

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Система менеджменту ІТ-проектів»

Виконав студент групи ЗППЗ-21-1
ДІХТІЄВСЬКА Марія Сергіївна

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
ВАКАЛЮК Тетяна Анатоліївна

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ДІХТІЄВСЬКА Марія Сергіївна**

Керівник роботи: **ЧИЖМОТРИЯ Олексій Володимирович**

Тема роботи: **«Система менеджменту ІТ-проектів»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №62/с**

Вихідні дані для роботи: Об'єктом дослідження є система управління проектами в сучасних організаціях. Предметом дослідження виступають інформаційні системи, які підтримують управління проектами.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О. В.	14.02.2025	14.02.2025
2	Чижмотря О. В.	14.02.2025	14.02.2025
3	Чижмотря О. В.	14.02.2025	14.02.2025

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з вебдодатку системи управління проєктами та пояснювальної записки. Пояснювальна записка містить 72 сторінки, 26 ілюстрацій, 7 таблиць, 5 додатків та 30 використаних джерел.

Метою роботи є створення вебсистеми для автоматизації керування проєктами, завданнями та ресурсами з використанням сучасних вебтехнологій.

Архітектура базується на клієнт-серверній моделі, де PHP і MySQL забезпечують серверну логіку, а HTML, CSS, JavaScript — інтерактивний інтерфейс. AJAX використовується для асинхронного оновлення даних, а нормалізована база даних гарантує цілісність.

Додаток має зручний інтерфейс із картковим дизайном, модальними формами та адаптивним відображенням. Система підтримує авторизацію, створення проєктів і завдань, їх фільтрацію та призначення виконавців.

КЛЮЧОВІ СЛОВА: ВЕБДОДАТОК, УПРАВЛІННЯ ПРОЄКТАМИ, PHP, MYSQL, JAVASCRIPT, AJAX, АДАПТИВНИЙ ДИЗАЙН, АВТОРИЗАЦІЯ.

					ІПЗ.КР.Б – 121 – 25 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Діхтієвська М.С.			Система менеджменту ІТ-проєктів Пояснювальна записка		Літ.	Арк.
Керівник		Чижмотря О. В.						Аркушів
							3	72
Рецензент		Вакалюк Т. А.					Житомирська політехніка, група ЗІПЗ-21-1	
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's thesis consists of a web application for a project management system and an explanatory note. The explanatory note comprises 72 pages, 26 illustrations, 7 tables, 5 Annex and 30 references.

The aim of the work is to create a web-based system for automating the management of projects, tasks, and resources using modern web technologies.

The architecture is based on a client-server model, where PHP and MySQL handle server-side logic, and HTML, CSS, and JavaScript provide an interactive interface. AJAX is used for asynchronous data updates, and a normalized database ensures data integrity.

The application features a user-friendly interface with a card-based design, modal forms, and adaptive display. The system supports authentication, project and task creation, filtering, and assignee allocation.

KEYWORDS: WEB APPLICATION, PROJECT MANAGEMENT, PHP, MYSQL, JAVASCRIPT, AJAX, ADAPTIVE DESIGN, AUTHENTICATION.

					ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT.....	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У СИСТЕМАХ УПРАВЛІННЯ ПРОЄКТАМИ.....	14
1.1. Постановка задачі.....	14
1.2. Аналіз існуючих систем управління проєктами	11
1.3. Вибір архітектури веборієнтованої системи управління проєктами	18
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення.....	22
Висновки до розділу 1.....	26
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ.....	28
2.1. Визначення варіантів використання системи та об'єктно-орієнтованої структури	28
2.2. Розробка бази даних системи управління проєктами.....	32
2.3. Проєктування та реалізація ключових алгоритмів	35
2.4. Реалізація вебдодатку системи управління проєктами	38
Висновки до розділу 2.....	42
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СИСТЕМОЮ УПРАВЛІННЯ ПРОЄКТАМИ	44
3.1. Установка та налаштування параметрів системи	44
3.2. Опис інтерфейсу користувача	45
3.3. Тестування та перевірка функціональності системи.....	52
Висновки до розділу 3.....	56

ВИСНОВКИ	59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	61
ДОДАТКИ	64

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface

БД - база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

SDK – Software Development Kit

MVC – Model-View-Controller

CRUD – Create, Read, Update, Delete

DApps – Decentralized Application

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність. У сучасному світі управління проектами є ключовим елементом забезпечення ефективності організаційних процесів у різних сферах діяльності. Зростання складності проєктів та потреба в оптимізації ресурсів, термінів і комунікацій зумовлюють необхідність використання інформаційних технологій для автоматизації процесів управління. Веборієнтовані системи управління проектами дозволяють забезпечити централізований доступ до даних, підвищити прозорість процесів і полегшити взаємодію між учасниками проєкту [1].

Мета дослідження полягає в розробці веборієнтованої системи управління проектами, яка забезпечує ефективне планування, моніторинг і контроль проєктних завдань, ресурсів і термінів, а також у створенні зручного інтерфейсу для взаємодії користувачів із системою.

Об'єкт дослідження – процеси управління проектами, які включають планування, організацію, контроль і координацію завдань, ресурсів і термінів у проєктній діяльності.

Предмет дослідження – веборієнтовані інформаційні системи, призначені для автоматизації процесів управління проектами, зокрема їх архітектура, функціональність і алгоритми реалізації.

Завдання дослідження:

1. Провести аналіз напрямків використання інформаційних технологій у системах управління проектами та сформулювати постановку задачі.
2. Виконати аналіз існуючих систем управління проектами, визначивши їх переваги та недоліки.
3. Обґрунтувати вибір архітектури веборієнтованої системи управління проектами.
4. Визначити інструментальні засоби розробки та вимоги до апаратного забезпечення.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

5. Розробити об'єктно-орієнтовану структуру системи та визначити варіанти її використання.
6. Спроекувати базу даних для забезпечення функціональності системи.
7. Розробити та реалізувати ключові алгоритми управління задачами, ресурсами й термінами.
8. Реалізувати вебдодаток системи управління проєктами.
9. Розробити та описати інтерфейс користувача, а також порядок роботи із системою.
10. Провести тестування системи для перевірки її функціональності та відповідності вимогам.

Кваліфікаційна робота бакалавра структурована відповідно до поставлених завдань і складається з трьох розділів. У першому розділі проведено аналіз використання інформаційних технологій у системах управління проєктами, розглянуто існуючі системи, обґрунтовано вибір архітектури та інструментів розробки. Другий розділ присвячено проєктуванню системи, зокрема розробці структури, бази даних, алгоритмів і реалізації вебдодатку. У третьому розділі описано інтерфейс користувача, процес встановлення системи та результати її тестування. Завершують роботу висновки та список використаних джерел.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ У СИСТЕМАХ УПРАВЛІННЯ ПРОЄКТАМИ

1.1. Постановка задачі

Управління проєктами є складним процесом, який вимагає ефективного планування, координації ресурсів, контролю термінів і забезпечення комунікації між учасниками. Сучасні організації все частіше використовують інформаційні технології для автоматизації цих процесів. Розробка веборієнтованої системи управління проєктами дозволить оптимізувати виконання проєктних завдань, підвищити прозорість процесів і забезпечити зручний доступ до даних для всіх учасників [2].

Основною задачею дослідження є розробка веборієнтованої системи управління проєктами, яка забезпечує автоматизацію ключових процесів управління, таких як планування завдань, розподіл ресурсів, контроль термінів і моніторинг прогресу. Система повинна мати інтуїтивно зрозумілий інтерфейс, бути доступною через веббраузер і відповідати сучасним вимогам до безпеки та масштабованості.

Система управління проєктами повинна відповідати функціональним і нефункціональним вимогам, які наведено в таблиці нижче (табл.1.1).

Таблиця 1.1

Вимоги до системи управління проєктами

№	Функціональні вимоги	Нефункціональні вимоги
1	Створення, редагування та видалення проєктів і завдань	Забезпечення швидкодії при обробці до 100 одночасних користувачів
2	Призначення завдань користувачам і розподіл ресурсів	Доступність системи через веббраузер на різних пристроях
3	Відстеження прогресу виконання завдань	Захист даних відповідно до стандартів GDPR
5	Генерація звітів про стан проєкту	Інтуїтивно зрозумілий інтерфейс користувача
5	Налаштування ролей і прав доступу для учасників	Масштабованість для роботи з проєктами різного обсягу

Система повинна підтримувати ключові функції управління проєктами, які деталізовано в таблиці нижче (табл.1.2).

Основні характеристики системи

№	Функція	Опис	Пріоритет
1	Управління завданнями	Створення, редагування, призначення та відстеження завдань	Високий
2	Управління ресурсами	Розподіл людських і матеріальних ресурсів	Високий
3	Контроль термінів	Нагадування та сповіщення про наближення дедлайнів	Середній
4	Генерація звітів	Формування звітів про прогрес і витрати ресурсів	Середній
5	Комунікація	Інтеграція чату або системи сповіщень для учасників	Низький

Результатом дослідження є повнофункціональний вебдодаток для управління проєктами, який включає:

- Модуль управління завданнями та ресурсами.
- Інтерфейс для моніторингу прогресу проєктів.
- Систему звітності та сповіщень.
- Документацію з описом інтерфейсу та порядку роботи.

Таким чином, постановка задачі визначає основні вимоги, характеристики та обмеження для розробки веборієнтованої системи управління проєктами, що буде реалізовано в рамках дипломної роботи.

1.2. Аналіз існуючих систем управління проєктами

У сучасному світі управління проєктами вимагає використання спеціалізованих інформаційних систем, які забезпечують автоматизацію планування, координації, моніторингу та контролю проєктних завдань. Аналіз існуючих систем управління проєктами дозволяє виявити їхні сильні та слабкі сторони, а також визначити ключові вимоги до розробки нової системи. У цьому розділі розглянуто п'ять популярних систем управління проєктами: Trello, Asana, Jira, Microsoft Project і Monday.com. Для кожної системи наведено опис, переваги, недоліки та посилання на офіційні вебсайти.

1. Trello

Trello – це веборієнтована система управління проєктами, яка використовує методологію Kanban для організації завдань (рис.1.1). Користувачі створюють дошки, списки та картки для відстеження прогресу проєктів. Система підходить для невеликих команд і проєктів із простою структурою [4].

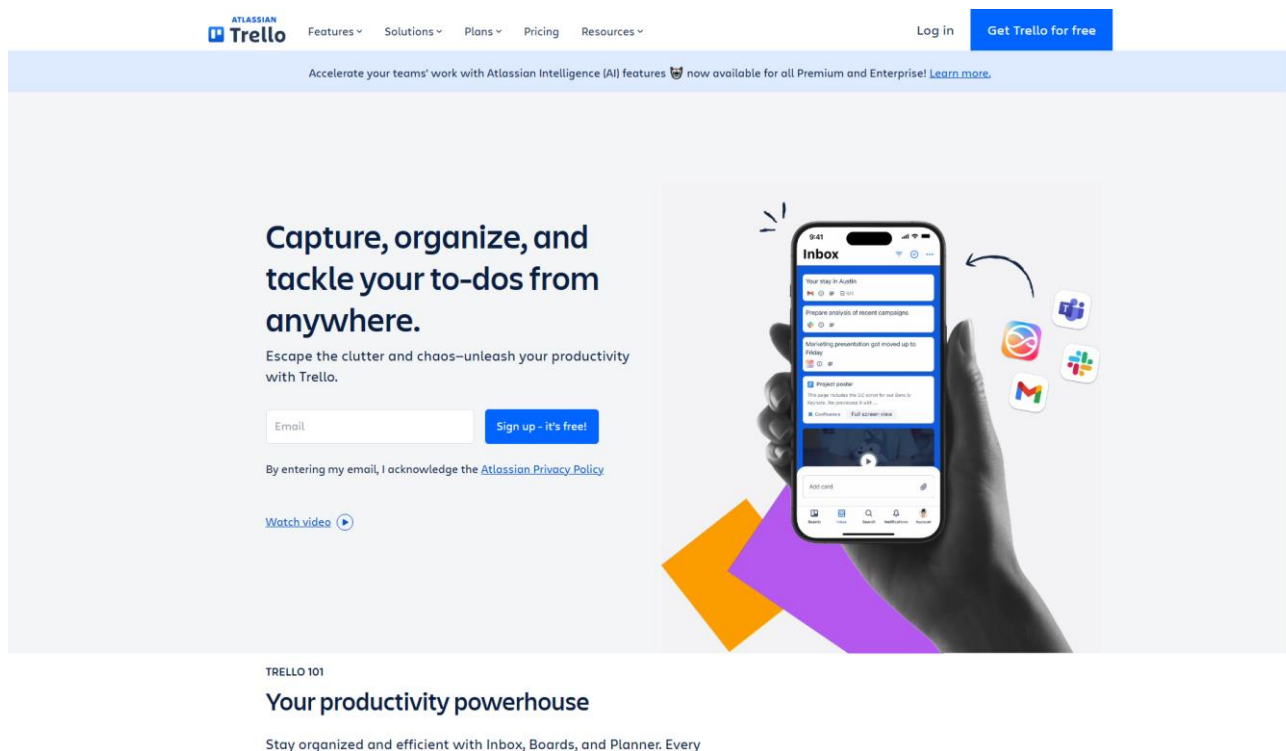


Рис.1.1 Сайт Trello

Переваги цієї системи включають інтуїтивно зрозумілий інтерфейс, заснований на Kanban-дошках, що значно спрощує організацію роботи. Безкоштовний базовий план надає достатню функціональність для малих команд, а інтеграція з популярними сервісами, такими як Slack, Google Drive і Dropbox, дозволяє легко об'єднати всі робочі процеси. Крім того, є підтримка мобільних додатків для iOS і Android, що забезпечує зручність у використанні на різних пристроях.

