Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

# КВАЛІФІКАЦІЙНА РОБОТА

**на здобуття ступеня вищої освіти бакалавра**

зі спеціальності 121 – Інженерія програмного забезпечення

На тему: Розробка клієнтської частини веб-застосунку для управління проєктами з використанням Vue.js

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ-22-1
_____ / Д. Ю. Климовський /

Керівник
кваліфікаційної роботи _____ / А. М. Гриценко /

Завідувач кафедри _____ / А. М. Стрюк /

Кривий Ріг
2026

Криворізький національний університет

Факультет: Інформаційних технологій
Кафедра: Моделювання та програмного забезпечення
Ступінь вищої освіти: бакалавр
Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Завідувач кафедри
_____А. М. Стрюк
«___» _____ 20__ р.

# ЗАВДАННЯ
## на кваліфікаційну роботу

студенту групи ІПЗ-22-1 Климовському Данилу Юрійовичу

1. Тема: Розробка клієнтської частини веб-застосунку для управління проєктами з використанням Vue.js
   затверджено наказом по КНУ № 284 с від 28.05.2026.
2. Термін подання студентом закінченої роботи: «10» червня 2026 р.
3. Вихідні дані: веб-додаток повинен забезпечувати управління проєктами з адаптивним інтерфейсом, що коректно відображається на пристроях з різними розмірами екрана, підтримувати автентифікацію користувачів, Kanban-дошку, персональний список завдань та управління командою.
4. Зміст пояснювальної записки: виконати аналіз існуючих аналогів та обґрунтувати актуальність теми; обґрунтувати вибір технологій та спроектувати архітектуру системи; розробити діаграми UML, BPMN та макети інтерфейсу у Figma; реалізувати клієнтську частину додатку з підключенням EmailJS; провести тестування функціоналу та адаптивності на мобільних пристроях.
5. Перелік ілюстративного матеріалу: діаграма прецедентів UML, модель бізнес-процесу в нотації BPMN 2.0, архітектурна схема системи, макети інтерфейсу з Figma, знімки екранних форм десктопної та мобільної версій діючого веб-додатку.

Календарний план:

| № | Найменування етапів кваліфікаційної роботи | Термін виконання |
|---|---|---|
| 1 | Огляд і аналіз предметної області | 27.04.2026 – 30.04.2026 |
| 2 | Аналіз існуючих аналогів, порівняльна характеристика | 01.05.2026 – 04.05.2026 |
| 3 | Підготовка матеріалів першого розділу роботи | 05.05.2026 – 07.05.2026 |
| 4 | Обґрунтування технологічного стеку, розробка архітектури системи | 08.05.2026 – 11.05.2026 |
| 5 | Розробка діаграм UML, BPMN та макетів інтерфейсу у Figma | 12.05.2026 – 15.05.2026 |
| 6 | Підготовка матеріалів другого розділу роботи | 16.05.2026 – 18.05.2026 |
| 7 | Програмна реалізація клієнтської частини (Vue 3, Tailwind CSS, Pinia) | 19.05.2026 – 25.05.2026 |
| 8 | Реалізація зберігання даних через LocalStorage, підключення EmailJS | 26.05.2026 – 30.05.2026 |
| 9 | Підготовка матеріалів третього розділу роботи, тестування адаптивності | 31.05.2026 – 03.06.2026 |
| 10 | Тестування додатку, оформлення пояснювальної записки | 04.06.2026 – 10.06.2026 |

Дата видачі завдання: «__» _____ 2026 р.

Студент _____ / Д. Ю. Климовський /

Керівник роботи _____ / А. М. Гриценко /

# РЕФЕРАТ

ВЕБ-ДОДАТОК, АДАПТИВНИЙ ІНТЕРФЕЙС, УПРАВЛІННЯ ПРОЕКТАМИ, KANBAN-ДОШКА, VUE 3, LOCALSTORAGE, TAILWIND CSS, PINIA, АВТЕНТИФІКАЦІЯ, SERVERLESS.

Пояснювальна записка: 80 с., 2 табл., 28 рис., 1 дод., 15 джерел.

Метою цієї кваліфікаційної роботи є створення і розробка веб-додатку для управління проєктами з адаптивним інтерфейсом, що забезпечує коректне відображення та зручну взаємодію на пристроях із різними розмірами екрана.

Об'єктом розробки є процес управління проєктами та завданнями в командах розробки програмного забезпечення.

У роботі проведено аналіз предметної області управління проєктами, розглянуто методології Agile, Scrum і Kanban. Виконано порівняльний огляд існуючих програмних рішень: Jira, Trello та Asana, виявлено їхні недоліки в адаптивності інтерфейсу на мобільних пристроях.

Для досягнення цілей спроектована архітектура системи, створені UML-діаграма і модель бізнес-процесів у BPMN 2.0, розроблені макети інтерфейсу у Figma для десктопів і мобільних пристроїв. Клієнтська частина реалізована на Vue 3 із Composition API, стилі — за допомогою Tailwind CSS, управління станом — через Pinia, навігація — Vue Router. Дані зберігаються у LocalStorage, що автоматично серіалізує стан у JSON. Email-сповіщення надсилає сервіс EmailJS, що виконує мережеві запити без власного серверного коду.

Додаток ProjectFlow дозволяє командам створювати проєкти, керувати завданнями через Kanban-дошку з перетягуванням карток, переглядати особистий список завдань, відслідковувати учасників та редагувати профілі. Він призначений для малих і середніх команд з розробки ПЗ, стартапів та проєктних груп.

Проведено тестування функціоналу додатка та перевірено коректність відображення інтерфейсу на пристроях з різними розмірами екрана.

# ABSTRACT

WEB APPLICATION, ADAPTIVE INTERFACE, PROJECT MANAGEMENT, KANBAN BOARD, VUE 3, LOCALSTORAGE, TAILWIND CSS, PINIA, AUTHENTICATION, SERVERLESS.

Thesis: 80 p., 2 tab., 28 fig., 1 app., 15 references.

The goal of the qualification thesis is to design and develop a project management web application with an adaptive interface that ensures proper display and convenient interaction across devices with varying screen sizes.

The object of development is the process of managing projects and tasks in software development teams.

The thesis includes an analysis of the project management domain, covering Agile, Scrum, and Kanban methodologies. A comparative review of existing solutions was conducted, and their shortcomings regarding mobile interface adaptability were identified.

The system architecture included a UML use case diagram, a BPMN 2.0 process model, and interface mockups in Figma for desktop and mobile. Client-side uses Vue 3 with the Composition API, Tailwind CSS, Pinia, and Vue Router. Data storage relies on LocalStorage with a Pinia watcher for serialization. EmailJS handles email notifications, making network requests directly from the client without custom server code.

The ProjectFlow application allows teams to create projects, manage tasks on a Kanban board with drag-and-drop, view individual task lists, oversee team members, and customize user profiles. Designed for small to medium-sized software development teams, startups, and project groups.

Functional testing of the application was conducted, and the correctness of the interface display on devices with various screen sizes was verified.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

| | |
|---|---|
| **API** | Application Programming Interface — програмний інтерфейс для взаємодії застосунків |
| **BPMN** | Business Process Model and Notation — нотація моделювання бізнес-процесів |
| **Composition API** | підхід до організації логіки компонентів у Vue 3 на основі функцій |
| **CSS** | Cascading Style Sheets — каскадні таблиці стилів |
| **JSON** | JavaScript Object Notation — текстовий формат обміну даними |
| **Kanban** | метод організації роботи за допомогою дошки з колонками станів завдань |
| **UML** | Unified Modeling Language — уніфікована мова моделювання |
| **URL** | Uniform Resource Locator — уніфікований локатор ресурсу |

# ВСТУП

Сучасний розвиток індустрії програмного забезпечення швидко веде до впровадження гнучких методологій управління проєктами й зростання популярності дистанційної роботи. В таких умовах ефективність проектів в значній мірі залежить від інструментів для відстеження завдань. Наразі існуючі платформи, такі як Jira, Asana і Trello, мають широкий функціонал, але їхній мобільний інтерфейс недостатньо адаптивний, що створює незручності. Відповідно, виникає нагальна потреба у створенні легкого, швидкого і повністю адаптивного веб-додатку для управління проєктами.

Метою роботи є проєктування та розробка веб-додатку ProjectFlow для управління проєктами з адаптивним інтерфейсом, що забезпечує зручну роботу на пристроях з різними розмірами екрану без втрати функціональності та швидкодії.

Для досягнення цієї мети виконується кілька завдань: аналіз предметної області та існуючих програмних аналогів, обґрунтування вибору технологічного стеку, проектування архітектури системи, створення діаграм UML і BPMN, розробка макетів інтерфейсу, реалізація клієнтської частини на Vue 3 з використанням LocalStorage для збереження даних та підключенням EmailJS для email-сповіщень, а також тестування розробленого додатку.

Область застосування результатів роботи охоплює малі та середні команди розробки програмного забезпечення, стартапи і проєктні групи, які потребують зручного кросплатформного інструменту для управління завданнями без потреби встановлювати окремі застосунки.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Критичний аналіз існуючих рішень

Ринок програмного забезпечення для управління проєктами сьогодні представлений великою кількістю рішень різного масштабу та складності. Серед них стійкі позиції займають три платформи: Jira, Trello та Asana. Кожна з них орієнтована на певний тип команд і має власний набір переваг та обмежень.

Jira позиціонується як інструмент корпоративного рівня з розширеними можливостями: кастомізовані робочі процеси, глибока аналітика, інтеграція з системами контролю версій і гнучке управління доступом [9]. Проте така функціональна насиченість стає недоліком для малих команд. Інтерфейс Jira має багато вкладених меню, бічних панелей і налаштувань, через що новачки витрачають багато часу на його освоєння. Також виникає проблема з мобільним використанням: веб-версія на смартфоні ускладнює досвід через малі елементи керування і складну навігацію, а повноцінне використання можливе лише через нативний мобільний додаток, який потребує окремого встановлення і оновлень.

Trello базується на концепції Kanban і відомий своєю простотою освоєння. Користувачі одразу бачать дошку з колонками та картками завдань, без необхідності додаткового налаштування [10]. Однак у командному середовищі виникають певні обмеження: при створенні більше ніж 4–5 колонок мобільна веб-версія переходить у горизонтальний режим прокручування, що ускладнює роботу на малих екранах.

Asana займає проміжну позицію між двома описаними рішеннями та пропонує кілька режимів відображення проєкту: список, Kanban-дошка та календар [11]. Адаптивність веб-версії реалізована дещо краще, ніж у Trello, проте надмірна кількість анімацій і декоративних елементів негативно впливає на продуктивність на бюджетних мобільних пристроях. Для невеликих команд

або стартапів функціонал Asana часто виявляється надлишковим, а вартість платних тарифів не завжди виправдана.

Порівняльний аналіз трьох платформ наведено у таблиці 1.1.

Таблиця 1.1 – Порівняльна характеристика існуючих рішень

| Критерій | Jira | Trello | Asana |
|---|---|---|---|
| Адаптивність веб-версії | Низька | Низька | Середня |
| Поріг входження | Високий | Низький | Середній |
| Функціональність | Висока | Базова | Середня |
| Робота без встановлення | Частково | Так | Так |
| Придатність для малих команд | Низька | Середня | Середня |

Отже, жодне з розглянутих рішень не забезпечує одночасно низький поріг входження, повну адаптивність веб-інтерфейсу та достатній функціонал для роботи невеликої команди без необхідності встановлювати додатки.

### 1.2 Актуальність теми кваліфікаційної роботи

Широке впровадження дистанційної та гібридної роботи кардинально змінило вимоги до інструментів управління проєктами. Якщо раніше таск-менеджери переважно використовувалися на стаціонарних комп’ютерах, тепер учасники команд частіше використовують їх з смартфонів, щоб перевірити статус завдання під час руху, змінити пріоритети між зустрічами або переглянути склад команди без ноутбука. Тому якість мобільного досвіду стала ключовою вимогою до такого програмного забезпечення.

Водночас, як показав аналіз у підрозділі 1.1, лідери ринку не вирішують цю проблему на рівні веб-інтерфейсу. Їхні мобільні веб-версії або суттєво

урізані за функціоналом або незручні через горизонтальне прокручування та дрібні елементи керування. Повноцінна робота з більшістю популярних платформ на смартфоні фактично вимагає встановлення нативного застосунку, що є додатковим бар'єром для нових учасників команди.

Розвиток сучасних веб-технологій, зокрема реактивних JavaScript-фреймворків і утилітарних CSS-бібліотек, відкриває можливість створити веб-додаток, що за якістю мобільного інтерфейсу не поступається нативним рішенням. Такий застосунок працює безпосередньо в браузері, не потребує встановлення та доступний з будь-якого пристрою за посиланням.

Таким чином, існує реальна і незакрита потреба у легкому, швидкому веб-додатку для управління проєктами, орієнтованому на невеликі команди, з якісним адаптивним інтерфейсом, що однаково зручно працює як на десктопі, так і на мобільних пристроях, без необхідності встановлення додаткового програмного забезпечення.

### 1.3 Цілі та завдання кваліфікаційної роботи

Метою кваліфікаційної роботи є проєктування та розробка веб-додатку ProjectFlow для управління проєктами з адаптивним інтерфейсом, який однаково зручно працює на комп'ютерах і на мобільних пристроях.

Для досягнення поставленої мети необхідно вирішити такі завдання:

а) проаналізувати існуючі програмні рішення для управління проєктами та визначити їх недоліки щодо роботи на мобільних пристроях;

б) обґрунтувати вибір технологій для розробки додатку та описати його архітектуру;

в) розробити діаграму прецедентів UML [14] і модель бізнес-процесів BPMN 2.0 [15], а також створити макети інтерфейсу у Figma [13];

г) реалізувати клієнтську частину додатка на Vue 3 [1], орієнтуючись на кілька ключових вимог: швидкість роботи інтерфейсу на мобільних пристроях з обмеженими ресурсами, зручність у розробці та підтримці компонентної

архітектури, а також мінімальні витрати на серверну інфраструктуру і пристрої.

Об'єктом роботи є процес управління проєктами та завданнями в командах розробки програмного забезпечення. Ми зосереджені на створенні інтуїтивно зрозумілого веб-інтерфейсу для управління проєктами на базі Vue 3, щоб зробити роботу ще зручнішою та ефективнішою.

# 2 ПРОЄКТУВАННЯ ВЕБ-ДОДАТКУ

## 2.1 Вибір та обґрунтування технологічного стеку

Вибір технологій для проєкту базувався на кількох практичних вимогах: високій швидкості роботи інтерфейсу на мобільних пристроях з обмеженими ресурсами, зручності в розробці та підтримці компонентної архітектури, мінімальних витратах на серверну інфраструктуру та можливості швидкого розгортання.

Основу клієнтської частини становить фреймворк Vue 3. У порівнянні з попередньою версією, Vue 3 має оновлений рушій рендерингу, що забезпечує швидше оновлення інтерфейсу і менше навантаження на пристрій. Однією з головних переваг є Composition API — підхід до структури коду компонентів, який дає можливість винести логіку в окремі функції та повторно використовувати її в різних частинах застосунку.

Для збирання проєкту та організації середовища розробки використовується інструмент Vite [5]. Він забезпечує миттєве відображення змін у браузері під час розробки завдяки технології гарячої заміни модулів, що значно прискорює процес написання та налагодження коду.

Адаптивна верстка інтерфейсу реалізована за допомогою Tailwind CSS [4]. На відміну від класичного підходу з написанням окремих CSS-файлів, Tailwind надає набір готових класів, які прописуються безпосередньо в шаблонах компонентів, що дозволяє швидко реалізовувати адаптивні правила для різних розмірів екрана без розростання таблиць стилів.

Управління глобальним станом застосунку реалізовано за допомогою бібліотеки Pinia [2]. Всі дані зберігаються у трьох окремих глобальних сховищах: authStore відповідає за автентифікацію та сесії користувачів, boardStore зберігає дошки, колонки та завдання, а userStore керує профілями учасників команди.

Навігація між екранами реалізована за допомогою Vue Router [3]. Його інтеграція дозволила побудувати односторінковий застосунок, у якому перехід між розділами відбувається миттєво без перезавантаження сторінки в браузері. На мобільних пристроях це забезпечує поведінку, характерну для нативних застосунків.

Для реалізації функції перетягування карток між колонками Kanban-дошки підключено бібліотеку vuedraggable [7], яка повністю сумісна з Vue 3 та не потребує додаткових налаштувань для коректної роботи на сенсорних екранах.

Іконки інтерфейсу реалізовано за допомогою бібліотеки Lucide Vue [8], яка надає уніфікований набір векторних іконок у форматі SVG із можливістю гнучкого масштабування.

Для реалізації функції email-сповіщень використано сервіс EmailJS [6], який дозволяє надсилати листи з клієнтської частини застосунку через публічне API без необхідності розгортання власного поштового сервера. EmailJS застосовується для автоматичного повідомлення користувачів про створення нових проєктів і завдань.

### 2.2 Розробка архітектури системи

Архітектура спроєктована за принципом розподілу відповідальності між клієнтською частиною та хмарною інфраструктурою. Застосунок є односторінковим — інтерфейс завантажується один раз під час першого відкриття, а всі наступні переходи між екранами відбуваються без перезавантаження сторінки в браузері.

Клієнтська частина побудована на компонентній архітектурі Vue 3. Кожен екран застосунку складається з набору ізольованих компонентів, кожен з яких відповідає за конкретний елемент інтерфейсу: картку завдання, колонку Kanban-дошки, форму створення проєкту тощо. Компоненти не залежать один від одного безпосередньо — обмін даними між ними відбувається через централізовані сховища стану Pinia.

Маршрутизація між екранами організована за допомогою Vue Router. Застосунок має такі основні маршрути: сторінка входу та реєстрації, Kanban-дошка проєкту, персональний список завдань, сторінка команди та сторінка налаштувань.

Система складається з 4 основних сховищ. authStore відповідає за процеси реєстрації, входу та виходу користувачів, зберігає активну сесію та передає інформацію про поточного користувача іншим компонентам. boardStore є головним сховищем робочих даних: він містить список дошок, колонки й завдання, а також обробляє переміщення карток між колонками як за допомогою перетягування через vuedraggable, так і за допомогою функції moveTaskDirectly. userStore зберігає профілі всіх учасників команди, їхні посади та налаштування. notificationStore відповідає за систему сповіщень: формує тости, веде журнал подій у LocalStorage [12] та підраховує кількість непрочитаних повідомлень.

Постійне зберігання даних між сесіями реалізовано через LocalStorage браузера. Кожне сховище під час ініціалізації зчитує свої дані з відповідного ключа LocalStorage, а глибокий спостерігач watch відстежує будь-які зміни реактивного стану та автоматично записує оновлені дані назад у LocalStorage у форматі JSON. Така схема повністю імітує поведінку клієнт-серверної бази даних у межах локального середовища.

Маршрутизація між екранами організована за допомогою Vue Router. Застосунок має такі основні маршрути: сторінка входу та реєстрації, Kanban-дошка проєкту, персональний список завдань, сторінка команди та сторінка налаштувань.

Єдиний зовнішній сервіс у проєкті — EmailJS, що відправляє email-сповіщення через публічне API безпосередньо з клієнтського боку. Всі інші функції виконуються локально у браузері без мережевих запитів до власних серверів. Загальну схему компонентів системи наведено на рисунку 2.1.

