці 10

Папка/файл	Призначення/вміст
vite.config.js	Файл конфігурації Vite (якщо використовується цей інструмент), який дозволяє налаштовувати поведінку збирача, додавати плагіни або змінювати процес збірки.

Також, слід зауважити, що для розробки у роботі були встановлені і використані пакети, що забезпечують як ефективність розробки, так і функціональність застосунку, полегшуючи управління станом, маршрутизацію, стилізацію та взаємодію з користувачем. Перелік пакетів і їх призначення наведені в таблиці 11.

Таблиця 11 – Пакети проєкту

Пакет	Призначення пакету
react	Основна бібліотека для створення інтерфейсів користувача з використанням компонентного підходу. (Встановлюється за замовчуванням при створенні проєкту).
react-dom	Додаткова бібліотека для роботи з DOM у React, що дозволяє рендерити компоненти в документі HTML. (Встановлюється за замовчуванням при створенні проєкту).
react-beautiful-dnd	Бібліотека для реалізації функціональності drag-and-drop, забезпечує зручний та інтуїтивно зрозумілий інтерфейс для переміщення елементів.
react-icons	набір іконок, дозволяє легко додавати іконки в проєкти, використовуючи популярні набори іконок, такі як Font Awesome та Material Icons.
react-router	Бібліотека для маршрутизації, забезпечує основну функціональність маршрутизації.
react-router-dom	Бібліотека для маршрутизації, додає специфічні для вебдодатків компоненти, такі як <BrowserRouter> та <Link>.
react-tiny-popover	Бібліотека для створення спливаючих вікон (popover), дозволяє відображати контент у спливаючих вікнах при наведенні або кліку.
tailwindcss	CSS-фреймворк для швидкого створення адаптивних інтерфейсів за допомогою утилітного підходу до стилізації, що дозволяє легко налаштовувати стилі без написання кастомного CSS.

Кінець таблиці 11

Пакет	Призначення пакету
vite	Збирач проєктів, що забезпечує швидке завантаження та гаряче оновлення модулів під час розробки.

3.2 Інтерфейс користувача

Структурна схема інтерфейсу наведена в роботі на рисунку 8. При завантаженні застосунку, попередньо перед завантаженням основного вікна, з’являється вікно авторизації (див. рис. 12).

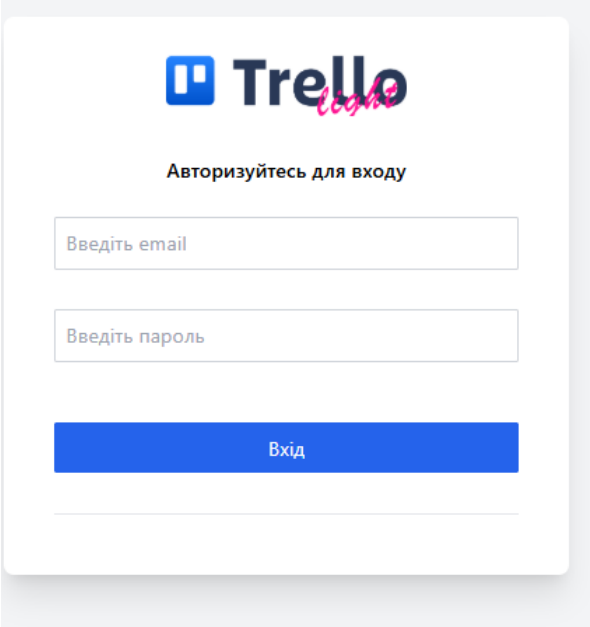


Рисунок 12 – Вікно авторизації додатку

Основне вікно розробленого вебзастосунку з’являється у разі успішної авторизації і має вигляд, наведений на рисунку 13.

Вікно, відповідно до спроєктованої структури (див. рис. 8), містить наступні елементи:

– хедер;
– бокова панель;
– робоча область.

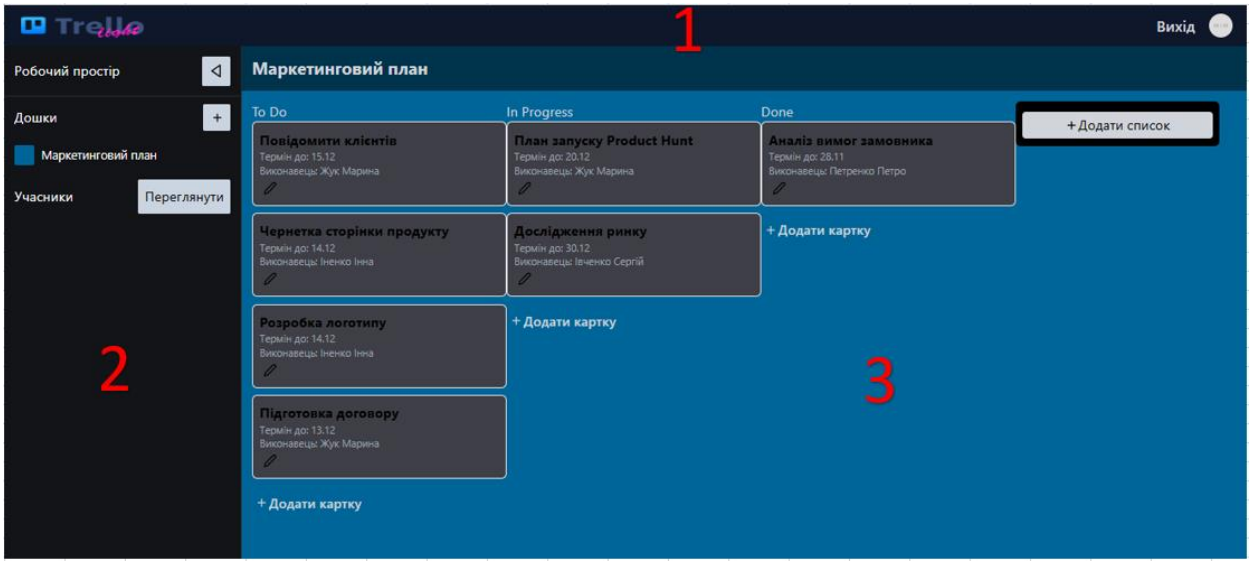


Рисунок 13 – Вікно вебзастосунку для управління проєктами

Хедер (див. рис. 14). Хедер містить логотип і назву застосунку та кнопку виходу з облікового запису. За наявності аватару у користувача, він буде відображений у правій частині компоненту.