Однак система має деякі недоліки. Вона обмежена у функціональності для складних проєктів, особливо коли йдеться про управління великою кількістю залежностей. Відсутні вбудовані інструменти для ефективного управління ресурсами та бюджетами, що може бути проблемою для більш комплексних

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

проектів. Також система має обмежені можливості для аналітики та звітності, що може бути важливим для команд, які потребують глибокого аналізу даних.

2. Asana

Asana – це хмарна платформа для управління проектами, яка підтримує різні методи організації роботи (списки, дошки, таймлайни) (рис.1.2). Вона дозволяє командам планувати завдання, відстежувати прогрес і налаштовувати робочі процеси [6].

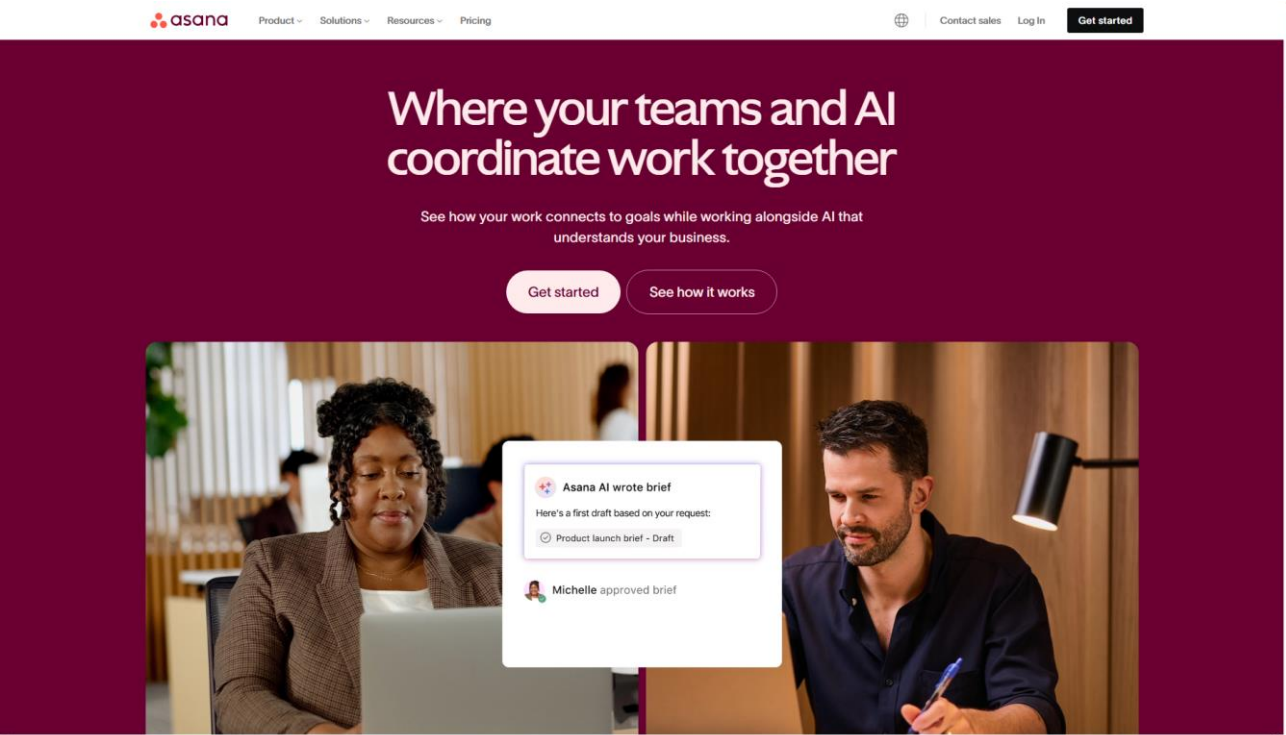


Рис.1.2 Сайт Asana

Ця система має кілька переваг, серед яких гнучкість у виборі формату відображення проектів, що дозволяє користувачам вибирати між Kanban, Gantt або списками, в залежності від потреб. Потужні інструменти для автоматизації повторюваних завдань значно знижують час, витрачений на рутинні операції. Крім того, інтеграція з популярними інструментами, такими як Zoom, Microsoft Teams і Salesforce, робить систему зручною для використання в різних робочих середовищах. Також є можливість налаштувати права доступу для різних ролей, що дозволяє точніше контролювати доступ до важливої інформації.

З іншого боку, є кілька недоліків. Преміум-плани для великих команд можуть бути досить дорогими, що обмежує доступність для деяких організацій.

Також система може бути складною для освоєння новими користувачами через велику кількість функцій і можливостей. І, незважаючи на велику кількість функціоналу, підтримка офлайн-режиму є обмеженою, що може стати проблемою у разі необхідності роботи без підключення до Інтернету.

3. Jira

Jira – це система управління проєктами, розроблена компанією Atlassian, яка орієнтована на IT-команди та розробку програмного забезпечення (рис.1.3). Вона підтримує гнучкі методології (Scrum, Kanban) і дозволяє налаштовувати робочі процеси [8].

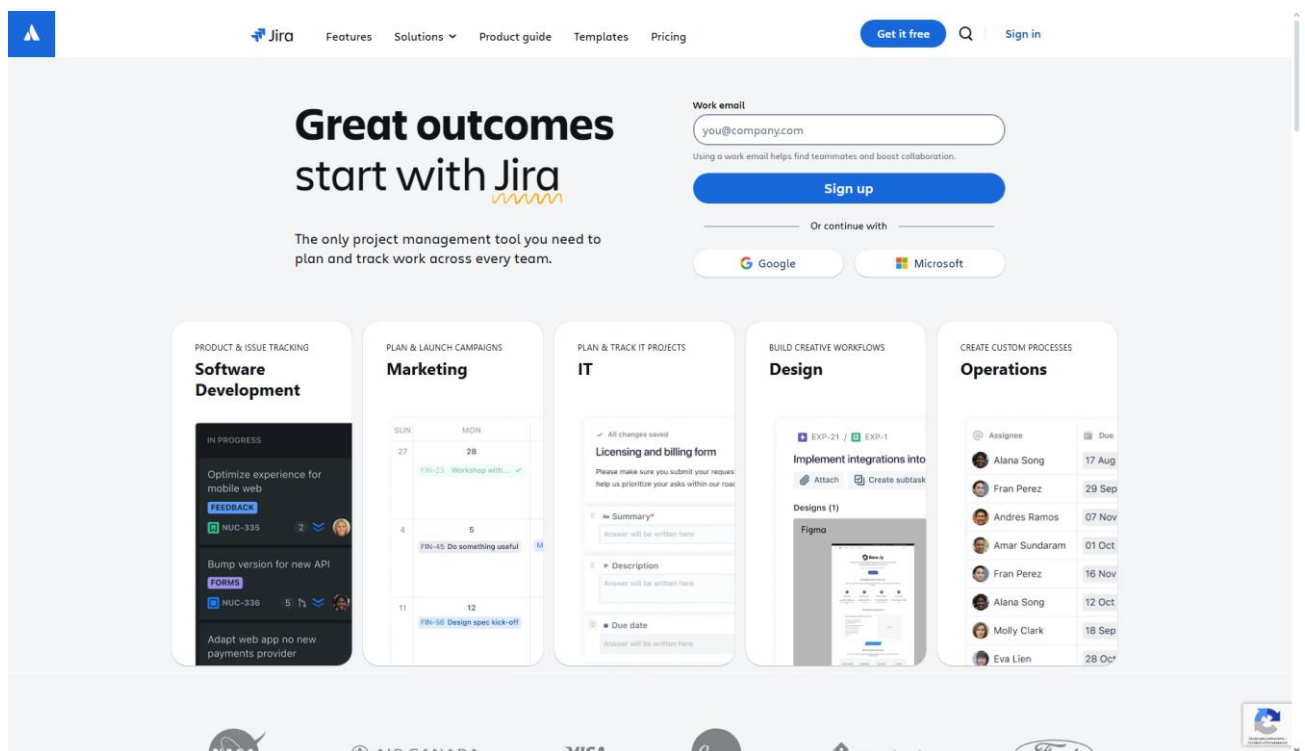


Рис.1.3 Сайт Jira

Ця система пропонує значну гнучкість завдяки глибокій кастомізації робочих процесів, що дозволяє адаптувати інструменти під специфічні потреби команди. Потужні інструменти для управління помилками та відстеження завдань допомагають ефективно координувати роботу і забезпечують якість виконання проєктів. Інтеграція з популярними інструментами розробки, такими як Bitbucket, GitHub і Confluence, робить її ідеальним вибором для технічних команд. Крім того, система надає розширені можливості аналітики та звітності, що дозволяє отримати детальну інформацію про прогрес і ефективність.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

Однак, є і деякі недоліки. Налаштування та адміністрування системи може бути складним, що вимагає додаткових ресурсів і часу. Система орієнтована переважно на технічні команди, що обмежує її використання в інших сферах, де потрібна більш універсальна платформа. Також її висока ціна для великих команд або організацій може стати значним бар'єром для впровадження в малих і середніх компаніях.

4. Microsoft Project

Microsoft Project – це потужне програмне забезпечення для управління проєктами, яке підтримує детальне планування, управління ресурсами та складання діаграм Ганта (рис.1.4). Воно доступне як у вигляді десктопного додатку, так і в хмарній версії [10].

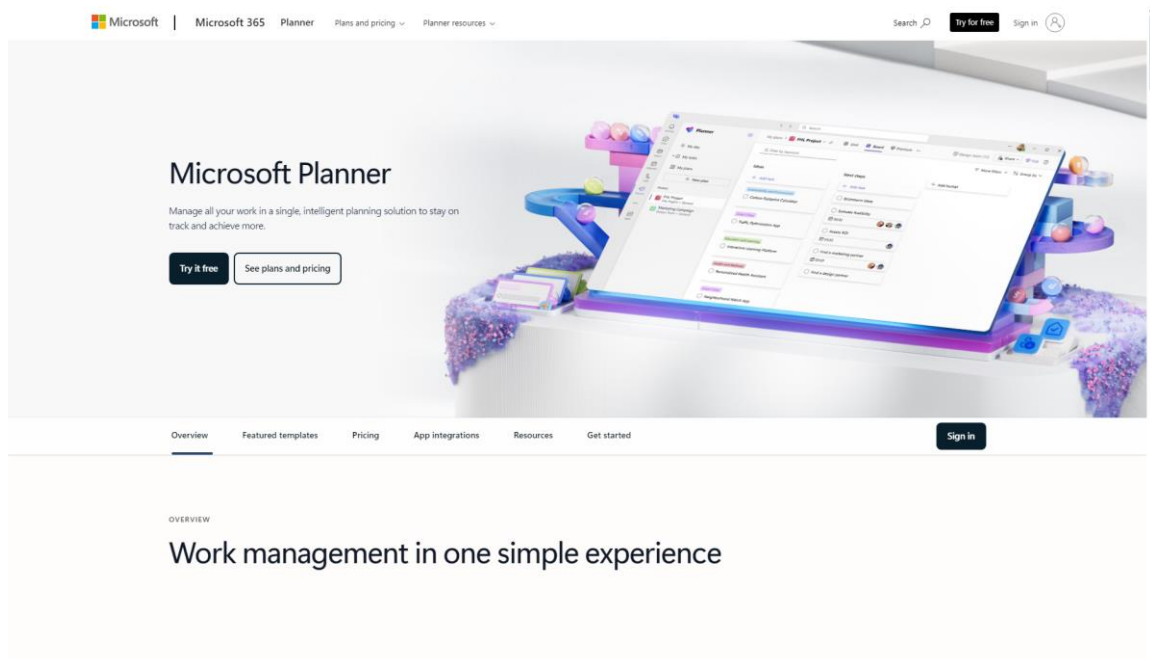


Рис.1.4 Сайт Microsoft Project

Ця система пропонує значну гнучкість завдяки глибокій кастомізації робочих процесів, що дозволяє адаптувати інструменти під специфічні потреби команди. Потужні інструменти для управління помилками та відстеження завдань допомагають ефективно координувати роботу і забезпечують якість виконання проєктів. Інтеграція з популярними інструментами розробки, такими як Bitbucket, GitHub і Confluence, робить її ідеальним вибором для технічних

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

команд. Крім того, система надає розширені можливості аналітики та звітності, що дозволяє отримати детальну інформацію про прогрес і ефективність.

Однак, є і деякі недоліки. Налаштування та адміністрування системи може бути складним, що вимагає додаткових ресурсів і часу. Система орієнтована переважно на технічні команди, що обмежує її використання в інших сферах, де потрібна більш універсальна платформа. Також її висока ціна для великих команд або організацій може стати значним бар'єром для впровадження в малих і середніх компаніях.

5. Monday.com

Monday.com – це хмарна платформа для управління проєктами, яка дозволяє командам налаштовувати робочі процеси, відстежувати прогрес і співпрацювати в реальному часі (рис.1.5). Вона підходить для різних галузей і типів проєктів [12].