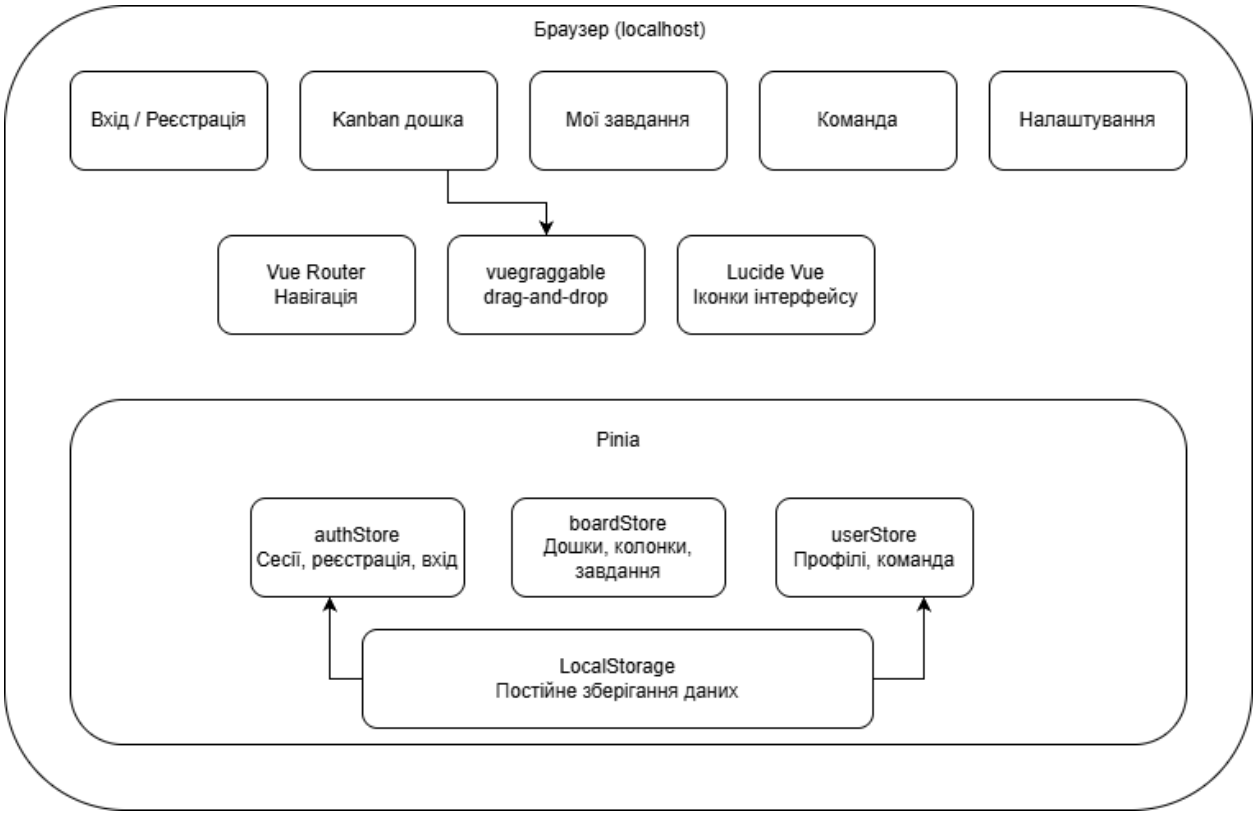


Рисунок 2.1 – Архітектура системи ProjectFlow

### 2.3 Моделювання процесів та функціональних схем

Для опису функціональних вимог і поведінки системи ProjectFlow застосовано два інструменти моделювання: діаграму прецедентів у нотації UML та модель бізнес-процесу у нотації BPMN 2.0.

Діаграма прецедентів ілюструє, як користувач взаємодіє з системою та її функціями. У рамках даного проєкту визначений один актор — зареєстрований користувач, який після входу отримує доступ до всіх можливостей застосунку. Сценарії прецедентів включають реєстрацію, вхід до системи, створення й видалення проєкту, перегляд Kanban-дошки, створення та редагування завдань, переміщення карток між колонками, перегляд особистого списку завдань, перегляд команди, редагування профілю користувача та налаштування email-сповіщень. Діаграма наведена на рисунку 2.2.


Рисунок 2.2 – Діаграма прецедентів UML системи ProjectFlow

Модель бізнес-процесу у нотації BPMN 2.0 описує сценарій роботи користувача з системою — від моменту відкриття застосунку до завершення роботи із завданням. Процес починається з перевірки активної сесії у

LocalStorage. У разі відсутності сесії, користувач потрапляє на сторінку входу, де вводить свої облікові дані. Система перевіряє їхню відповідність збереженим даним у authStore та, у разі успіху, надає доступ, або ж повертає повідомлення про помилку. Після успішного входу користувач потрапляє на Kanban-дошку активного проєкту. Далі він може обрати один із доступних сценаріїв: створити нове завдання, створити нову дошку, перемістити наявне завдання між колонками або перейти до іншого розділу застосунку. Під час створення завдання система зберігає його у boardStore, а за умов увімкнення email-сповіщень формує запит до EmailJS і надсилає листа на електронну пошту користувача. Процес завершується виходом з акаунту або закриттям браузера, при цьому дані залишаються збереженими у LocalStorage. Модель бізнес-процесу наведено на рисунку 2.3.

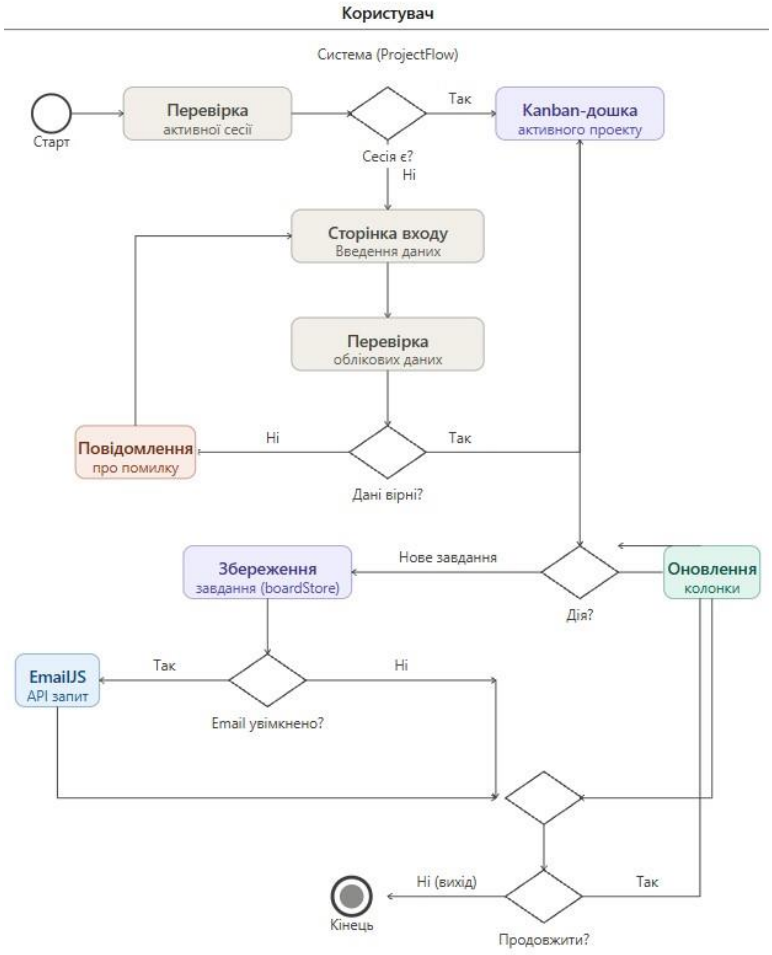


Рисунок 2.3 – Модель бізнес-процесу у нотації BPMN 2.0

Розроблені моделі допомагають чітко зафіксувати, що система має вміти робити та в якому порядку відбуваються основні процеси. Це корисно як на етапі проєктування, так і під час подальшої доробки чи розширення функціоналу.

### 2.4 Проєктування інтерфейсу користувача

Проєктування інтерфейсу розпочалося зі створення макетів у Figma ще до початку написання коду. Такий підхід дозволив заздалегідь визначити структуру кожного екрана, розташування елементів керування та поведінку інтерфейсу на різних розмірах екрана, не витрачаючи час на переробку вже написаного коду.

Під час розробки макетів враховувалися кілька ключових принципів. Перше — інтерфейс має бути зручним для користування як на широкому екрані, так і на смартфоні. Друге — навігація між розділами має бути інтуїтивною та зрозумілою без додаткових пояснень. Третє — кількість необхідних дій для основних операцій має бути мінімальною.

Візуальна концепція додатка створена так, щоб бути зручною і привабливою: темна бічна панель навігації гармонійно контрастує зі світлою робочою зоною. Це допомагає користувачу легко орієнтуватися та швидко знаходити потрібні елементи керування і основний контент. Головний акцентний колір — фіолетовий — яскраво підкреслює активні кнопки, підтвердження та поточний розділ у меню.

Бічна панель навігації містить 4 основні розділи: Дошка, Мої завдання, Команда та Налаштування. У нижній частині панелі відображаються аватар і ім'я поточного користувача. На вузьких екранах панель приховується, а навігація переходить у компактний режим.

Сторінка входу зліва містить брендинг та короткий опис можливостей додатка, справа — форму з полями email і пароль, функцією запам'ятовування та посиланням на реєстрацію.

Головний екран представляє собою горизонтальний ряд колонок: До виконання, В роботі, Тестування та Готово. У кожній колонці розміщені картки завдань з заголовком, описом, датою створення, дедлайном, рівнем пріоритету та аватаром відповідального учасника. Картки можна перетягувати між колонками або переміщати за допомогою контекстного меню. Вгорі розміщено назву поточного проекту з можливістю його перемикання, сортування за категоріями, а також кнопки для створення нової дошки та завдання.

Екран персональних завдань відображає лише ті завдання, які призначені поточному користувачу. Вони розділені на дві вкладки: Активні та Виконані.

Сторінка команди містить список усіх учасників проєкту з їхніми аватарами, іменами, посадами та електронними адресами. Кожен учасник представлений у вигляді картки з можливістю перегляду детальної інформації.

Сторінка налаштувань розділена на кілька блоків: редагування профілю з полями імені, посади та email, перемикач зовнішніх email-сповіщень, розділи сповіщень і зміни пароля, кнопка виходу з акаунту.

Макети розроблені для двох варіантів відображення: десктопного та мобільного. Екрани авторизації у Figma наведено на рисунку 2.4.

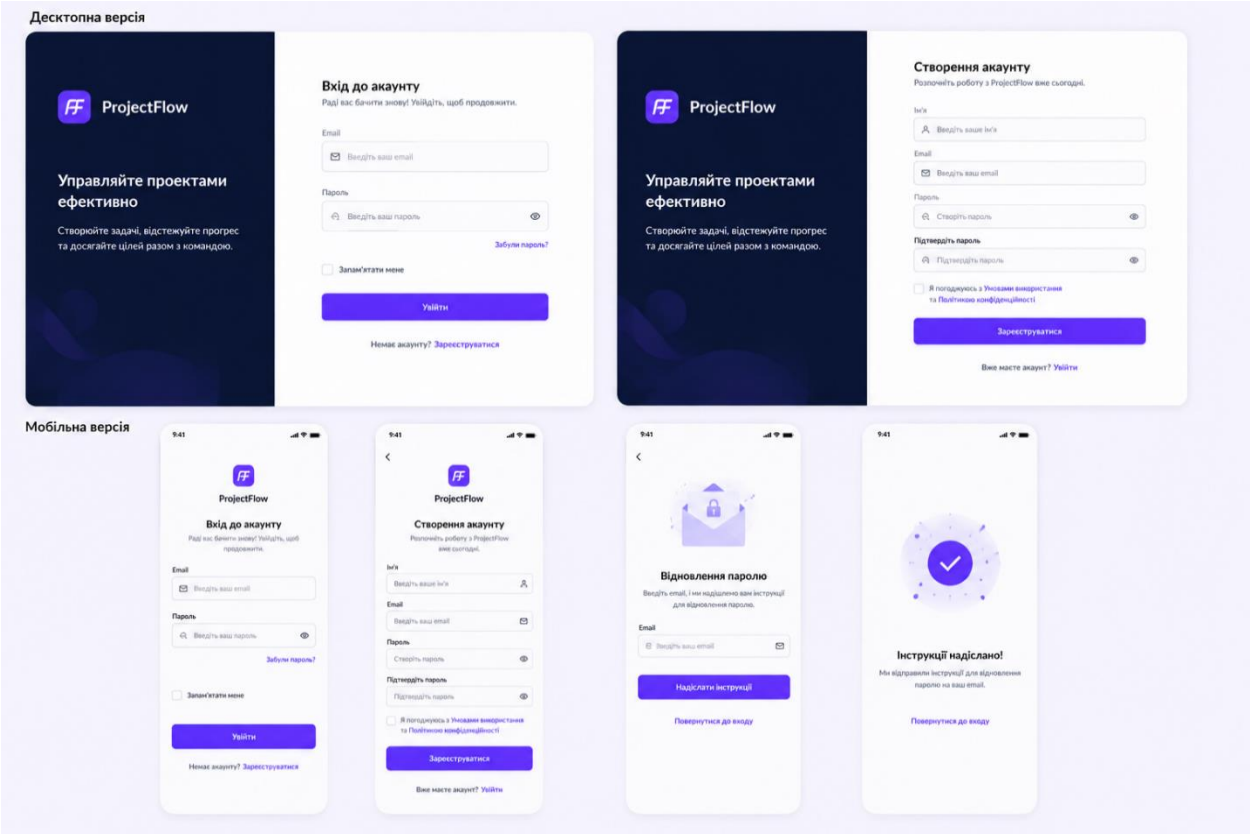


Рисунок 2.4 – Макети екранів авторизації (десктопна і мобільна версії)

Десктопна версія складається з двох частин: ліворуч розташований блок з логотипом і коротким описом додатка, праворуч — форма для введення email та пароля. На мобільних пристроях ці дві частини об'єднуються в один вертикальний екран: логотип розміщується зверху, а під ним — форма входу на всю ширину екрану. Також передбачено окремий екран відновлення пароля, де користувач вводить свій email і отримує підтвердження, що інструкції для скидання пароля надіслано.

Десктопна версія Kanban-дошки відображає 4 колонки з картками завдань, бокову навігаційну панель та верхню панель із назвою проєкту. У мобільній версії колонки перетворюються на горизонтальні вкладки, а навігація переноситься у бокове меню, яке відкривається жестом. Також представлені мобільні версії екранів Мої завдання, Команда та Налаштування. Макети основних робочих екранів наведені на рисунку 2.5.

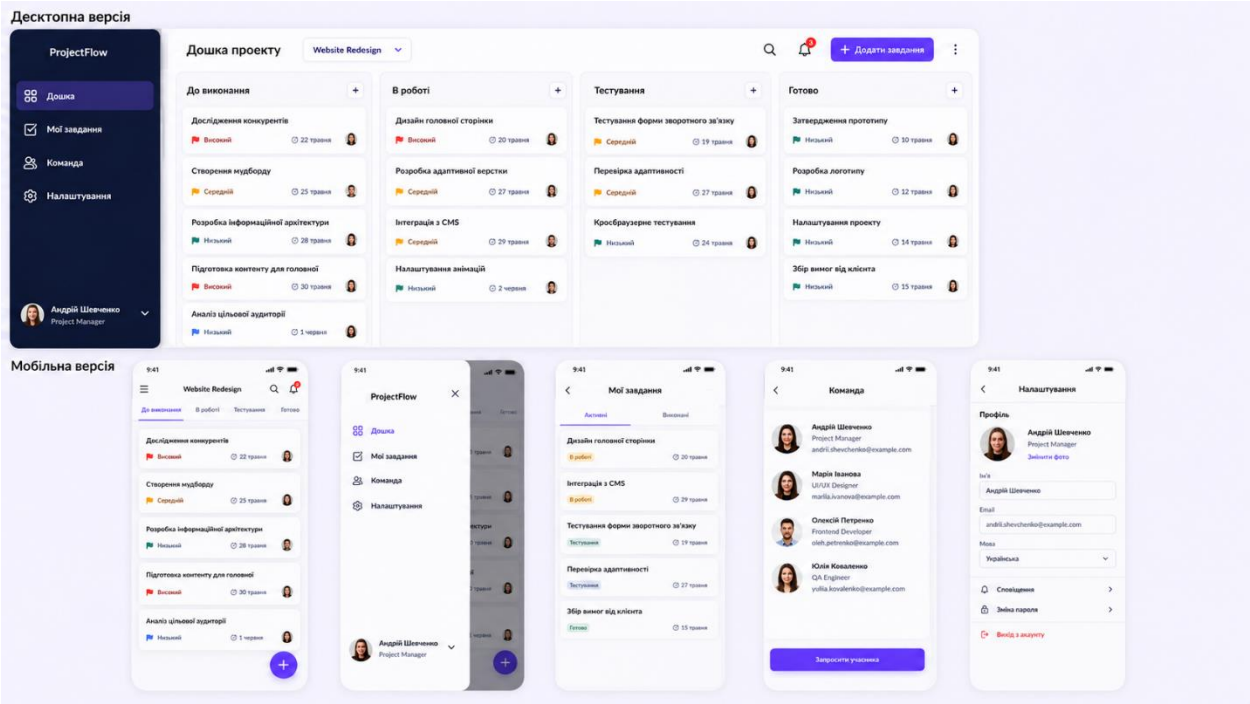


Рисунок 2.5 – Макети основних робочих екранів (десктопна і мобільна версії)

Розроблені у Figma макети стали основою для подальшої програмної реалізації: кожен елемент інтерфейсу, його розташування та поведінка на різних розмірах екрану були визначені ще до написання коду, що дозволило уникнути значних переробок верстки на етапі розробки та забезпечило візуальну узгодженість усіх екранів застосунку.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ДОДАТКУ

## 3.1 Структура та організація бази даних

Роль бази даних у нашому проєкті виконує механізм LocalStorage браузера разом із реактивними сховищами Pinia. Такий підхід дозволив нам повністю відмовитися від серверної інфраструктури та миттєво зберігати всі зміни без будь-яких мережевих затримок.

Дані зберігаються у вигляді набору незалежних ключів у LocalStorage. Кожен ключ відповідає окремій сутності системи та містить серіалізований JSON-рядок. Детальна інформація про ключі та їх призначення наведена у таблиці 3.1.

Таблиця 3.1 – Структура ключів LocalStorage у ProjectFlow

| Ключ | Тип даних | Призначення |
|---|---|---|
| projectflow_users | Масив об'єктів | Список зареєстрованих користувачів |
| projectflow_logged_in | Рядок (true/false) | Маркер активної сесії |
| projectflow_current_email | Рядок | Email поточного користувача |
| projectflow_boards | Масив об'єктів | Дошки, колонки та завдання |
| projectflow_active_board_id | Рядок | Ідентифікатор активної дошки |
| projectflow_notifications_history | Масив об'єктів | Журнал сповіщень |
| projectflow_email_notifications | Рядок (true/false) | Налаштування email-сповіщень |

Масив projectflow_users зберігає об'єкти, кожен з яких описує окремого зареєстрованого користувача. У кожному об'єкті знаходяться поля: name —

повне ім'я користувача, email — електронна пошта, яка є унікальним ідентифікатором, password — пароль у відкритому вигляді, `role` — посада учасника проєкту, та avatar — URL зображення профілю. Під час реєстрації система перевіряє, чи цей email унікальний у межах масиву. Якщо збігу не знайдено, додається новий запис і оновлений масив зберігається в LocalStorage. Аватар автоматично генерується через сервіс ui-avatars.com на основі імені під час реєстрації, але його можна замінити на особисте зображення у форматі base64.

Ключ projectflow_logged_in зберігає рядкове значення 'true' або 'false' і виконує роль маркера активної сесії. Саме це значення зчитує authStore під час ініціалізації, щоб визначити стан автентифікації. При успішному вході записується значення 'true', а при виході з акаунту запис повністю видаляється з LocalStorage. Vue Router перевіряє цей маркер перед кожним переходом між сторінками.

Ключ projectflow_current_email зберігає email-адресу користувача, який наразі залогінений у системі. Цей ключ використовується в userStore для визначення поточного користувача шляхом пошуку відповідного запису в масиві projectflow_users. При виході з акаунту цей запис також видаляється разом з маркером сесії.