Бокова панель (див. рис. 15), що реалізує навігацію по сторінках застосунку. На боковій панелі розташований перелік доступних дошок і посилання для переходу на сторінку учасників. За необхідності, панель може бути прихована відповідною кнопкою.

Робоча область, в якій виводиться обрана дошка (див. рис. 16) або список учасників (див. рис. 17).



Рисунок 14 – Хедер застосунку

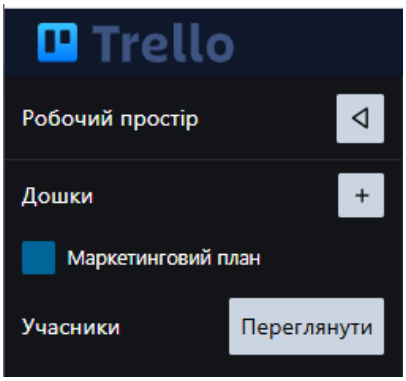


Рисунок 15 – Бокова панель застосунку

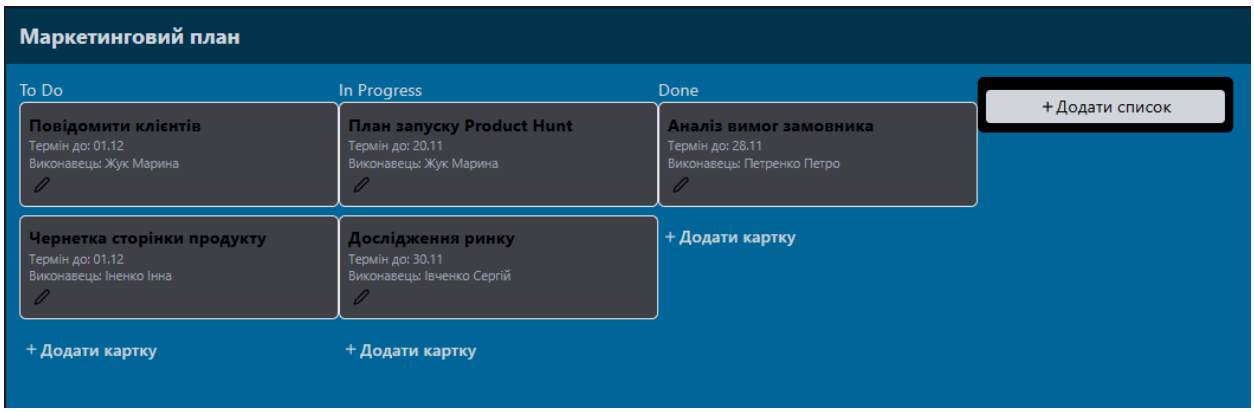


Рисунок 16 – Робоча область вікна застосунку зі списком активної дошки

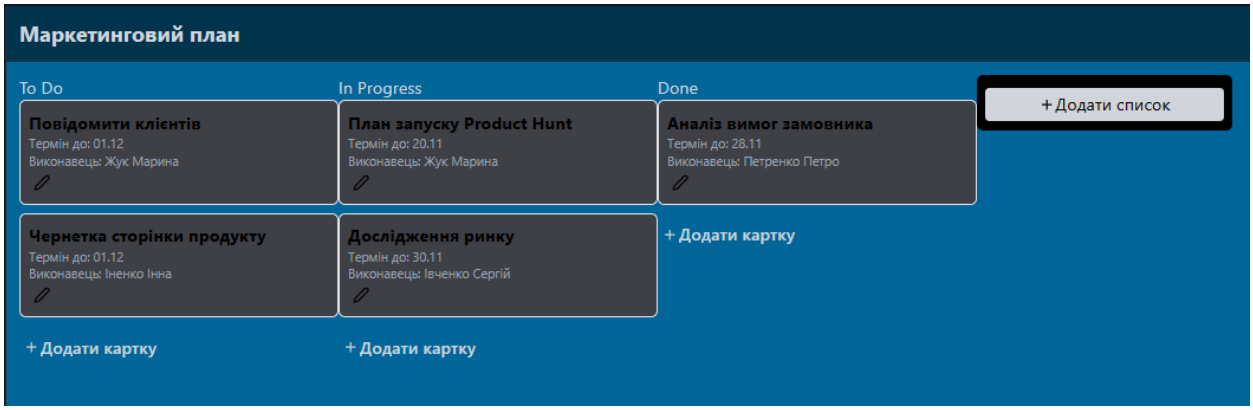


Рисунок 17 – Робоча область вікна застосунку зі списком учасників

3.3 Робота користувача в застосунку (Керівництво користувача)

Після завантаження застосунку, користувач попадає на сторінку авторизації (див. рис. 12). На сторінці він має ввести адресу електронної пошти та пароль. У разі внесення помилкових облікових даних, користувач одержить відповідне повідомлення у цьому ж вікні. У вікні авторизації здійснюється валідація введених даних, чи введено в поле email саме адресу електронної пошти.