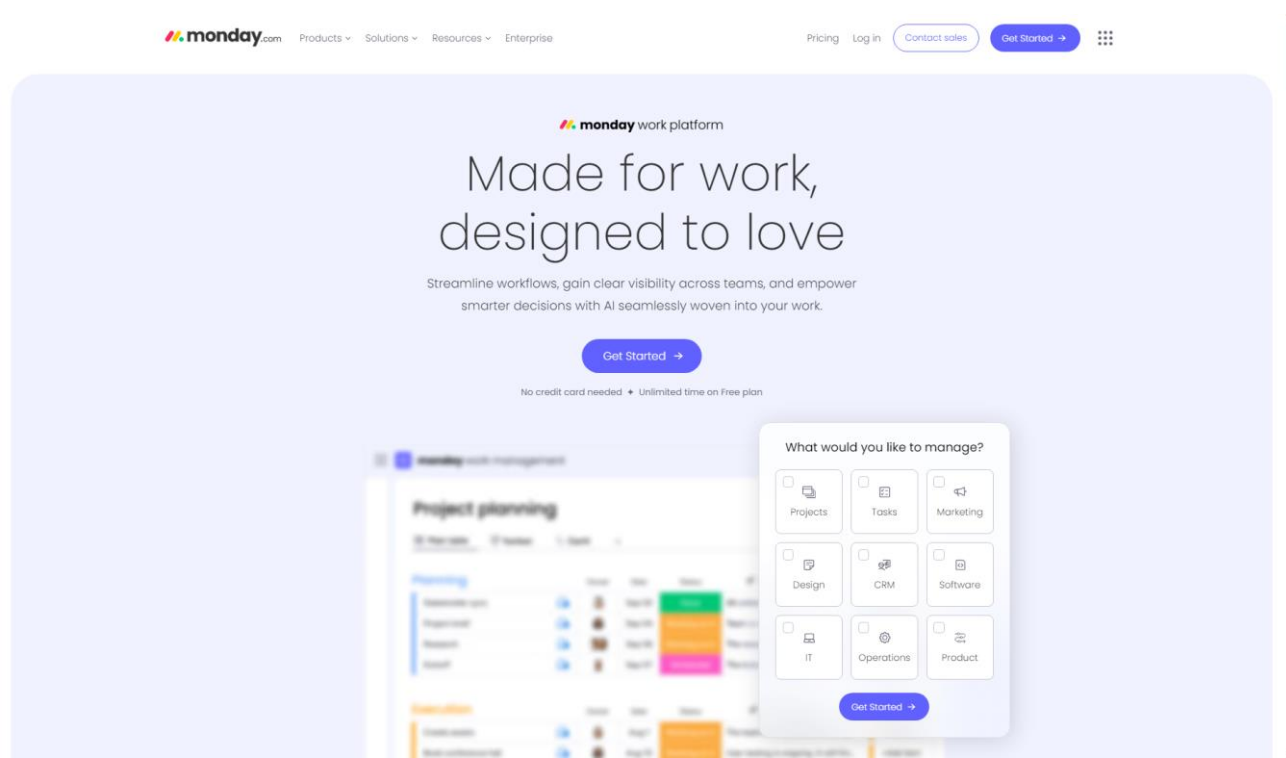


Рис.1.5 Сайт [Monday.com](https://monday.com)

Система пропонує кілька значних переваг, таких як візуально привабливий і гнучкий інтерфейс, що дозволяє легко кастомізувати його під потреби користувача. Широкий набір шаблонів для різних типів проєктів допомагає

швидко адаптувати інструмент для різних завдань. Інтеграція з популярними сервісами, такими як Slack, Zapier і Google Calendar, дозволяє зручно організувати робочі процеси, а підтримка автоматизації та сповіщень значно підвищує ефективність.

Проте є й деякі недоліки. Висока вартість, особливо для великих команд або при використанні преміум-функцій, може бути перешкодою для деяких організацій. Базовий план має обмежену функціональність, що змушує користувачів часто звертатися до платних опцій для отримання додаткових можливостей. Крім того, при управлінні великою кількістю дощок система може стати складною і потребувати додаткових ресурсів для налаштування та підтримки.

Для кращого розуміння відмінностей між розглянутими системами наведено порівняльну таблицю їхніх характеристик (табл.1.3).

Таблиця 1.3

Порівняння систем управління проєктами

Система	Тип	Основні функції	Ціна	Складність освоєння	Підходить для
Trello [4]	Хмарна	Kanban-дощки, управління завданнями	Безкоштовно/ Платно від \$5/міс	Низька	Малі команди, прості проєкти
Asana [6]	Хмарна	Списки, таймлайни, автоматизація	Безкоштовно/ Платно від \$10.99/міс	Середня	Команди середнього розміру
Jira [8]	Хмарна/Локальна	Управління Agile, звіти	Від \$7.75/міс	Висока	ІТ-команди, розробка ПЗ
Microsoft Project [10]	Хмарна/Десктоп	Діаграми Ганта, управління ресурсами	Від \$10/міс	Висока	Великі проєкти, корпорації
Monday.com [12]	Хмарна	Кастомізація, автоматизація	Від \$8/міс	Середня	Різні галузі, гнучкі проєкти

Аналіз існуючих систем управління проєктами показав, що кожна з них має свої сильні та слабкі сторони. Trello підходить для простих проєктів завдяки легкості використання, але не справляється зі складними задачами. Asana і

Monday.com пропонують гнучкість і підходять для різних типів команд, але їхня висока вартість може бути обмеженням. Jira є ідеальним вибором для ІТ-проектів, але вимагає значних зусиль для налаштування. Microsoft Project найкраще підходить для великих організацій із детальним плануванням, але його складність і ціна обмежують використання в малих командах. Отримані дані будуть використані для обґрунтування вибору архітектури та функціональності нової системи управління проектами в рамках дипломної роботи.

1.3. Вибір архітектури веборієнтованої системи управління проектами

Розробка веборієнтованої системи управління проектами вимагає ретельного вибору архітектури, яка забезпечить ефективну взаємодію між компонентами, масштабованість, безпеку та зручність підтримки. У цьому розділі розглянуто вибір архітектури для системи управління проектами з використанням стеку технологій HTML, CSS, JavaScript, PHP і MySQL.

Архітектура системи повинна забезпечувати підтримку ключових функцій, таких як управління проектами, завданнями, ресурсами та термінами. Важливою вимогою є здатність системи обробляти до 100 одночасних користувачів без втрат у продуктивності. Окрім того, система повинна відповідати вимогам безпеки, захищаючи дані згідно зі стандартами GDPR, що включає шифрування даних та контроль доступу для захисту конфіденційної інформації. Важливо, щоб система була доступною через веббраузери на різних пристроях, таких як десктопи, планшети та смартфони, що забезпечує зручний доступ для користувачів у будь-який час і з будь-якого місця. Останньою вимогою є проста підтримка системи, що досягається за допомогою використання технологій без складних фреймворків, що полегшує розробку та подальші модифікації системи.

Для реалізації системи обрано клієнт-серверну архітектуру [14] з використанням моделі MVC (Model-View-Controller) [15] на серверній стороні.

Ця архітектура забезпечує чіткий поділ логіки даних, інтерфейсу користувача та бізнес-логіки, що полегшує розробку, тестування та масштабування системи.

Основні компоненти архітектури включають клієнтську сторону, де використовується HTML для структурування контенту, CSS для створення стилів та JavaScript для забезпечення інтерактивності. Серверна сторона побудована на PHP, яке відповідає за обробку запитів, бізнес-логіку та взаємодію з базою даних. Для зберігання інформації про проєкти, завдання, користувачів і ресурси використовується база даних MySQL.

Система побудована на трирівневій архітектурі, яка складається з трьох основних рівнів. Перший рівень — це рівень представлення, де клієнтська частина реалізована за допомогою HTML, CSS і JavaScript. Цей рівень відповідає за відображення даних і взаємодію з користувачем. Другий рівень — бізнес-логіки, що працює на сервері через PHP, виконує обробку запитів, обчислення та взаємодіє з базою даних. Останній рівень — рівень даних, де зберігається вся інформація в базі даних MySQL.

Нижче наведено діаграму, яка показує загальну архітектуру системи та взаємодію між клієнтом, сервером і базою даних (рис.1.6).

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

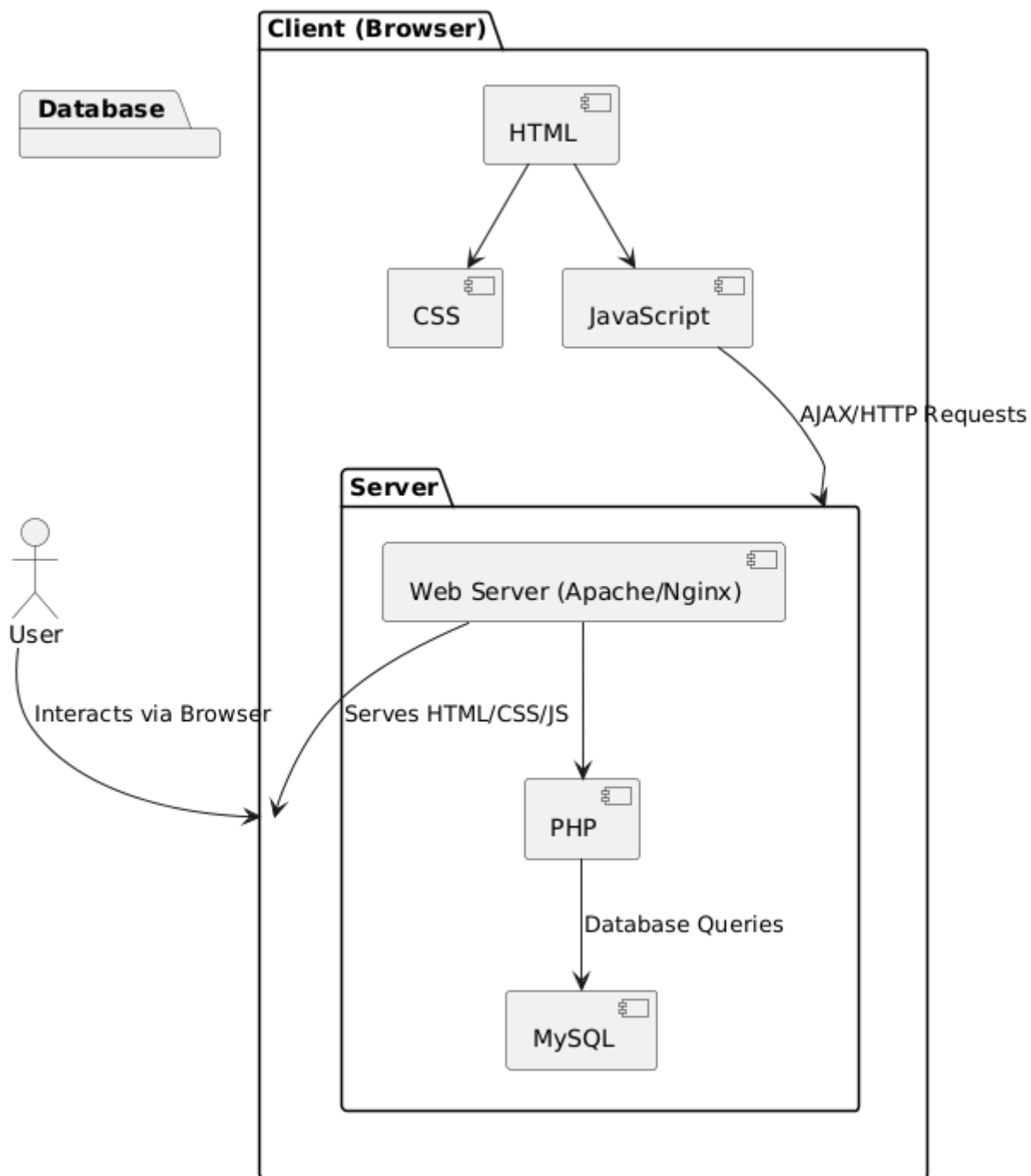


Рис.1.6 Архітектура системи управління проєктами

Для деталізації серверної логіки використовується модель MVC. Діаграма нижче показує, як компоненти MVC взаємодіють на сервері (рис.1.7).

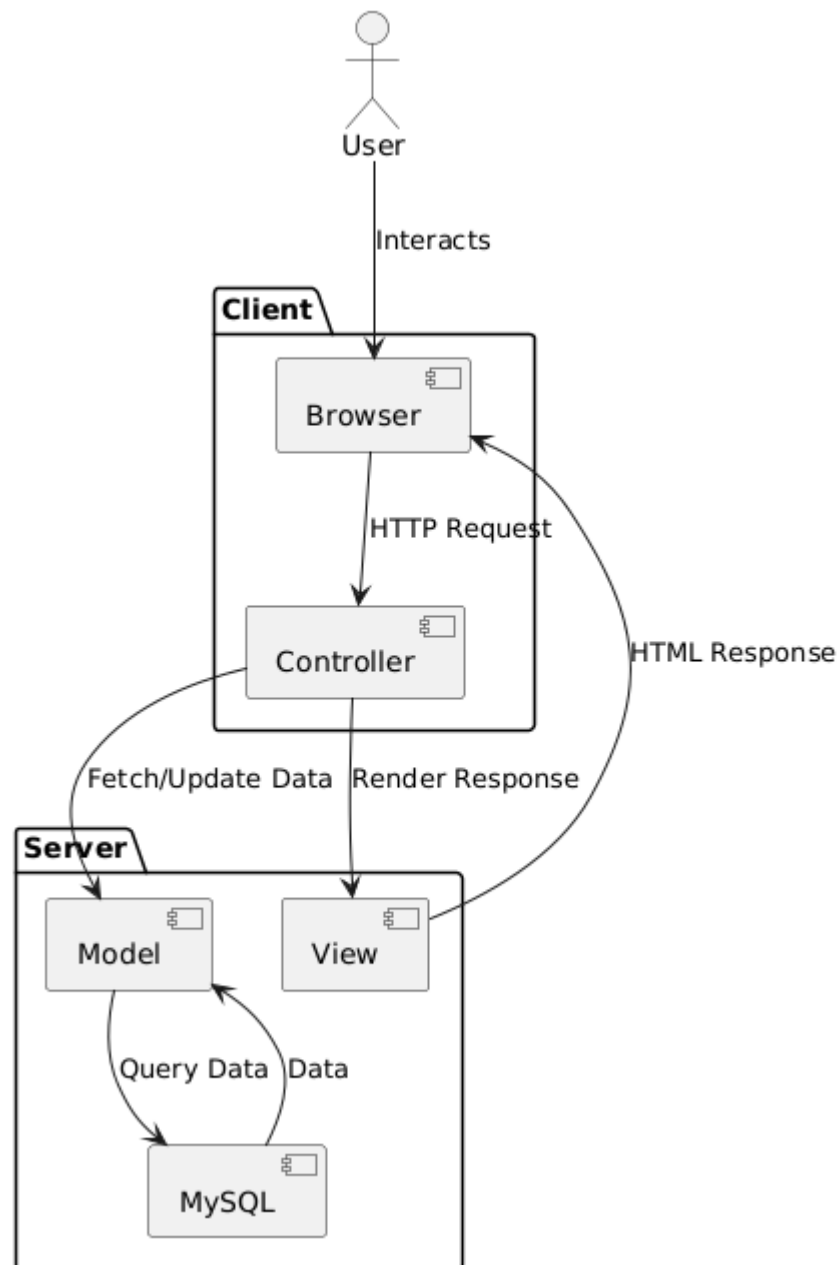


Рис.1.7 MVC для системи управління проектами

Переваги обраної архітектури включають простоту реалізації, оскільки використання базового стеку технологій, таких як HTML, CSS, JavaScript, PHP і MySQL, без застосування складних фреймворків значно скорочує час на освоєння і налаштування системи. Це дозволяє зосередитись на функціональності без зайвих ускладнень. Гнучкість цієї архітектури також є важливою перевагою, оскільки модель MVC дає змогу легко модифікувати окремі компоненти системи без впливу на інші частини, що робить процес оновлення і розширення функцій зручнішим. Масштабованість забезпечується

завдяки MySQL і PHP, які підтримують масштабування через оптимізацію запитів і використання кешування, наприклад, за допомогою Memcached. Крім того, кросплатформність системи дозволяє отримувати доступ до неї через будь-який веббраузер на різних пристроях, що робить її доступною з будь-якої точки світу.