Ключовий projectflow_boards — найбільший за обсягом, він зберігає всі робочі дані системи. Це масив об'єктів-дошок, кожна з яких має поля: id — унікальний ідентифікатор, title — назва дошки та columns — масив колонок. Кожна дошка містить 4 фіксовані колонки з ідентифікаторами: todo, in-progress, testing та done. Кожна колонка зберігає масив карток із завданнями, що мають поля: id у форматі task, title, description, priority, assignee, reporter, deadline та createdAt. Завдяки глибокому спостерігачу watch з параметром deep: true у boardStore будь-яка зміна в цій структурі автоматично серіалізується та зберігається у LocalStorage.

Ключ projectflow_active_board_id зберігає ідентифікатор дошки, яка відображається користувачу в цей момент. Під час завантаження застосунку

boardStore зчитує це значення та встановлює відповідну дошку як активну. Якщо збережений ідентифікатор не відповідає жодній з існуючих дошок, автоматично обирається перша дошка зі списку.

Ключ projectflow_notifications_history зберігає журнал усіх системних сповіщень у вигляді масиву об'єктів. Кожен запис має поля: id, message — текст повідомлення, type — тип (success або error), timestamp — час створення в форматі ISO й isRead — булевий прапор, що показує, чи переглянув користувач повідомлення. За цим прапором визначається лічильник непрочитаних сповіщень, який відображається на іконці дзвіночка у шапці застосунку.

Ключ projectflow_email_notifications зберігає рядкове значення 'true' або 'false' і визначає, чи надсилати листи через EmailJS під час створення дошок і завдань. Значення зчитується у boardStore перед кожним викликом функції sendRealEmail. Користувач може змінити це налаштування в розділі Налаштування через відповідний перемикач, після чого нове значення одразу записується в LocalStorage.

Синхронізація між Pinia-сховищами та LocalStorage реалізована через механізм глибокого спостерігача watch з параметром deep: true. Це означає, що будь-яка зміна у вкладеній структурі автоматично фіксується та записується в LocalStorage без явного виклику функції збереження.

### 3.2 Програмна реалізація основних функцій

Програмна реалізація ProjectFlow побудована на взаємодії чотирьох сховищ Pinia, кожне з яких відповідає за окрему функціональну область застосунку.

Структура проєкту побудована відповідно до стандартів Vue-додатків і включає декілька зручних директорій у папці src. Директорія views містить 5 компонентів-сторінок: AuthView, BoardView, TasksView, TeamView і SettingsView. Кожен з них відповідає за окремий екран і легко підключається до відповідного маршруту. У директорії stores розташовані 4 сховища Pinia: authStore, boardStore, userStore і notificationStore. Там же у папці router

знаходиться файл index.js, що описує маршрути й навігаційний охоронець. У директорії services знаходиться файл emailService.js з функцією для відправки листів через EmailJS. У папці components розміщені допоміжні компоненти, зокрема Sidebar.vue — зручна бічна навігаційна панель, що використовується на всіх захищених сторінках. Точка входу в застосунок — це файл main.js, який створює екземпляр Vue, підключає Pinia та Vue Router і монтує головний компонент App.vue в елемент з ідентифікатором app в index.html.

Автентифікація реалізована в authStore. При вході користувач вводить email і пароль, система шукає збіг у масиві зареєстрованих користувачів, збереженому у LocalStorage. У разі успіху зберігаються два маркери: булевий прапор сесії та email користувача, після чого відбувається перенаправлення на головну сторінку. Під час реєстрації спочатку перевіряється унікальність email, потім новий користувач додається до масиву і вхід виконується автоматично. Зміна пароля — окремий метод, що перевіряє поточний пароль перед збереженням нового.

Захист маршрутів організовано через router.beforeEach. Перед кожним переходом між сторінками зчитується значення isAuthenticated з authStore і приймається одне з трьох рішень: дозволити перехід, перенаправити незалогіненого користувача на сторінку входу або перенаправити вже авторизованого користувача на головну сторінку.

Сховище notificationStore реалізує дворівневу систему сповіщень. Перший рівень — це тости, невеликі спливаючі повідомлення в нижній частині екрана, які автоматично зникають через 4 секунди. Вони повідомляють користувача про результат поточної дії, наприклад, про успішне збереження завдання або помилку під час надсилання листа. Другий рівень — це постійний журнал подій, що зберігається в LocalStorage за ключем projectflow_notifications_history. Кожен запис журналу містить текст повідомлення, тип, час створення та прапорець isRead.

Управління дошками та завданнями зосереджено у boardStore. Створення нової дошки генерує унікальний ідентифікатор на основі поточного

часу та ініціалізує 4 порожні колонки зі стандартними назвами. Після створення нова дошка одразу стає активною. Додавання завдання відбувається через метод addTask, який приймає ідентифікатор цільової колонки та об'єкт з даними завдання. Дата створення фіксується автоматично у момент виклику методу.

Переміщення карток між колонками здійснюється двома зручними способами. Перший — з використанням бібліотеки vuedraggable, яка плавно обробляє перетягування та миттєво оновлює масиви завдань у кожній колонці. Другий — через метод moveTaskDirectly, створений спеціально для зручної роботи на мобільних пристроях: він знаходить необхідне завдання за ідентифікатором у вихідній колонці, швидко видаляє його та додає до цільової колонки за один виклик.

Сортування завдань у колонках доступне за трьома критеріями. Сортування за дедлайном розташовує картки від найближчого терміну до найвіддаленішого. Сортування за датою створення показує найновіші завдання на початку. Сортування за пріоритетом розташовує картки від найвищого пріоритету до найнижчого.

Система сповіщень реалізована у notificationStore і працює на двох рівнях. Перший рівень — тости, короткі повідомлення, що з'являються в нижній частині екрана й зникають через 4 секунди. Другий рівень — постійний журнал сповіщень, який зберігається в LocalStorage та доступний через іконку дзвіночка в шапці застосунку. Кожен запис журналу має прапор isRead, що дозволяє відображати лічильник непрочитаних сповіщень.

Відправка email-сповіщень у файлі emailService.js реалізована через функцію sendRealEmail, яка формує JSON-запит з ідентифікаторами сервісу та шаблону EmailJS, і надсилає його на API методом POST. Функція повертає true при успіху або false при помилках. Перед викликом функції перевіряється emailNotificationsEnabled з userStore для вмикання/вимикання сповіщень. Під час створення завдання лист надсилається виконавцю, а при створенні дошки — користувачеві.

Управління профілем реалізоване через userStore. Метод updateProfile оновлює ім'я, email і посаду одночасно в реактивному стані сховища та в масиві користувачів у LocalStorage. Якщо змінюється ім'я користувача, викликається додатково метод updateOwnerName з boardStore, який проходить по всіх дошках і колонках та оновлює поля виконавця і постановника у всіх завданнях, де використовувалося старе ім'я. Завантаження аватара здійснюється через конвертацію файлу у форматі base64, після чого цей рядок зберігається у полі avatar у LocalStorage.

### 3.3 Опис інтерфейсу додатку та методика роботи користувача

Інтерфейс ProjectFlow складається з п'яти основних екранів, між якими користувач переміщується через бічну навігаційну панель. На десктопі панель завжди видима зліва, на мобільних пристроях вона згортається і відкривається через іконку меню у верхньому куті екрану. У нижній частині панелі відображаються аватар та ім'я поточного користувача, що дозволяє одразу розуміти, під яким акаунтом виконується робота.

Сторінка авторизації відкривається автоматично, якщо користувач ще не залогінений. Вона поділена на дві частини: ліва містить темний фон з логотипом та коротким описом можливостей застосунку, права — форму входу з полями email і пароль. Під полем пароля є посилання «Забули пароль?», яке веде на окремий екран відновлення доступу. Якщо акаунту ще немає, внизу форми є посилання «Зареєструватися». При введенні невірних даних під кнопкою входу з'являється повідомлення про помилку з поясненням причини. Сторінку авторизації наведено на рисунку 3.1.

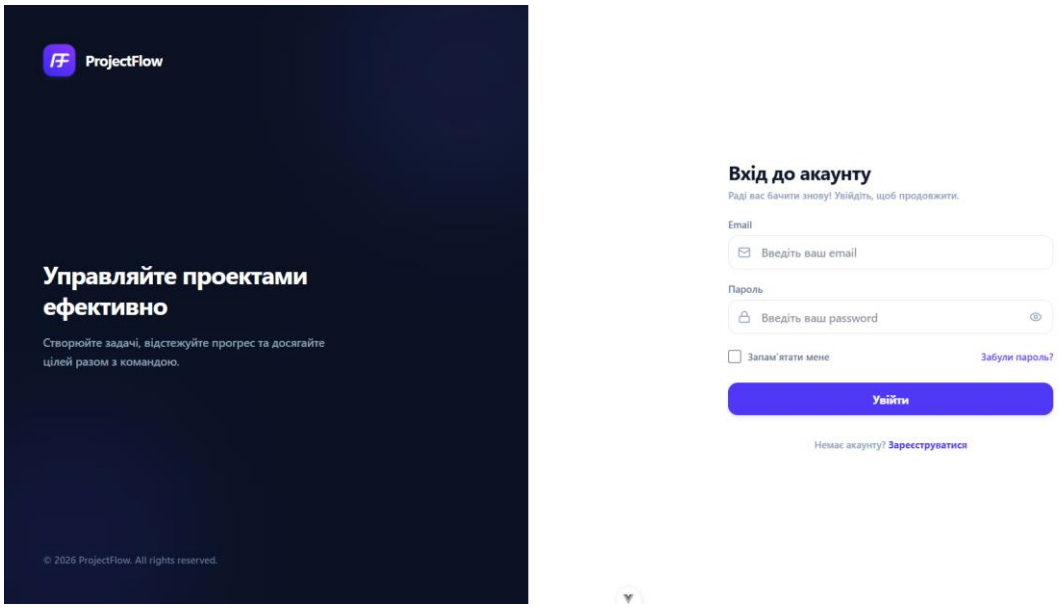


Рисунок 3.1 – Сторінка авторизації ProjectFlow

Сторінка реєстрації має схожий вигляд, проте форма містить більше полів: ім'я, електронна пошта, пароль і підтвердження пароля. Після заповнення всіх полів і натискання кнопки «Зареєструватися» система перевіряє, чи вже існує акаунт з цим email. Якщо пошта вільна, новий користувач одразу потрапляє на головну сторінку без додаткового кроку входу. Аватар автоматично генерується з введеного імені. Сторінку реєстрації показано на рисунку 3.2.

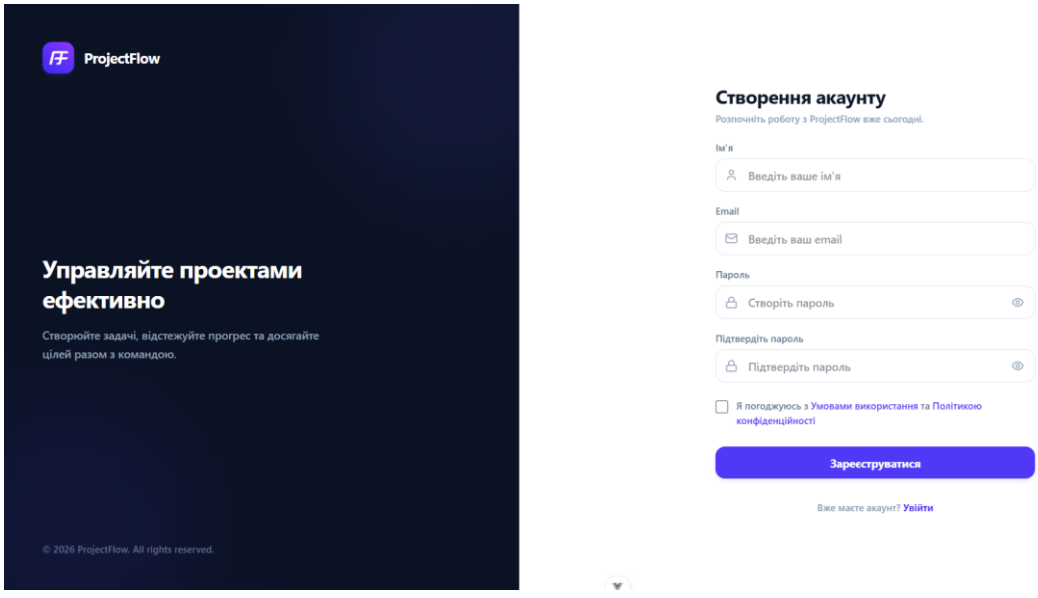


Рисунок 3.2 – Сторінка реєстрації ProjectFlow

Головний екран застосунку — Kanban-дошка. Саме тут відбувається основна робота над проєктом. У верхній частині екрану відображається назва активного проєкту, поруч із якою є кнопка для перемикання між існуючими дошками через випадаючий список. Там само розташовані кнопки «Нова дошка» для створення нового проєкту та «Завдання» для додавання нової картки. Праворуч у шапці знаходиться іконка дзвіночка з числовим лічильником непрочитаних сповіщень. Канбан-дошку наведено на рисунку 3.3.

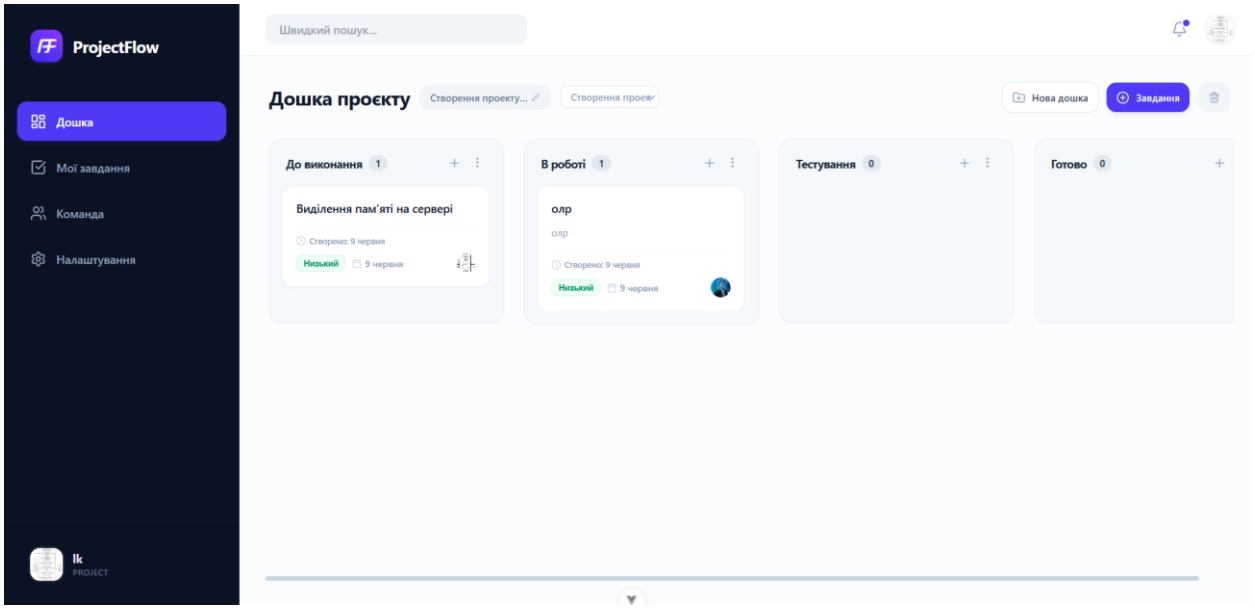


Рисунок 3.3 – Kanban-дошка проєкту

Робоча область дошки складається з чотирьох колонок: До виконання, В роботі, Тестування та Готово. Кожна колонка має заголовок з лічильником карток та кнопку додавання завдання безпосередньо в цю колонку. У заголовку також є меню з трьома крапками, через яке доступні опції сортування та очищення колонки.

Кожна картка завдання відображає такі дані: заголовок завдання, короткий опис, дату створення, дедлайн у вигляді іконки календаря з датою, рівень пріоритету у вигляді кольорової мітки та аватар відповідального виконавця. Пріоритет позначається кольором: червоний — високий, жовтий — середній, зелений — низький. Натискання на картку відкриває модальне вікно

з повними деталями завдання та можливістю його редагування. Модальне вікно завдання наведено на рисунку 3.4.

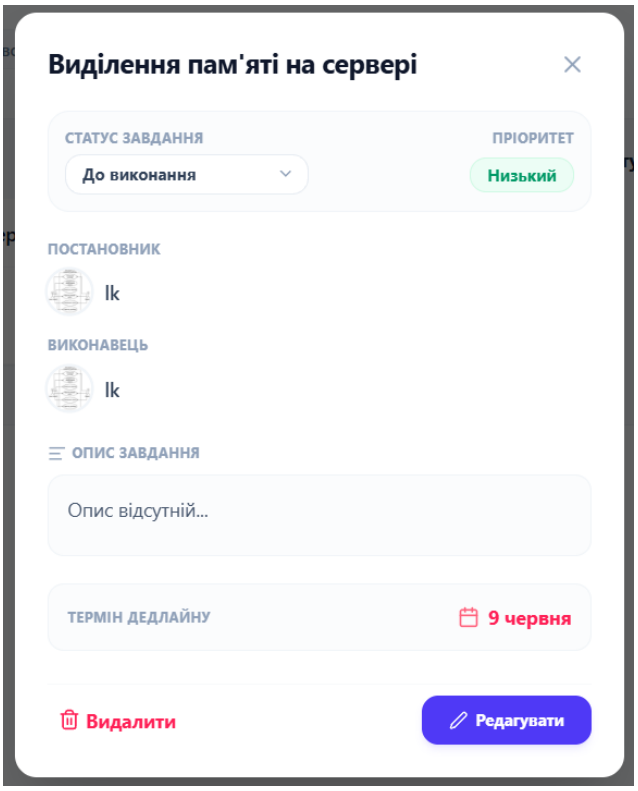


Рисунок 3.4 – Модальне вікно завдання

Щоб створити нове завдання, відкривається форма з полями: назва, опис, пріоритет (у вигляді випадаючого списку), виконавець, постановник і дедлайн. Після збереження картка миттєво з'являється в відповідній колонці. Якщо в профілі активовано email-сповіщення, виконавцю автоматично надходить лист із назвою завдання, ім'ям постановника та датою дедлайну.

Переміщення карток між колонками на десктопі виконується перетягуванням: користувач затискає картку і перетягує її в потрібну колонку. На мобільних пристроях перетягування замінено на кліковий механізм: при натисканні на картку з'являється меню, де можна обрати цільову колонку одним дотиком. Після переміщення у журналі сповіщень з'являється запис про те, з якої колонки в яку було переміщено завдання.

У верхній частині кожного екрана розташований рядок швидкого пошуку. Під час введення тексту він фільтрує картки на активній дошці за заголовком завдання в реальному часі, приховуючи ті, що не відповідають запиту. Пошук працює без затримок і не потребує натискання кнопки підтвердження.

При натисканні на іконку дзвіночка в шапці відкривається панель сповіщень із повним журналом подій. Кожен запис показує текст події та час її виникнення. Непрочитані сповіщення виділені візуально. Користувач може позначити окреме сповіщення як прочитане, видалити його або очистити весь журнал одразу. Після перегляду лічильник на іконці дзвіночка скидається до нуля. Екран панелі сповіщень наведено на рисунку 3.5.