Після успішної авторизації відкривається основне вікно застосунку (див. рис. 13). В основному вікні відображається активна дошка зі списками і картками, розташованими в цих списках. Кількість списків може бути довільною, тому, для зручності користувача на бокову панель додана кнопка,

що дозволяє приховати панель, щоб збільшити простір для виведення наявних списків.

Для створення нової дошки необхідно натиснути кнопку «+» на боковій панелі. У формі необхідно ввести назву нової дошки, обрати колір дошки і натиснути кнопку «Створити» (див. рис. 18).

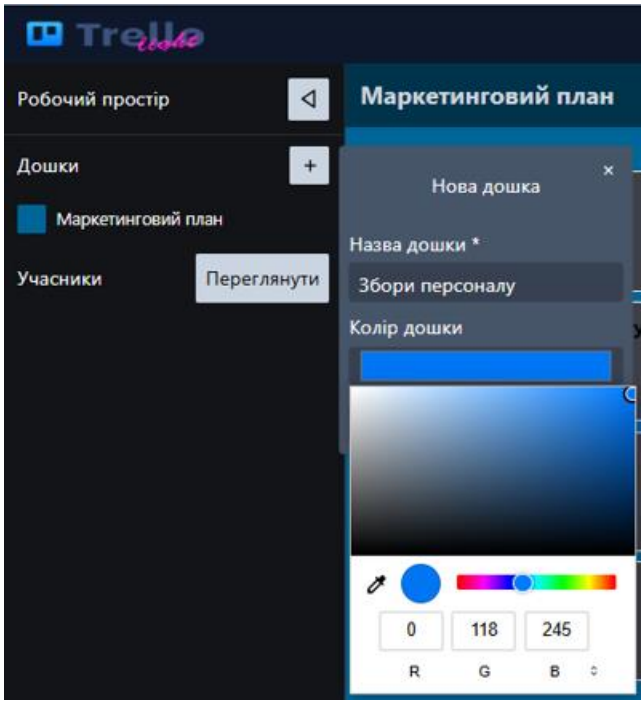


Рисунок 18 – Форма створення нової дошки

Після створення нова дошка з’явиться в списку дошок на боковій панелі (див. рис. 19).

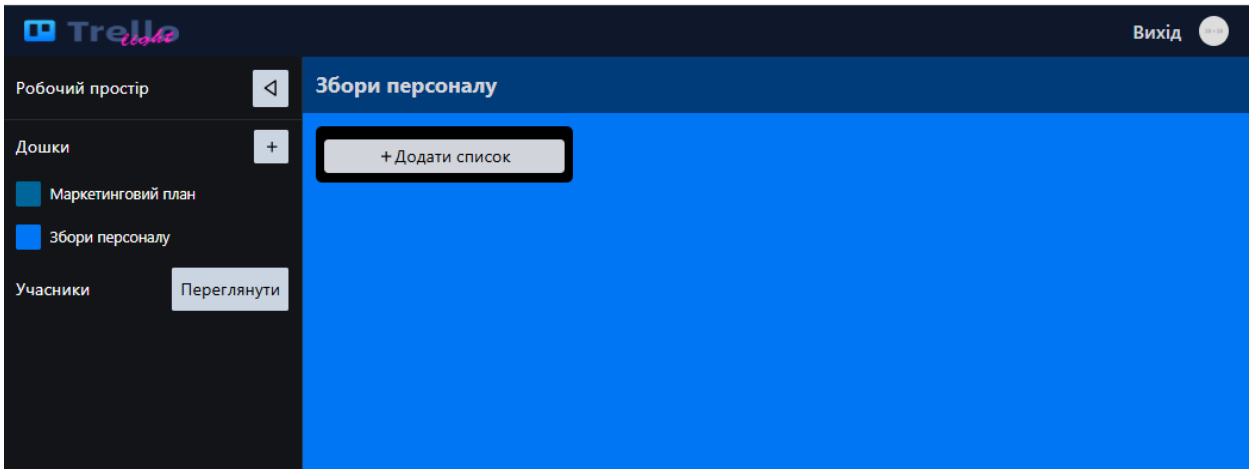


Рисунок 19 – Нова дошка на боковій панелі

Після створення нової дошки, потрібно додати новий список. Списки можна додавати і до існуючих дошок, їх кількість не обмежується. Щоб додати новий список на дошку, необхідно натиснути кнопку «+Додати список», після цього з’явиться форма створення нового списку, в якій достатньо ввести його назву (див. рис. 20).

Після введення назви необхідно натиснути кнопку форми «Зберегти» і новий список з’явиться на дошці (див. рис. 21). Одночасно у вікні буде відображено і кнопку «+Додати список», що дозволить створити наступний список на дошці.

