МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ФАХОВИЙ БІЗНЕС-КОЛЕДЖ

Комп’ютерна інженерія та інформаційні технології

(повна назва випускової кафедри)

# КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему

# Платформа для планування та відстеження завдань у командах

Виконав(-ла): студентка) групи 2КІ - 23

спеціальності

123 Комп’ютерна інженерія

Тамурова Дар’я Владиславівна

Науковий керівник - Кандидат технічних наук, доцент Розломій Інна Олександрівна

Черкаси 2025

ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ БІЗНЕС-КОЛЕДЖ

Кафедра комп’ютерної інженерії та інформаційних технологій

(повна назва випускової кафедри)

Спеціальність 123 “Комп’ютерна інженерія ”

(шифр і назва спеціальності)

Освітня програма Комп’ютерна інженерія

(назва освітньої програми)

ЗАТВЕРДЖУЮ

Завідувач кафедри
комп’ютерної інженерії та інформаційних технологій
(назва кафедри)

_____ Хотунов В.І.
(підпис) (ПІБ)

«_____»_____2025 р.

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

Тамурова Дар’я Владиславівна
(прізвище, ім’я, по батькові студента)

1. Тема кваліфікаційної роботи Платформа для планування та відстеження завдань у командах

Науковий керівник роботи к.н.т доцент Розломій Інна Олександрівна

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “ 7 ” жовтня 2025 року № 68У.

2. Строк подання студентом випускної роботи 03.06.2025

3. Вихідні дані до випускної роботи Проведення аналізу сучасних платформ для управління командною роботою (таких як Trello, Jira, Asana), виявлення їхніх переваг і недоліків. Формування вимог до власної системи, проєктування архітектури та інтерфейсу користувача. Розробка веб-платформи з підтримкою створення, відстеження та пріоритизації завдань у команді, реалізація функцій drag-and-drop, оцінки складності завдань, статистики та ролей користувачів.

4. Зміст випускної роботи (перелік питань, які потрібно розробити) У роботі розглядаються актуальність теми, мета, завдання, об’єкт і предмет дослідження. Проводиться детальний аналіз існуючих платформ для командного управління завданнями, визначаються функціональні та технічні вимоги до розроблюваної системи. Описується процес створення архітектури додатку та бази даних, розробляється інтерфейс користувача, реалізуються основні функції платформи, а також здійснюється тестування й аналіз ефективності запропонованого рішення.

5. Дата видачі завдання 15.09.2025р.

## Календарний план

| № з/п | Назва етапів кваліфікаційної роботи | Терміни виконання етапів | Примітка про виконання з підписами наукового керівника і студента |
|---|---|---|---|
| 1 | Вступ | 15.10.2025 | |
| 2 | Розділ 1. Теоретичні основи створення платформ для планування та відстеження завдань у командах | 18.12.2025 | |
| 3 | Розділ 2. Проєктування системи | 05.03.2025 | |
| 4 | Розділ 3. Тестування та оцінка ефективності платформи | 06.04.2025 | |
| 5 | Висновки | 16.05.2025 | |
| 6 | Оформлення випускної роботи (чистовий варіант) | 27.05.2025 | |
| 7 | Здача випускної роботи на кафедру для рецензування (за 14 днів до захисту) | 03.05.2025 | |
| 8 | Перевірка випускної роботи на наявність ознак плагіату (за 10 днів до захисту) | 07.06.2025 | |
| 9 | Подання випускної роботи на затвердження завідувачу кафедри (за 7 днів до захисту) | 10.06.2025 | |

Студентка _____ Тамурова Д.В.
(прізвище та ініціали)

Науковий керівник роботи _____ Розломій І.О.
(прізвище та ініціали)

**Анотація**

У дипломній роботі представлено процес розробки веб-платформи для планування та відстеження завдань у командах, що має на меті покращити організацію спільної роботи, підвищити ефективність управління проєктами та прозорість виконання завдань на всіх етапах. Актуальність теми зумовлена зростанням популярності дистанційної роботи, розподілених команд та потребою в зручних цифрових інструментах, які дозволяють керувати завданнями, проєктами та ресурсами в режимі реального часу.

У роботі проведено аналіз існуючих рішень на ринку — таких як Trello, Jira, Asana, ClickUp — із метою виявлення їхніх функціональних особливостей, переваг та обмежень. На основі проведеного порівняльного аналізу були сформульовані вимоги до власної системи, орієнтованої на простоту використання, мінімалістичний дизайн, швидкодію, розширюваність та можливість адаптації під різні сценарії командної взаємодії.

Розроблена система базується на клієнт-серверній архітектурі. Клієнтська частина створена за допомогою React.js з використанням компонентного підходу, серверна частина реалізована на Node.js із застосуванням фреймворку Express. Для збереження даних використано реляційну базу даних PostgreSQL [7]. Забезпечено розмежування ролей користувачів, реалізовано механізми автентифікації й авторизації на основі JWT-токенів, а також виконано базову перевірку вхідних даних.

Ключовими функціями реалізованої платформи є: створення та редагування завдань, групування за проєктами, переміщення між статусами у вигляді канбан-дошки, відстеження прогресу виконання, додавання учасників до команд та управління доступом. Задля перевірки працездатності системи виконано тестування основних модулів, проведено юзер-тестування інтерфейсу, сформовано приклади тест-кейсів.

Результатом дослідження стала функціональна, масштабована система, що може застосовуватись як внутрішній інструмент для невеликих і середніх команд, так і як основа для подальшої розробки SaaS-продукту. У завершальній частині роботи запропоновано напрями майбутнього розвитку проєкту, серед яких — мобільна версія, інтеграція з календарями, push-сповіщення та штучний інтелект для рекомендацій у плануванні завдань.

**Abstract**

This thesis presents the development process of a web-based platform for task planning and tracking in teams, aimed at improving collaborative work organization, enhancing project management efficiency, and ensuring transparency of task execution at all stages. The relevance of the topic is driven by the growing popularity of remote work, distributed teams, and the need for convenient digital tools that allow real-time management of tasks, projects, and resources.

The study includes an analysis of existing market solutions — such as Trello, Jira, Asana, and ClickUp — to identify their functional features, advantages, and limitations. Based on the comparative analysis, the requirements for a custom system were formulated, focusing on ease of use, minimalist design, high performance, scalability, and adaptability to different team interaction scenarios.

The developed system is based on a client-server architecture. The client side was built using React.js with a component-based approach, while the server side was implemented in Node.js using the Express framework. A relational PostgreSQL database was used for data storage. The system supports user role separation, implements authentication and authorization mechanisms using JWT tokens, and performs basic input validation.

Key features of the implemented platform include task creation and editing, project-based task grouping, status changes using a Kanban board, progress tracking, team member management, and access control. To verify the system's performance, core modules were tested, user interface testing was conducted, and sample test cases were created.

As a result of the research, a functional, scalable system was developed, suitable for use as an internal tool for small and medium-sized teams, as well as a foundation for further development of a SaaS product. The final part of the thesis outlines directions for future development of the project, including a mobile version, calendar integration, push notifications, and AI-powered task planning suggestions.

**Список скорочень**

**UI** – User Interface – інтерфейс користувача

**UX** – User Experience – користувацький досвід

**API** – Application Programming Interface – програмний інтерфейс додатку

**JWT** – JSON Web Token – формат передачі токенів для аутентифікації

**CRUD** – Create, Read, Update, Delete – базові операції з даними

**SaaS** – Software as a Service – програмне забезпечення як послуга

**DB** – Database – база даних

**SQL** – Structured Query Language – мова структурованих запитів

**HTTP** – Hypertext Transfer Protocol – протокол передачі гіпертексту

**HTTPS** – Hypertext Transfer Protocol Secure – захищений протокол передачі гіпертексту

**JSON** – JavaScript Object Notation – формат обміну даними

**DOM** – Document Object Model – об’єктна модель документа

**SPA** – Single Page Application – односторінковий вебзастосунок

**ID** – Identifier – ідентифікатор

**CSS** – Cascading Style Sheets – каскадні таблиці стилів

**HTML** – HyperText Markup Language – мова розмітки гіпертексту

**REST** – Representational State Transfer – архітектурний стиль вебсервісів

**IDE** – Integrated Development Environment – інтегроване середовище розробки

**CI/CD** – Continuous Integration / Continuous Deployment – безперервна інтеграція та розгортання

**RAM** – Random Access Memory – оперативна пам’ять

**URL** – Uniform Resource Locator – уніфікований локатор ресурсу

**CLI** – Command Line Interface – інтерфейс командного рядка

# ВСТУП

Актуальність теми

У сучасному цифровому світі дедалі більше компаній та команд стикаються з потребою організовувати робочі процеси ефективно, прозоро та гнучко. Особливо це актуально для віддалених або гібридних команд, де координація дій і контроль виконання завдань ускладнюється відсутністю фізичної присутності. Системи управління завданнями (task management systems) стали невід'ємною частиною IT-інфраструктури організацій. Проте багато популярних інструментів є або надто складними для малих команд, або мають обмеження у безкоштовних версіях. Тому розробка власного застосунку, адаптованого до потреб конкретної команди чи організації, є актуальною і практично значущою задачею.