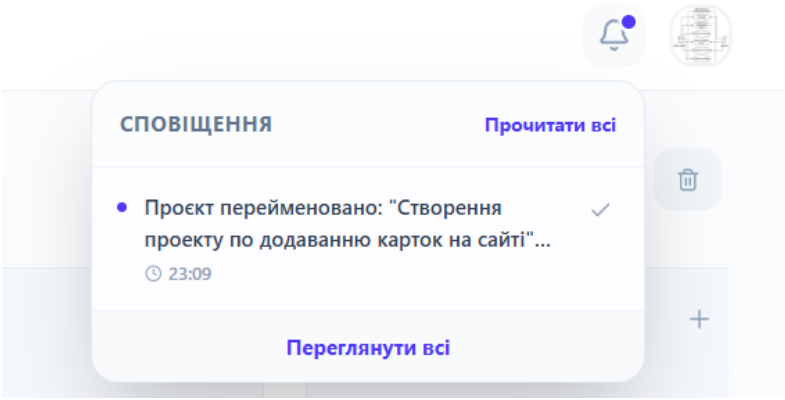


Рисунок 3.5 – Екран панелі сповіщень

Екран персональних завдань показує лише ті завдання, для яких поточний користувач призначений виконавцем. Вони розподілені на дві вкладки: «Активні» та «Виконані». До активних належать картки з колонок До виконання, В роботі та Тестування, до виконаних — картки з колонки Готово. Поруч із назвою кожної вкладки відображається кількість завдань у ній. Якщо завдань немає, відображається відповідне повідомлення із зображенням порожньої скриньки. Екран персональних завдань наведено на рисунку 3.6.

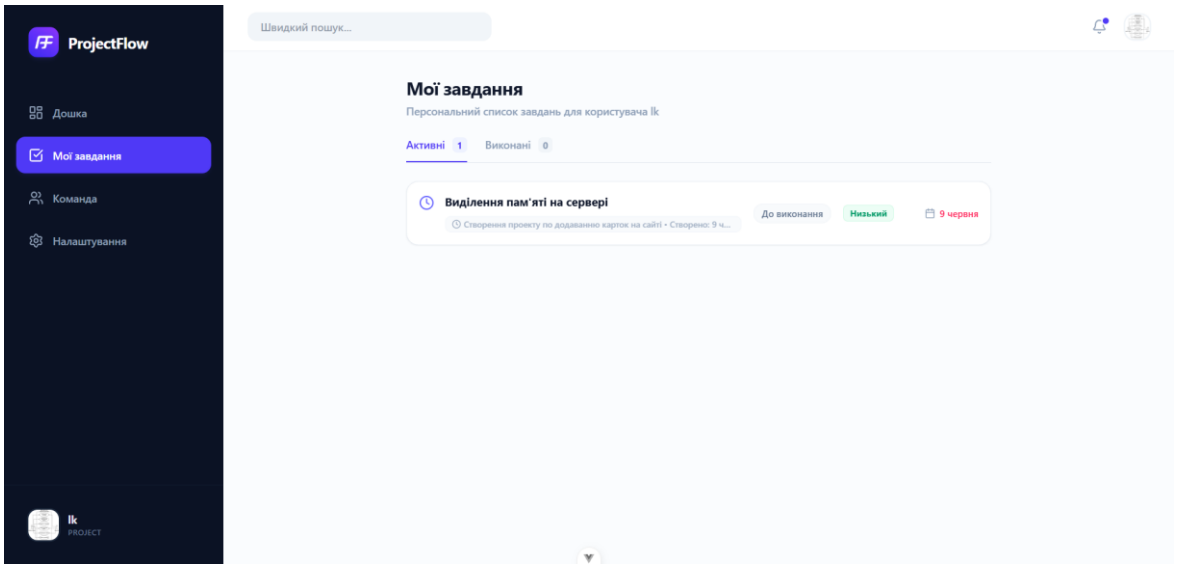


Рисунок 3.6 – Екран персональних завдань

Екран команди відображає всіх учасників, зареєстрованих у системі. Кожен учасник представлений карткою з аватаром, іменем, посадою та електронною адресою. Список створюється автоматично на основі масиву користувачів з LocalStorage, тому нові учасники миттєво з'являються у списку після реєстрації без потреби будь-яких додаткових дій з боку адміністратора. Натискання на картку учасника відкриває детальну панель з усією потрібною інформацією. Екран команди ілюстрований на рисунку 3.7.

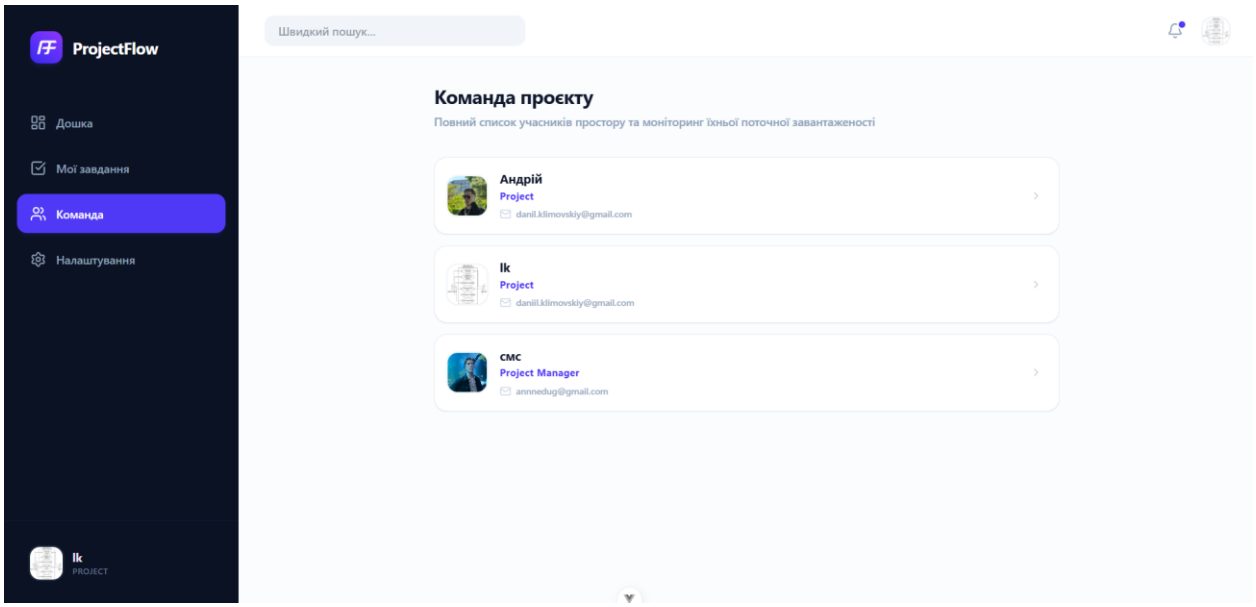


Рисунок 3.7 – Екран команди проєкту

Екран налаштувань дозволяє керувати особистим профілем і параметрами застосунку. У верхньому блоці відображається поточний аватар з кнопкою для завантаження нового зображення, а також поля для редагування імені, посади та email. Зміна імені впливає не лише на профіль, а й на всі завдання в системі: скрізь, де користувач вказаний як виконавець або постановник, ім'я оновлюється автоматично. Верхню частину сторінки налаштувань наведено на рисунку 3.8.

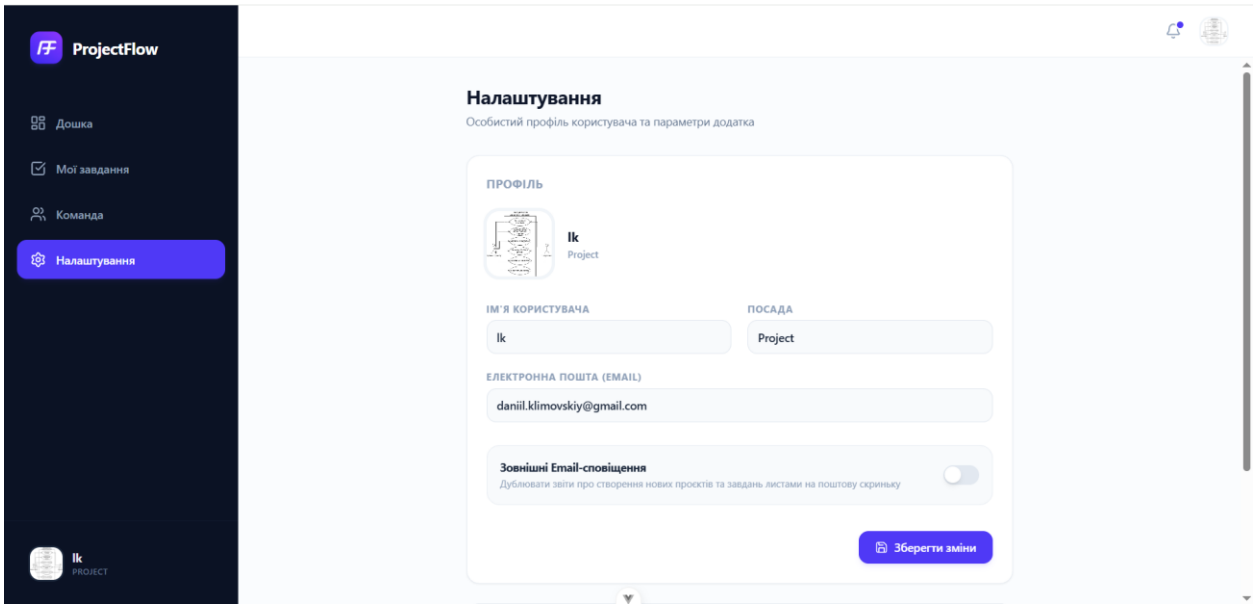


Рисунок 3.8 – Екран налаштувань користувача, блок профілю

При прокручуванні екрану вниз розташований перемикач зовнішніх email-сповіщень. Коли він увімкнений, під час створення нових дошок і завдань система надсилає реальні листи через сервіс EmailJS. Коли він вимкнений, сповіщення відображаються лише в журналі застосунку. Нижче розташовані кнопка збереження змін, а також окремі пункти меню, що відкривають розділ перегляду журналу сповіщень і форму зміни пароля. У самому низу сторінки знаходиться кнопка виходу з акаунту, яка очищує маркери сесії та повертає користувача на сторінку входу. Нижню частину сторінки налаштувань наведено на рисунку 3.9.

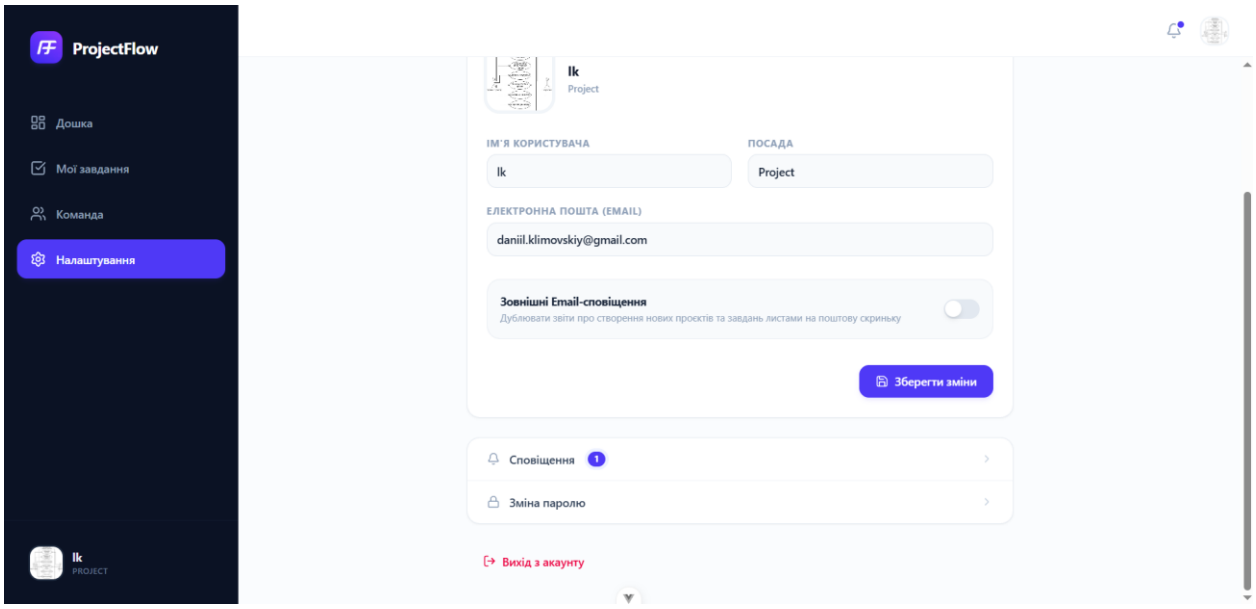


Рисунок 3.9 – Сторінка налаштувань, сповіщення та вихід з акаунту

Перехід до журналу сповіщень здійснюється через відповідний пункт у налаштуваннях або через іконку дзвіночка у шапці застосунку. Журнал відображає хронологічний список усіх подій системи: створення та видалення дошок, додавання завдань, переміщення карток між колонками, зміну профілю. Кожен запис містить текст події та час її виникнення. Непрочитані сповіщення позначені кольоровою міткою. У верхній частині екрана розташовані кнопки повернення до налаштувань та очищення всього журналу одним натисканням. Також є можливість видалити повідомлення або позначити його як прочитане. Екран журналу сповіщень наведено на рисунку 3.10.

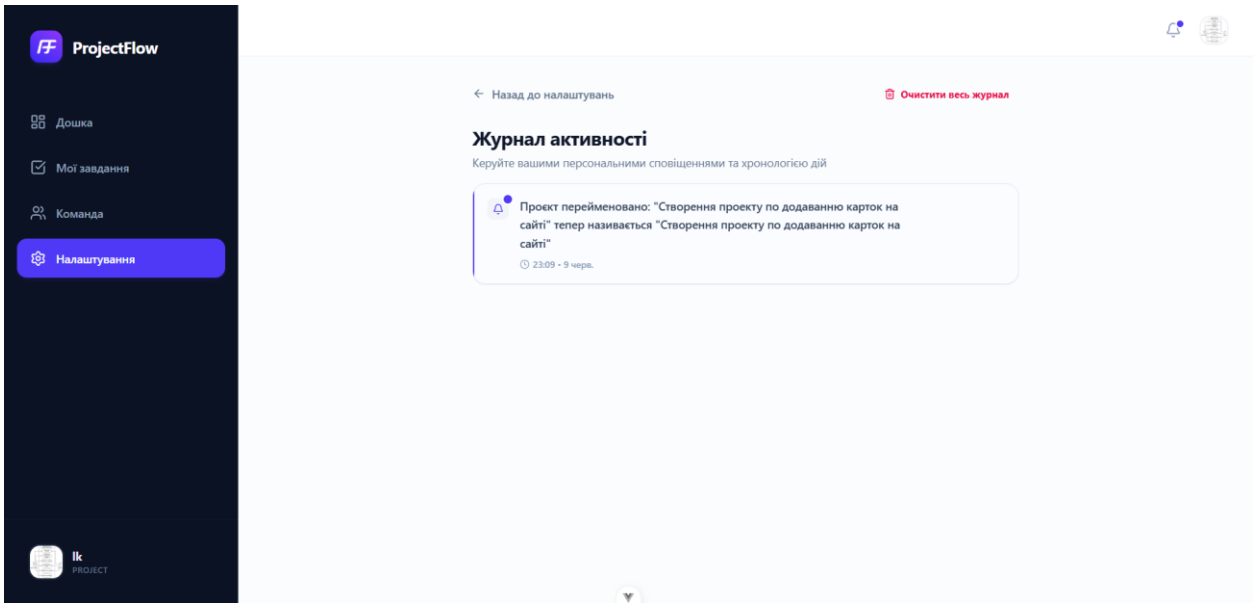


Рисунок 3.10 – Журнал сповіщень

Форма зміни пароля доступна окремим пунктом у налаштуваннях і містить три поля: поточний пароль, новий пароль та підтвердження нового пароля. Кожне поле має іконку ока, що дозволяє тимчасово показати введений текст для перевірки правильності. Перед збереженням нового пароля система перевіряє коректність поточного пароля і лише після успішної перевірки записує оновлене значення в LocalStorage. Форму зміни пароля наведено на рисунку 3.11.

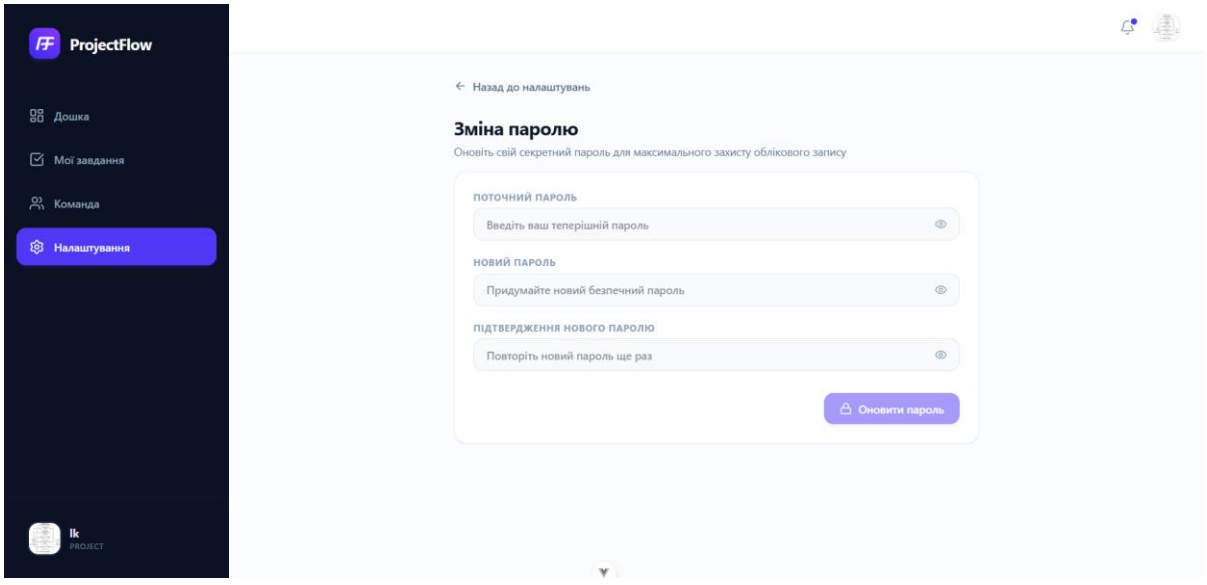


Рисунок 3.11 – Форма зміни паролю

Таким чином, інтерфейс ProjectFlow побудований так, що кожен з п'яти основних екранів вирішує одну чітко визначену задачу, а перехід між ними займає не більше одного натискання через бічну панель навігації. Це робить роботу з додатком зрозумілою навіть для користувача, який звертається до системи вперше.

### 3.4 Тестування адаптивності інтерфейсу на мобільних пристроях

Для перевірки якості адаптивної верстки проведено тестування ProjectFlow на екрані з шириною, характерною для типового смартфона. Метою тестування було підтвердити, що всі функції, доступні на десктопі, повноцінно працюють і на малому екрані без втрати зручності використання.

Сторінки авторизації та реєстрації на мобільному екрані перебудовуються у вертикальний одноколонковий формат. Двочастинний split-layout, характерний для десктопної версії, замінюється на компактну форму з логотипом у верхній частині та полями введення нижче. Усі елементи керування — поля email і пароль, чекбокс «Запам'ятати мене», кнопка «Увійти» — залишаються повністю функціональними та зручними для натискання пальцем. Мобільну версію сторінки входу наведено на рисунку 3.12.