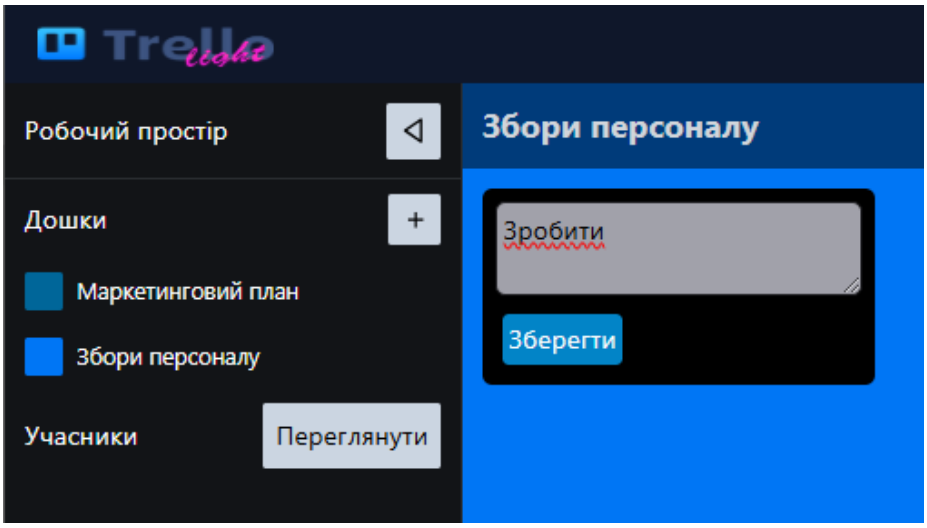


Рисунок 20 – Форма створення нового списку

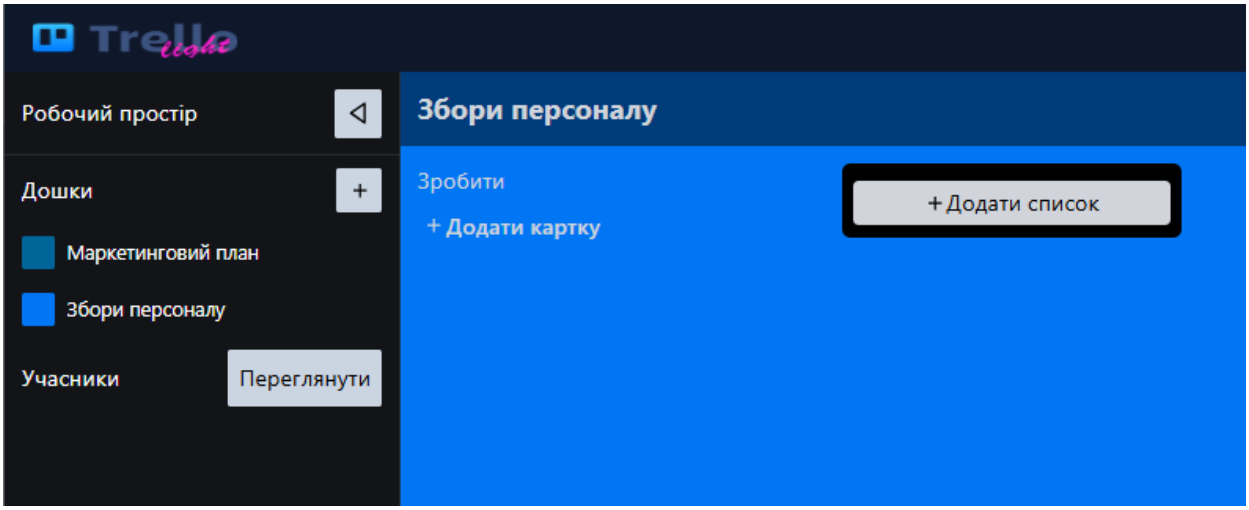


Рисунок 21 – Створений список «Зробити» на новій дошці «Збори персоналу»

Після створення списку до нього можна додавати картки. Для цього необхідно натиснути кнопку «+Додати картку» і в формі створення картки (див. рис. 22) ввести завдання, обрати дату дедлайну (див. рис. 23) і обрати відповідального за виконання завдання зі спадного списку учасників (див. рис. 24).