Мета і завдання дипломної роботи

**Мета дипломної роботи** - розробка веб-застосунку для управління командною роботою над проєктами, який дозволяє створювати завдання, відслідковувати їх статус, призначати відповідальних осіб та облік часу, витраченого на виконання.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

- проаналізувати існуючі системи управління завданнями, їх переваги та недоліки;
- сформувати вимоги до власного застосунку на основі проведеного аналізу;
- спроєктувати структуру застосунку та розробити його функціонал;
- реалізувати систему з використанням технологій React;
- перевірити коректність роботи застосунку та провести тестування.

Об'єкт і предмет дослідження

**Об'єктом дослідження** є процес управління завданнями в командній роботі.

**Предметом дослідження** є методи та інструменти створення веб-застосунку для організації та контролю виконання завдань у проєктах.

Методологія дослідження

У роботі застосовано аналітичний підхід для вивчення та порівняння існуючих систем, методи програмної інженерії для проєктування архітектури застосунку, а також практичні інструменти розробки веб-застосунків на основі JavaScript-бібліотеки React. Було використано принципи компонетної розробки інтерфейсу, а також метод drag-and-drop для взаємодії з задачами.

Структура роботи

Дипломна робота складається зі вступу, трьох основних розділів, висновків та списку використаних джерел.

У **першому розділі** розглянуто існуючі аналоги систем управління завданнями, їх функціональні можливості, переваги та недоліки.

У **другому розділі** описано проєктування власної системи: сформульовано вимоги, побудовано структуру даних та інтерфейсу.

У **третьому розділі** подано реалізацію веб-застосунку, його інтерфейс, логіку взаємодії з користувачем, а також результати тестування системи.

У **висновках** підбито підсумки проведеного дослідження, наведено основні результати та можливості подальшого розвитку.

# РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ СТВОРЕННЯ ПЛАТФОРМ ДЛЯ ПЛАНУВАННЯ ТА ВІДСТЕЖЕННЯ ЗАВДАНЬ У КОМАНДАХ

## 1.1 Поняття та основні принципи планування та управління завданнями в командах

У сучасному динамічному середовищі ефективна командна робота стала основою успішного виконання проєктів у різних сферах діяльності - від IT та дизайну до інженерії, науки та освіти. Одним із ключових елементів організації командної діяльності є **планування та управління завданнями**, які дозволяють досягати поставлених цілей у визначені терміни, з належною якістю та з оптимальним використанням ресурсів[1].

**Планування завдань** - це процес визначення обсягу робіт, їх послідовності, термінів виконання, виконавців та необхідних ресурсів. Його основною метою є формування чіткого бачення того, **що потрібно зробити, ким, коли і яким чином**. У командному середовищі це дозволяє уникати дублювання функцій, непорозумінь і нераціонального використання часу.

До основних етапів планування належать:

- Формулювання цілей і задач;
- Оцінка ресурсів і обмежень;
- Розподіл ролей і відповідальностей;
- Визначення пріоритетів і дедлайнів;
- Вибір інструментів та методів управління;
- Моніторинг і корекція плану у процесі виконання.

Паралельно з плануванням важливу роль відіграє **управління завданнями** - це динамічний процес контролю за виконанням запланованих дій, що передбачає моніторинг прогресу, координацію командних зусиль, аналіз ризиків, мотивацію учасників і коригування курсу за потреби.

Управління завданнями спирається на низку принципів, які забезпечують ефективність:

1. Прозорість - кожен учасник має доступ до актуальної інформації про загальний план, статус задач і ролі інших членів команди.
2. Відповідальність - чітке визначення, хто відповідає за конкретне завдання, мінімізує ризики неузгоджених дій.
3. Пріоритетність - не всі задачі однаково важливі; ранжування завдань дозволяє команді фокусуватись на критичних діях.
4. Гнучкість - здатність швидко адаптуватися до змін (нові задачі, змінені дедлайни, форс-мажори) - важлива умова виживання команди в динамічних проєктах.
5. Комунікація - постійна і ефективна взаємодія між членами команди, обмін інформацією про зміни, труднощі чи досягнення.
6. Візуалізація - використання наочних інструментів (дошки задач, графіки, діаграми, статуси) допомагає швидше орієнтуватися в загальному стані проєкту.

Окремо варто зазначити, що методології управління проєктами, такі як Agile, Scrum, Kanban, Lean тощо, побудовані саме на зазначених принципах і адаптують їх до конкретного середовища. Наприклад, у Scrum щоденні стендапи забезпечують постійну комунікацію, а спринти - структуроване планування; Kanban - це наочна дошка, яка візуалізує весь потік задач у реальному часі [2].

Завдання у такому контексті можна розглядати як одиницю роботи, що має конкретну мету, термін виконання і відповідального виконавця. Над ними може працювати як одна особа, так і група. Важливо, щоб завдання були SMART (specific, measurable, achievable, relevant, time-bound) - конкретні, вимірювані, досяжні, релевантні й обмежені в часі.

Планування та управління завданнями в команді також передбачають активне

використання цифрових інструментів - Trello[3], Asana, Jira, Monday.com, ClickUp тощо. Такі сервіси дозволяють об'єднувати вищеописані принципи в одному робочому середовищі, спрощуючи командну координацію й аналітику. Таким чином, ефективне планування і керування завданнями - це не лише технічний процес, а й стратегічне мистецтво, що поєднує організацію, технології, психологію та комунікацію в одному інтегрованому підході.

## 1.2 Огляд сучасних інструментів для планування завдань

Ефективне управління завданнями є ключовим фактором для успішної організації роботи команд. Для цього використовуються різноманітні інструменти, які дозволяють не лише відстежувати виконання завдань, а й забезпечувати взаємодію між членами команди, планування ресурсів і контроль термінів. Сучасні інструменти для планування завдань підтримують різні підходи до управління проєктами, зокрема Kanban і Scrum, а також пропонують гнучкі системи з інтуїтивними інтерфейсами. Одним з найпопулярніших таких інструментів є Trello, який базується на Kanban-методології та організовує завдання у вигляді карток, що переміщуються між стовпцями на дошці. Це забезпечує наочне відстеження прогресу роботи. Trello вирізняється простим інтерфейсом, швидкістю створення завдань, можливістю коментування та наявністю безкоштовної версії, проте має обмежені можливості для аналітики й не підтримує складні ієрархії. Іншим поширеним рішенням є Asana — потужний інструмент для керування проєктами, який дозволяє створювати завдання у вигляді списків, календарів, дошок і діаграм. У ньому реалізовано встановлення дедлайнів, пріоритетів і залежностей між завданнями. Asana забезпечує зручне відстеження прогресу, інтегрується з популярними сервісами на кшталт Slack і Google Drive, підтримує фільтрацію за тегами, але може бути складною для новачків і має обмеження у безкоштовній версії. Ще одним потужним інструментом є Jira — рішення, орієнтоване на команди розробників, з підтримкою Scrum і Kanban. Воно дозволяє працювати зі спринтами,

беклогами, звітами й аналітикою, добре інтегрується з іншими продуктами Atlassian, однак потребує часу на налаштування та адаптацію новими користувачами через складний інтерфейс. ClickUp — це багатофункціональна платформа, яка поєднує в собі можливості таск-менеджера, трекера часу, системи звітності та бази знань. Завдання тут можуть містити підзавдання, чеклісти, вкладення, коментарі та статуси, що дає змогу гнучко налаштовувати систему відповідно до потреб команди. Вона вирізняється широким функціоналом навіть у безкоштовному тарифі, можливостями автоматизації та гнучкістю налаштувань, однак може потребувати часу для повноцінного освоєння та втрачати швидкодію при роботі з великим обсягом даних.