Обрана клієнт-серверна архітектура з моделлю MVC і стеком HTML, CSS, JavaScript, PHP і MySQL відповідає вимогам до системи управління проєктами. Вона забезпечує простоту розробки, достатню продуктивність для 100 одночасних користувачів і можливість подальшого масштабування.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Стек технологій HTML, CSS, JavaScript, PHP і MySQL є оптимальним для створення динамічного, функціонального та економічно вигідного веборієнтованого програмного рішення для пошуку та бронювання спортивних заходів. Ці технології широко використовуються в розробці вебдодатків, мають велику спільноту, багатий набір бібліотек і інструментів, а також забезпечують гнучкість для реалізації всіх необхідних функцій, таких як пошук заходів, бронювання, особистий кабінет і адмін-панель. Visual Studio Code, як основна технічна платформа, є універсальним і потужним середовищем розробки, яке підтримує всі обрані технології, забезпечує зручність написання коду, налагодження та інтеграцію з додатковими інструментами.

Основні переваги стеку полягають у тому, що HTML забезпечує семантичну розмітку, яка дозволяє створювати чітку і логічну структуру сторінок, що, у свою чергу, полегшує SEO-оптимізацію і підвищує доступність. CSS дозволяє створювати адаптивний та естетично привабливий дизайн, що відповідає тематиці туризму, використовуючи природні кольори та інтуїтивно зрозумілу навігацію. Завдяки JavaScript, система отримує необхідну інтерактивність, дозволяючи створювати динамічні елементи, такі як фільтри,

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

модальні вікна та асинхронне оновлення кошика. PHP відповідає за серверну логіку, обробку запитів і інтеграцію з базою даних, що забезпечує динамічний контент, необхідний для коректної роботи системи. MySQL гарантує надійне зберігання та швидкий доступ до важливих даних про товари, користувачів і замовлення, що забезпечує стабільну роботу всього додатку.

VS Code є легким, швидким і кросплатформним редактором коду, який підтримує широкий спектр мов програмування, включаючи HTML, CSS, JavaScript, PHP і SQL. Завдяки великій кількості розширень, інтеграції з Git, підтримці автодоповнення, дебагінгу та терміналу, VS Code ідеально підходить для розробки вебдодатків. Його модульність дозволяє налаштувати середовище під потреби проєкту, а активна спільнота забезпечує регулярні оновлення та підтримку [20].

Детальний опис інструментарію:

1. HTML

HTML (HyperText Markup Language) є основою для створення структури вебсторінок. HTML5 пропонує семантичні теги (<header>, <nav>, <main>, <footer>, <article>), які покращують читабельність коду, доступність для пошукових систем і підтримку стандартів WCAG 2.1. Він також підтримує мультимедійні елементи (зображення, відео).

Переваги цієї технології полягають у її універсальності та сумісності з усіма сучасними браузерами, що забезпечує коректне відображення сайту для всіх користувачів. Завдяки простоті освоєння та використання, навіть новачки можуть швидко адаптуватися і почати працювати з цією технологією без значних труднощів. Крім того, підтримка SEO завдяки семантичним тегам дає змогу покращити видимість сайту в пошукових системах, що сприяє кращому ранжуванню і залученню трафіку.

2. CSS

CSS (Cascading Style Sheets) відповідає за стилізацію вебсторінок, включаючи кольори, шрифти, компоновання та анімації. CSS3 підтримує

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

адаптивний дизайн (медіа-запити), гнучкі макети (Flexbox, Grid) і сучасні ефекти (переходи, трансформації).

Інструменти та підходи, які використовуються в проєкті, включають фреймворк Bootstrap 5, який надає можливість швидко створювати адаптивну сітку, різні компоненти, такі як кнопки, модальні вікна, каруселі, а також прискорює процес розробки завдяки готовим елементам і стилям. Перевагою є висока гнучкість при створенні візуально привабливого дизайну, що дозволяє легко налаштовувати зовнішній вигляд елементів для відповідності специфічним вимогам. Крім того, використання Bootstrap забезпечує підтримку кросбраузерності, що дозволяє додатку коректно працювати на різних браузерах. Важливою перевагою є також прискорення розробки завдяки використанню SCSS і вже готових компонентів, що значно знижує час, необхідний для створення елементів інтерфейсу.

3. JavaScript

JavaScript є мовою програмування для створення інтерактивних елементів вебсайту. Вона дозволяє обробляти дії користувачів, виконувати асинхронні запити та динамічно оновлювати контент без перезавантаження сторінки.

Переваги цієї технології включають високу інтерактивність і швидкість відгуку інтерфейсу, що дозволяє користувачам отримувати швидкі й чіткі реакції на їхні дії. Велика кількість доступних бібліотек допомагає значно спростити процес розробки, оскільки можна використовувати готові рішення для різних завдань. Крім того, вона забезпечує сумісність із усіма сучасними браузерами, що гарантує коректну роботу додатка на будь-якому пристрої.

4. PHP

PHP є серверною мовою програмування, яка широко використовується для створення динамічних вебдодатків. PHP 8.2 пропонує покращену продуктивність, нові можливості (наприклад, перелічувані типи) і підвищену безпеку.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

Переваги цієї технології включають високу продуктивність, що дозволяє швидко обробляти запити та забезпечувати ефективну роботу системи навіть за великих навантажень. Зручна робота з базою даних забезпечує простоту взаємодії з даними, а також дозволяє ефективно зберігати і обробляти інформацію. Крім того, активна спільнота та детальна документація надають цінну підтримку для розробників, що значно спрощує процес розробки та вирішення проблем, які можуть виникнути під час роботи з цією технологією.

5. MySQL

MySQL – це реляційна система управління базами даних (СУБД) із відкритим кодом, яка забезпечує швидке та надійне зберігання даних. MySQL 8.0 підтримує JSON-дані, покращені індекси та транзакції.

Переваги цієї технології включають високу швидкість обробки запитів, що дозволяє ефективно працювати з середнім обсягом даних, забезпечуючи мінімальні затримки при виконанні операцій. Вона також відзначається надійністю та підтримкою транзакцій, що гарантує цілісність і коректність даних під час складних операцій. Окрім цього, проста інтеграція з PHP дозволяє безперешкодно підключати систему до вебдодатків, що робить її зручним вибором для багатьох розробників.

Visual Studio Code – це безкоштовний редактор коду від Microsoft, який підтримує розробку на десятках мов програмування, включаючи HTML, CSS, JavaScript, PHP і SQL. VS Code вирізняється легкістю, швидкодією, підтримкою розширень і вбудованим терміналом, що робить його ідеальним вибором для проєкту [20].

Переваги використання VS Code для проєкту полягають у його кросплатформності, оскільки він працює на Windows, macOS і Linux, що дає розробникам можливість працювати на різних операційних системах без обмежень. Крім того, VS Code має тисячі плагінів для підтримки всіх технологій стеку, таких як HTML, CSS, JavaScript, PHP, MySQL, що значно полегшує роботу з різними мовами програмування. Інструменти дебагінгу для JavaScript (через

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Chrome) і PHP допомагають ефективно знаходити і виправляти помилки в коді. Також, завдяки можливості кастомізації, можна налаштувати теми, шрифти та гарячі клавіші, що дозволяє підвищити продуктивність і зробити роботу з редактором більш комфортною.

Стек технологій HTML, CSS, JavaScript, PHP і MySQL у поєднанні з технічною платформою Visual Studio Code є оптимальним вибором для розробки веборієнтованого програмного рішення для пошуку та бронювання спортивних заходів. HTML і CSS забезпечують структуру й адаптивний дизайн, JavaScript додає інтерактивність, PHP відповідає за серверну логіку та інтеграцію, а MySQL гарантує надійне зберігання даних. VS Code, завдяки своїм розширенням створює зручне середовище для розробки. Цей підхід дозволяє створити сучасне, функціональне і масштабоване веборієнтоване програмне рішення, яке відповідатиме потребам цільової аудиторії, ринковим стандартам і вимогам проєкту.

Висновки до розділу 1

У першому розділі дипломної роботи проведено комплексний аналіз напрямків використання інформаційних технологій у системах управління проєктами, що дозволило сформулювати теоретичну та практичну основу для розробки веборієнтованої системи.

У підрозділі 1.1 сформульовано постановку задачі, визначено мету розробки – створення веборієнтованої системи управління проєктами, яка забезпечує автоматизацію планування, моніторингу та контролю завдань, ресурсів і термінів. Встановлено функціональні та нефункціональні вимоги.

Підрозділ 1.2 присвячено аналізу п'яти популярних систем управління проєктами (Trello, Asana, Jira, Microsoft Project, Monday.com). Виявлено їхні переваги, такі як інтуїтивний інтерфейс (Trello), гнучкість (Asana, Monday.com), підтримка Agile (Jira) та детальне планування (Microsoft Project), а також недоліки, зокрема високу вартість, складність освоєння чи обмежену функціональність для певних типів проєктів.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

У підрозділі 1.3 обґрунтовано вибір клієнт-серверної архітектури з моделлю MVC, яка забезпечує чіткий поділ логіки даних, інтерфейсу та бізнес-логіки. Використання стеку технологій HTML, CSS, JavaScript, PHP і MySQL дозволяє досягти простоти розробки, масштабованості та кросплатформності.

Підрозділ 1.4 містить обґрунтування вибору інструментальних засобів (PHP для серверної логіки, MySQL для бази даних, HTML/CSS/JavaScript для клієнтської частини) та вимоги до апаратного забезпечення, які забезпечують швидкодію та надійність системи. Відмова від фреймворків зменшує залежності, але вимагає ретельної обробки безпеки та оптимізації продуктивності.

Таким чином, проведений аналіз дозволив визначити основні вимоги, обрати оптимальну архітектуру та стек технологій для розробки веборієнтованої системи управління проєктами. Отримані результати створюють підґрунтя для подальшого проєктування та реалізації системи в наступних розділах дипломної роботи.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ

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

Для ефективного функціонування системи управління проєктами необхідно чітко визначити взаємодію користувачів із системою через ідентифікацію основних варіантів використання. Система призначена для авторизованих користувачів, які виконують функції управління проєктами, завданнями, коментарями та інтерфейсом. Основним актором є користувач, який після аутентифікації отримує доступ до всіх функціональних можливостей системи.

Основні варіанти використання включають:

Аутентифікація. Користувач може увійти в систему, ввівши email і пароль, або зареєструватися, вказавши ім'я, email і пароль. Цей процес є обов'язковим для доступу до інших функцій.

Управління проєктами. Користувач має можливість створювати нові проєкти, вказуючи назву, опис, дедлайн і статус ("Не розпочато", "В процесі", "Завершено", "Заморожено"); редагувати існуючі проєкти; видаляти проєкти, що автоматично видаляє пов'язані завдання; переглядати список проєктів; фільтрувати проєкти за статусом для зручного аналізу, наприклад, перегляду завершених проєктів.

Управління завданнями. Користувач може створювати завдання, призначаючи їх до проєкту, вказуючи назву, опис, виконавця та статус ("Нове", "В процесі", "Завершено", "Заморожено"); редагувати або видаляти завдання; переглядати список завдань; фільтрувати завдання за проєктом, виконавцем або статусом для швидкого пошуку потрібної інформації.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

Коментарі до завдань. Користувач може додавати коментарі до конкретного завдання для обговорення деталей або уточнень, а також переглядати всі коментарі, пов’язані із завданням.

Оновлення статусу завдання. Користувач може змінювати статус завдання, наприклад, переводити його з "Нове" у "В процесі" або "Завершено", що сприяє відстеженню прогресу.

Згортання/розгортання проєктів. Користувач може приховувати або показувати список проєктів, щоб зосередитися на завданнях, що підвищує зручність інтерфейсу.

Для наочного представлення взаємодії користувача з системою розроблено діаграму варіантів використання (Use Case Diagram) (рис.2.1).