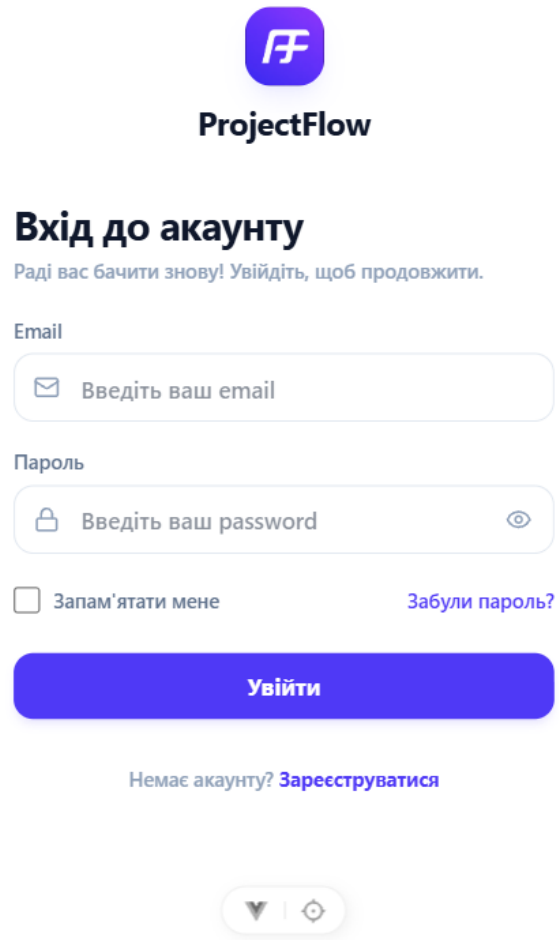


Рисунок 3.12 – Мобільна версія: сторінка входу

Аналогічно перебудовується форма реєстрації: поля імені, email, пароля та підтвердження пароля розташовуються одне під одним на всю ширину екрану, а чекбокс згоди з умовами використання залишається легко доступним для натискання. Форму реєстрації на мобільному екрані наведено на рисунку 3.13.

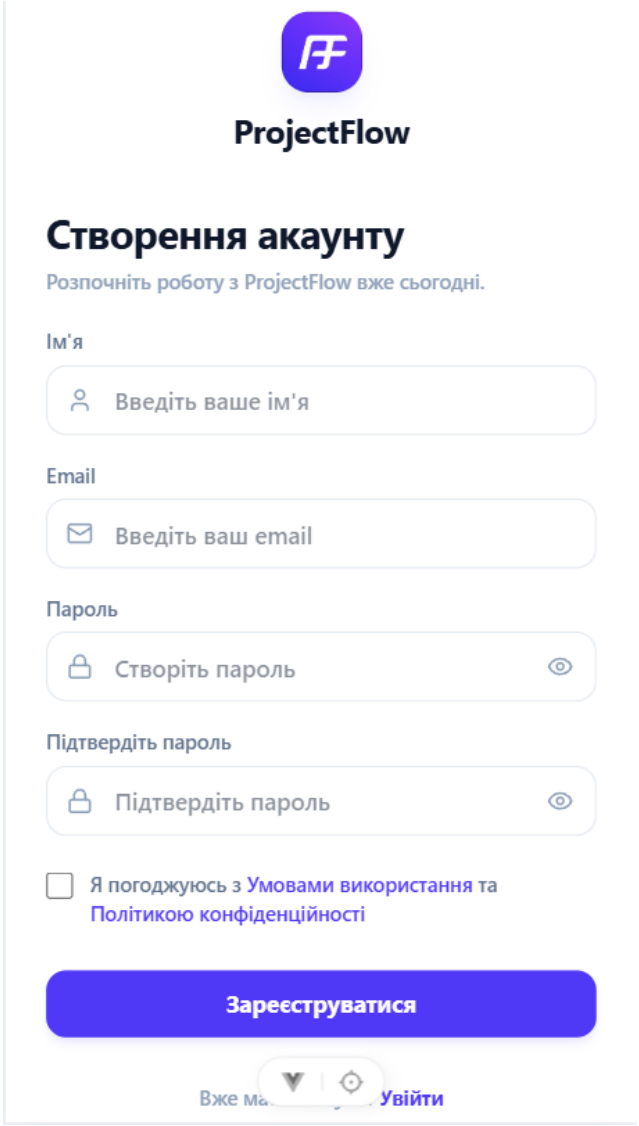


Рисунок 3.13 – Мобільна версія: сторінка реєстрації

Бічна навігаційна панель, яка на десктопі постійно відображається зліва, на мобільному пристрої прихована за замовчуванням і відкривається через іконку меню у верхньому лівому куті екрану. Панель з'являється поверх основного контенту у вигляді висувної шторки з затемненим фоном позаду, що є стандартним патерном мобільної навігації. Усі 4 пункти меню — Дошка, Мої завдання, Команда, Налаштування — залишаються доступними, а закрити панель можна через кнопку у вигляді хрестика у її верхній частині. Висувне навігаційне меню наведено на рисунку 3.14.

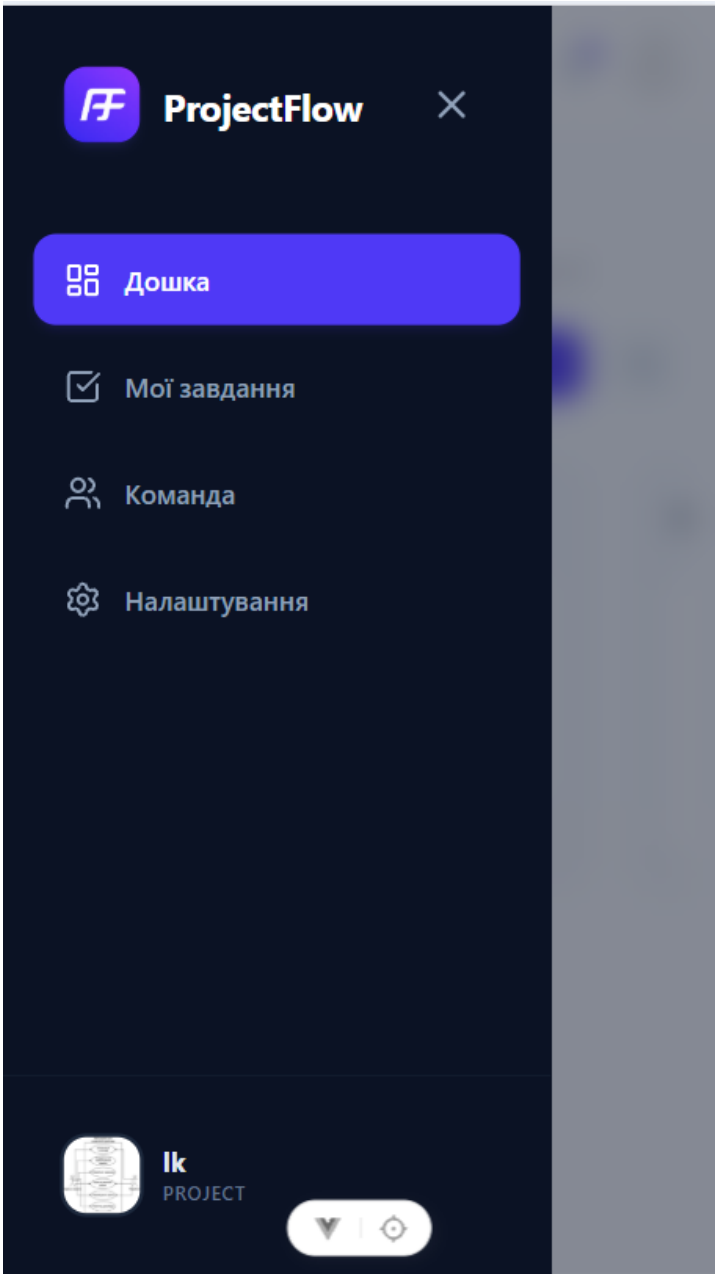


Рисунок 3.14 – Мобільна версія: висувне навігаційне меню

Kanban-дошка на мобільному екрані відображає колонки послідовно зі скролом, а не в один ряд, як на десктопі. Кожна картка завдання зберігає всю інформацію: заголовок, дату створення, рівень пріоритету та дедлайн. Кнопки «Нова дошка» та «Завдання» залишаються доступними у верхній панелі керування. Мобільний вигляд Kanban-дошки наведено на рисунку 3.15.

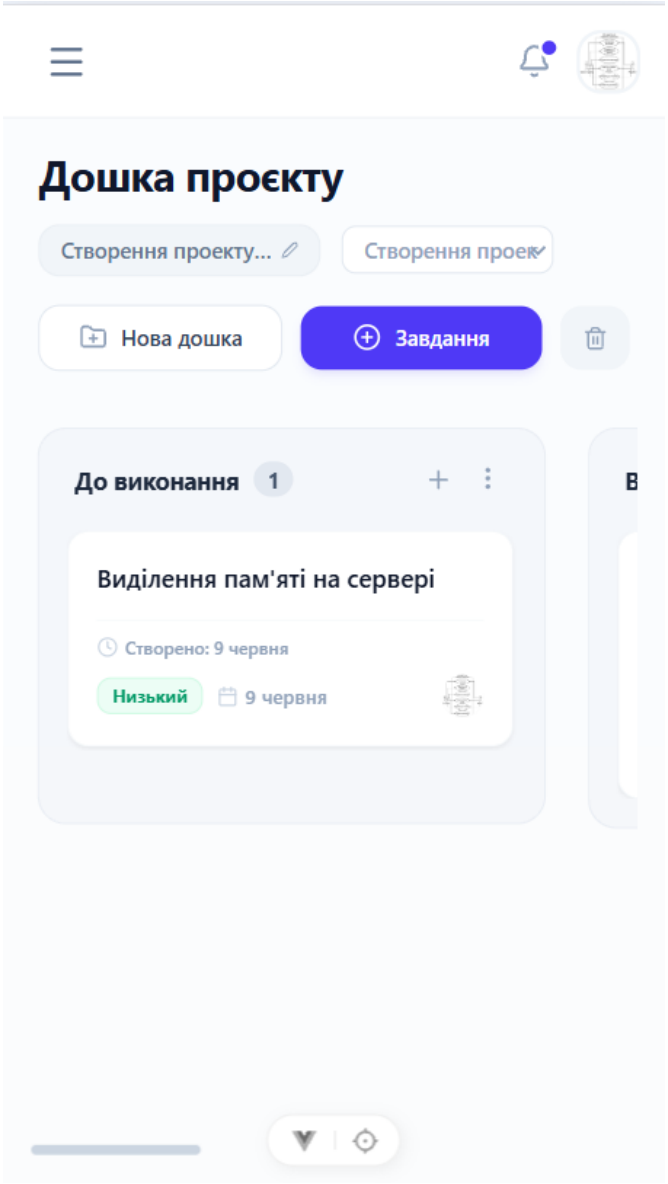


Рисунок 3.15 – Мобільна версія: Kanban-дошка

Модальне вікно створення завдання адаптується під ширину екрану: усі поля форми — назва, опис, пріоритет, дедлайн, виконавець, постановник — розташовуються вертикально та залишаються зручними для заповнення, а кнопки «Скасувати» та «Зберегти» розміщені в нижній частині вікна. Форму створення завдання на мобільному екрані наведено на рисунку 3.16.

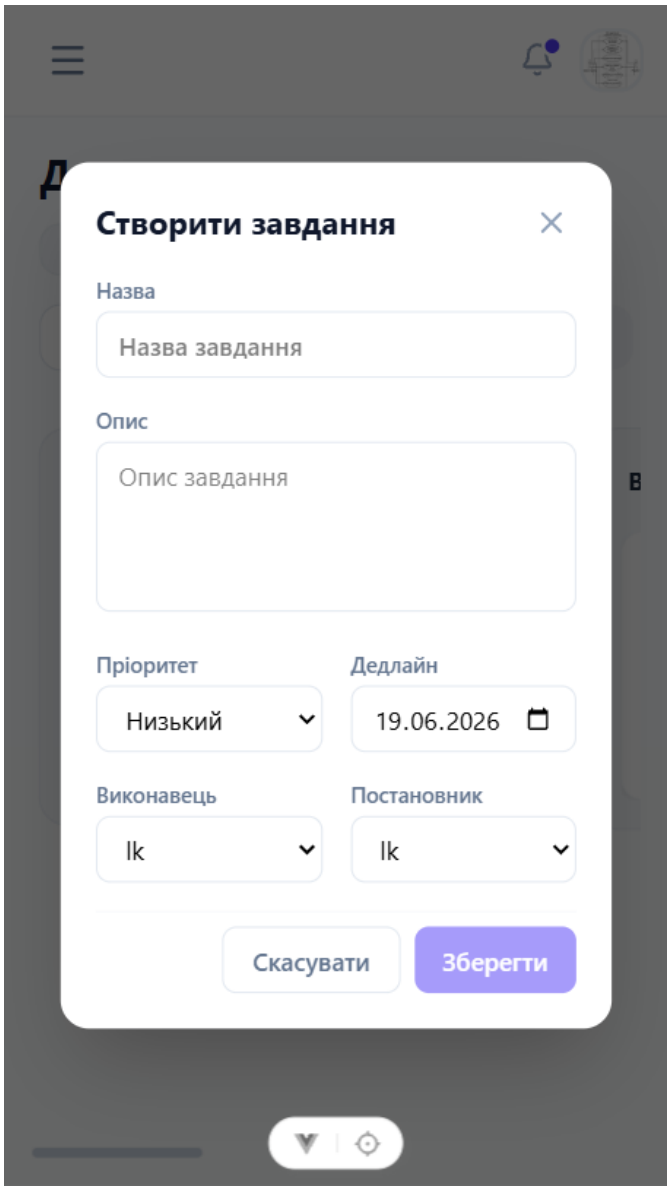


Рисунок 3.16 – Мобільна версія: створення завдання

Екран персональних завдань на мобільному пристрої зберігає поділ на вкладки Активні та Виконані з лічильниками кількості завдань у кожній. Картка завдання відображає всі ключові дані: назву, короткий опис, статус колонки, рівень пріоритету та дедлайн, при цьому текст не обрізається і повністю вміщується на екрані. Екран персональних завдань наведено на рисунку 3.17.

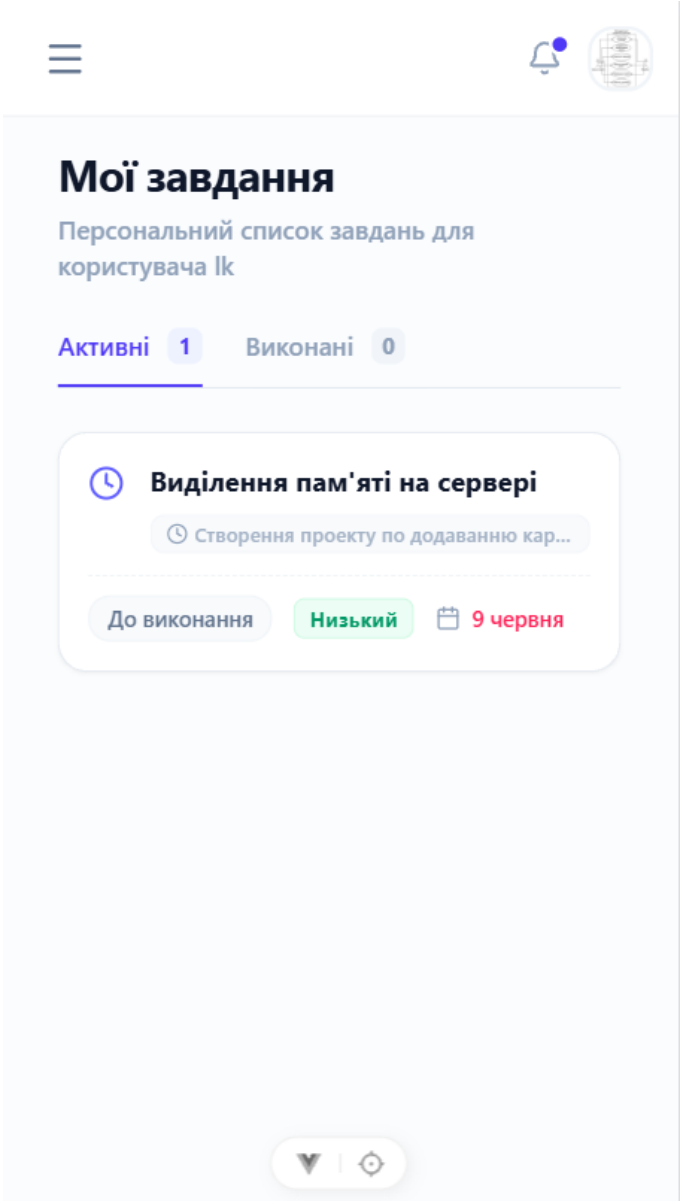


Рисунок 3.17 – Мобільна версія: персональні завдання

Сторінка команди на малому екрані показує список учасників у вигляді вертикальних карток на всю ширину, кожна з яких містить аватар, ім'я, посаду та email учасника. Список команди на мобільному екрані наведено на рисунку 3.18.

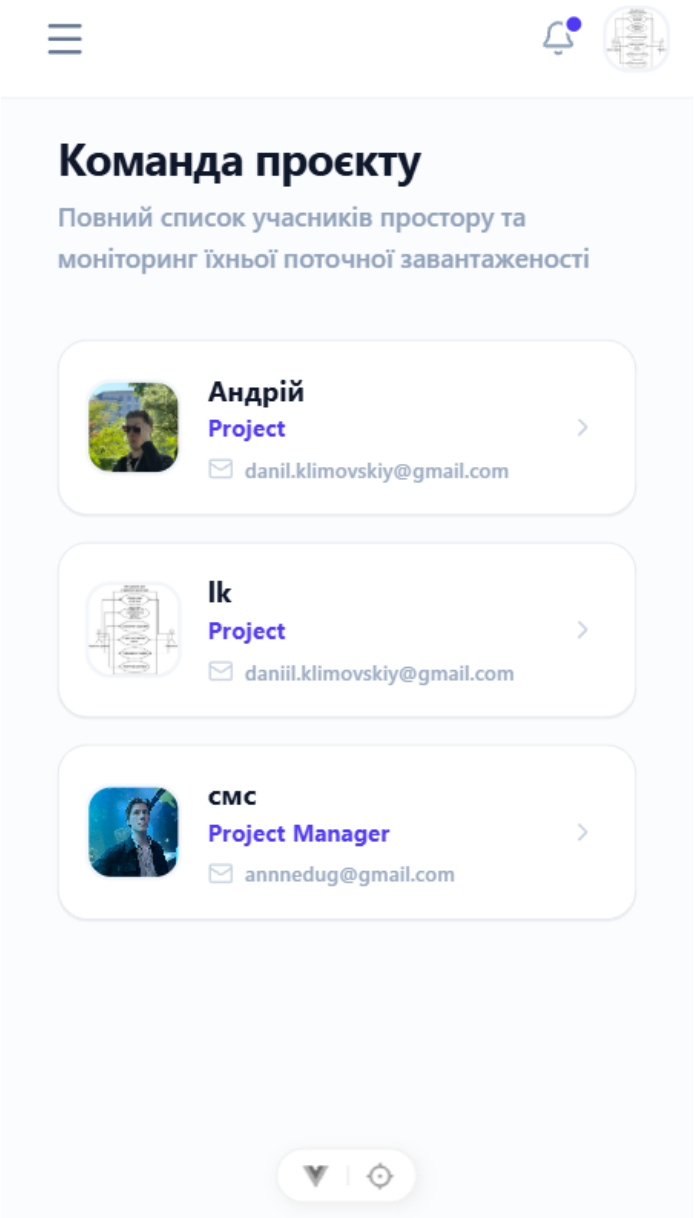


Рисунок 3.18 – Мобільна версія: список команди

При натисканні на картку учасника відкривається модальне вікно з детальною інформацією: збільшеним аватаром, ім'ям, посадою, email та блоком активних завдань цього учасника. Якщо активних завдань немає, відображається відповідне порожнє повідомлення. Детальну картку учасника команди наведено на рисунку 3.19.