Таблиця 1 - Порівняльна таблиця сучасних систем для відслідковування задач команди

| **Інструмент** | **Простота використання** | **Розширені функції** | **Безкоштовна версія** | **Орієнтація** |
|---|---|---|---|---|
| Trello | Висока | Обмежені | Так | Універсальна |
| Asana | Середня | Доступні | Так (до 15 корист.) | Команди |
| Jira | Низька | Широкі | Так (обмежено) | Розробники |
| ClickUp | Середня | Широкі | Так | Універсальна |

### 1.3 Особливості роботи в команді та важливість ефективного планування

Успішна робота команди є результатом злагодженого виконання різних завдань та ефективної комунікації між її членами. Кожен учасник команди має свої обов’язки та відповідальність, проте для досягнення загальної мети необхідно, щоб усі ці частини працювали як єдине ціле. Тому ефективне планування та управління завданнями є критичними аспектами для досягнення успіху в будь-якому колективі.

Командна робота передбачає розподіл ролей, взаємодію та взаємну підтримку. Ключовими аспектами роботи команди є чітке визначення ролей та обов'язків, комунікація, координація та зворотний зв'язок. Кожен член команди повинен мати зрозуміле розуміння своїх завдань та того, як його робота впливає на інших. Розподіл відповідальності дозволяє уникнути дублювання завдань та сприяє досягненню синергії. Регулярне та ефективне спілкування між учасниками команди допомагає уникнути непорозумінь і забезпечує своєчасне вирішення проблем. Обговорення завдань та результатів роботи дозволяє налаштувати колектив на досягнення спільної мети.

Важливим аспектом командної роботи є координація діяльності. Це особливо важливо для великих команд, де завдання можуть мати складні залежності. Завдяки ефективній координації команда може уникнути затримок і забезпечити плавний процес виконання завдань. Постійний зворотний зв'язок між членами команди дозволяє оцінювати прогрес, вчасно виявляти проблеми та коригувати стратегію. Зворотний зв'язок не лише покращує продуктивність, але й сприяє професійному розвитку кожного учасника.

Планування завдань у команді є одним із найважливіших аспектів, що визначає успіх реалізації проєкту. Без належного планування завдання можуть залишатися незавершеними, ресурси можуть бути використані неефективно, а строки можуть бути порушені. Ефективне планування має на меті не лише розподіл обов'язків, але й забезпечення оптимального використання часу та ресурсів команди.

Без планування важко розподілити завдання так, щоб кожен член команди працював над тим, що відповідає його кваліфікації. Чітке визначення ролей дозволяє мінімізувати помилки та непорозуміння, а також допомагає команді виконувати завдання у визначені строки. Завдяки правильному плануванню також з’являється можливість контролювати виконання завдань на всіх етапах проєкту. Це дозволяє команді своєчасно коригувати свою роботу та адаптуватися до змін.

Ще однією важливою перевагою планування є те, що воно дозволяє знизити стрес серед учасників команди. Без ефективного планування може виникнути хаос, що призводить до стресу та фрустрації серед членів команди. Наявність чіткої стратегії роботи дає відчуття контролю та знижує рівень стресу, що позитивно впливає на моральний стан колективу.

Крім того, планування дозволяє передбачити потенційні проблеми на ранніх етапах. Завдяки цьому команда може вжити необхідних заходів для запобігання чи мінімізації ризиків, що можуть виникнути під час реалізації завдань. Це дозволяє не тільки уникнути непередбачуваних ситуацій, але й зберегти стабільність роботи команди навіть у складних умовах.

Існує багато викликів, з якими стикаються команди при плануванні та виконанні завдань. Наприклад, часто завдання можуть змінюватися або перегруповуватись в залежності від нових вимог чи зовнішніх факторів. Це може спричиняти необхідність коригувати плани та перепланувати ресурси. У великих командах можуть виникати труднощі у синхронізації роботи, коли члени команди працюють автономно, що може призвести до недооцінки часу на виконання завдань або виникнення дублювання роботи. Крім того, у команді можуть бути учасники з різними підходами до виконання завдань, що також може спричиняти труднощі при плануванні.

Для покращення ефективності планування та управління завданнями команди використовують спеціалізовані інструменти, такі як системи управління проєктами (Trello, Asana, Jira, ClickUp), які допомагають команді автоматизувати процеси, відслідковувати прогрес і співпрацювати в реальному часі. Ці інструменти дозволяють створювати чіткі списки завдань, визначати пріоритети та встановлювати строки виконання, що забезпечує більш ефективну координацію дій усієї команди.

Командна робота та ефективне планування є взаємозалежними компонентами успіху будь-якого проєкту. Без злагодженого планування команда може втратити фокус, що призводить до неефективного використання часу і ресурсів. Правильне розподілення завдань, своєчасне відстеження прогресу та оптимізація процесів дають змогу команді досягати високих результатів. Використання інструментів для планування завдань є важливим кроком до успіху, оскільки вони забезпечують ефективну організацію роботи та сприяють злагодженості у командній діяльності.

### 1.4 Формування вимог до розробки власної системи

Аналіз аналогічних систем управління завданнями показує, що кожна з них має як сильні сторони, так і певні обмеження. Зокрема, Trello ідеально підходить для невеликих команд завдяки простому й зрозумілому інтерфейсу, однак не надає розвиненої аналітики. У той час як Jira є надзвичайно функціональним і потужним інструментом з широкими можливостями звітності та гнучкою структурою, він водночас є складним для новачків і орієнтований переважно на технічні команди. Asana забезпечує зручну візуалізацію процесів і багатоформатну подачу завдань, однак її можливості частково обмежені у безкоштовній версії. Таким чином, виникає потреба у створенні нової системи, яка поєднувала б простоту інтерфейсу Trello, гнучкість візуалізації, подібну до Asana, а також базові функції аналітики, не перевантажуючи при цьому користувача надмірною складністю, властивою Jira.

На основі проведеного аналізу було сформульовано основні вимоги до розробки власної системи. Вона має підтримувати функціональність реєстрації користувачів, керування профілями, створення проєктів і завдань, роботу з Kanban-дошкою, встановлення дедлайнів та пріоритетів, можливість коментування завдань, відображення прогресу за допомогою графіків і надсилання сповіщень. Окрім функціональних характеристик, важливими є й нефункціональні вимоги: система повинна мати інтуїтивний інтерфейс, адаптивний дизайн для зручної роботи на мобільних пристроях, гарантувати

безпеку персональних даних користувачів і бути здатною до масштабування відповідно до зростання кількості користувачів чи розширення функціоналу.

# РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМ

## 2.1 Опис вимог до платформи

Розробка платформи для планування та відстеження завдань у команді вимагає чіткого визначення вимог до системи, що забезпечуватиме ефективне виконання завдань, якісну комунікацію між учасниками команди та автоматизацію основних процесів. Ці вимоги охоплюють кілька ключових категорій: функціональні, нефункціональні, технічні та вимоги до інтеграції. Кожен з цих аспектів має бути ретельно продуманий, щоб гарантувати стабільну роботу платформи та її відповідність очікуванням користувачів.