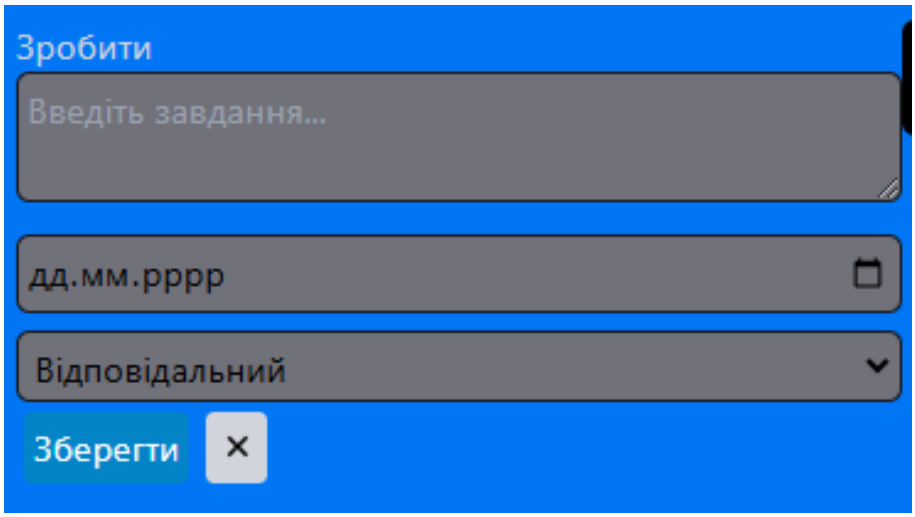


Рисунок 22 – Форма створення картки

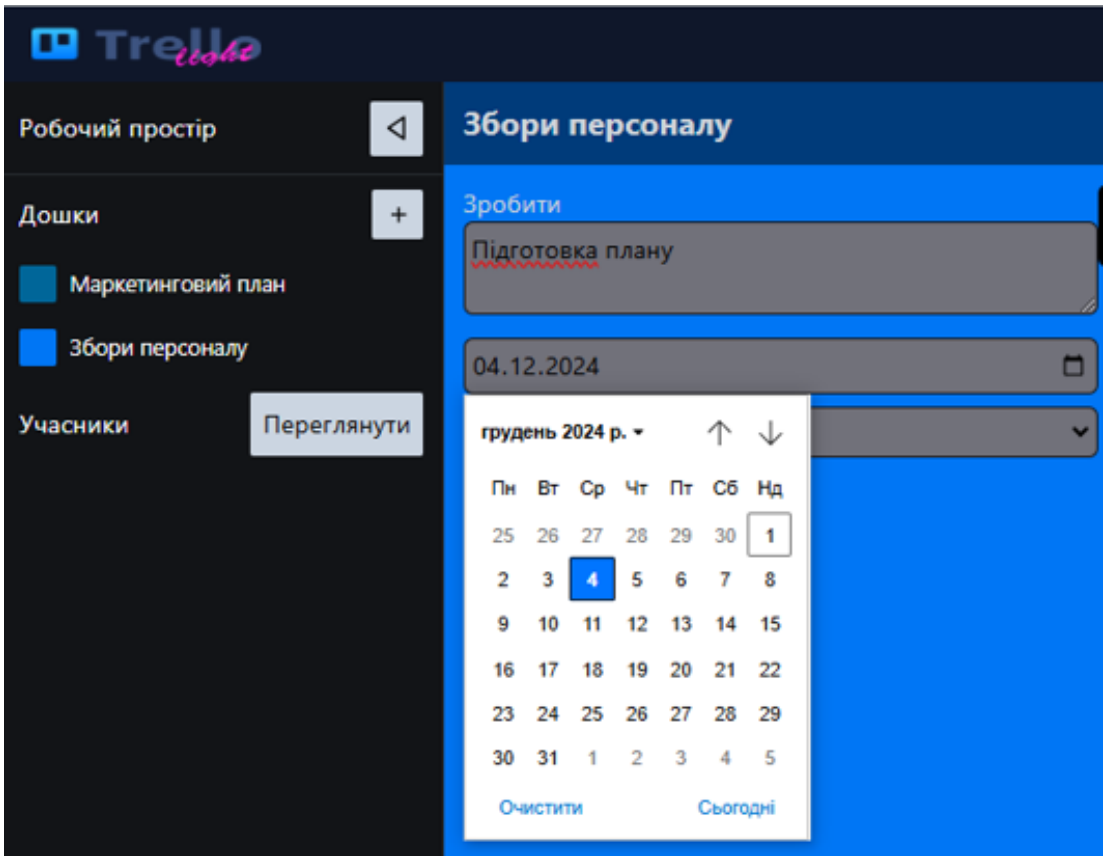


Рисунок 23 – Вибір дати на формі створення картки

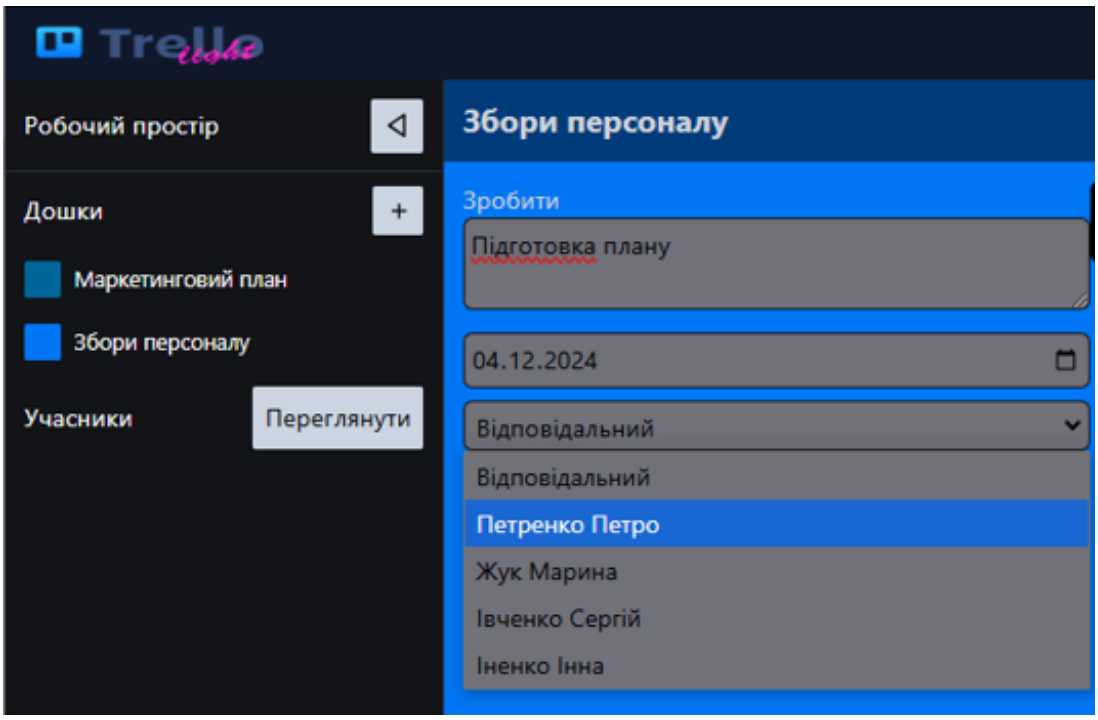


Рисунок 24 – Вибір відповідального зі спадного списку учасникі на формі створення картки

Список учасників системи можна переглянути, натиснувши кнопку «Переглянути» на боковій панелі. До списку можна додавати нових учасників. Також передбачена можливість редагування і видалення існуючих учасників (див. рис. 25).



Рисунок 25 – Список учасників

ВИСНОВКИ

В ході виконання кваліфікаційної роботи «Розробка вебзастосунку для управління проектами засобами React» було проведене дослідження, що охоплює огляд і аналіз систем управління проєктами, аналіз існуючих програмних рішень, проєктування та програмну реалізацію фронт-енд частини вебзастосунку для управління проєктами.