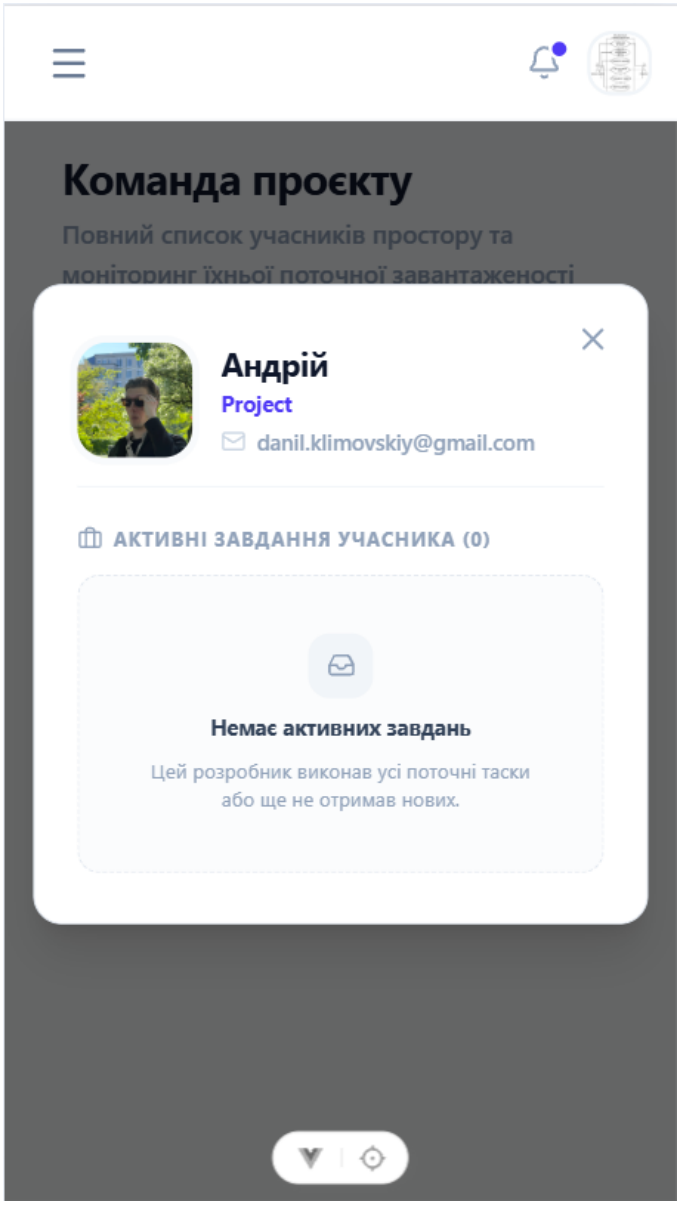


Рисунок 3.19 – Мобільна версія: детальна інформація про учасника команди

Сторінка налаштувань на мобільному пристрої зберігає повну функціональність десктопної версії, розподілену по вертикалі. Верхня частина екрану містить блок профілю з аватаром, ім'ям, посадою та полями для редагування. Початок сторінки налаштувань наведено на рисунку 3.20.

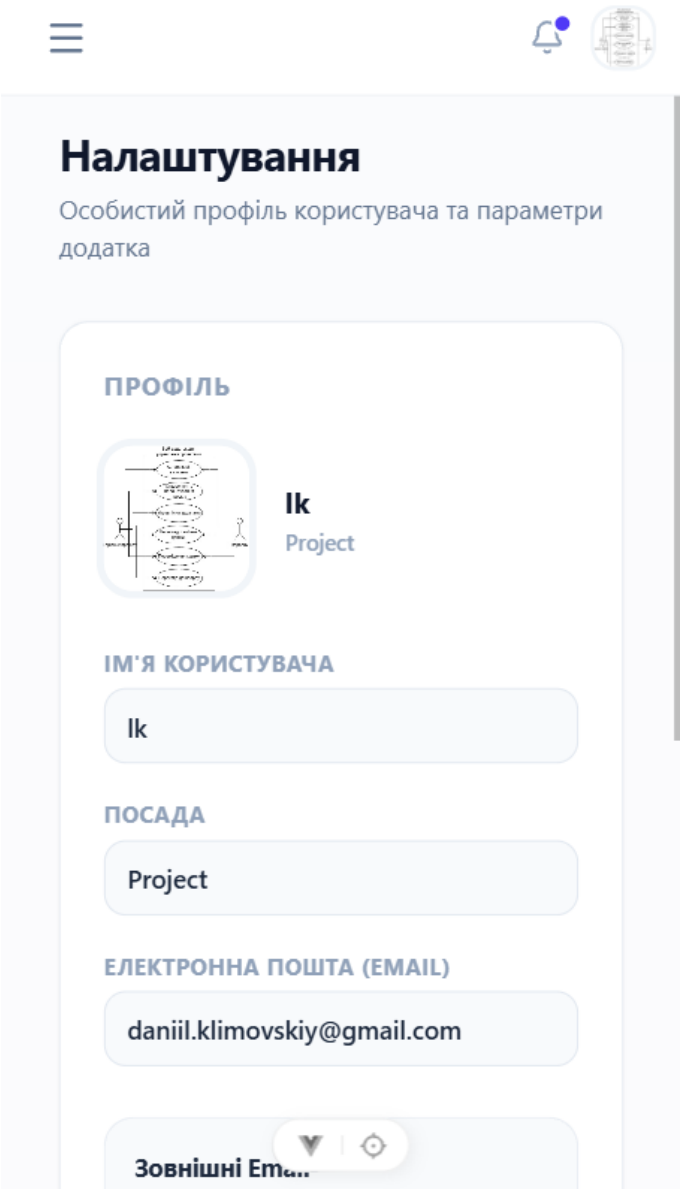


Рисунок 3.20 – Мобільна версія: сторінка налаштувань, блок профілю

При прокручуванні екрану вниз стають доступними поле електронної пошти, перемикач зовнішніх email-сповіщень через EmailJS, кнопка збереження змін, а також пункти переходу до журналу сповіщень, зміни пароля та виходу з акаунту. Нижню частину сторінки налаштувань наведено на рисунку 3.21.

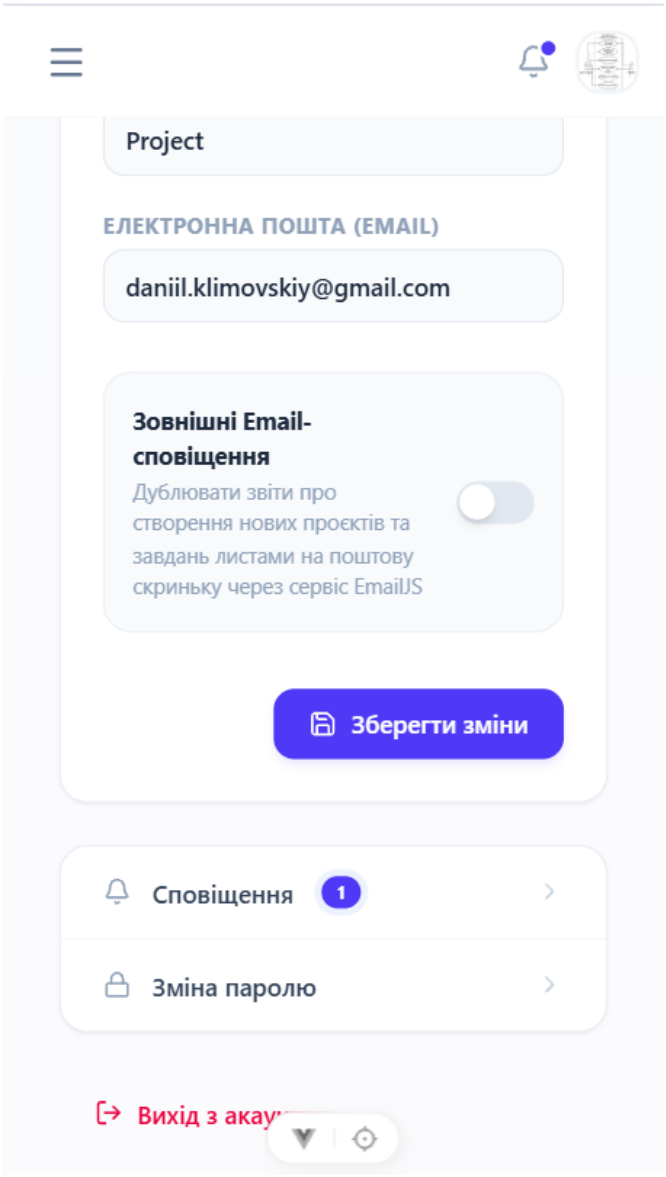


Рисунок 3.21 – Мобільна версія: сторінка налаштувань, сповіщення та вихід

Перехід до журналу активності відкриває хронологічний список подій із датою та часом кожного запису, а також кнопки для очищення журналу та повернення до налаштувань. Журнал активності на мобільному екрані наведено на рисунку 3.22.

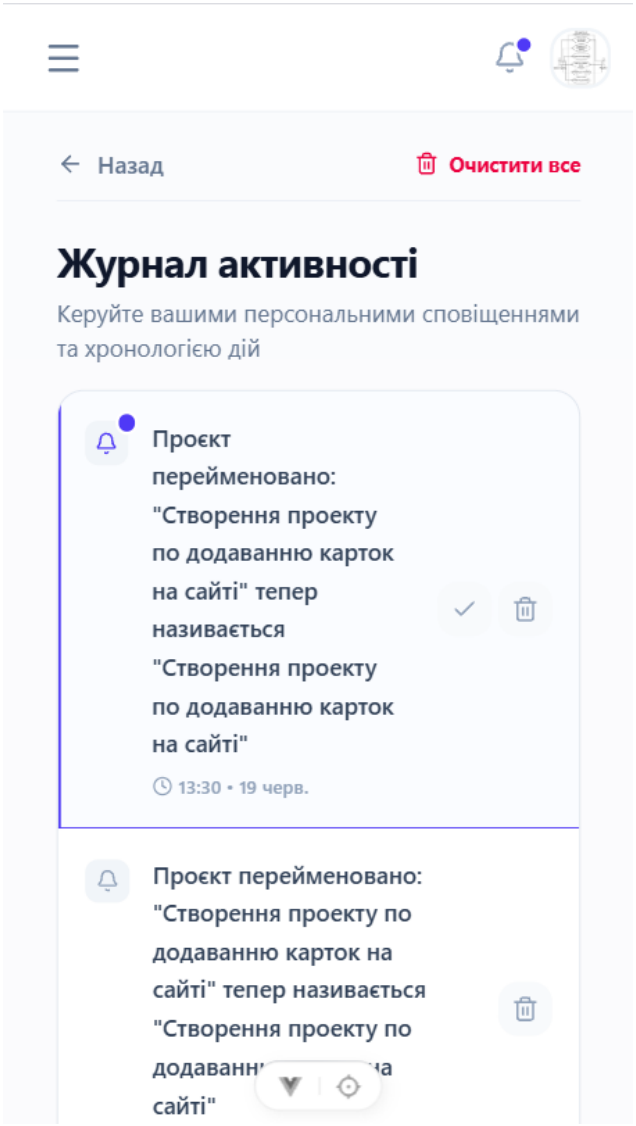


Рисунок 3.22 – Мобільна версія: журнал активності

Форма зміни пароля містить три поля: поточний пароль, новий пароль та підтвердження нового пароля, кожне з можливістю показати або приховати введений текст через іконку ока. Форму зміни пароля на мобільному екрані наведено на рисунку 3.23.

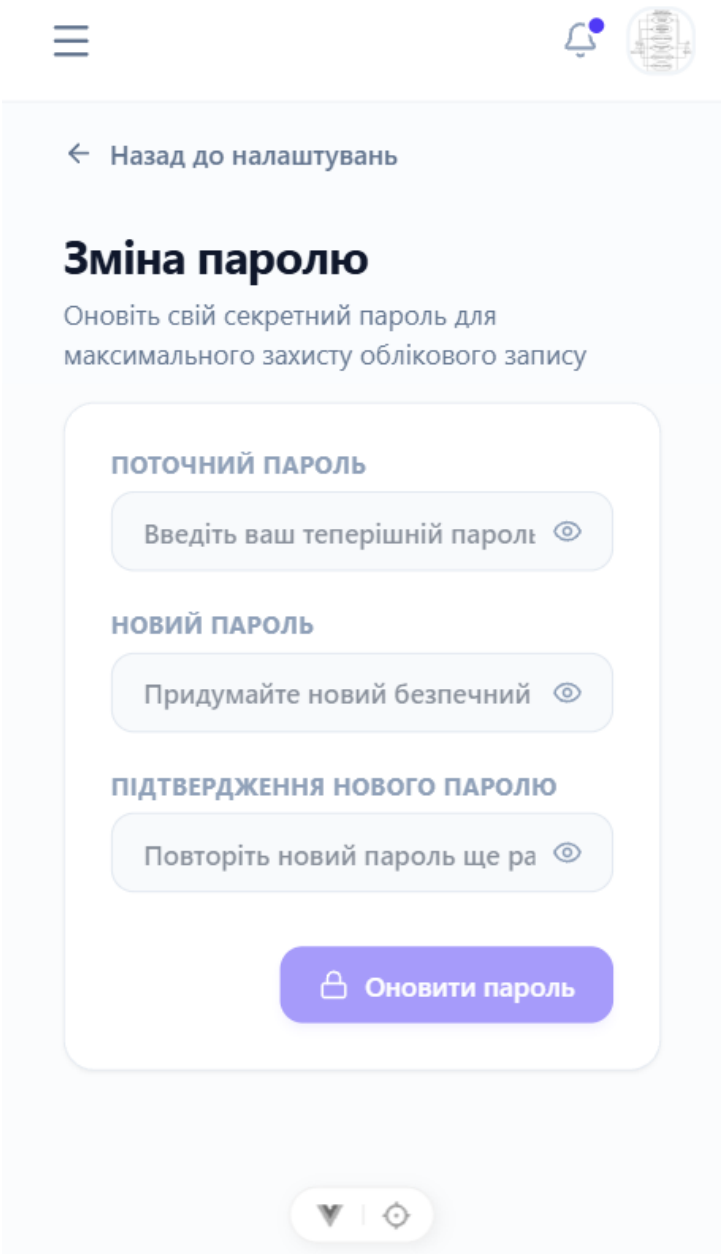


Рисунок 3.23 – Мобільна версія: зміна паролю

За результатами тестування підтверджено, що інтерфейс ProjectFlow коректно адаптується під мобільний розмір екрану: жоден елемент керування не виходить за межі видимої області, текст залишається читабельним без масштабування, а всі функції, реалізовані для десктопної версії, доступні і на смартфоні без обмежень.

# ВИСНОВКИ

Під час роботи над кваліфікаційною роботою було спроєктовано та розроблено веб-додаток ProjectFlow для керування проєктами. Він має адаптивний інтерфейс, який однаково зручно працює як на комп'ютерах, так і на мобільних пристроях.

У першому розділі проаналізовано сферу управління проєктами та розглянуто три популярні програмні рішення: Jira, Trello та Asana. З’ясовано, що всі вони мають недоліки у мобільних веб-інтерфейсах. Jira занадто складна для малих команд, Trello обмежений у безкоштовній версії та незручний на малих екранах, а Asana має зайвий функціонал і працює повільно на бюджетних мобільних пристроях. Це підкреслює необхідність створення легкого, повністю адаптивного інструменту для командної роботи.

У другому розділі описано вибір технологічного стеку, спроєктовано архітектуру системи та розроблено проєктну документацію. Для клієнтської частини обрано Vue 3 з Composition API, Tailwind CSS, Pinia, Vue Router, vuedraggable та Lucide Vue. Збереження даних здійснюється через LocalStorage браузера за допомогою механізму глибокого спостерігача Pinia. Для відправки email-сповіщень використовується EmailJS, що дозволяє надсилати листи безпосередньо з клієнтської частини без власного серверного коду. Розроблено UML-діаграму прецедентів і модель бізнес-процесу у нотації BPMN 2.0. Макети інтерфейсу створено у Figma для десктопної та мобільної версії.

У третьому розділі викладено структуру зберігання даних за допомогою 7 ключів LocalStorage, реалізовано 4 Pinia-сховища з чітким розподілом відповідальності між ними, а також описано методику роботи користувача з кожним екраном застосунку як у десктопній, так і в мобільній версії. Особливу увагу приділено тестуванню адаптивності інтерфейсу: перевірено коректність відображення сторінок авторизації, навігаційного меню, Kanban-дошки, персональних завдань, команд і налаштувань на екрані з шириною, типовою

для смартфона. Тестування підтвердило, що всі функції, доступні на десктопі, зберігаються й на мобільному пристрої у повному обсязі, без обмежень щодо функціоналу або зручності використання.

Завдяки результатам розробки ProjectFlow забезпечує такі можливості: створення та керування кількома проєктами одночасно, організація завдань у зручні 4 колонки з можливістю легкого переміщення карток за допомогою перетягування або одним натисканням на мобільному пристрої. Ви можете сортувати завдання за дедлайном, пріоритетом або датою створення, а також швидко фільтрувати картки за допомогою зручного пошукового рядка в реальному часі. Автоматична відправка email-сповіщень виконавцям при додаванні нових завдань допомагає бути в курсі всього, а дворівнева система сповіщень з тостами та журналом подій, повна адаптивність інтерфейсу для пристроїв з різними розмірами екрану, підтверджена практичне тестування, забезпечуючи високу якість та зручність роботи.

Усі завдання, поставлені на початку роботи, виконано в повному обсязі. Розроблений застосунок може використовуватися малими та середніми командами розробки програмного забезпечення, стартапами та проєктними групами, яким потрібен простий і зручний інструмент для управління завданнями без необхідності встановлювати додаткове програмне забезпечення.