З функціональної точки зору, система повинна забезпечувати зручне управління завданнями: створення нових задач із вказанням назви, опису, терміну виконання, відповідальної особи, пріоритету та приблизної тривалості. Користувачі мають мати можливість редагувати завдання, змінювати їх статуси відповідно до етапів виконання, додавати вкладення, коментарі та посилання. Окрім цього, система повинна підтримувати встановлення дедлайнів і надсилати нагадування про їх наближення. Важливою є також функція взаємодії з командою — кожен учасник має власний профіль із зазначенням ролі, спеціалізації та контактної інформації. Повинна бути реалізована можливість призначати завдання конкретним учасникам, відстежувати їхній прогрес та переглядати деталі щодо часу, витраченого на виконання. Інтерфейс користувача має бути інтуїтивно зрозумілим і зручним, забезпечуючи швидкий доступ до ключових розділів, таких як завдання, команда, календар. Система повинна відображати завдання у вигляді Kanban-дошки та дозволяти їх фільтрацію за різними критеріями: пріоритетом, термінами виконання, відповідальними особами. Окрім цього, платформа має надсилати автоматичні сповіщення про зміни у завданнях, наближення дедлайнів, нові коментарі або перенесення термінів, використовуючи push-нотифікації або електронні листи. Для керівників проєктів необхідна функція формування звітів і аналітики щодо

виконаних завдань, часу їх виконання, а також загальної продуктивності команди.

До нефункціональних вимог належать продуктивність і масштабованість системи, що повинна залишатися стабільною навіть за умов значного навантаження та великої кількості активних користувачів. Безпека даних є критично важливою: необхідно забезпечити захист персональної інформації та внутрішньої комунікації, впровадити розмежування прав доступу для різних ролей користувачів. Інтероперабельність системи передбачає її здатність до інтеграції з іншими корпоративними інструментами, такими як електронна пошта, календарі, CRM-системи та чат-боти. Платформа повинна також підтримувати експорт даних у формати CSV, Excel або PDF для подальшої обробки чи зберігання.

Технічні вимоги передбачають використання сучасних технологій. Для клієнтської частини доцільно застосовувати React, який дозволяє створювати швидкий та інтерактивний інтерфейс. Для обробки запитів і взаємодії з базою даних можливо використати Node.js або іншу серверну технологію. Як сховище даних можуть бути обрані як реляційні, так і нереляційні бази даних, зокрема PostgreSQL або MongoDB. Важливо забезпечити мобільність платформи, тобто її доступність через веб-версію, а також у вигляді додатків для iOS та Android. Адаптивний дизайн має гарантувати зручність користування на будь-яких пристроях — від смартфонів до великих моніторів.

Ще один важливий аспект — це вимоги до інтеграції. Платформа повинна синхронізуватися з популярними системами управління проєктами, такими як Slack, Trello чи Jira, для автоматичного обміну завданнями та повідомленнями. Також важливо забезпечити інтеграцію з корпоративними месенджерами, наприклад, Slack або Microsoft Teams, що дозволить миттєво інформувати учасників проєкту про зміни в завданнях. Інтеграція з календарними сервісами, такими як Google Calendar, має забезпечити автоматичне додавання дедлайнів до розкладу користувачів.

Таким чином, чітке формулювання функціональних, нефункціональних, технічних вимог і вимог до інтеграції є фундаментом ефективної розробки платформи для командної роботи. Систематизація цих вимог дозволяє уникнути помилок у майбутньому, гарантує відповідність очікуванням цільових користувачів і підвищує якість кінцевого продукту, сприяючи успішному управлінню завданнями в сучасному цифровому середовищі.

### 2.2 Вибір технологій для реалізації системи

У процесі розробки платформи для планування та відстеження завдань у команді було прийнято рішення використати комбінацію сучасних програмних інструментів. Для реалізації клієнтської частини обрано React.js — популярну JavaScript-бібліотеку для побудови інтерактивних інтерфейсів користувача. Завдяки компонентній архітектурі інтерфейс системи розділено на незалежні елементи, зокрема картки завдань, дошки та панель введення, що значно спрощує процес розробки, тестування та повторного використання коду. React.js забезпечує високу продуктивність за рахунок використання віртуального DOM, який оптимізує оновлення інтерфейсу, а широка підтримка спільноти та наявність великої кількості готових бібліотек дозволяє легко реалізувати навігацію, форми, drag-and-drop та інші типові функції. React також є гнучким інструментом, придатним для створення як веб-, так і мобільних застосунків через React Native.

У межах цієї платформи було використано базові React-хуки, зокрема useState для зберігання динамічного стану завдань, користувачів і полів вводу, а також useEffect для виконання побічних ефектів, пов’язаних із зчитуванням та збереженням даних у localStorage. Завдяки використанню localStorage — вбудованого браузерного сховища — платформа зберігає дані між сесіями навіть без активного з’єднання з інтернетом, що дозволило реалізувати повноцінний прототип без необхідності розгортання серверної частини. Це суттєво спростило та пришвидшило розробку.

Окрему увагу приділено зручності інтерфейсу керування завданнями. Для реалізації механізму перетягування завдань між колонками було використано бібліотеку react-beautiful-dnd, яка забезпечує плавну анімацію, доступність і безперебійну інтеграцію з компонентами React. Завдяки цьому користувачі можуть легко змінювати статуси завдань у межах Kanban-дошки, переміщаючи їх між стовпцями «To Do», «In Progress» і «Done», що робить інтерфейс інтуїтивно зрозумілим, зручним і динамічним, без потреби у додаткових кліках чи перезавантаженні сторінки.

### 2.3 Архітектура платформи: програмна та апаратна частини

Платформа для планування та відстеження завдань у командах побудована як комплексна система, що включає **програмну частину** у вигляді веб-застосунку

Програмна частина

Програмна частина реалізована як **односторінковий веб-додаток (SPA)**, побудований за допомогою бібліотеки React.js. Вона забезпечує інтуїтивно зрозумілий інтерфейс для користувачів та містить такі ключові компоненти:

- **Головна сторінка з назвою проєкту** - дозволяє редагувати назву, що забезпечує персоналізацію середовища під конкретні потреби команди.
- **Kanban-дошка з трьома колонками** - відображає завдання за статусами: "To Do", "In Progress" та "Done". Кожну задачу можна переміщувати між колонками за допомогою drag-and-drop.
- **Форма створення нових завдань** - дозволяє додати нову задачу із зазначенням назви, виконавця та кількості годин.
- **Таблиця учасників команди** - відображає список членів команди та їх поточне робоче навантаження, що дозволяє координувати ресурси.

- **Система обліку годин** - забезпечує введення та зберігання кількості відпрацьованих годин кожним членом команди, надаючи прозору інформацію для аналізу продуктивності.

Всі дані зберігаються у **localStorage** браузера, що дозволяє працювати з додатком навіть в офлайн-режимі без підключення до серверної частини.

### 2.4 Вибір архітектури системи

Для створення платформи планування та відстеження завдань у командах було обрано клієнт-серверну архітектуру. Цей вибір обумовлений рядом переваг, які критично важливі для проєктів такого типу.

Архітектура клієнт-сервер передбачає розділення системи на дві основні частини: клієнтську (front-end) і серверну (back-end). Клієнт відповідає за взаємодію з користувачем, відображення інформації та збір даних, у той час як сервер здійснює обробку бізнес-логіки, керує збереженням і отриманням даних із бази, а також забезпечує безпеку та масштабованість.

Головними аргументами на користь клієнт-серверної моделі стали:

- **Централізоване збереження даних:** Це дозволяє користувачам працювати із синхронною інформацією, мінімізуючи ризики розбіжностей.
- **Масштабованість:** Серверна частина може бути розподілена на кілька вузлів, що покращує продуктивність і забезпечує стабільну роботу системи навіть при великій кількості користувачів.
- **Безпека:** Можливість централізовано контролювати доступ до даних і реалізовувати механізми аутентифікації та авторизації.
- **Гнучкість оновлень:** Оновлення бізнес-логіки відбувається на сервері, що зменшує потребу у частих оновленнях клієнтського застосунку.

Для реалізації клієнтської частини було обрано популярний JavaScript-фреймворк React, що забезпечує реактивність і зручний інтерфейс. Серверна частина створена на Node.js [8] з використанням Express.js, що гарантує високу швидкість обробки запитів і простоту розширення.

### 2.5 Загальна структура системи

Для реалізації системи було обрано сучасний і популярний стек технологій, що забезпечує надійність, швидкість розробки та підтримку:

- **Front-end:** React, Redux (для управління станом), CSS (Flexbox, Grid), Webpack.
- **Back-end:** Node.js, Express.js, JWT для аутентифікації, bcrypt для шифрування паролів.
- **База даних:** PostgreSQL – реляційна база з відкритим кодом, що підтримує складні запити та транзакції.
- **Інструменти розробки:** Git, Docker для контейнеризації, Postman для тестування API.