Системи управління проєктами на сьогоднішній день є невід'ємною частиною сучасного бізнесу, що забезпечують ефективну організацію, планування та контроль за виконанням проєктів.

Аналіз існуючих рішень продемонстрував, що ринок програмного забезпечення пропонує широкий спектр інструментів управління проєктами, але здебільшого вони мають обмежений функціонал на безкоштовних планах використання, натомість переобтяжені зайвим функціоналом на платних, що робить їх використання неефективним на безкоштовних версіях та ускладненим і фінансово недоцільним на платних версіях, особливо для малих проєктних груп та груп, що тільки починають свою діяльність.

На етапі визначення вимог до вебзастосунку було визначено мінімально необхідний набір функціоналу, що забезпечує управління проєктом. Він включає в себе створення дошки, створення списків на дошці, створення і редагування карток та їх переміщення між спискам, робота зі списком користувачів, авторизація користувача в системі. Окрім того, користувачі потребують інтуїтивно зрозумілого інтерфейсу. Проєктування архітектури вебзастосунку здійснювалось з урахуванням принципів модульності та масштабованості, що дозволяє просто розширювати функціонал та швидко адаптувати систему до змінюваних вимог.

Для реалізації застосунку був обраний React – JavaScript-бібліотека, орієнтована на розробку інтерфейсів користувача. Вибір ґрунтується на продуктивності, модульності та гнучкості React, що робить його ідеальним інструментом розробки сучасних вебсервісів.

Програмна реалізація застосунку включала розробку програмного коду компонентів, що забезпечують реалізацію функціоналу для управління проєктами. Опис використання застосунку продемонстрував наявність інтуїтивно зрозумілого інтерфейсу та простоту його використання. Окрім того, на цьому етапі була перевірена коректність роботи застосунку на тестовому наборі даних.

Результати роботи продемонстрували доцільність розробки на базі React вебзастосунку для управління проєктами. Запропоноване рішення забезпечує мінімально необхідний набір функціоналу для організації проєктної діяльності та відповідає вимогам ринку і може стати корисним інструментом для команд різного масштабу. Водночас, представлений застосунок, має перспективи подальшого вдосконалення і розвитку в напрямку розширення інструментів візуалізації проєктів і завдань, а також інтеграції з іншими технологіями, в т.ч. і такими як штучний інтелект, для автоматизації та оптимізації процесів управління проєктами.

ПЕРЕЛІК ПОСИЛАНЬ

1. Єпіфанова І. Ю., Джеджула В. В. Місце процесу управління проектами в стратегії розвитку підприємств. *Innovation and Sustainability*. 2022. № 3. С. 8–13.

2. П’ятничук І. Д. Інформаційні системи в управлінні проектами: онлайн-платформи і сервіси. *Економіка та суспільство*. 2022. № 42. URL: https://doi.org/10.32782/2524-0072/2022-42-21 (дата звернення: 08.08.2024).

3. Галушка В. Теоретико-методичні засади управління проєктами *Підприємництво, господарство і право.* 2020. № 7. С. 430–434.

4. Салій І. В., Кобилкін Д. С. Життєвий цикл проекту, як важлива складова в управлінні ІТ-проектами. *Інновінг сучасних трендів в менеджменті безпеки* : збірник наукових праць Всеукраїнської науково-практичної конференції (Львів, 26 травня 2023). Львів : ЛДУ БЖД, 2023. С. 17–18.

5. Козлова І. М. Теоретичні аспекти управління проектами в діяльності підприємства. *Регіональна економіка та управління*. 2020. № 2 (28). С. 63–70. URL: http://repository.hneu.edu.ua/handle/123456789/23643 (дата звернення: 07.09.2024).

6. Шевченко Т. Популярні методології управління проєктами: від гнучкої Agile до вимогливої PRINCE2. *Wizeclub Education*. URL: https://wizeclub.education/blog/populyarni-metodologiyi-upravlinnya-proyektami-vid-gnuchkoyi-agile-do-vimoglivoyi-prince2/ (дата звернення: 11.09.2024).

7. Методології управління проєктами та завданнями. *CoreWin*. 2024. URL: https://corewin.ua/blog/which_project_tasks_management_methodology_should_you_choose/ (дата звернення: 10.09.2024).

8. Храпкін О., Кіндрат О., Чопей Р. Управління проєктами в ІТ-галузі: методики, інструменти та керування ризиками. *Економіка та суспільство*. 2023. № 55. URL: https://doi.org/10.32782/2524-0072/2023-55-110 (дата звернення: 11.09.2024).

9. Шашкова Н., Фадєєва І., Казакова Т. Управління проєктами в ІТ сфері: застосування гнучких методологій. *Наукові записки Львівського університету бізнесу та права*. 2021. Вип. 28. С. 166–172. URL: https://nzlubp.org.ua/index.php/journal/article/view/402 (дата звернення: 10.09.2024).

10. Aston B. 15 Best Project Management Software Picked For 2024. *The Digital Project Manager*. URL: https://thedigitalprojectmanager.com/tools/best-project-management-software/ (дата звернення: 16.09.2024).

11. Johnson H., Adams D., Howard L. 10 Best Project Management Software & Tools Of 2024. *Forbes Advisor*. URL: https://www.forbes.com/uk/advisor/business/software/best-project-management-software/ (дата звернення: 16.09.2024).

12. Худенко Д. ТОП 7 інструментів для управління проектами у 2023 році. *Worksection*. URL: https://worksection.com/ua/blog/5-project-management-tools.html (дата звернення: 16.09.2024).