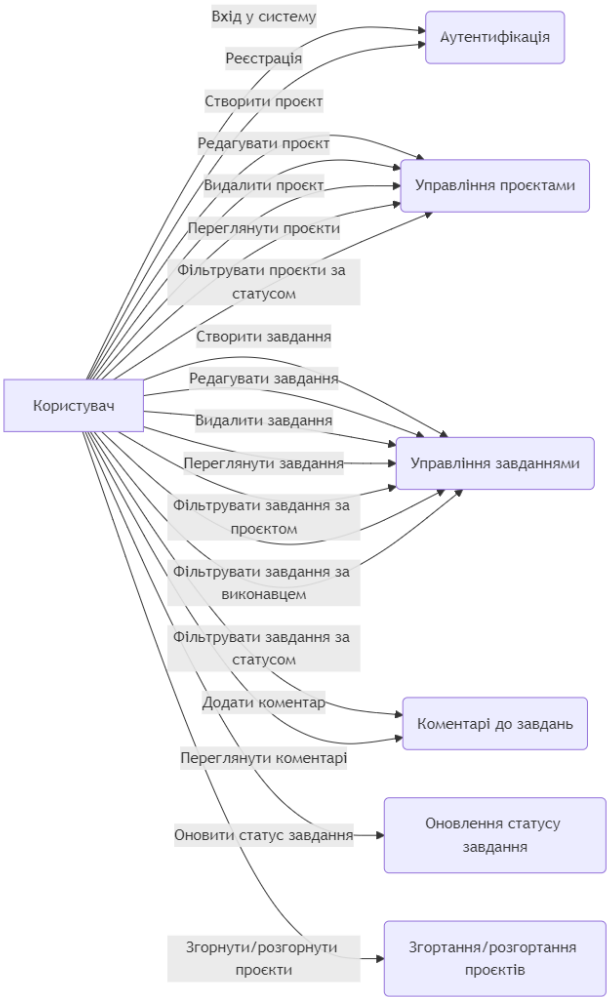


Рис.2.1 Use Case Diagram

Об'єктно-орієнтована структура системи управління проєктами базується на ключових сутностях, які відображають модель даних і їх взаємозв'язки. Ця структура забезпечує модульність, масштабованість і підтримку функціональних вимог системи. Основними класами системи є User, Project, Task і Comment, кожен із яких має свої атрибути та методи для реалізації функціоналу.

Клас User містить атрибути, такі як id, username, email і password, які дозволяють ідентифікувати користувача в системі. У цьому класі є методи для створення проєктів, завдань, додавання коментарів та оновлення статусу завдань. Призначення цього класу — представляти авторизованого користувача, який взаємодіє з системою для управління своїми завданнями та проєктами.

Клас Project зберігає інформацію про проєкти, включаючи атрибути, такі як id, name, description, deadline, status і user_id (ідентифікатор власника проєкту). Він включає методи для створення, редагування, видалення проєктів та фільтрації за статусом. Призначення цього класу — управляти даними про проєкти та забезпечувати їх обробку відповідно до різних статусів.

Клас Task містить атрибути, як-от id, project_id, title, description, assignee_id і status. Завдання можуть мати різні статуси: "Нове", "В процесі", "Завершено" або "Заморожено". Клас забезпечує методи для створення, редагування та видалення завдань, а також фільтрації за проєктом, виконавцем та статусом. Його основне призначення — зберігати інформацію про завдання та підтримувати їх управління.

Клас Comment зберігає інформацію про коментарі до завдань. Він має атрибути, такі як id, task_id, user_id, comment_text і created_at, що дозволяють відстежувати, до якого завдання належить коментар, хто є його автором, а також текст коментаря і дату його створення. Клас містить методи для додавання коментарів та отримання всіх коментарів до певного завдання. Його основне призначення — забезпечити комунікацію між користувачами шляхом коментування завдань, що дозволяє полегшити взаємодію та обговорення.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

Відношення між класами такі: кожен користувач може створювати багато проєктів, зв'язок між ними здійснюється через user_id. Один користувач може бути виконавцем багатьох завдань, зв'язок між ними реалізується через assignee_id. Користувач може писати багато коментарів, зв'язок між ними визначається через user_id. Проєкт може містити багато завдань, зв'язок між ними формується через project_id. Одне завдання може мати багато коментарів, зв'язок між ними забезпечується через task_id.

Для наочного представлення об'єктно-орієнтованої структури розроблено діаграму класів (Class Diagram) (рис.2.2).

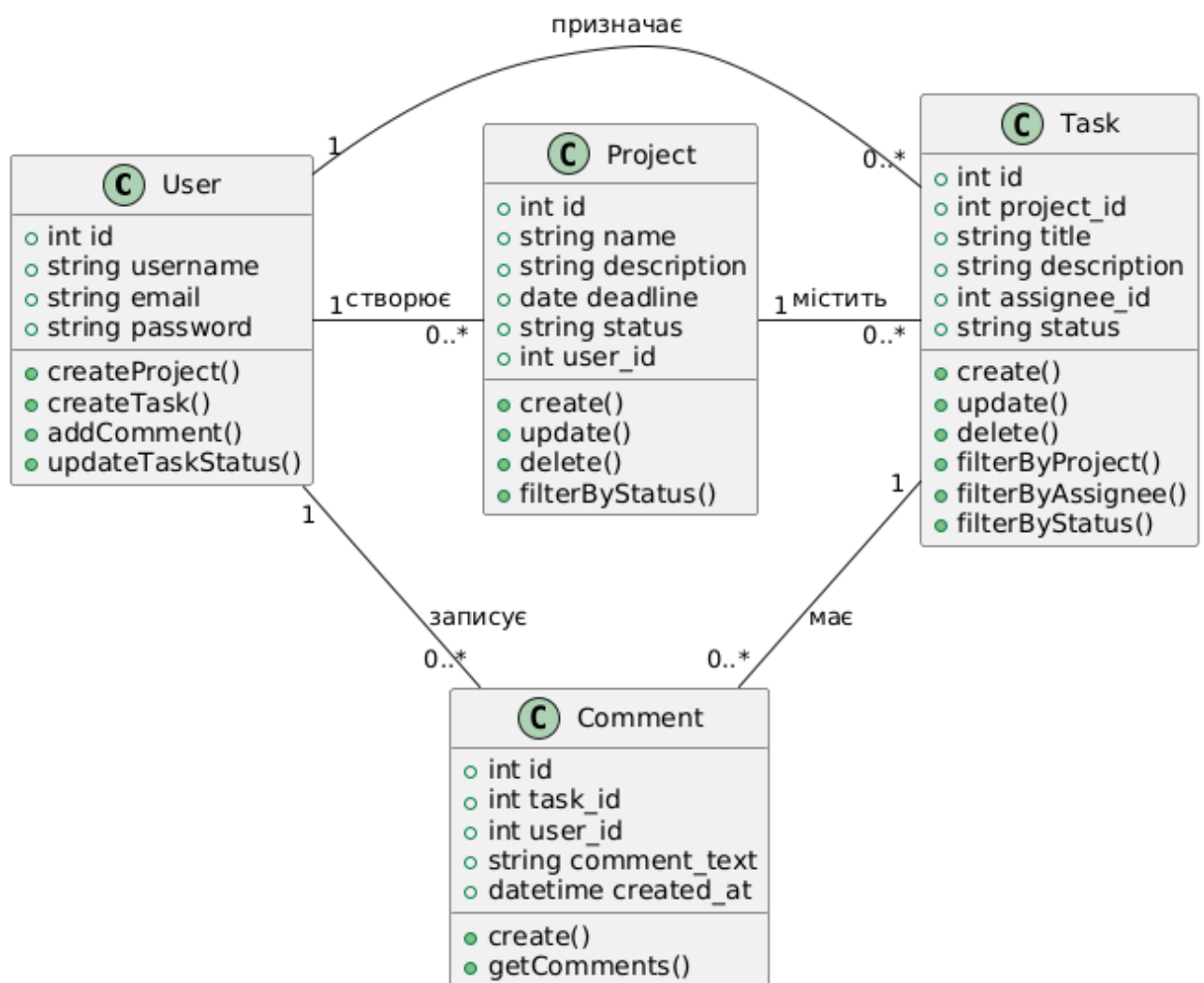


Рис.2.2 Class Diagram

2.2. Розробка бази даних системи управління проєктами

База даних системи управління проєктами призначена для зберігання інформації про користувачів, проєкти, завдання та коментарі, а також для забезпечення їхньої взаємодії. Вона реалізована як реляційна база даних, що використовує принципи нормалізації для уникнення надмірності даних і забезпечення цілісності. База даних підтримує основні функції системи, такі як аутентифікація користувачів, створення та управління проєктами і завданнями, фільтрація за статусом, додавання коментарів і оновлення статусів.

Основні сутності бази даних включають Users, яка зберігає дані авторизованих користувачів, які створюють проєкти, виконують завдання та пишуть коментарі. Projects містить інформацію про проєкти, включаючи назву, опис, дедлайн і статус. Tasks зберігає дані про завдання, пов'язані з проєктами, вказуючи виконавця та статус. Comments зберігає коментарі до завдань для підтримки комунікації.

База даних спроектована відповідно до третьої нормальної форми (3NF), що дозволяє усунути дублювання даних, завдяки використанню зовнішніх ключів, які забезпечують правильні зв'язки між таблицями. Це також гарантує цілісність даних через застосування обмежень, таких як PRIMARY KEY, FOREIGN KEY і NOT NULL, що забезпечують правильність і повноту інформації. Крім того, така структура бази даних сприяє ефективному виконанню запитів для фільтрації та пошуку, оскільки дані зберігаються у більш структурованому вигляді, що зменшує час на обробку великих обсягів інформації.

Для наочного представлення структури бази даних розроблено діаграму сутність-зв'язок (Entity-Relationship Diagram, ERD) (рис.2.3).

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

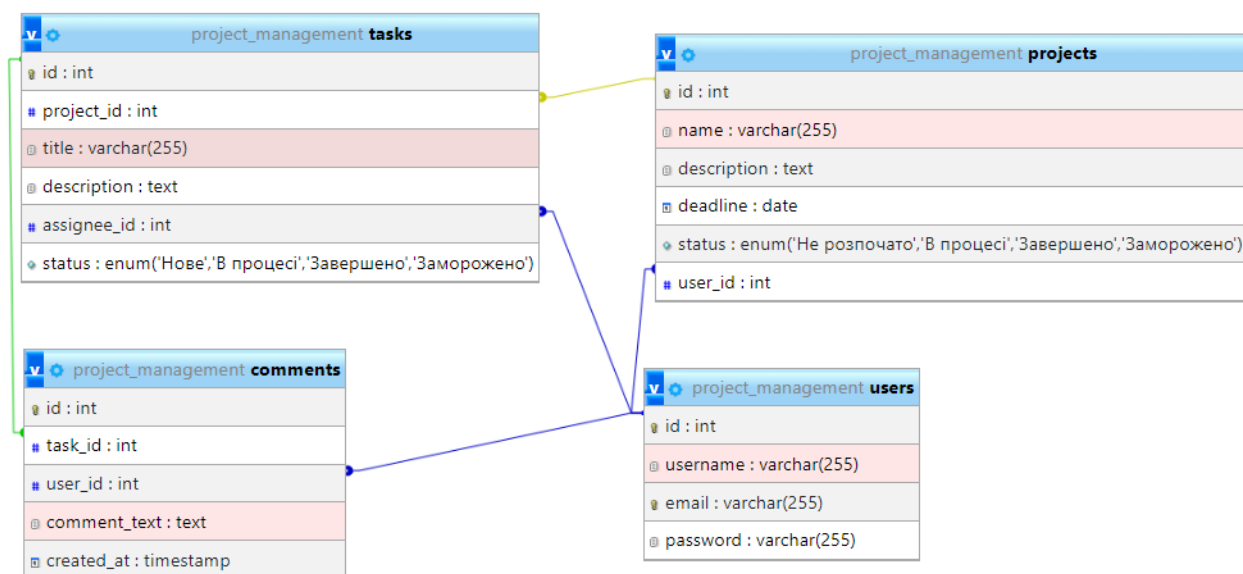


Рис.2.3 ERD діаграма

Схема бази даних включає чотири основні таблиці: users, projects, tasks і comments. Зв'язки між таблицями реалізовано через зовнішні ключі, що забезпечують референційну цілісність. Наприклад, поле user_id у таблиці projects посиляється на id у таблиці users, а поле project_id у таблиці tasks — на id у таблиці projects.

Таблиця Users зберігає дані авторизованих користувачів, які створюють проєкти, виконують завдання та додають коментарі

Таблиця 2.1

Таблиця Users

Поле	Тип даних	Обмеження	Опис
id	INT	PRIMARY KEY, AUTO_INCREMENT	Унікальний ідентифікатор користувача
username	VARCHAR(50)	NOT NULL, UNIQUE	Ім'я користувача для відображення
email	VARCHAR(100)	NOT NULL, UNIQUE	Електронна пошта для входу
password	VARCHAR(255)	NOT NULL	Хеш пароля для аутентифікації

Таблиця Projects зберігає інформацію про проєкти, включаючи їхні характеристики та зв'язок із власником (табл.2.2).

Таблиця 2.2

Таблиця Projects

Поле	Тип даних	Обмеження	Опис
id	INT	PRIMARY KEY, AUTO_INCREMENT	Унікальний ідентифікатор проєкту
name	VARCHAR(100)	NOT NULL	Назва проєкту
description	TEXT	NOT NULL	Опис проєкту
deadline	DATE	NOT NULL	Дедлайн проєкту
status	VARCHAR(20)	NOT NULL	Статус ("Не розпочато", "В процесі", "Завершено", "Заморожено")
user_id	INT	FOREIGN KEY (Users.id)	Ідентифікатор користувача-власника

Таблиця Tasks зберігає дані про завдання, пов'язані з проєктами та виконавцями (табл.2.3).