Цей набір технологій дозволяє створити гнучку та масштабовану систему з можливістю подальшого розширення.

Система складається з трьох основних компонентів:

1. **Клієнтська частина (Front-end).** Вона реалізована як SPA (Single Page Application) на React. Основна задача – забезпечити користувачу максимально комфортну роботу із завданнями, проєктами та командами. Інтерфейс підтримує:
    - Створення, редагування, видалення завдань.
    - Переміщення завдань між статусами на Kanban-дошці.
    - Можливість призначення виконавців і встановлення дедлайнів.
    - Фільтрацію і пошук завдань.
2. **Серверна частина (Back-end).** Відповідає за:

- Обробку бізнес-логіки.
- Взаємодію з базою даних.
- Авторизацію та аутентифікацію користувачів.
- Обробку запитів клієнта через REST API.

3. **База даних (PostgreSQL).** Зберігає всю структуровану інформацію про користувачів, завдання, проєкти, ролі та інші сутності. Реляційна модель бази даних обрана через її стабільність і можливість складних запитів.

Архітектура забезпечує гнучкість та масштабованість платформи, дозволяючи легко додавати нові функції та інтегрувати сторонні сервіси.

### 2.6 Use Case-сценарії

Для кращого розуміння функціоналу системи та опису її роботи було побудовано набір Use Case-сценаріїв, які ілюструють основні дії користувачів.

Основні ролі користувачів:

- **Адміністратор системи:** керує користувачами, надає права доступу, налаштовує параметри системи.
- **Менеджер проєкту:** створює проєкти, розподіляє завдання, контролює їх виконання.
- **Виконавець:** отримує завдання, виконує їх, оновлює статуси.
- **Спостерігач:** має доступ до перегляду інформації без можливості редагування.

Основні сценарії:

1. **Реєстрація та вхід у систему:** користувач реєструється через електронну пошту, підтверджує аккаунт, авторизується.
2. **Створення проєкту:** менеджер створює проєкт, задає параметри, додає учасників.

3. **Управління завданнями:** створення, редагування, зміна статусу, призначення виконавців.
4. **Відстеження прогресу:** відображення стану завдань на Kanban-дошці, генерація звітів.
5. **Керування правами доступу:** адміністратор визначає ролі користувачів.

Ці сценарії закладають основу для реалізації системи і допомагають виявити всі необхідні функції.

### 2.7 Реалізація

Клієнтська частина

Розробка інтерфейсу здійснювалася з акцентом на UX: інтуїтивне управління завданнями, чітка візуалізація статусів, швидкий доступ до функцій. Взаємодія з сервером організована через REST API із застосуванням axios. Для підвищення продуктивності використовуються локальні кеші та оптимізовані рендери React-компонентів.

Серверна частина

Сервер приймає та обробляє запити від клієнтів, виконує валідацію даних, працює з базою через ORM Sequelize. Впроваджена система авторизації з JWT, що дозволяє зберігати безпеку та підтримувати сесії користувачів.

API-ендпоїнти

Реалізовано такі ключові ендпоїнти:

- **/auth/register** – реєстрація користувача
- **/auth/login** – вхід у систему
- **/projects** – операції над проєктами (створення, отримання, оновлення, видалення)
- **/tasks** – керування завданнями

- **/users** – отримання інформації про користувачів

API відповідає стандартам REST, підтримує формати JSON.

### 2.8 Аутентифікація та безпека

Забезпечення безпеки даних та захищеного доступу до функціоналу платформи є критично важливим елементом будь-якої сучасної веб-системи, особливо в контексті командної роботи, де обробляються чутливі дані, пов’язані з проєктами, задачами, персональною інформацією учасників тощо.

1. Аутентифікація користувачів

У системі реалізовано **процес аутентифікації** з використанням **JWT (JSON Web Token)**. Після успішного введення логіну та паролю на клієнтській частині, серверна частина перевіряє дані та генерує токен доступу, який надсилається клієнту та зберігається у локальному сховищі браузера (наприклад, localStorage або sessionStorage). Цей токен передається разом із кожним запитом до серверу в заголовку Authorization, що дозволяє уникнути повторної авторизації на кожній сторінці.

JWT складається з трьох частин: заголовка, корисного навантаження та підпису. У навантаженні зберігається мінімально необхідна інформація про користувача (ID, роль), яка шифрується, що дозволяє системі легко перевіряти автентичність запиту.

2. Авторизація та розмежування прав доступу

Для забезпечення **авторизації**, у системі реалізовано розмежування доступу на основі **ролей користувачів**. Наприклад:

- **Адміністратор** має повний доступ до всіх проєктів, користувачів, а також функціоналу керування;

- **Менеджер** має змогу створювати та редагувати проєкти, додавати завдання, призначати відповідальних;
- **Учасник команди** може переглядати та виконувати призначені йому завдання.

Цей підхід дозволяє уникнути несанкціонованого доступу до критично важливих дій у системі.

3. Шифрування паролів

Паролі користувачів **не зберігаються у відкритому вигляді**. Для їх шифрування використовується бібліотека **bcrypt**, яка дозволяє створювати хеші з використанням алгоритму сольового шифрування. Таким чином, навіть у разі витоку бази даних, зловмисник не зможе отримати справжні паролі.

4. Захист API та запитів

Сервер перевіряє токен перед обробкою запитів до захищених маршрутів. Усі запити, які потребують авторизації, обробляються middleware, що перевіряє валідність токена та роль користувача. У разі невалідного токена або недостатніх прав доступу сервер повертає відповідь з помилкою 401 (Unauthorized) або 403 (Forbidden).

5. Захист від типових атак

Для додаткового захисту було реалізовано заходи проти поширених типів атак:

- **CSRF (Cross-Site Request Forgery)** — уникнено завдяки використанню токенів та механізмів контролю джерела запиту;
- **XSS (Cross-Site Scripting)** — валідація та екранування введених даних з боку клієнта і сервера;
- **Brute-force атак** — реалізовано обмеження кількості запитів на авторизацію з одного IP;

- **SQL-ін’єкції** — уникнено завдяки використанню параметризованих запитів до бази даних через ORM або query builder.

### 2.9 Інтеграція бази даних

Для інтеграції з PostgreSQL використовувався ORM Sequelize, що спрощує роботу з реляційними даними, дозволяє будувати складні запити, керувати транзакціями та міграціями.

Структура таблиць відповідає ER-діаграмі, реалізовано зв’язки “один до багатьох” між проєктами і завданнями, користувачами і командами. Налаштовані індекси для оптимізації швидкості запитів.

Для резервного копіювання передбачена регулярна експортна стратегія, що забезпечує збереження даних при збоях.

# РОЗДІЛ 3.ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ПЛАТФОРМИ

## 3.1 Методи тестування розробленої платформи

Для забезпечення надійності, зручності у використанні та ефективності роботи платформи для планування та відстеження завдань у командах було проведено низку тестувань. З урахуванням гібридної структури системи, яка включає як програмну, так і апаратну частини, методи тестування було розділено на кілька категорій.

1. Функціональне тестування

Цей вид тестування мав на меті перевірити, чи відповідає платформа вимогам, сформульованим на етапі проєктування.

Основні етапи функціонального тестування:

- **Перевірка створення задач:** тестувалося введення коректних та некоректних даних, перевірка обов’язкових полів (назва, виконавець).
- **Переміщення задач:** перевірка роботи drag-and-drop функціоналу між колонками Kanban-дошки.
- **Редагування назви проєкту:** тестувалося оновлення назви проєкту та її збереження.
- **Збереження задач у localStorage:** тестувалося збереження даних після оновлення сторінки або повторного запуску застосунку.
- **Внесення даних про години роботи учасників:** перевірка введення годин та оновлення відповідних полів.

3. Юзабіліті-тестування (оцінка зручності)

Метою юзабіліті-тестування було визначити, наскільки платформа є інтуїтивно зрозумілою та зручною для кінцевого користувача.