# ПЕРЕЛІК ПОСИЛАНЬ

1. Evan You. Vue.js 3 Documentation [Електронний ресурс]. – Режим доступу: https://vuejs.org/guide/introduction.html

2. Pinia Documentation. The intuitive store for Vue.js [Електронний ресурс]. – Режим доступу: https://pinia.vuejs.org/introduction.html

3. Vue Router Documentation. The official router for Vue.js [Електронний ресурс]. – Режим доступу: https://router.vuejs.org/introduction.html

4. Tailwind CSS Documentation. A utility-first CSS framework [Електронний ресурс]. – Режим доступу: https://tailwindcss.com/docs/installation

5. Vite Documentation. Next Generation Frontend Tooling [Електронний ресурс]. – Режим доступу: https://vitejs.dev/guide/

6. EmailJS Documentation. Send emails directly from your code [Електронний ресурс]. – Режим доступу: https://www.emailjs.com/docs/

7. vuedraggable. Vue drag-and-drop component [Електронний ресурс]. – Режим доступу: https://github.com/SortableJS/vue.draggable.next

8. Lucide Icons Documentation [Електронний ресурс]. – Режим доступу: https://lucide.dev/guide/

9. Atlassian. Jira Software [Електронний ресурс]. – Режим доступу: https://www.atlassian.com/software/jira

10. Trello. Manage your team's projects from anywhere [Електронний ресурс]. – Режим доступу: https://trello.com/en

11. Asana. Work on big ideas, without the busywork [Електронний ресурс]. – Режим доступу: https://asana.com/product

12. MDN Web Docs. Window: localStorage property [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

13. Figma. The collaborative interface design tool [Електронний ресурс]. – Режим доступу: https://www.figma.com

14. Lucidchart. What is BPMN? Business Process Model and Notation explained [Електронний ресурс]. – Режим доступу: https://www.lucidchart.com/pages/bpmn

15. Visual Paradigm. What is UML? Unified Modeling Language explained [Електронний ресурс]. – Режим доступу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/

# ДОДАТКИ

## Додаток А

### Лістинг основних модулів програмного забезпечення

Скрипт authStore.js - Сховище автентифікації користувача

```
import { ref } from 'vue'
import { defineStore } from 'pinia'
import { useRouter } from 'vue-router'
import { useUserStore } from './userStore'

export const useAuthStore = defineStore('auth', () => {
  const router = useRouter()
  const userStore = useUserStore()

  const isAuthenticated =
ref(localStorage.getItem('projectflow_logged_in') === 'true')

  const getRegisteredUsers = () => {
    const users = localStorage.getItem('projectflow_users')
    return users
      ? JSON.parse(users)
      : [
          {
            name: 'Андрій р',
            email: 'danil.klimovskiy@gmail.com',
            password: '123',
            role: 'Project Manager',
            avatar: 'https://images.unsplash.com/photo-
1535713875002-d1d0cf377fde?q=80&w=150',
          },
        ]
  }

  const login = (emailVal, password) => {
    const users = getRegisteredUsers()
    const user = users.find(
      (u) => u.email === emailVal.toLowerCase().trim() &&
u.password === password,
    )

    if (user) {
      isAuthenticated.value = true
      localStorage.setItem('projectflow_logged_in', 'true')
      localStorage.setItem('projectflow_current_email',
user.email)

      // Наказуємо оновити базу реактивних користувачів для
перемальовки Команди
```

```
      userStore.loadUsers()

      router.push('/')
      return { success: true }
    }
    return { success: false, message: 'Невірний Email або
пароль' }
  }

  const register = (name, emailVal, password) => {
    const users = getRegisteredUsers()
    const exists = users.some((u) => u.email ===
emailVal.toLowerCase().trim())

    if (exists) {
      return { success: false, message: 'Користувач з таким
Email вже існує' }
    }

    const trimmedName = name.trim()
    const dynamicAvatarStub = `https://ui-
avatars.com/api/?name=${encodeURIComponent(trimmedName)}&backgro
und=eef2ff&color=4f46e5&bold=true`

    users.push({
      name: trimmedName,
      email: emailVal.toLowerCase().trim(),
      password,
      role: 'Project Manager',
      avatar: dynamicAvatarStub,
    })

    localStorage.setItem('projectflow_users',
JSON.stringify(users))

    return login(emailVal, password)
  }

  const logout = () => {
    isAuthenticated.value = false
    localStorage.removeItem('projectflow_logged_in')
    localStorage.removeItem('projectflow_current_email')

    userStore.loadUsers() // Скидаємо списки до дефолту при
розлогіненні
    router.push('/auth')
  }

  return {
    isAuthenticated,
    login,
    register,
```

```
    changePassword: (emailVal, currentPassword, newPassword) =>
{
      const users = getRegisteredUsers()
      const userIndex = users.findIndex(
        (u) => u.email.toLowerCase() ===
emailVal.toLowerCase().trim(),
      )
      if (userIndex !== -1) {
        if (users[userIndex].password !== currentPassword)
          return { success: false, message: 'Поточний пароль
невірний!' }
        users[userIndex].password = newPassword
        localStorage.setItem('projectflow_users',
JSON.stringify(users))
        return { success: true }
      }
      return { success: false, message: 'Користувача не
знайдено' }
    },
    logout,
  }
})
```

Скрипт boardStore.js - Сховище дошок, колонок та завдань

```
import { ref, watch, computed } from 'vue'
import { defineStore } from 'pinia'
import { useUserStore } from './userStore'
import { useNotificationStore } from './notificationStore'
import { sendRealEmail } from '../services/emailService'

export const useBoardStore = defineStore('board', () => {
  const savedBoards = localStorage.getItem('projectflow_boards')
  const savedActiveBoardId =
localStorage.getItem('projectflow_active_board_id')

  const defaultBoards = []

  // Стан
  const boards = ref(savedBoards ? JSON.parse(savedBoards) :
defaultBoards)
  const activeBoardId = ref(
    savedActiveBoardId && boards.value.length > 0
      ? savedActiveBoardId
      : boards.value.length > 0
        ? boards.value[0].id
        : null,
  )
  const searchQuery = ref('')

  const formatStoreDate = (dateStr) => {
    if (!dateStr) return 'Не вказано'
    try {
```

```
      const date = new Date(dateStr)
      return new Intl.DateTimeFormat('uk-UA', { day: 'numeric',
month: 'long' }).format(date)
    } catch (e) {
      return dateStr
    }
  }

  const columns = computed(() => {
    if (!activeBoardId.value) return []
    const board = boards.value.find((b) => b.id ===
activeBoardId.value)
    return board ? board.columns : []
  })

  // Синхронізація з LocalStorage
  watch(
    boards,
    (newBoards) => {
      localStorage.setItem('projectflow_boards',
JSON.stringify(newBoards))
    },
    { deep: true },
  )

  watch(activeBoardId, (newId) => {
    if (newId)
localStorage.setItem('projectflow_active_board_id', newId)
    else localStorage.removeItem('projectflow_active_board_id')
  })

  // Створення дошки
  const addBoard = async (title) => {
    const newId = `board-${Date.now()}`
    boards.value.push({
      id: newId,
      title,
      columns: [
        { id: 'todo', title: 'До виконання', tasks: [] },
        { id: 'in-progress', title: 'В роботі', tasks: [] },
        { id: 'testing', title: 'Тестування', tasks: [] },
        { id: 'done', title: 'Готово', tasks: [] },
      ],
    })
    activeBoardId.value = newId

    const userStore = useUserStore()
    const notificationStore = useNotificationStore()

    const isEmailEnabled =
      typeof userStore.emailNotificationsEnabled === 'object'
        ? userStore.emailNotificationsEnabled.value
```

```
        : userStore.emailNotificationsEnabled

    if (isEmailEnabled && String(isEmailEnabled) !== 'false') {
      notificationStore.addToastOnly(`⏳ Відправляємо лист про
створення дошки...`, 'info')
      const success = await sendRealEmail(userStore.email, {
        recipient_name: userStore.name,
        name: userStore.name,
        to_name: userStore.name,
        message_text: `Ви успішно створили нову робочу дошку
проєктів "${title}" у системі ProjectFlow.`,
      })
      if (success) {
        notificationStore.addNotification(`Лист успішно
надіслано на ${userStore.email}!`)
      } else {
        notificationStore.addNotification(`Лист на пошту не
пішов через помилку API.`, 'error')
      }
    } else {
      notificationStore.addNotification(
        `Створено нову робочу дошку "${title}" (Email-сповіщення
вимкнені)`,
      )
    }
  }

  const updateBoardTitle = (boardId, newTitle) => {
    const board = boards.value.find((b) => b.id === boardId)
    if (board && newTitle.trim()) {
      const oldTitle = board.title
      board.title = newTitle.trim()

      const notificationStore = useNotificationStore()
      notificationStore.addNotification(
        `Проєкт перейменовано: "${oldTitle}" тепер називається
"${board.title}"`,
        'info',
      )
    }
  }

  const deleteBoard = (boardId) => {
    const index = boards.value.findIndex((b) => b.id ===
boardId)
    if (index !== -1) {
      const deletedTitle = boards.value[index].title
      boards.value.splice(index, 1)

      if (activeBoardId.value === boardId) {
        activeBoardId.value = boards.value.length > 0 ?
boards.value[0].id : null
```

```
      }

      const notificationStore = useNotificationStore()
      notificationStore.addNotification(`Видалено робочий проєкт
"${deletedTitle}"`)
    }
  }

  // Керування завданнями
  const addTask = async (
    columnId,
    { title, description, priority, deadline, assignee, reporter
},
  ) => {
    const column = columns.value.find((col) => col.id ===
columnId)
    if (column) {
      column.tasks.push({
        id: `task-${Date.now()}`,
        title,
        description,
        priority,
        assignee,
        reporter,
        deadline: deadline || new
Date().toISOString().split('T')[0],
        createdAt: new Date().toISOString(),
      })

      const userStore = useUserStore()
      const notificationStore = useNotificationStore()

      const targetMember = userStore.team.find((m) => m.name ===
assignee)
      const recipientEmail = targetMember ? targetMember.email :
userStore.email
      const recipientName = targetMember ? targetMember.name :
assignee

      const isEmailEnabled =
        typeof userStore.emailNotificationsEnabled === 'object'
          ? userStore.emailNotificationsEnabled.value
          : userStore.emailNotificationsEnabled

      if (isEmailEnabled && String(isEmailEnabled) !== 'false')
{
        notificationStore.addToastOnly(`Сповіщаємо виконавця
${recipientName} по email...`, 'info')
        const success = await sendRealEmail(recipientEmail, {
          recipient_name: recipientName,
          name: recipientName,
          to_name: recipientName,
```

```
          message_text: `Вас призначено виконавцем нового
завдання: "${title}". Постановник: ${reporter}. Дедлайн:
${formatStoreDate(deadline)}.`,
        })
        if (success) {
          notificationStore.addNotification(`Повідомлення
доставлено на пошту ${recipientEmail}!`)
        } else {
          notificationStore.addNotification(
            `Не вдалося надіслати email через збій мережі.`,
            'error',
          )
        }
      } else {
        notificationStore.addNotification(
          `Завдання "${title}" закріплено за ${recipientName}
(Email-сповіщення вимкнені)`,
        )
      }
    }
  }

  const updateTask = (
    taskId,
    { title, description, priority, deadline, assignee,
reporter, columnId },
  ) => {
    let foundTask = null
    let oldColumnTitle = ''
    let originalCreatedAt = new Date().toISOString()

    columns.value.forEach((col) => {
      const index = col.tasks.findIndex((t) => t.id === taskId)
      if (index !== -1) {
        foundTask = col.tasks.splice(index, 1)[0]
        oldColumnTitle = col.title
        originalCreatedAt = foundTask.createdAt
      }
    })

    if (foundTask) {
      foundTask.title = title
      foundTask.description = description
      foundTask.priority = priority
      foundTask.deadline = deadline
      foundTask.assignee = assignee
      foundTask.reporter = reporter
      foundTask.createdAt = originalCreatedAt

      const targetColumn = columns.value.find((col) => col.id
=== columnId)
      if (targetColumn) {
```

```
        targetColumn.tasks.push(foundTask)
        if (oldColumnTitle && oldColumnTitle !==
targetColumn.title) {
          const notificationStore = useNotificationStore()
          notificationStore.addNotification(
            `Завдання "${title}" переведено з
"${oldColumnTitle}" до "${targetColumn.title}"`,
            'info',
          )
        }
      }
    }
  }

  // Кліковий перенос завдань (Tap-to-Move) для смартфонів
  const moveTaskDirectly = (taskId, fromColumnId, toColumnId) =>
{
    if (fromColumnId === toColumnId) return

    const sourceColumn = columns.value.find((c) => c.id ===
fromColumnId)
    const targetColumn = columns.value.find((c) => c.id ===
toColumnId)

    if (sourceColumn && targetColumn) {
      const taskIndex = sourceColumn.tasks.findIndex((t) => t.id
=== taskId)
      if (taskIndex !== -1) {
        const [task] = sourceColumn.tasks.splice(taskIndex, 1)
        targetColumn.tasks.push(task)
        localStorage.setItem('projectflow_boards',
JSON.stringify(boards.value))
      }
    }
  }

  const deleteTask = (taskId) => {
    columns.value.forEach((column) => {
      column.tasks = column.tasks.filter((task) => task.id !==
taskId)
    })
  }

  const clearColumn = (columnId) => {
    const column = columns.value.find((col) => col.id ===
columnId)
    if (column) column.tasks = []
  }

  const sortColumnByDeadline = (columnId) => {
    const column = columns.value.find((col) => col.id ===
columnId)
```

```
    if (column) column.tasks.sort((a, b) => new Date(a.deadline)
- new Date(b.deadline))
  }

  const sortColumnByCreatedAt = (columnId) => {
    const column = columns.value.find((col) => col.id ===
columnId)
    if (column) column.tasks.sort((a, b) => new
Date(b.createdAt) - new Date(a.createdAt))
  }

  const sortColumnByPriority = (columnId) => {
    const column = columns.value.find((col) => col.id ===
columnId)
    if (column) {
      const weight = { Високий: 3, Середній: 2, Низький: 1 }
      column.tasks.sort((a, b) => (weight[b.priority] || 0) -
(weight[a.priority] || 0))
    }
  }

  const updateOwnerName = (oldName, newName) => {
    boards.value.forEach((board) => {
      board.columns.forEach((column) => {
        column.tasks.forEach((task) => {
          if (task.assignee === oldName) task.assignee = newName
          if (task.reporter === oldName) task.reporter = newName
        })
      })
    })
  }

  return {
    boards,
    activeBoardId,
    columns,
    searchQuery,
    addBoard,
    updateBoardTitle,
    deleteBoard,
    addTask,
    updateTask,
    moveTaskDirectly,
    deleteTask,
    clearColumn,
    sortColumnByDeadline,
    sortColumnByCreatedAt,
    sortColumnByPriority,
    updateOwnerName,
  }
})
```