Таблиця 2.3

Таблиця Tasks

Поле	Тип даних	Обмеження	Опис
id	INT	PRIMARY KEY, AUTO_INCREMENT	Унікальний ідентифікатор завдання
project_id	INT	FOREIGN KEY (Projects.id)	Ідентифікатор проєкту
title	VARCHAR(100)	NOT NULL	Назва завдання
description	TEXT	NOT NULL	Опис завдання
assignee_id	INT	FOREIGN KEY (Users.id)	Ідентифікатор виконавця
status	VARCHAR(20)	NOT NULL	Статус ("Нове", "В процесі", "Завершено", "Заморожено")

Таблиця Comments зберігає коментарі до завдань для підтримки комунікації (табл.2.4).

Таблиця 2.4

Таблиця Comments

Поле	Тип даних	Обмеження	Опис
id	INT	PRIMARY KEY, AUTO_INCREMENT	Унікальний ідентифікатор коментаря
task_id	INT	FOREIGN KEY (Tasks.id)	Ідентифікатор завдання
user_id	INT	FOREIGN KEY (Users.id)	Ідентифікатор автора коментаря
comment_text	TEXT	NOT NULL	Текст коментаря
created_at	DATETIME	NOT NULL	Дата і час створення коментаря

База даних відповідає принципам нормалізації до третьої нормальної форми (3NF). Вона має першу нормальну форму (1NF), що означає, що всі поля є атомарними, без повторюваних груп або масивів. Наприклад, у таблицях Projects і Tasks поле "status" зберігає тільки одне значення. Друга нормальна форма (2NF) передбачає, що всі некритичні атрибути залежать від усього первинного ключа. Оскільки всі таблиці мають прості первинні ключі (id), це вимога виконується автоматично. Третя нормальна форма (3NF) означає, що в базі немає транзитивних залежностей. Наприклад, у таблиці Tasks поле assignee_id залежить від id таблиці Users, а не від інших полів, таких як project_id. Зовнішні ключі (user_id, project_id, task_id, assignee_id) забезпечують референційну цілісність, а обмеження UNIQUE на поля username і email у таблиці Users запобігають дублюванню даних.

2.3. Проєктування та реалізація ключових алгоритмів

У системі управління проєктами реалізовано ключові алгоритми для керування задачами та ресурсами. Ці алгоритми створені та працюють у межах проєкту, інтегровані в серверну (PHP, MySQL) та клієнтську (JavaScript, AJAX) логіку. Вони забезпечують безпечну обробку даних і асинхронне оновлення інтерфейсу.

Реалізовані алгоритми:

Керування задачами включає створення, редагування, видалення та фільтрацію завдань за проєктом, виконавцем або статусом.

Керування ресурсами полягає в призначенні виконавців на завдання через вибір із списку користувачів.

1. Керування задачами

Алгоритм керування задачами дозволяє створювати, редагувати, видаляти та фільтрувати завдання. Створення відбувається через форму, фільтрація — через асинхронні запити з динамічним SQL-запитом.

Алгоритм керування задачами є основним у системі та забезпечує:

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

Користувач створює завдання, заповнюючи форму з полями `project_id`, `title`, `description`, `assignee_id` та `status`. Дані перевіряються на валідність і зберігаються в таблиці `tasks`. Користувач редагує атрибути завдання, а сервер оновлює відповідний запис у базі даних. Користувач видаляє завдання, а сервер видаляє запис із бази. Користувач обирає критерії фільтрації, такі як проєкт, виконавець і статус, а система повертає відфільтрований список завдань через асинхронний запит.

Фільтрація є ключовою функцією, оскільки дозволяє швидко знаходити завдання, наприклад, відобразити лише завдання зі статусом "В процесі" у певному проєкті. Реалізація використовує підготовлені запити PDO для безпеки та AJAX для динамічного оновлення інтерфейсу без перезавантаження сторінки.

```
if ($action === 'create') {
    $project_id = filter_input(INPUT_POST, 'project_id',
    FILTER_SANITIZE_NUMBER_INT);
    $title = filter_input(INPUT_POST, 'title', FILTER_SANITIZE_STRING);
    $description = filter_input(INPUT_POST, 'description',
    FILTER_SANITIZE_STRING);
    $assignee_id = filter_input(INPUT_POST, 'assignee_id',
    FILTER_SANITIZE_NUMBER_INT);
    $status = filter_input(INPUT_POST, 'status', FILTER_SANITIZE_STRING);

    $stmt = $pdo->prepare("INSERT INTO tasks (project_id, title,
    description, assignee_id, status) VALUES (?, ?, ?, ?, ?)");
    if ($stmt->execute([$project_id, $title, $description, $assignee_id,
    $status])) {
        echo json_encode(['success' => true]);
    } else {
        echo json_encode(['success' => false, 'message' => 'Помилка
    створення завдання']);
    }
}
```

Код створює нову задачу, отримує дані з форми, фільтрує їх і зберігає в базі даних. Якщо успішно, повертає повідомлення про успіх, в іншому випадку — про помилку.

2. Керування ресурсами

Алгоритм керування ресурсами дозволяє призначати виконавців на завдання, використовуючи список зареєстрованих користувачів. Валідність призначення забезпечується обмеженнями бази даних.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

Алгоритм керування ресурсами забезпечує призначення виконавців на завдання. Реалізовані функції:

Система витягує всіх зареєстрованих користувачів із таблиці users для заповнення випадającego списку у формі створення або редагування завдання. Користувач обирає виконавця, а його id зберігається в полі assignee_id таблиці tasks. Обмеження FOREIGN KEY у базі даних гарантує, що assignee_id відповідає існуючому користувачу.

Алгоритм є базовим, але ефективним, забезпечуючи коректне призначення людських ресурсів.

```
<!-- Форма створення завдання -->
<div id="task-modal" class="modal">
  <div class="modal-content">
    <span class="close-btn" onclick="closeModal('task-
modal')">x</span>
    <h2>Створити завдання</h2>
    <form id="task-form" action="tasks.php" method="POST">
      <input type="hidden" name="action" value="create">
      <label for="task-project-id">Проект:</label>
      <select id="task-project-id" name="project_id" required>
        <?php foreach ($projects as $project): ?>
          <option value="<?php echo $project['id']; ?>"><?php
echo htmlspecialchars($project['name']); ?></option>
        <?php endforeach; ?>
      </select>
      <label for="task-title">Назва:</label>
      <input type="text" id="task-title" name="title" required>
      <label for="task-description">Опис:</label>
      <textarea id="task-description" name="description"
required></textarea>
      <label for="task-assignee">Виконавець:</label>
      <select id="task-assignee" name="assignee_id" required>
        <?php foreach ($users as $user): ?>
          <option value="<?php echo $user['id']; ?>"><?php echo
htmlspecialchars($user['username']); ?></option>
        <?php endforeach; ?>
      </select>
      <label for="task-status">Статус:</label>
      <select id="task-status" name="status" required>
        <option value="Нове">Нове</option>
        <option value="В процесі">В процесі</option>
        <option value="Завершено">Завершено</option>
        <option value="Заморожено">Заморожено</option>
      </select>
      <button type="submit">Створити завдання</button>
    </form>
  </div>
</div>
```

2.4. Реалізація вебдодатку системи управління проектами

Вебдодаток системи управління проектами розроблено для автоматизації процесів керування проектами, завданнями та ресурсами. Додаток дозволяє користувачам створювати проекти, додавати завдання, призначати виконавців, фільтрувати завдання за різними критеріями та відстежувати їхній статус. Система підтримує авторизацію користувачів для забезпечення безпеки даних і персоналізованого доступу.

Вебдодаток побудовано за клієнт-серверною архітектурою. Клієнтська частина відповідає за інтерфейс користувача, відображення даних і взаємодію через асинхронні запити (AJAX), і реалізована з використанням HTML, CSS і JavaScript. Серверна частина обробляє запити, взаємодіє з базою даних і забезпечує бізнес-логіку, реалізована на PHP із підключенням до реляційної бази даних MySQL. База даних зберігає інформацію про користувачів, проекти, завдання та коментарі.

Додаток розгортається на вебсервері і доступний через браузер, що забезпечує кросплатформність і легкість використання.

Для реалізації вебдодатку використано такі технології. PHP 7.4+ є серверною мовою програмування, яка відповідає за обробку запитів, взаємодію з базою даних і генерацію динамічного контенту. MySQL використовується як реляційна база даних для зберігання інформації про користувачів, проекти, завдання та коментарі. JavaScript (ES6) реалізує клієнтську логіку, асинхронні запити через fetch і динамічне оновлення інтерфейсу. HTML5 визначає структуру вебсторінок, зокрема форми для створення проектів і завдань. CSS3 відповідає за стилізацію інтерфейсу, адаптивний дизайн і візуальні ефекти, такі як виділення прострочених проектів. AJAX використовується для асинхронної взаємодії між клієнтом і сервером, що дозволяє оновлювати дані без перезавантаження сторінки. PDO (PHP Data Objects) є інтерфейсом для безпечного доступу до бази даних із захистом від SQL-ін'єкцій. Додатково використовуються бібліотеки, такі як Bootstrap 5 для швидкого створення

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

адаптивного інтерфейсу (опціонально, залежно від реалізації) і jQuery для спрощення роботи з DOM і подіями.

Проект має модульну структуру, що полегшує підтримку та розширення. Основні файли виконують наступні функції:

- Файл `index.php` відповідає за сторінку авторизації та реєстрації користувачів.
- Файл `dashboard.php` є головною панеллю користувача, де відображаються списки проєктів і завдань, а також форми для їх створення та фільтрації.
- Файл `projects.php` обробляє операції з проєктами, зокрема створення, редагування та видалення.
- Файл `tasks.php` відповідає за операції із завданнями, включаючи їх створення, фільтрацію та видалення.
- Файл `script.js` містить клієнтську логіку, включаючи асинхронні запити та обробку подій.
- Файл `style.css` визначає стилі інтерфейсу, зокрема адаптивний дизайн.
- Файл `config.php` містить налаштування підключення до бази даних, зокрема параметри для PDO.

Схема взаємодії компонентів:

- Користувач взаємодіє з інтерфейсом (`dashboard.php`, HTML/CSS).
- JavaScript (`script.js`) надсилає асинхронні запити до серверних скриптів (`projects.php`, `tasks.php`).
- Сервер обробляє запити, використовуючи PDO для взаємодії з MySQL, і повертає JSON-відповідь.
- Клієнт оновлює інтерфейс на основі отриманої відповіді.

Реалізація ключових компонентів:

1. Аутентифікація користувачів

Аутентифікація забезпечує безпечний доступ до системи. Користувач вводить email і password на сторінці `index.php`, після чого сервер перевіряє дані в

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

таблиці users. У разі успіху створюється сесія (\$_SESSION['user_id']), і користувач перенаправляється на dashboard.php.

```
<?php
session_start();
require_once 'config.php';

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['action']) &&
$_POST['action'] === 'login') {
    $email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_EMAIL);
    $password = $_POST['password'];

    $stmt = $pdo->prepare("SELECT id, password FROM users WHERE email =
?");
    $stmt->execute([$email]);
    $user = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($user && password_verify($password, $user['password'])) {
        $_SESSION['user_id'] = $user['id'];
        header('Location: dashboard.php');
        exit;
    } else {
        $error = "Невірний email або пароль";
    }
}
?>
<form method="POST">
    <input type="hidden" name="action" value="login">
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>
    <label for="password">Пароль:</label>
    <input type="password" id="password" name="password" required>
    <button type="submit">Увійти</button>
    <?php if (isset($error)) echo "<p>$error</p>"; ?>
</form>
```

Форми обробляються за допомогою функції filter_input для забезпечення безпеки. Пароль перевіряється через password_verify, оскільки в базі даних зберігається його хеш, наприклад, створений за допомогою password_hash. Сесія зберігає user_id, що дозволяє здійснювати авторизацію на інших сторінках.

2. Керування задачами

Компонент керування задачами дозволяє створювати, редагувати, видаляти та фільтрувати завдання. Основна логіка реалізована в tasks.php (сервер) і script.js (клієнт). Користувач створює завдання через форму на dashboard.php, а фільтрація виконується асинхронно через AJAX.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

У tasks.php обробляються запити на створення (action=create) і фільтрацію (action=get) завдань. Динамічний SQL-запит із підготовленими параметрами забезпечує гнучку фільтрацію.

У script.js функція filterTasks формує URL із параметрами фільтрації, надсилає запит через fetch і оновлює список завдань у DOM.

3. Керування ресурсами

Компонент керування ресурсами дозволяє призначати виконавців на завдання. Список користувачів витягується з таблиці users і відображається у формі створення завдання на dashboard.php.

4. Стилiзація інтерфейсу

Інтерфейс стилізовано за допомогою CSS для забезпечення зручності та естетичного вигляду. Стилi включають оформлення списків, форм і модальних вікон.

```
body {
  font-family: Arial, sans-serif;
  margin: 0;
  padding: 20px;
  background-color: #f4f4f4;
}

.project-item, .task-item {
  background: white;
  padding: 15px;
  margin-bottom: 10px;
  border-radius: 5px;
  box-shadow: 0 2px 5px rgba(0,0,0,0.1);
}

.modal {
  display: none;
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0,0,0,0.5);
}

.modal-content {
  background: white;
  margin: 15% auto;
  padding: 20px;
  width: 80%;
  max-width: 500px;
}
```

```

        border-radius: 5px;
    }

    button {
        padding: 10px 20px;
        background-color: #007bff;
        color: white;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }

    button:hover {
        background-color: #0056b3;
    }

```

Стилі для `.project-item` і `.task-item` формують картковий вигляд елементів списків. Класи `.modal` і `.modal-content` відповідають за оформлення модальних вікон для форм. Стилі кнопок додають інтерактивні ефекти, зокрема ефект наведення.