13. Коросташовець А. Найкраще програмне забезпечення для автоматизації робочих процесів у 2024 році. Блог системи управління проектами. *Worksection*. URL: https://worksection.com/ua/blog/best-workflow-automation-software-2024.html (дата звернення: 16.09.2024).

14. Google Trends. URL: https://trends.google.com/trends/explore?geo=UA&q=Trello,ClickUp,monday.com%20-%20Work%20Management,Jira,Asana&hl=uk (дата звернення: 01.10.2024).

15. Башинська І. О., Хрістова А. В. Використання сучасних інформаційних технологій управління проектами. *Економічний журнал Одеського політехнічного університету*. 2017. № 1. С. 16–22. URL: http://nbuv.gov.ua/UJRN/ejopu_2017_1_6 (дата звернення: 01.10.2024).

16. Trello Guides: Help Getting Started With Trello. Manage Your Team’s Projects From Anywhere. *Trello*. URL: https://trello.com/guide (дата звернення: 11.10.2024).

17. Товма О. Основні типи архітектури програмного забезпечення. *ArtofBA*. URL: https://www.artofba.com/uk/post/main-types-of-software-architecture (дата звернення: 11.10.2024).

18. Волошин О., Добринін М. Архітектура веб-додатків – як вибрати? : стаття з блогу IT-школи Hillel. *Комп'ютерна школа Hillel*. URL: https://blog.ithillel.ua/articles/web-application-architecture (дата звернення: 12.10.2024).

19. Гузенко О. Що таке інтерфейс користувача та як він впливає на продажі. *Wezom*. URL: https://wezom.com.ua/ua/blog/chto-takoe-ui-i-kak-polzovatelskij-interfejs-vliyaet-na-prodazhi-internet-magazina (дата звернення: 17.10.2024).

20. Поради щодо інтерфейсу користувача, щоб вивести ваш веб-сайт на нові висоти. *DIZZ*. URL: https://dizz.in.ua/uk/poradi-shhodo-interfejsu-koristuvacha-shhob-vivesti-vash-veb-sajt-na-novi-visoti/ (дата звернення: 18.10.2024).

21. De Sousa M., Gonçalves A. Humanportal – A React.js case study. *CISTI* : 15th Iberian Conference on Information Systems and Technologies (Seville, Spain, 2020). IEEE, 2020. P. 1–6.

22. Швидкий старт – React. *React*. URL: https://uk.react.dev/learn (дата звернення: 19.10.2024).

23. Красинська М. Навіщо і як вивчати React. *Robot_dreams*. URL: https://robotdreams.cc/uk/blog/237-zachem-i-kak-uchit-react-js (дата звернення: 19.10.2024).

ДОДАТОК А

Компонент APP.JSX

```
import { useEffect, useState } from 'react';
import './App.css';
import Body from './components/Body';
import { BoardContext } from './context/BoardContext';
import FunctionsProvider from './context/FunctionsContext';
import data from './data.json';

function App() {
  const [allboard, setAllBoard] = useState(null);

  useEffect(() => {
    const storedData = sessionStorage.getItem('boardData');

    if (storedData) {
      setAllBoard(JSON.parse(storedData));
    } else {
      sessionStorage.setItem('boardData', JSON.stringify(data));
      setAllBoard(data);
    }
  }, []);

  if (!allboard) {
    return <div>Loading...</div>;
  }

  return (
    <FunctionsProvider>
      <BoardContext.Provider value={{ allboard, setAllBoard }}>
        <div className="App">
          <Body />
        </div>
      </BoardContext.Provider>
    </FunctionsProvider>
  );
}

export default App;
```

ДОДАТОК Б

Компонент BODY.JSX

```
import { createBrowserRouter, RouterProvider } from "react-router-dom";
import Browse from "./Browse";
import Login from "./Login";
import Participants from "./Participants";

const Body = () => {
  const appRouter = createBrowserRouter([
    { path: "/",
     element: <Login />
    },
    { path: "/browse",
     element: <Browse />
    },
    { path: "/participants",
     element: <Participants />
    },
  ],
    {
      future: {
        v7_skipActionErrorRevalidation: true,
      },
    }
  );

  return (
    <div>
      <RouterProvider router={appRouter} />
    </div>
  );
}

export default Body;
```

ДОДАТОК В

Компонент BOARD.JSX

```
import React from 'react';
import { DragDropContext, Droppable } from 'react-beautiful-dnd';
import List from './List';
import { BoardContext } from '../context/BoardContext';

const Board = ({ bdata, onDragEnd }) => {
  return (
    <div className="flex flex-col w-full" style={{ backgroundColor: `${bdata.bgcolor}` }}>
      <div className="p-3 bg-black flex justify-between w-full bg-opacity-50">
        <h2 className="text-gray-300 font-bold text-xl">{bdata.name}</h2>
      </div>
      <div className="flex flex-col w-full flex-grow relative">
        <div className="absolute left-0 right-0 top-0 bottom-0 p-3 flex overflow-x-scroll
overflow-y-hidden">
          <DragDropContext onDragEnd={onDragEnd}>
            <Droppable droppableId="board" direction="horizontal">
              {(provided) => (
                <div
                  className="flex"
                  ref={provided.innerRef}
                  {...provided.droppableProps}
                >
                  {bdata.list?.map((list, index) => (
                    <List
                      key={list.id}
                      list={list}
                      index={index}
                    />
                  ))}
                  {provided.placeholder}
                </div>
              )}
            </Droppable>
          </DragDropContext>
        </div>
      </div>
    </div>
  );
};

export default Board;
```