Скрипт BoardView.vue - Компонент Kanban-дошки проекту

```
<script setup>
import { ref, computed, onMounted, onUnmounted } from 'vue'
import { useBoardStore } from '@/stores/boardStore'
import { useUserStore } from '@/stores/userStore'
import { useNotificationStore } from
'@/stores/notificationStore'
import draggable from 'vuedraggable'
import {
  Plus,
  MoreVertical,
  Trash2,
  Calendar,
  Clock,
  Edit2,
  Check,
  X,
  KanbanSquare,
  AlignLeft,
  ChevronDown,
  FolderPlus,
  PlusCircle,
} from '@lucide/vue'

const boardStore = useBoardStore()
const userStore = useUserStore()
const notificationStore = useNotificationStore()

const isEditingBoardTitle = ref(false)
const editedBoardTitle = ref('')

const startEditingTitle = (currentTitle) => {
  editedBoardTitle.value = currentTitle
  isEditingBoardTitle.value = true
}

const saveBoardTitle = () => {
  if (editedBoardTitle.value.trim() && boardStore.activeBoardId)
{
    boardStore.updateBoardTitle(boardStore.activeBoardId,
editedBoardTitle.value)
    isEditingBoardTitle.value = false
  }
}

const currentBoard = computed(() => {
  return boardStore.boards.find((b) => b.id ===
boardStore.activeBoardId)
})

const activeMenuColumnId = ref(null)
const toggleMenu = (columnId) => {
```

```
  activeMenuColumnId.value = activeMenuColumnId.value ===
columnId ? null : columnId
}

const closeDropdownMenu = () => {
  activeMenuColumnId.value = null
}

onMounted(() => {
  window.addEventListener('click', closeDropdownMenu)
})

onUnmounted(() => {
  window.removeEventListener('click', closeDropdownMenu)
})

const handleSortColumn = (type, columnId) => {
  if (type === 'priority')
boardStore.sortColumnByPriority(columnId)
  if (type === 'deadline')
boardStore.sortColumnByDeadline(columnId)
  if (type === 'createdAt')
boardStore.sortColumnByCreatedAt(columnId)
  activeMenuColumnId.value = null
}

const handleClearColumn = (columnId) => {
  boardStore.clearColumn(columnId)
  activeMenuColumnId.value = null
}

const handleDragChange = (event, columnTitle) => {
  if (event.added) {
    const task = event.added.element
    notificationStore.addNotification(
      `Завдання "${task.title}" переміщено в статус
"${columnTitle}"`,
      'info',
    )
  }
}

// Перемикання колонок з модального вікна за два кліки
const handleStatusChange = (taskId, currentColumnId, event) => {
  const targetColId = event.target.value
  if (currentColumnId === targetColId) return

  boardStore.moveTaskDirectly(taskId, currentColumnId,
targetColId)
  targetColumnId.value = targetColId
  isModalOpen.value = false
```

```
  const targetColumnTitle = boardStore.columns.find((c) => c.id
=== targetColId)?.title
  notificationStore.addNotification(
    `Завдання "${taskTitle.value}" переміщено в статус
"${targetColumnTitle}"`,
    'info',
  )
}

const isNewBoardModalOpen = ref(false)
const newBoardName = ref('')

const handleCreateBoard = () => {
  if (newBoardName.value.trim()) {
    boardStore.addBoard(newBoardName.value)
    newBoardName.value = ''
    isNewBoardModalOpen.value = false
  }
}

const isModalOpen = ref(false)
const modalMode = ref('create')

const targetColumnId = ref('')
const currentTaskId = ref('')

const taskTitle = ref('')
const taskDescription = ref('')
const taskPriority = ref('Низький')
const taskDeadline = ref('')
const taskAssignee = ref('')
const taskReporter = ref('')

const openCreateModal = (columnId = 'todo') => {
  modalMode.value = 'create'
  targetColumnId.value = columnId
  taskTitle.value = ''
  taskDescription.value = ''
  taskPriority.value = 'Низький'
  taskDeadline.value = new Date().toISOString().split('T')[0]
  taskAssignee.value = userStore.name
  taskReporter.value = userStore.name
  isModalOpen.value = true
}

const openViewModal = (columnId, task) => {
  modalMode.value = 'view'
  targetColumnId.value = columnId
  currentTaskId.value = task.id
  taskTitle.value = task.title
  taskDescription.value = task.description
  taskPriority.value = task.priority
```

```
  taskDeadline.value = task.deadline
  taskAssignee.value = task.assignee || userStore.name
  taskReporter.value = task.reporter || userStore.name
  isModalOpen.value = true
}

const switchToEditMode = () => {
  modalMode.value = 'edit'
}

const handleSaveTask = () => {
  if (!taskTitle.value.trim()) return

  const taskData = {
    title: taskTitle.value,
    description: taskDescription.value,
    priority: taskPriority.value,
    deadline: taskDeadline.value,
    assignee: taskAssignee.value,
    reporter: taskReporter.value,
    columnId: targetColumnId.value,
  }

  if (modalMode.value === 'create') {
    boardStore.addTask(targetColumnId.value, taskData)
  } else {
    boardStore.updateTask(currentTaskId.value, taskData)
  }
  isModalOpen.value = false
}

const handleDeleteTask = () => {
  boardStore.deleteTask(currentTaskId.value)
  isModalOpen.value = false
}

const getPriorityClass = (priority) => {
  switch (priority) {
    case 'Високий':
      return 'bg-rose-50 text-rose-500 border-rose-100'
    case 'Середній':
      return 'bg-amber-50 text-amber-500 border-amber-100'
    case 'Низький':
      return 'bg-emerald-50 text-emerald-600 border-emerald-100'
    default:
      return 'bg-slate-50 text-slate-500 border-slate-200'
  }
}

const getDateColorClass = (priority) => {
  return priority === 'Низький' ? 'text-slate-400 font-medium' :
'text-rose-500 font-semibold'
```

```
}

const getAvatar = (name) => {
  const member = userStore.team.find((m) => m.name === name)
  return member ? member.avatar : userStore.avatar
}

const formatStringDate = (dateStr) => {
  if (!dateStr) return ''
  const date = new Date(dateStr)
  return date.toLocaleDateString('uk-UA', { day: 'numeric',
month: 'long' }).replace(' р.', '')
}
</script>

<template>
  <div class="h-full flex flex-col w-full min-w-0 overflow-
hidden">
    <div
      v-if="boardStore.boards.length === 0"
      class="flex-1 flex flex-col items-center justify-center
py-16 animate-fade-in px-4"
    >
      <div
        class="w-16 h-16 bg-slate-100 text-slate-400 rounded-2xl
flex items-center justify-center mb-4 border border-slate-
200/40"
      >
        <KanbanSquare class="w-8 h-8" />
      </div>
      <h2 class="text-xl font-bold text-slate-900 tracking-
tight">Робочий простір порожній</h2>
      <p class="text-sm text-slate-400 mt-1 max-w-sm text-center
leading-normal">
        Ви видалили всі робочі дошки. Створіть новий проєкт, щоб
розпочати планування завдань.
      </p>
      <button
        @click="isNewBoardModalOpen = true"
        class="mt-6 bg-indigo-600 hover:bg-indigo-700 text-white
px-5 py-2.5 rounded-xl text-sm font-semibold inline-flex items-
center gap-2 transition-all shadow-md shadow-indigo-600/10
cursor-pointer w-full xs:w-auto justify-center"
      >
        <Plus class="w-4 h-4" /> Створити першу дошку
      </button>
    </div>

    <div v-else class="flex-1 flex flex-col min-h-0 w-full
overflow-hidden">
      <div
```

```
        class="flex flex-col sm:flex-row sm:items-center
justify-between gap-4 mb-8 shrink-0 w-full px-1"
      >
        <div class="flex flex-wrap items-center gap-3 min-w-0">
          <h1 class="text-2xl font-bold text-slate-900 tracking-
tight shrink-0">Дошка проєкту</h1>

          <div v-if="isEditingBoardTitle" class="flex items-
center gap-1.5">
            <input
              v-model="editedBoardTitle"
              type="text"
              @keyup.enter="saveBoardTitle"
              class="bg-white border border-indigo-500 px-3 py-1
rounded-lg text-sm font-semibold text-slate-700 outline-none
shadow-xs"
              autofocus
            />
            <button
              @click="saveBoardTitle"
              class="p-1.5 bg-emerald-50 text-emerald-600
rounded-lg hover:bg-emerald-100 transition-colors cursor-
pointer"
            >
              <Check class="w-3.5 h-3.5" />
            </button>
            <button
              @click="isEditingBoardTitle = false"
              class="p-1.5 bg-slate-50 text-slate-400 rounded-lg
hover:bg-slate-100 transition-colors cursor-pointer"
            >
              <X class="w-3.5 h-3.5" />
            </button>
          </div>

          <div
            v-else
            @click="startEditingTitle(currentBoard?.title)"
            class="bg-slate-100 text-slate-500 font-semibold
text-xs px-3 py-1.5 rounded-xl border border-slate-200/40
cursor-pointer hover:bg-slate-200/60 transition-colors flex
items-center gap-1 max-w-[160px] xs:max-w-none truncate"
          >
            <span class="truncate">{{ currentBoard?.title
}}</span>
            <Edit2 class="w-2.5 h-2.5 text-slate-400 shrink-0"
/>
          </div>

          <select
            v-model="boardStore.activeBoardId"
```

```
            class="bg-white border border-slate-200 px-2 py-1
rounded-lg text-xs font-semibold text-slate-400 outline-none
focus:border-slate-300 transition-all cursor-pointer max-w-
[120px] xs:max-w-none"
          >
            <option v-for="b in boardStore.boards" :key="b.id"
:value="b.id">{{ b.title }}</option>
          </select>
        </div>

        <div class="flex items-center gap-2.5 shrink-0 w-full
sm:w-auto">
          <button
            @click="isNewBoardModalOpen = true"
            class="flex-1 sm:flex-none bg-white border border-
slate-200 hover:bg-slate-50 text-slate-700 px-3 py-2.5 rounded-
xl text-xs font-semibold inline-flex items-center justify-center
gap-2 transition-all cursor-pointer"
          >
            <FolderPlus class="w-4 h-4 text-slate-400 shrink-0"
/>
            <span>Нова дошка</span>
          </button>

          <button
            @click="openCreateModal('todo')"
            class="flex-1 sm:flex-none bg-indigo-600 hover:bg-
indigo-700 text-white px-3 py-2.5 rounded-xl text-xs font-
semibold inline-flex items-center justify-center gap-2
transition-all shadow-md shadow-indigo-600/10 cursor-pointer"
          >
            <PlusCircle class="w-4 h-4 shrink-0" />
            <span>Завдання</span>
          </button>

          <button

@click="boardStore.deleteBoard(boardStore.activeBoardId)"
            class="bg-slate-100 hover:bg-rose-50 hover:text-
rose-600 hover:border-rose-100 text-slate-400 border border-
transparent px-3 py-2.5 rounded-xl text-xs font-semibold inline-
flex items-center gap-1 transition-all cursor-pointer shrink-0"
            title="Видалити цю дошку"
          >
            <Trash2 class="w-3.5 h-3.5 shrink-0" />
          </button>
        </div>
      </div>

      <div class="flex-1 w-full overflow-x-auto custom-scrollbar
snap-x snap-mandatory">
        <div class="flex gap-6 pb-4 items-start pr-4 pl-1">
```

```
          <div
            v-for="column in boardStore.columns"
            :key="column.id"
            class="w-[288px] xs:w-[310px] sm:w-72 max-h-full bg-
slate-100/70 border border-slate-200/50 rounded-2xl p-4 flex
flex-col shrink-0 snap-center"
          >
            <div class="flex justify-between items-center mb-4
shrink-0 px-1">
              <div class="flex items-center gap-2">
                <h2 class="font-bold text-slate-800 text-sm
tracking-tight">{{ column.title }}</h2>
                <span
                  class="bg-slate-200 text-slate-600 font-bold
text-xs px-2 py-0.5 rounded-full"
                >
                  {{ column.tasks.length }}
                </span>
              </div>

              <div class="flex items-center gap-1">
                <button
                  @click="openCreateModal(column.id)"
                  class="text-slate-400 hover:text-indigo-600 p-
1 rounded-lg hover:bg-slate-200/40 transition-colors cursor-
pointer"
                >
                  <Plus class="w-4 h-4" />
                </button>

                <div class="relative">
                  <button
                    @click.stop="toggleMenu(column.id)"
                    class="text-slate-400 hover:text-slate-600
p-1 rounded-lg hover:bg-slate-200/40 transition-colors cursor-
pointer"
                  >
                    <MoreVertical class="w-4 h-4" />
                  </button>

                  <div
                    v-if="activeMenuColumnId === column.id"
                    @click.stop
                    class="absolute right-0 mt-1 w-48 bg-white
border border-slate-200 rounded-xl shadow-xl z-30 py-1 font-
medium text-xs"
                  >
                    <button
                      @click="handleSortColumn('priority',
column.id)"
                      class="w-full px-4 py-2 flex items-center
gap-2 hover:bg-slate-50 text-slate-600"
```

```
                    >
                      Сортувати за пріоритетом
                    </button>
                    <button
                      @click="handleSortColumn('deadline',
column.id)"
                      class="w-full px-4 py-2 flex items-center
gap-2 hover:bg-slate-50 text-slate-600"
                    >
                      Сортувати за дедлайном
                    </button>
                    <button
                      @click="handleSortColumn('createdAt',
column.id)"
                      class="w-full px-4 py-2 flex items-center
gap-2 hover:bg-slate-50 text-slate-600"
                    >
                      Сортувати за новістю
                    </button>
                    <hr class="border-slate-100 my-1" />
                    <button
                      @click="handleClearColumn(column.id)"
                      class="w-full px-4 py-2 flex items-center
gap-2 hover:bg-slate-50 text-rose-600"
                    >
                      Очистити колонку
                    </button>
                  </div>
                </div>
              </div>

              <draggable
                v-model="column.tasks"
                group="tasks"
                item-key="id"
                class="flex-1 overflow-y-auto space-y-3 pr-0.5
custom-scrollbar min-h-[150px]"
                ghost-class="opacity-40"
                drag-class="rotate-2"
                :animation="200"
                @change="handleDragChange($event, column.title)"
              >
                <template #item="{ element: task }">
                  <div
                    v-show="

task.title.toLowerCase().includes(boardStore.searchQuery.toLower
Case()) ||
                      (task.description &&
```

```
task.description.toLowerCase().includes(boardStore.searchQuery.t
oLowerCase()))
                  "
                  @click="openViewModal(column.id, task)"
                  class="bg-white p-4 rounded-xl shadow-xs
border border-slate-100/60 flex flex-col gap-2.5 cursor-pointer
hover:shadow-md transition-all duration-150"
                >
                  <h3 class="font-medium text-slate-900 text-sm
leading-snug">
                    {{ task.title }}
                  </h3>
                  <p
                    v-if="task.description"
                    class="text-xs text-slate-400 line-clamp-2
leading-relaxed"
                  >
                    {{ task.description }}
                  </p>

                  <div class="flex flex-col gap-1 mt-1 pt-2
border-t border-slate-50">
                    <div
                      v-if="task.createdAt"
                      class="flex items-center gap-1 text-[10px]
text-slate-400 font-medium"
                    >
                      <Clock class="w-3 h-3 text-slate-300" />
                      <span>Створено: {{
formatStringDate(task.createdAt) }}</span>
                    </div>

                    <div class="flex justify-between items-
center mt-1">
                      <div class="flex items-center gap-2">
                        <span

:class="getPriorityClass(task.priority)"
                          class="text-[10px] font-bold px-2 py-
0.5 rounded-lg border"
                        >
                          {{ task.priority }}
                        </span>
                        <div
                          class="flex items-center gap-1 text-
[11px]"

:class="getDateColorClass(task.priority)"
                        >
                          <Calendar class="w-3 h-3 text-slate-
300" />
```

```
                          <span>{{
formatStringDate(task.deadline) }}</span>
                        </div>
                      </div>
                      <img
                        :src="getAvatar(task.assignee)"
                        alt="Avatar"
                        class="w-6 h-6 rounded-full object-cover
ring-2 ring-white shadow-2xs"
                      />
                    </div>
                  </div>
                </div>
              </template>
            </draggable>
          </div>
        </div>
      </div>

      <div
        v-if="isNewBoardModalOpen"
        class="fixed inset-0 bg-black/60 flex items-center
justify-center z-50 animate-fade-in"
        @click="isNewBoardModalOpen = false"
      >
        <div
          class="bg-white rounded-2xl border border-slate-200
shadow-2xl p-6 w-full max-w-sm mx-4 animate-scale-up"
          @click.stop
        >
          <h2 class="text-base font-bold text-slate-900 mb-
4">Створення нового проєкту</h2>
          <div class="space-y-4">
            <div class="space-y-1.5">
              <label class="block text-xs font-bold uppercase
tracking-wider text-slate-400"
                >Назва дошки</label
              >
              <input
                v-model="newBoardName"
                type="text"
                placeholder="Наприклад, Маркетинг..."
                class="w-full bg-white border border-slate-200 px-
3 py-2 rounded-xl text-sm outline-none focus:border-indigo-500
transition-all text-slate-800 font-medium"
                @keyup.enter="handleCreateBoard"
              />
            </div>
            <div class="flex justify-end gap-2.5 pt-2">
              <button
                @click="isNewBoardModalOpen = false"
```

```
              class="px-4 py-2 border border-slate-200 hover:bg-
slate-50 rounded-xl text-xs font-semibold text-slate-500
transition-colors cursor-pointer"
            >
              Скасувати
            </button>
            <button
              @click="handleCreateBoard"
              :disabled="!newBoardName.trim()"
              class="px-4 py-2 bg-indigo-600 text-white rounded-
xl text-xs font-semibold transition-colors cursor-pointer"
            >
              Створити
            </button>
          </div>
        </div>
      </div>
    </div>

    <div
      v-if="isModalOpen && modalMode === 'view'"
      class="fixed inset-0 bg-black/60 flex items-center
justify-center z-50 p-4 animate-fade-in"
      @click="isModalOpen = false"
    >
      <div
        class="bg-white rounded-2xl p-5 sm:p-6 w-full max-w-md
mx-4 shadow-2xl relative animate-scale-up flex flex-col gap-5
max-h-[90vh] overflow-y-auto custom-scrollbar"
        @click.stop
      >
        <div class="flex justify-between items-start gap-4">
          <h2 class="text-xl font-bold text-slate-900 leading-
tight tracking-tight">
            {{ taskTitle }}
          </h2>
          <button
            @click="isModalOpen = false"
            class="text-slate-400 hover:text-slate-600 p-1
rounded-lg transition-colors cursor-pointer"
          >
            <X class="w-5 h-5" />
          </button>
        </div>

        <div
          class="grid grid-cols-1 sm:grid-cols-2 gap-4 bg-slate-
50/60 p-3 rounded-2xl border border-slate-100"
        >
          <div class="space-y-1">
            <label class="block text-[10px] font-bold text-
slate-400 uppercase tracking-wider"
```

```
              >Статус завдання</label
            >
            <div class="relative flex items-center">
              <select
                :value="targetColumnId"
                @change="handleStatusChange(currentTaskId,
targetColumnId, $event)"
                class="w-full bg-white border border-slate-
200/80 pl-3 pr-8 py-1.5 rounded-xl text-xs font-bold text-slate-
700 outline-none focus:border-indigo-500 transition-all
appearance-none cursor-pointer"
              >
                <option v-for="col in boardStore.columns"
:key="col.id" :value="col.id">
                  {{ col.title }}
                </option>
              </select>
              <ChevronDown
                class="w-3.5 h-3.5 text-slate-400 absolute
right-2.5 pointer-events-none"
              />
            </div>
          </div>

          <div class="space-y-1">
            <label
              class="block text-[10px] font-bold text-slate-400
uppercase tracking-wider sm:text-right"
              >Пріоритет</label
            >
            <div class="sm:text-right pt-0.5">
              <span
                :class="getPriorityClass(taskPriority)"
                class="text-xs font-bold px-3 py-1 rounded-xl
border inline-block"
              >
                {{ taskPriority }}
              </span>
            </div>
          </div>
        </div>

        <div class="grid grid-cols-1 xs:grid-cols-2 gap-4 pb-1">
          <div>
            <span class="block text-[10px] font-bold text-slate-
400 uppercase tracking-wider mb-2"
              >Постановник</span
            >
            <div class="flex items-center gap-2.5">
              <img
                :src="getAvatar(taskReporter)"
```

```
                class="w-8 h-8 rounded-full object-cover ring-2
ring-slate-100"
              />
              <span class="text-sm font-semibold text-slate-700
truncate">{{ taskReporter }}</span>
            </div>
          </div>
          <div>
            <span class="block text-[10px] font-bold text-slate-
400 uppercase tracking-wider mb-2"
              >Виконавець</span
            >
            <div class="flex items-center gap-2.5">
              <img
                :src="getAvatar(taskAssignee)"
                class="w-8 h-8 rounded-full object-cover ring-2
ring-slate-100"
              />
              <span class="text-sm font-semibold text-slate-700
truncate">{{ taskAssignee }}</span>
            </div>
          </div>
        </div>

        <div class="space-y-2">
          <span
            class="text-[10px] font-bold text-slate-400
uppercase tracking-wider flex items-center gap-1"
          >
            <AlignLeft class="w-3.5 h-3.5" /> Опис завдання
          </span>
          <div
            class="bg-slate-50/70 border border-slate-100 p-3.5
rounded-xl text-sm text-slate-600 leading-relaxed min-h-[60px]
whitespace-pre-wrap"
          >
            {{ taskDescription || 'Опис відсутній...' }}
          </div>
        </div>

        <div
          class="bg-slate-50/70 border border-slate-100 p-3.5
rounded-xl flex items-center justify-between"
        >
          <span class="text-[10px] font-bold text-slate-400
uppercase tracking-wider"
            >Термін дедлайну</span
          >
          <div class="flex items-center gap-1.5 text-rose-500
font-bold text-sm">
            <Calendar class="w-4 h-4 text-rose-400" />
            <span>{{ formatStringDate(taskDeadline) }}</span>
```

```
          </div>
        </div>

        <div class="flex items-center justify-between pt-2
border-t border-slate-50 mt-1">
          <button
            @click="handleDeleteTask"
            class="text-sm font-bold text-rose-500 hover:text-
rose-600 flex items-center gap-1 px-2 py-1 rounded-lg
transition-colors cursor-pointer"
          >
            <Trash2 class="w-4 h-4" /> Видалити
          </button>
          <button
            @click="switchToEditMode"
            class="bg-indigo-600 hover:bg-indigo-700 text-white
px-5 py-2.5 rounded-xl text-xs font-bold flex items-center gap-
1.5 transition-all shadow-md shadow-indigo-600/10 cursor-
pointer"
          >
            <Edit2 class="w-3.5 h-3.5" /> Редагувати
          </button>
        </div>
      </div>
    </div>

    <div
      v-if="isModalOpen && (modalMode === 'create' || modalMode
=== 'edit')"
      class="fixed inset-0 bg-black/60 flex items-center
justify-center z-50 p-4 animate-fade-in"
      @click="isModalOpen = false"
    >
      <div
        class="bg-white rounded-2xl p-5 sm:p-6 w-full max-w-md
mx-4 shadow-2xl relative animate-scale-up max-h-[90vh] overflow-
y-auto custom-scrollbar"
        @click.stop
      >
        <div class="flex justify-between items-center mb-4">
          <h2 class="text-lg font-bold text-slate-900">
            {{ modalMode === 'create' ? 'Створити завдання' :
'Редагування завдання' }}
          </h2>
          <button
            @click="isModalOpen = false"
            class="text-slate-400 hover:text-slate-600 p-1
rounded-lg cursor-pointer"
          >
            <X class="w-5 h-5" />
          </button>
        </div>
```

```
        <div class="space-y-4">
          <div>
            <label class="block text-xs font-semibold text-
slate-500 mb-1">Назва</label>
            <input
              v-model="taskTitle"
              type="text"
              placeholder="Назва завдання"
              class="w-full border border-slate-200 px-3 py-2
rounded-lg text-sm outline-none focus:border-indigo-500 font-
medium"
            />
          </div>

          <div>
            <label class="block text-xs font-semibold text-
slate-500 mb-1">Опис</label>
            <textarea
              v-model="taskDescription"
              placeholder="Опис завдання"
              class="w-full border border-slate-200 px-3 py-2
rounded-lg text-sm outline-none focus:border-indigo-500 h-24
resize-none"
            ></textarea>
          </div>

          <div class="grid grid-cols-2 gap-4">
            <div>
              <label class="block text-xs font-semibold text-
slate-500 mb-1">Пріоритет</label>
              <select
                v-model="taskPriority"
                class="w-full border border-slate-200 px-3 py-2
rounded-lg text-sm outline-none focus:border-indigo-500 cursor-
pointer"
              >
                <option value="Низький">Низький</option>
                <option value="Середній">Середній</option>
                <option value="Високий">Високий</option>
              </select>
            </div>
            <div>
              <label class="block text-xs font-semibold text-
slate-500 mb-1">Дедлайн</label>
              <input
                v-model="taskDeadline"
                type="date"
                class="w-full border border-slate-200 px-3 py-2
rounded-lg text-sm outline-none focus:border-indigo-500 cursor-
pointer"
              />
```

```
            </div>
          </div>

          <div class="grid grid-cols-2 gap-4">
            <div>
              <label class="block text-xs font-semibold text-
slate-500 mb-1">Виконавець</label>
              <select
                v-model="taskAssignee"
                class="w-full border border-slate-200 px-3 py-2
rounded-lg text-sm outline-none focus:border-indigo-500 cursor-
pointer"
              >
                <option v-for="member in userStore.team"
:key="member.name" :value="member.name">
                  {{ member.name }}
                </option>
              </select>
            </div>
            <div>
              <label class="block text-xs font-semibold text-
slate-500 mb-1">Постановник</label>
              <select
                v-model="taskReporter"
                class="w-full border border-slate-200 px-3 py-2
rounded-lg text-sm outline-none focus:border-indigo-500 cursor-
pointer"
              >
                <option v-for="member in userStore.team"
:key="member.name" :value="member.name">
                  {{ member.name }}
                </option>
              </select>
            </div>
          </div>

          <div class="flex justify-end gap-2 pt-2 border-t
border-slate-50">
            <button
              @click="isModalOpen = false"
              class="px-4 py-2 border border-slate-200 rounded-
lg text-sm font-semibold text-slate-500 hover:bg-slate-50
cursor-pointer"
            >
              Скасувати
            </button>
            <button
              @click="handleSaveTask"
              :disabled="!taskTitle.trim()"
              class="px-4 py-2 bg-indigo-600 text-white rounded-
lg text-sm font-semibold hover:bg-indigo-700 disabled:opacity-50
cursor-pointer"
```

```
            >
              Зберегти
            </button>
          </div>
        </div>
      </div>
    </div>
  </div>
</template>

<style scoped>
@keyframes fadeIn {
  from {
    opacity: 0;
  }
  to {
    opacity: 1;
  }
}
@keyframes scaleUp {
  from {
    opacity: 0;
    transform: scale(0.96);
  }
  to {
    opacity: 1;
    transform: scale(1);
  }
}
.animate-fade-in {
  animation: fadeIn 0.15s ease-out forwards;
}
.animate-scale-up {
  animation: scaleUp 0.15s cubic-bezier(0.16, 1, 0.3, 1)
forwards;
}
.custom-scrollbar::-webkit-scrollbar {
  width: 4px;
  height: 6px;
}
.custom-scrollbar::-webkit-scrollbar-track {
  background: transparent;
}
.custom-scrollbar::-webkit-scrollbar-thumb {
  background: #cbd5e1;
  border-radius: 10px;
}
</style>
```