Висновки до розділу 2

У розділі було здійснено комплексне проектування веборієнтованої системи управління проєктами, що охопило визначення функціональних вимог, розробку структури бази даних, створення ключових алгоритмів і технічну реалізацію вебдодатку.

На етапі визначення варіантів використання було сформовано набір сценаріїв взаємодії користувачів із системою, включаючи авторизацію, створення та керування проєктами й завданнями, а також призначення виконавців. Об'єктно-орієнтована структура системи дозволила чітко визначити основні сутності (користувачі, проєкти, завдання, коментарі) та їхні взаємозв'язки, що стало основою для подальшого проектування.

Розробка бази даних забезпечила створення реляційної моделі з таблицями `users`, `projects`, `tasks` і `comments`, пов'язаними через зовнішні ключі. Використання MySQL і нормалізація даних гарантують ефективне зберігання, швидкий доступ і цілісність інформації.

Також було спроектовано та реалізовано ключові алгоритми системи, зокрема керування задачами (створення, редагування, видалення, фільтрація) і

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

ресурсами (призначення виконавців). Алгоритми розроблено з урахуванням безпеки (захист від SQL-ін'єкцій через PDO) і продуктивності (асинхронні запити AJAX), що підтверджено їхньою працездатністю в робочому прототипі.

Реалізація вебдодатку охопила створення клієнтської (HTML, CSS, JavaScript) та серверної (PHP, MySQL) частин. Додаток підтримує авторизацію, керування проєктами й завданнями, а також зручний інтерфейс із модальними формами та картковим відображенням даних.

Таким чином, у розділі закладено фундамент для функціонування системи управління проєктами, що відповідає поставленим вимогам щодо зручності, безпеки та ефективності.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ СИСТЕМОЮ УПРАВЛІННЯ ПРОЄКТАМИ

3.1. Установка та налаштування параметрів системи

Установка веборієнтованої системи управління проєктами передбачає розгортання серверного середовища, розміщення файлів проєкту, ініціалізацію бази даних і налаштування конфігураційних параметрів. Система розроблена для роботи на вебсервері з підтримкою PHP і MySQL, що забезпечує кросплатформність і легкість розгортання. Процес установки включає підготовку серверного стеку, копіювання файлів проєкту, створення бази даних і налаштування підключення до неї.

Система протестована на локальному сервері (Apache 2.4, PHP 7.4, MySQL 8.0) і може бути розгорнута як локально, так і на віддаленому хостингу.

Для роботи системи потрібен наступний програмний стек. вебсервер Apache 2.4 обробляє HTTP-запити. PHP версії 7.4 або новіша з модулями pdo_mysql і mbstring забезпечує роботу з базою даних і обробку рядків. MySQL версії 5.7 або 8.0 використовується для зберігання даних. Для доступу до інтерфейсу потрібен сучасний веббраузер, наприклад, Google Chrome або Mozilla Firefox. Опціонально для локального тестування можна використовувати XAMPP, який включає Apache, PHP та MySQL.

Для адміністрування бази даних використовується phpMyAdmin. Для редагування коду застосовується текстовий редактор Visual Studio Code.

Для початку потрібно завантажити XAMPP з офіційного сайту (<https://www.apachefriends.org>) і встановити його, вибираючи компоненти Apache, PHP і MySQL. Після цього потрібно відкрити XAMPP Control Panel і запустити модулі Apache та MySQL.

Далі потрібно скопіювати файли проєкту (index.php, dashboard.php, projects.php, tasks.php, script.js, style.css, config.php) в директорію вашого вебсервера. Якщо ви використовуєте XAMPP, розмістіть їх у папці

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

C:\xampp\htdocs\project_management (для Windows) або
/opt/lampp/htdocs/project_management (для Linux).

Тепер потрібно відкрити браузер і ввести адресу http://localhost/project_management, щоб відкрити проєкт локально.

Щоб налаштувати базу даних, потрібно зайти в phpMyAdmin (зазвичай доступний за адресою <http://localhost/phpmyadmin>), потрібно створити нову базу даних (деталі є в Додатку А). Не забудьте, що таблиці users, projects, tasks і comments мають бути пов'язані зовнішніми ключами для забезпечення цілісності.

Останній крок — потрібно налаштувати файл config.php, де вказано підключення до бази даних через PDO. Це потрібно зробити перед тим, як запускати систему.

```
<?php
$host = 'localhost';
$dbname = 'project_management';
$username = 'root';
$password = '';

try {
    $pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8",
$username, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
    die("Connection failed: " . $e->getMessage());
}
?>
```

Цей пункт детально описує процес установки та налаштування системи.

3.2. Опис інтерфейсу користувача

Інтерфейс користувача вебдодатку системи управління проєктами розроблено з урахуванням принципів зручності, інтуїтивності та мінімалізму. Основною метою інтерфейсу є забезпечення швидкого доступу до функціоналу керування проєктами, завданнями та ресурсами, а також надання чіткої візуальної інформації про статус проєктів і завдань. Цільова аудиторія — менеджери проєктів, члени команд і адміністратори, які потребують ефективного інструменту для організації роботи.

Інтерфейс побудовано на основі клієнт-серверної архітектури, де клієнтська частина (HTML, CSS, JavaScript) відповідає за відображення даних і взаємодію з користувачем, а серверна частина (PHP) забезпечує обробку запитів і доступ до бази даних. Дизайн інтерфейсу є адаптивним, що дозволяє коректно відображатися на різних пристроях (десктоп, планшет, смартфон). Використовуються сучасні вебтехнології (HTML5, CSS3, JavaScript ES6) для створення динамічного та інтерактивного досвіду.

Інтерфейс користувача складається з кількох основних сторінок, кожна з яких виконує певну функцію:

`index.php` — це сторінка для авторизації та реєстрації. Вона містить форму, де можна ввести email і пароль або створити новий обліковий запис.

`dashboard.php` є головною панеллю користувача, на якій відображаються списки проєктів і завдань, а також є форми для їх створення. Це центральна сторінка, де користувач взаємодіє з системою.

`tasks.php` служить для перегляду та фільтрації завдань. Тут можна вибирати різні критерії, такі як проєкт, виконавець і статус, щоб швидше знайти потрібну інформацію.

`projects.php` дає змогу керувати проєктами: створювати нові, редагувати наявні або видаляти їх.

Кожна сторінка має єдину навігаційну панель із посиланнями на основні розділи (наприклад, "Проєкти", "Завдання", "Вихід"). Інтерфейс використовує картковий дизайн для відображення проєктів і завдань, що полегшує сприйняття інформації.

Опис ключових елементів інтерфейсу:

1. Реєстрація користувача

Користувач створює новий обліковий запис, заповнюючи форму на сторінці `index.php` з полями `username`, `email` і `password` (рис.3.1). Дані зберігаються в таблиці `users` із захешованим паролем.

Рис.3.1 Форма реєстрації користувача

2. Форма авторизації на сторінці index.php дозволяє користувачам увійти в систему. Вона включає поля для введення email і password, а також кнопку "Увійти". У разі помилки відображається повідомлення (наприклад, "Невірний email або пароль").

```
<form method="POST">
  <input type="hidden" name="action" value="login">
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required>
  <label for="password">Пароль:</label>
  <input type="password" id="password" name="password" required>
  <button type="submit">Увійти</button>
  <?php if (isset($error)) echo "<p>$error</p>"; ?>
</form>
```

Форма має мінімалістичний дизайн із чіткими мітками. Поле email вимагає валідного формату, а password приховує введені символи. Повідомлення про помилку відображається червоним кольором, щоб привернути увагу.

The image shows a user login interface. At the top, there are two buttons: 'Вхід' (Login) in green and 'Реєстрація' (Registration) in grey. Below them is the heading 'Вхід'. There are two input fields: 'Email:' with the text 'user@gmail.com' and 'Пароль:' (Password) with masked characters. A green 'Увійти' (Login) button is at the bottom.

Рис.3.2 Авторизація користувача

У разі успіху створюється сесія, і користувач перенаправляється на dashboard.php (рис.3.3).

The image shows a system dashboard titled 'Панель управління' (Management Panel). It includes buttons for 'Створити проект' (Create project) and 'Створити завдання' (Create task). A 'Вийти' (Logout) button is in the top right. The 'Проекти' (Projects) section has a filter dropdown set to 'Усі статуси' (All statuses) and a 'Згорнути' (Collapse) button. Two project cards are shown: 'Розробка сайту' (Website development) with a description, deadline (2025-06-30), status (In process), and buttons for 'Переглянути завдання' (View task) and 'Видалити' (Delete); and 'Мобільний додаток' (Mobile app) with a description and deadline (2025-08-15).

Рис.3.3 Дашбоард системи

3. Створення проєкту

Авторизований користувач створює проєкт через модальну форму на dashboard.php, вказуючи name, description і deadline. Дані зберігаються в таблиці projects (рис.3.4).

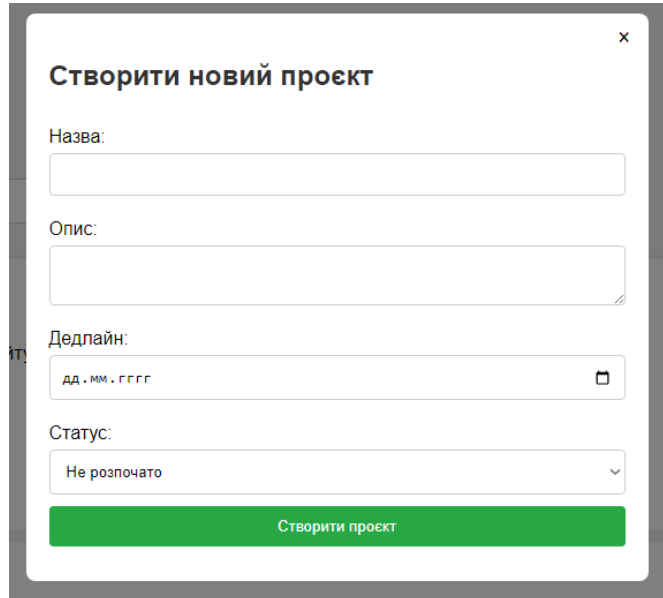
A modal window titled "Створити новий проєкт" (Create new project) with a close button (X) in the top right corner. It contains four input fields: "Назва:" (Name), "Опис:" (Description), "Дедлайн:" (Deadline) with a calendar icon, and "Статус:" (Status) with a dropdown menu showing "Не розпочато" (Not started). A green button labeled "Створити проєкт" (Create project) is at the bottom.

Рис.3.4 Модальне вікно створення проєкту

4. Створення завдання

Користувач створює завдання через модальну форму на dashboard.php, вказуючи title, description, assignee_id і status. Дані зберігаються в таблиці tasks (рис.3.5).

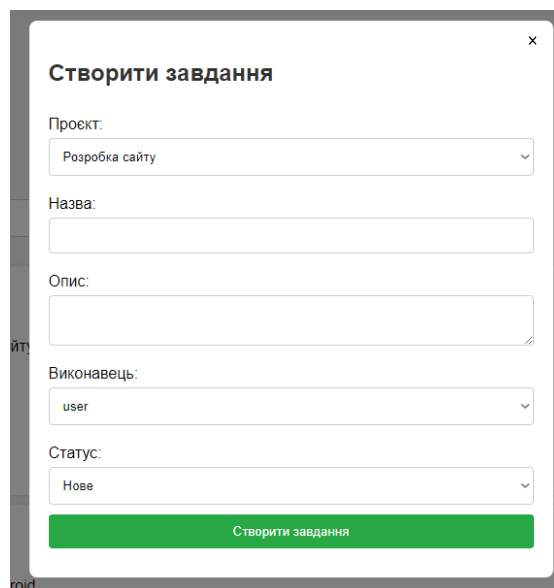
A modal window titled "Створити завдання" (Create task) with a close button (X) in the top right corner. It contains four input fields: "Проект:" (Project) with a dropdown menu showing "Розробка сайту" (Website development), "Назва:" (Name), "Опис:" (Description), and "Виконавець:" (Assignee) with a dropdown menu showing "user". The "Статус:" (Status) dropdown menu shows "Нове" (New). A green button labeled "Створити завдання" (Create task) is at the bottom.

Рис.3.5 Модальне вікно створення завдання

5. Призначення виконавців (керування ресурсами)

Користувач призначає виконавця на завдання через випадаючий список у формі створення/редагування завдання (рис.3.6). Поле assignee_id оновлюється в таблиці tasks.

The image shows a web form with a label "Виконавець:" (Assignee:). Below it is a dropdown menu. The currently selected option is "user". The dropdown is open, showing a list of options: "user" (highlighted in blue), "user1", "user2", and "Нове" (New). The dropdown has a small arrow icon on the right side.

Рис.3.6 Випадаючий список із користувачами для створення відповідального за завдання

6. Фільтрація завдань

Користувач фільтрує завдання на сторінці tasks.php за проектом, виконавцем або статусом, використовуючи форму з випадаючими списками (рис.3.7). Фільтрація виконується асинхронно через AJAX.