ДОДАТОК Г

Компонент ADDBOARD.JSX

```
import React, { useState } from 'react';
import { BoardContext } from '../context/BoardContext';

const AddBoard = ({ addBoard }) => {
  const [boardName, setBoardName] = useState('');
  const [bgColor, setBgColor] = useState('#069');

  const handleSubmit = (e) => {
    e.preventDefault();

    if (boardName.trim()) {
      const newBoard = {
        name: boardName,
        bgcolor: bgColor,
        list: [],
        active: 0,
      };

      addBoard(newBoard);

      setBoardName('');
      setBgColor('#069');
    }
  };

  return (
    <div className="add-board-container">
      <form onSubmit={handleSubmit} className="add-board-form">
        <input
          type="text"
          value={boardName}
          onChange={(e) => setBoardName(e.target.value)}
          placeholder="Enter board name"
          required
        />
        <input
          type="color"
          value={bgColor}
          onChange={(e) => setBgColor(e.target.value)}
        />
        <button type="submit">Add Board</button>
      </form>
    </div>
```

```
    );
};

export default AddBoard;
```

ДОДАТОК Д

Компонент LIST.JSX

```
import React from 'react';
import { Draggable, Droppable } from 'react-beautiful-dnd';
import Card from './Card';
import AddCard from './AddCard';
import { FiEdit2 } from 'react-icons/fi';

const List = ({ list, index, formatDeadline, isDeadlineClose, getCard }) => {

  const handleAddCard = (newCard) => {
    const boardData = JSON.parse(sessionStorage.getItem('boardData'));

    if (!boardData) {
      console.error("Дані boardData не знайдені у sessionStorage");
      return;
    }

    const boardIndex = boardData.boards.findIndex(board =>
      board.list.some(lst => lst.id === list.id)
    );

    if (boardIndex === -1) {
      console.error("Список не знайдено");
      return;
    }

    const updatedBoards = [...boardData.boards];
    const listIndex = updatedBoards[boardIndex].list.findIndex(lst => lst.id === list.id);

    if (listIndex !== -1) {
      updatedBoards[boardIndex].list[listIndex].items.push({
        id: Date.now().toString(),
        title: newCard.title,
        deadline: newCard.deadline,
        assignee: newCard.assignee,
      });

      console.log("Оновлений список:", updatedBoards[boardIndex].list[listIndex].items);

      const updatedBoardData = {
        ...boardData,
        boards: updatedBoards,
      };
```

```
        console.log("Підготовка до запису в sessionStorage:",
JSON.stringify(updatedBoardData));
        sessionStorage.setItem('boardData', JSON.stringify(updatedBoardData));
        console.log("Дані записано в sessionStorage");
        console.log("Оновлений boardData записаний у sessionStorage:", updatedBoardData);

        getCard(list.id, updatedBoards[boardIndex].list[listIndex].items);
      }
    };


    return (
      <Droppable droppableId={list.id} direction="vertical">
        {(provided) => (
          <div
            ref={provided.innerRef}
            {...provided.droppableProps}
            className="flex flex-col"
            style={{ minWidth: '300px', maxWidth: '400px' }}
          >
            <h3 className="text-gray-300">{list.title}</h3>
            {list.items?.map((item, index) => (
              <Draggable key={item.id} draggableId={item.id} index={index}>
                {(provided) => (
                  <div
                    ref={provided.innerRef}
                    {...provided.draggableProps}
                    {...provided.dragHandleProps}
                    className="item flex flex-col items-start bg-zinc-700 p-2 mb-2 rounded-md
border transition duration-200"
                    style={{ width: '100%' }}
                  >
                    <span className="font-bold">{item.title}</span>
                    {item.deadline && (
                      <span className="text-xs text-gray-400">
                        {`Термін до: ${formatDeadline(item.deadline)}`}
                      </span>
                    )}
                    {item.assignee?.name && (
                      <span className="text-xs text-gray-400">{`Виконавець:
${item.assignee.name}`}</span>
                    )}
                    <span className="flex justify-start items-start">
                      <button className="hover:bg-gray-600 p-1 rounded-sm">
                        <FiEdit2 size={16} />
                      </button>
                    </span>
                  </div>
                )}
```

```
                </Draggable>
              ))}
              {provided.placeholder}

              <AddCard onAddCard={handleAddCard} />
            </div>
          )}
        </Droppable>
    );
};

export default List;
```

ДОДАТОК Е

Компонент CARD.JSX

```
import React, { useContext } from 'react';
import { Draggable } from 'react-beautiful-dnd';
import { FiEdit2 } from 'react-icons/fi';
import { FunctionsContext } from '../context/FunctionsContext';

const Card = ({ item, index }) => {
  const { formatDeadline, isDeadlineClose } = useContext(FunctionsContext);

  return (
    <Draggable key={item.id} draggableId={item.id} index={index}>
      {(provided) => (
        <div
          ref={provided.innerRef}
          {...provided.draggableProps}
          {...provided.dragHandleProps}
          className={`item flex flex-col items-start bg-zinc-700 p-2 mb-2 rounded-md border
transition duration-200 ${
            item.deadline && isDeadlineClose(item.deadline) ? 'border-red-500' : 'border-
zinc-900'
          }`}
        >
          <span className="font-bold">{item.title}</span>
          {item.deadline && (
            <span className="text-xs text-gray-400">
              {`Термін до: ${formatDeadline(item.deadline)}`}
            </span>
          )}
          {item.assignee?.name && (
            <span className="text-xs text-gray-400">{`Виконавець:
${item.assignee.name}`}</span>
          )}
          <span className="flex justify-start items-start">
            <button className="hover:bg-gray-600 p-1 rounded-sm">
              <FiEdit2 size={16} />
            </button>
          </span>
        </div>
      )}
    </Draggable>
  );
};

export default Card;
```