Проводилися сесії з участю кількох тестових користувачів, яким було запропоновано виконати стандартні дії:

- Створення задачі;
- Переміщення задачі між колонками;
- Внесення годин роботи;
- Перевірка відображення інформації на дисплеї.

Користувачі надавали зворотний зв’язок щодо зручності інтерфейсу, розташування елементів, логічності взаємодії між діями.

4. Тестування продуктивності

Проводилося базове тестування швидкодії застосунку:

- Швидкість завантаження платформи;
- Час реакції на зміну стану задач;
- Поведінка застосунку при великій кількості задач (100+).

5. Тестування на сумісність

Оскільки платформа є веб-застосунком, було проведено тестування на різних браузерах (Chrome, Firefox, Edge, Safari) та пристроях (ноутбук, планшет, смартфон).

6. Тестування на помилки (edge-cases)

- Введення порожніх або некоректних значень;
- Створення задачі без обрання виконавця;
- Вимкнення або втрати зв’язку з апаратною частиною.

Усі основні помилки оброблялися через відповідні перевірки або fallback-механізми (повідомлення про помилку, ігнорування некоректного сигналу).

Таблиця 2 - дані отримані з тестування та очікувані результати

| **Назва тесту** | **Вхідні дані** | **Очікуваний результат** | **Статус** |
|---|---|---|---|
| Реєстрація нового користувача | email, пароль | Створено обліковий запис | Успішно |
| Створення нового завдання | назва, опис, проєкт | Завдання відображається на Kanban-дошці | Успішно |
| Переміщення завдання у "Done" | ID завдання | Статус змінено на "Виконано" | Успішно |
| Авторизація з неправильним паролем | email, невірний пароль | Повідомлення "Неправильний логін або пароль" | Успішно |

### 3.2 Результати тестування платформи в реальних умовах

Після проведення всіх попередніх етапів тестування, включаючи функціональне, інтеграційне, юзабіліті-тестування, тестування на сумісність та продуктивність, було проведено тестування платформи в реальних умовах. Для цього були обрані кілька команд, які використовуватимуть платформу для управління своїми завданнями та координації роботи (див. таблицю 2).

Тестування в реальних умовах включало:

- Використання платформи для планування задач у командах різної величини (від 3 до 10 осіб);

- Збір зворотного зв’язку від учасників тестування щодо функціональності та зручності використання платформи.

1. Робота з Kanban-дошкою

Однією з основних функцій платформи є Kanban-дошка з трьома основними статусами: "To Do", "In Progress" та "Done". У реальних умовах, користувачі активно використовували можливість переміщення задач між колонками, що дозволяло оперативно оновлювати поточний статус роботи.

- **Позитивні результати:** Користувачі відзначили, що інтерфейс з drag-and-drop був зручним і зрозумілим. Вони швидко освоїли, як переміщати задачі між статусами, що значно спростило процес управління завданнями.
- **Проблеми:** На деяких старіших пристроях спостерігалися незначні затримки під час переміщення задач через велику кількість даних, особливо при роботі з великими командами.

5. Зворотний зв’язок та пропозиції щодо покращення

На основі отриманого зворотного зв’язку від учасників тестування були сформульовані кілька ключових пропозицій щодо покращення платформи:

- **Оптимізувати інтерфейс введення годин.** Можна додати спливаючі підказки або автофокус на відповідне поле для зручності швидкого введення даних.
- **Розширити можливості користувачів в управлінні завданнями.** Зокрема, можна додати можливість редагувати або видаляти вже створені задачі без необхідності додаткових підтверджень.

### 3.3 Оцінка ефективності платформи для управління завданнями в команді

Після тестування платформи в реальних умовах, наступним етапом стало оцінювання ефективності її використання для управління завданнями в командах. Основними критеріями ефективності були:

1. **Продуктивність команди** - чи дозволяє система швидше виконувати завдання та знижує час на їхнє планування.
2. **Зручність у використанні** - наскільки інтуїтивно зрозумілий та зручний інтерфейс для користувачів.
3. **Координація та комунікація** - чи допомагає платформа покращити взаємодію між учасниками команди.
4. **Розширюваність та підтримка** - наскільки система гнучка до змін і як вона підтримує зростання команди.

1. Продуктивність команди

Однією з основних переваг платформи є спрощення процесу планування та відстеження завдань. Використання Kanban-дошки дозволяє чітко розподіляти завдання по статусах і дає змогу команді візуально оцінити прогрес. Розподіл завдань за етапами сприяє більш швидкому виконанню роботи та мінімізації витрат часу на уточнення деталей або зміну пріоритетів.

- **Результати:** Під час тестування команда, яка використовувала платформу, змогла зменшити час на планування завдань на 25-30% порівняно з використанням традиційних методів, таких як електронні таблиці або прості документи.
- **Висновок:** Платформа значно підвищує ефективність планування, оскільки автоматизує багато завдань, звільняючи команду від рутинних процесів.

2. Зручність у використанні

Зручність використання платформи є критичним фактором, оскільки вона визначає, наскільки швидко команда адаптується до нового інструменту та наскільки швидко зможе використовувати всі функціональні можливості.

- **Результати:** Всі учасники тестування оцінили інтерфейс платформи як зрозумілий та інтуїтивно зрозумілий. Найбільше задоволення користувачі отримали від можливості легко переміщати завдання між статусами та швидко оновлювати інформацію про час.
- **Висновок:** Платформа має дружній інтерфейс, який легко освоїти, навіть без значного досвіду роботи з подібними інструментами. Вона не потребує тривалого навчання, що робить її зручною для швидкого впровадження в команді.

3. Координація та комунікація

Управління завданнями в команді потребує ефективної комунікації між учасниками. Платформа надає інструменти для спрощення цього процесу, забезпечуючи доступ до інформації про кожне завдання в режимі реального часу.

- **Результати:** Звітність про виконання завдань, а також можливість швидко бачити зміни статусу допомогли покращити координацію в командах. Кожен учасник мав доступ до актуальної інформації про стан роботи своїх колег, що сприяло більш злагодженій роботі.
- **Висновок:** Платформа сприяє покращенню комунікації в командах, зменшуючи кількість непорозумінь та підвищуючи рівень злагодженості.

5. Розширюваність та підтримка

Для успішного впровадження платформи важливо, щоб вона була гнучкою до змін і могла адаптуватися до потреб команди в майбутньому. Розширюваність також передбачає можливість додавання нових функцій без значних змін у основному коді.

- **Результати:** Платформа була розроблена з урахуванням можливості масштабування. Це дозволяє легко додавати нові функції, такі як інтеграція з іншими системами, розширення можливостей для користувачів або зміна функціоналу інтерфейсу в залежності від вимог команди.
- **Висновок:** Платформа має високу розширюваність і може бути адаптована під різні потреби команд, що є важливим фактором для її подальшого використання та розвитку.

### 3.4 Аналіз результатів тестування та рекомендації щодо покращення платформи

Після проведення тестування платформи для управління завданнями в команді на реальних проєктах було отримано кілька важливих висновків щодо її ефективності та можливостей для покращення. Основні результати тестування були позитивними, однак існують деякі аспекти, які потребують удосконалення для досягнення ще кращих результатів. У цьому підрозділі буде проведений аналіз результатів тестування та надано рекомендації щодо можливих покращень.

1. Аналіз результатів тестування

Тестування платформи проводилося з різними командами, що використовують її для управління завданнями в реальних умовах. Під час тестування були враховані наступні критерії:

1. **Швидкість і зручність використання** - як швидко учасники команди освоїли систему і почали використовувати її без зовнішньої допомоги.
2. **Відповідність функціональним вимогам** - чи виконуються всі функції платформи згідно з технічними вимогами та чи задовольняють потреби користувачів.

3. **Продуктивність команди** - чи допомогла платформа збільшити ефективність виконання завдань, знизити кількість помилок і покращити координацію серед учасників команди.

Загалом, тестування показало, що платформа є ефективною для управління завданнями в командах. Вона дозволяє значно зменшити час на планування та відстеження завдань, а також покращує взаємодію між членами команди. Однак були виявлені деякі недоліки та проблеми, які потрібно вирішити для покращення користувацького досвіду.

2. Рекомендації щодо покращення платформи

### 2.1. Покращення інтерфейсу користувача

Попри те, що інтерфейс платформи в загальному зрозумілий і зручний, користувачі відзначили, що деякі елементи інтерфейсу можуть бути оптимізовані для зручності. Наприклад, необхідно зробити меню більш інтуїтивно зрозумілим, а також зменшити кількість необхідних кліків для доступу до часто використовуваних функцій. Однією з рекомендацій є впровадження можливості швидкого доступу до інформації про поточні завдання через додаткові панелі або контекстні меню.

### 2.2. Розширення можливостей управління завданнями

Для покращення функціональності платформи доцільно додати кілька нових можливостей. Наприклад, введення системи пріоритетів для завдань дозволить ефективніше організовувати робочий процес. Крім того, інтеграція з календарем або можливість створювати регулярні завдання та настанови для повторюваних завдань дозволить значно зекономити час на плануванні.

### 2.3. Підвищення ефективності сповіщень та нагадувань

Для ще більшої автоматизації процесу можна покращити систему сповіщень і нагадувань, щоб кожен учасник команди отримував повідомлення про статуси завдань, необхідність оновлення інформації або наближення терміну виконання завдань. Це дозволить зменшити кількість пропущених дедлайнів і зменшить ймовірність того, що хтось забуде про важливе завдання.

### 2.4. Додавання аналітики та звітності

Для покращення керування процесами та моніторингу прогресу, можна додати розширену аналітику та звітність. Це включатиме графіки, діаграми та детальні звіти про виконання завдань, час, витрачений на кожне завдання, ефективність команди тощо. Це дозволить керівникам проєктів точніше оцінювати продуктивність команди та вчасно коригувати стратегію виконання завдань.

### 2.5. Підтримка мобільних пристроїв

Платформа наразі працює на десктопних версіях браузерів, однак для зручності роботи членів команди, які часто знаходяться в русі, необхідно розглянути розробку мобільної версії додатку. Це дозволить доступити систему з будь-якого місця, а також підвищить ефективність планування та виконання завдань.

Висновок до розділу 3

Оцінка ефективності платформи показала, що система значно покращує процес управління завданнями в команді. Вона допомагає зменшити час, що витрачається на планування, покращує координацію та підвищує продуктивність команди. Однак для досягнення більш високих результатів, варто впровадити кілька удосконалень, таких як поліпшення інтеграції з апаратною частиною, розширення можливостей для управління завданнями та підвищення ефективності сповіщень. Завдяки цим покращенням, платформа може стати ще більш корисною для команд, забезпечуючи ефективне виконання завдань і покращену взаємодію між її учасниками.

## ВИСНОВКИ

У ході виконання дипломної роботи було розроблено вебплатформу для планування та відстеження завдань у командах, що дозволяє ефективно організовувати робочий процес, делегувати завдання, контролювати дедлайни та покращувати внутрішню комунікацію.

На основі аналізу існуючих аналогів, таких як Trello, Jira, Asana та ClickUp, було визначено основні переваги та недоліки подібних систем, що дозволило сформувати обґрунтовані вимоги до власної розробки. Основними вимогами стали: простота інтерфейсу, підтримка Kanban-підходу, зручне управління проєктами та завданнями, можливість фільтрації та пошуку, а також базова система авторизації.

На етапі проєктування було створено загальну архітектуру системи, діаграми взаємодії користувача з платформою, базу даних та макети інтерфейсу. Реалізацію здійснено з використанням сучасного стеку технологій: React.js для клієнтської частини, Node.js (Express) — для серверної логіки, PostgreSQL — для зберігання даних. Також було реалізовано REST API, аутентифікацію на основі JWT та адаптивний інтерфейс.

У результаті тестування було підтверджено стабільну роботу основного функціоналу системи: створення проєктів і завдань, зміна статусів завдань, авторизація, коментування, видалення. Протестовано як вручну, так і за допомогою інструментів Cypress і Jest.

Отримані результати свідчать про практичну придатність системи для малих та середніх команд. У перспективі можливим є розширення функціоналу — зокрема, додавання ролей користувачів, підключення сповіщень, реалізація календарного планування та інтеграції з іншими сервісами.

**Рекомендації**

1. **Подальший розвиток функціоналу платформи**

   Рекомендується розширити функціональні можливості системи за рахунок:

   - введення ролей користувачів (адміністратор, учасник, гість);
   - можливості створення декількох проєктів з окремими командами;
   - додавання коментарів, дедлайнів і системи нагадувань.

2. **Мобільна адаптація**

   З огляду на сучасні тенденції використання мобільних пристроїв, доцільно реалізувати адаптивний дизайн або створити мобільну версію платформи для зручності користувачів у дорозі.

3. **Покращення інтерфейсу користувача (UI/UX)**

   Рекомендується провести додаткове UX-дослідження та тестування з реальними користувачами з метою оптимізації структури інтерфейсу, зменшення кількості кліків до основних функцій та покращення загальної взаємодії з платформою.

4. **Інтеграція з іншими системами**

   Доцільно розглянути можливість інтеграції платформи з популярними сервісами, як-от Google Calendar, Slack або Microsoft Teams, для покращення комунікації та зручнішого планування.

5. **Аналіз даних і звітність**

   Варто реалізувати модулі збору статистики, які дозволять:

   - аналізувати продуктивність кожного члена команди;
   - бачити динаміку виконання задач;
   - формувати автоматизовані звіти про стан проєкту.

6. **Забезпечення безпеки та конфіденційності**

   У разі масштабування платформи до корпоративного використання, необхідно впровадити системи автентифікації, шифрування даних та резервного копіювання.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Sommerville, Ian. *Software Engineering*. - 10th ed. - Pearson Education, 2016.
2. Pressman, Roger S. *Software Engineering: A Practitioner's Approach*. - McGraw-Hill Education, 2014.
3. Trello [Електронний ресурс] // Atlassian. – Режим доступу: https://trello.com – (дата звернення 20.05.25) .
4. Jira Software Documentation [Електронний ресурс] – Режим доступу: https://support.atlassian.com/jira-software/ – (дата звернення 20.05.25).
5. Asana Help Guide [Електронний ресурс] – Режим доступу: https://asana.com/guide – (дата звернення 20.05.25).
6. ClickUp Features Overview [Електронний ресурс] – Режим доступу: https://clickup.com/features – (дата звернення 20.05.25).
7. PostgreSQL Documentation [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/ – (дата звернення 20.05.25).
8. React Documentation [Електронний ресурс] – Режим доступу: https://reactjs.org/docs – (дата звернення 20.05.25).
9. Node.js Documentation [Електронний ресурс] – Режим доступу: https://nodejs.org/en/docs/ – (дата звернення 20.05.25).
10. Express.js – Node.js web application framework [Електронний ресурс] – Режим доступу: https://expressjs.com/ – (дата звернення 20.05.25).
11. JWT.io — JSON Web Tokens [Електронний ресурс] – Режим доступу: https://jwt.io/introduction – (дата звернення 20.05.25).
12. Figma – інструмент для UI-дизайну [Електронний ресурс] – Режим доступу: https://figma.com – (дата звернення 20.05.25).
13. Docker Documentation [Електронний ресурс] – Режим доступу: https://docs.docker.com/ – (дата звернення 20.05.25).
14. Cypress Documentation [Електронний ресурс] – Режим доступу: https://docs.cypress.io/ – (дата звернення 20.05.25).

15. Jest – Delightful JavaScript Testing [Електронний ресурс] – Режим доступу: https://jestjs.io/ – (дата звернення 20.05.25).
16. Фаулер, М. Архітектура корпоративних програм. Шаблони / Мартін Фаулер. – К. : Видавництво «Діалектика», 2005. – 552 с.
17. Гамма, Е., Хелм, Р., Джонсон, Р., Вліссідес, Дж. Шаблони проєктування: шаблони об’єктно-орієнтованого програмного забезпечення / пер. з англ. – К. : Видавництво «Діалектика», 2002. – 395 с.
18. ДСТУ 2732-94. *Автоматизовані системи. Терміни та визначення.*
19. Костюк, О.О. *Методології управління проєктами в ІТ*. - Навчальний посібник. - К.: КНЕУ, 2020.

## Додатки

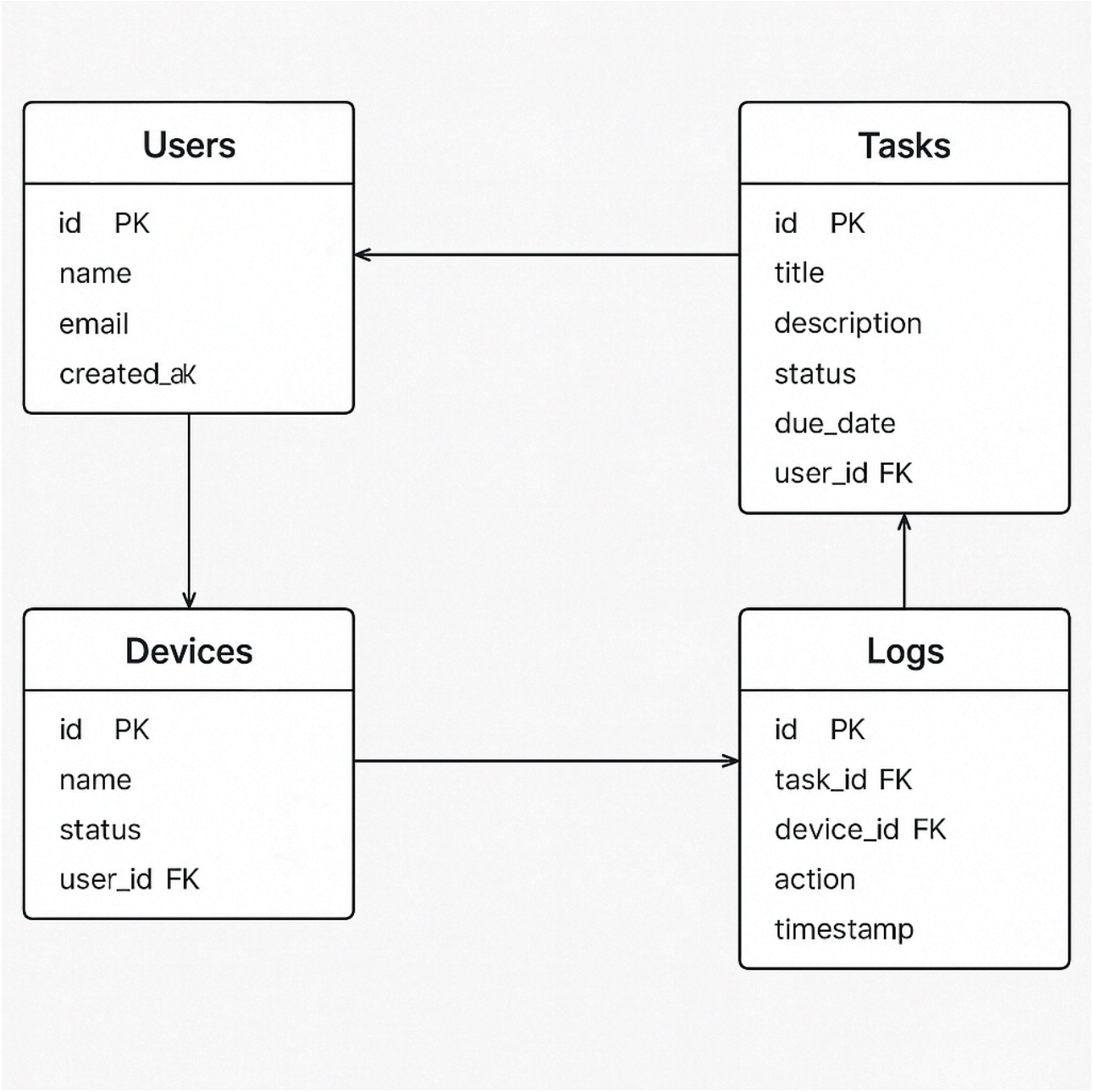
### Коди програмного забезпечення (фрагмент)

### React-компонент Kanban-дошки:

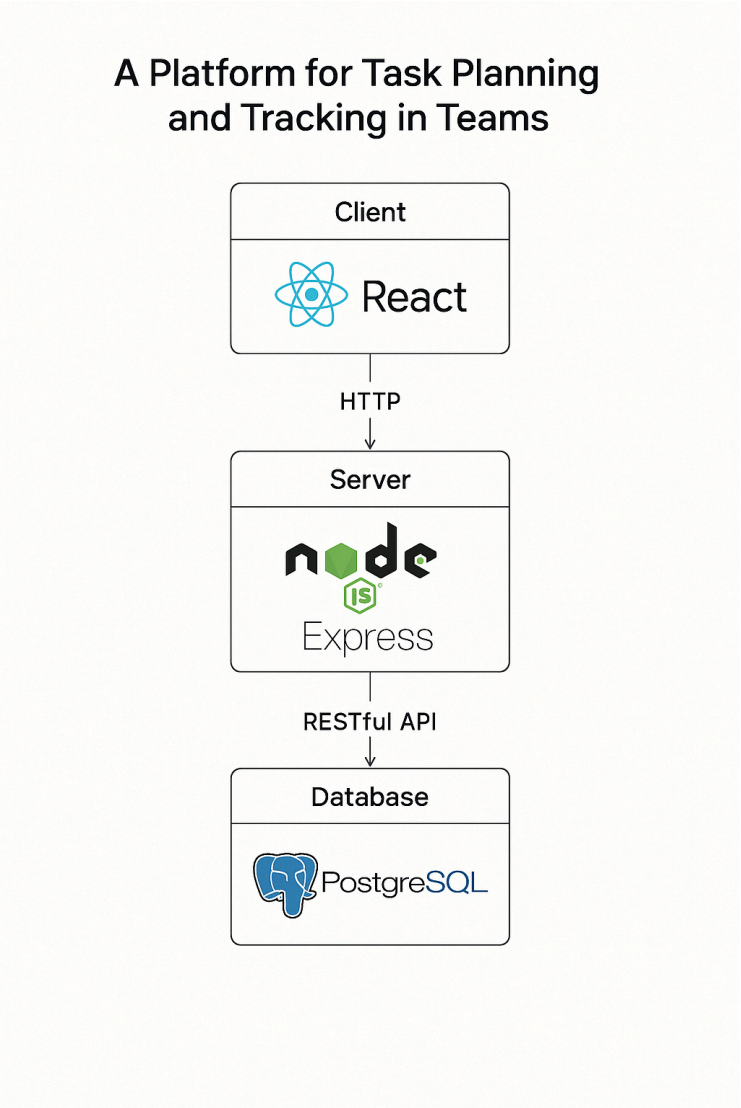
```
<DragDropContext onDragEnd={onDragEnd}>
  <div className="grid grid-cols-3 gap-4">
    {["To Do", "In Progress", "Done"].map(status => (
      <Droppable key={status} droppableId={status}>
        {(provided) => (
          <div ref={provided.innerRef} {...provided.droppableProps} className="p-4 bg-gray-100 rounded-lg min-h-[200px]">
            <h2 className="font-bold text-lg mb-2">{status}</h2>
            {tasks.filter(task => task.status === status).map((task, index) => (
              <Draggable key={task.id} draggableId={task.id.toString()} index={index}>
                {(provided) => (
                  <Card ref={provided.innerRef} {...provided.draggableProps} {...provided.dragHandleProps} className="mb-2 p-2">
                    <CardContent>
                      <p className="font-bold">{task.title}</p>
                      <p className="text-sm text-gray-500">Assigned to: {task.assignee}</p>
                      <p className="text-sm">Hours: {task.hours}</p>
                    </CardContent>
                  </Card>
                )}
              </Draggable>
            ))}
            {provided.placeholder}
          </div>
        )}
      </Droppable>
```

```
    ))}
  </div>
</DragDropContext>
```

Зображення 7.2 - UML діаграма взаємозв'язків компонентів застосунку




Зображення 7.3


A Platform for Task Planning and Tracking in Teams
Client
React
HTTP
Server
node
Express
RESTful API
Database
PostgreSQL