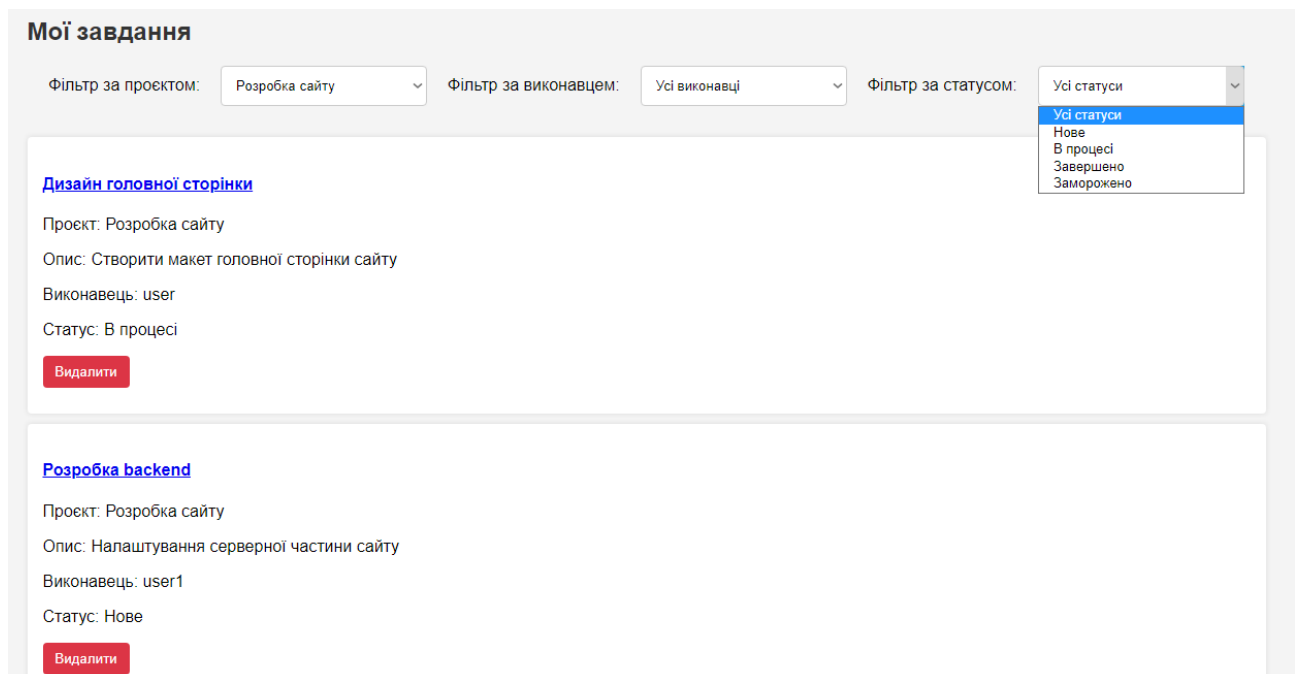
The image shows a web page titled "Мії завдання" (My tasks). At the top, there are three filter dropdowns: "Фільтр за проектом:" (Filter by project:) with "Розробка сайту" (Website development) selected, "Фільтр за виконавцем:" (Filter by assignee:) with "Усі виконавці" (All assignees) selected, and "Фільтр за статусом:" (Filter by status:) with "Усі статуси" (All statuses) selected. Below the filters is a list of tasks. The first task is "Дизайн головної сторінки" (Design of the main page) with project "Розробка сайту", description "Створити макет головної сторінки сайту", assignee "user", and status "В процесі" (In progress). It has a "Видалити" (Delete) button. The second task is "Розробка backend" (Backend development) with project "Розробка сайту", description "Налаштування серверної частини сайту", assignee "user1", and status "Нове" (New). It also has a "Видалити" (Delete) button. The status dropdown menu is open, showing options: "Усі статуси", "Нове", "В процесі", "Завершено", and "Заморожено".

Рис.3.7 Фільтрація завдань

Сторінка tasks.php містить форму фільтрації, яка дозволяє користувачам вибирати завдання за проектом, виконавцем або статусом. Фільтрація

виконується асинхронно через AJAX, що забезпечує швидке оновлення списку без перезавантаження сторінки.

```
window.filterTasks = async function() {
  const projectId = document.getElementById('project-filter').value;
  const assigneeId = document.getElementById('assignee-filter').value;
  const status = document.getElementById('status-filter').value;
  let url = 'tasks.php?action=get';
  if (projectId !== 'all') {
    url += `&project_id=${projectId}`;
  }
  if (assigneeId !== 'all') {
    url += `&assignee_id=${assigneeId}`;
  }
  if (status !== 'all') {
    url += `&status=${encodeURIComponent(status)}`;
  }

  const response = await fetch(url);
  const result = await response.json();
  const taskList = document.getElementById('task-list');
  taskList.innerHTML = '';
  if (result.success) {
    result.tasks.forEach(task => {
      const taskItem = document.createElement('div');
      taskItem.className = 'task-item';
      taskItem.dataset.taskId = task.id;
      taskItem.innerHTML = `
        <h4><a
href="task_details.php?task_id=${task.id}">${task.title}</a></h4>
        <p>Проект: ${task.project_name}</p>
        <p>Опис: ${task.description}</p>
        <p>Виконавець:      ${task.assignee_name      ||      'Не
призначено'}</p>
        <p>Статус: ${task.status}</p>
        <form      class="delete-task-form"      method="POST"
style="display:inline;"
          <input type="hidden" name="action" value="delete">
          <input type="hidden" name="id" value="${task.id}">
          <button      type="submit"      class="delete-
btn">Видалити</button>
        </form>
      `;
      taskList.appendChild(taskItem);
    });
  }
};
```

Форма фільтрації містить випадаючі списки для вибору критеріїв, таких як project-filter, assignee-filter і status-filter. Функція filterTasks надсилає асинхронний запит до tasks.php і динамічно оновлює список завдань. Кнопка "Видалити" у кожній картці завдання дає змогу швидко видаляти завдання.

7. Додавання коментарів до завдання

Користувач додає коментар до завдання через форму на сторінці деталей завдання (task_details.php) (рис.3.8). Коментар зберігається в таблиці comments.

Оновити статус

Коментарі

user: Потрібно внести зміни у макет
2025-05-23 14:02:07

user: Макет готовий на 50%, потрібен фідбек від клієнта
2025-05-23 13:57:50

user1: Чи враховано адаптивний дизайн?
2025-05-23 13:57:50

Додати коментар

Коментар:

Так **адаптивний** дизайн враховано

Додати коментар

Рис.3.8 Внесення коментаря в задачу

3.3. Тестування та перевірка функціональності системи

Тестування системи управління проєктами проводилося з метою перевірки працездатності всіх функціональностей, їхньої відповідності вимогам і стабільності роботи в реальних умовах. Використовувалася методологія функціонального тестування, яка охоплювала ручне тестування ключових сценаріїв використання. Тестування виконувалося на локальному сервері (XAMPP, Apache 2.4, PHP 7.4, MySQL 8.0).

Тестові сценарії розроблено для кожної функціональності:

1. Авторизація з невірним логіном або паролем

При введенні некоректних даних виводиться повідомлення про помилку введення даних (рис.3.9).

The screenshot shows a login interface with two tabs at the top: 'Вхід' (Login) in green and 'Реєстрація' (Registration) in grey. The 'Вхід' tab is active. Below the tabs, the title 'Вхід' is displayed. There are two input fields: 'Email:' containing 'myemail@test.com' and 'Пароль:' (Password) containing '.....'. A green button labeled 'Увійти' (Login) is positioned below the password field. At the bottom, a red error message 'Невірний email або пароль' (Incorrect email or password) is shown with a red arrow pointing to it.

Рис.3.9 Повідомлення про помилку при авторизації

2. Реєстрація без введення валідної пошти

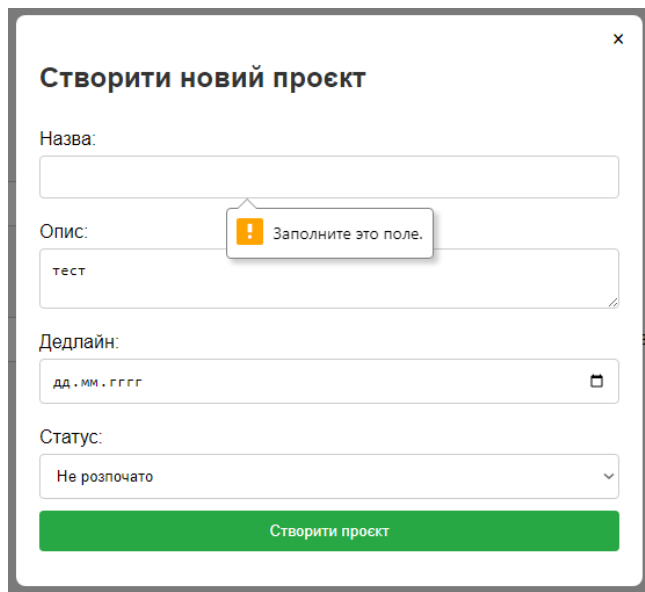
При введенні невалідного значення пошти виводиться повідомлення про помилку у введенні пошти (рис.3.10).

The screenshot shows a registration interface with two tabs at the top: 'Вхід' (Login) in grey and 'Реєстрація' (Registration) in green. The 'Реєстрація' tab is active. Below the tabs, the title 'Реєстрація' is displayed. There are two input fields: 'Ім'я:' (Name) containing 'testuser' and 'Email:' containing 'user1'. A green button labeled 'Зареєструватися' (Register) is positioned below the email field. A yellow error message box is overlaid on the form, stating: 'Адрес электронной почты должен содержать символ "@". В адресе "user1" отсутствует символ "@'.

Рис.3.10 Помилка про введення не валідної пошти

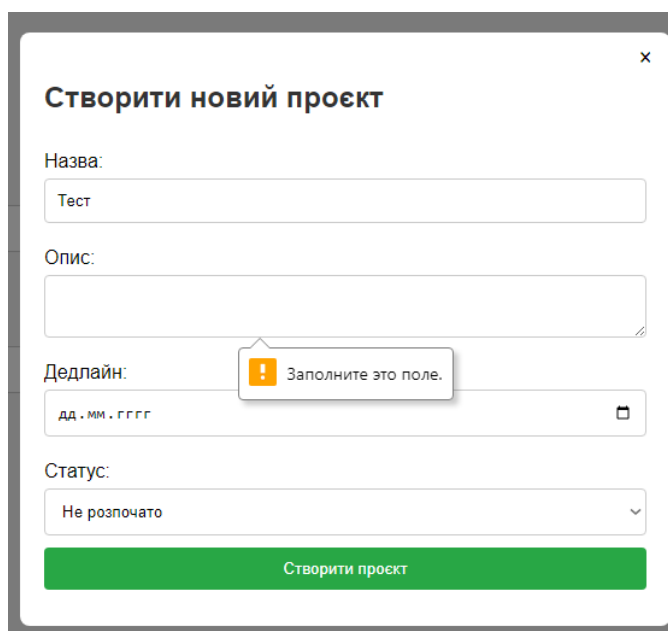
3. Створення проекту

При створенні нового проекту обов'язково заповнювати деякі поля. Якщо їх не заповнити, буде виведено повідомлення біля кожного обов'язкового поля про його заповнення (рис.3.11-3.13).



The screenshot shows a web form titled "Створити новий проект" (Create new project). It contains several input fields: "Назва:" (Name), "Опис:" (Description), "Дедлайн:" (Deadline), and "Статус:" (Status). The "Назва:" field is empty, and a yellow tooltip with an exclamation mark icon and the text "Заполните это поле." (Fill in this field.) is pointing to it. The "Опис:" field contains the text "тест". The "Дедлайн:" field has a date picker icon and the text "дд . мм . гggg". The "Статус:" field is a dropdown menu with "Не розпочато" (Not started) selected. At the bottom, there is a green button labeled "Створити проект" (Create project).

Рис.3.11 Повідомлення про заповнення назви проекту



The screenshot shows the same "Створити новий проект" form. In this version, the "Назва:" field contains the text "Тест", and the "Опис:" field is empty. The yellow tooltip with the exclamation mark icon and the text "Заполните это поле." is now pointing to the "Опис:" field. The other fields and the "Створити проект" button remain the same as in the previous screenshot.

Рис.3.12 Повідомлення про обов'язкове заповнення опису проекту

Рис.3.13 Повідомлення про обов’язкове заповнення поля дедлайну проекту

4. Створення завдання

При створенні завдання є обов’язкові поля для заповнення. Для кожного поля виводиться повідомлення про потребу його заповнити (рис.3.14-3.15).

Рис.3.14 Повідомлення про обов’язкове заповнення поля назви завдання

Рис.3.15 Повідомлення про обов'язкове заповнення поля опису завдання

5. Додавання коментаря

До кожного завдання можна додати коментар. Якщо натиснути кнопку додавання коментаря, а в полі коментаря тексту не буде, виведеться попередження, про те, що потрібно заповнити поле коментаря (рис.3.16).

Рис.3.16 Попередження про порожнє поле коментаря

Усі функціональності системи успішно пройшли тестування. Система коректно обробляє вхідні дані, забезпечує безпечну взаємодію з базою даних через PDO і відображає дані в інтерфейсі.

Висновки до розділу 3

Розробка веб-орієнтованої системи управління проєктами, представлена в даній дипломній роботі, є комплексним дослідженням, що охоплює аналіз, проєктування, реалізацію та тестування інформаційної системи, спрямованої на автоматизацію процесів керування проєктами. Робота структурована у три основні розділи, кожен з яких послідовно розкриває етапи створення системи, від теоретичних основ до практичної реалізації.

У першому розділі проведено аналіз сучасних інформаційних технологій у системах управління проєктами. Постановка задачі визначила ключові вимоги до системи, включаючи підтримку авторизації, керування проєктами, завданнями та ресурсами. Аналіз існуючих систем, таких як Trello, Jira та Asana, дозволив виявити їхні переваги й недоліки, що вплинуло на формування функціональних і нефункціональних вимог до розроблюваної системи. Вибір клієнт-серверної архітектури обґрунтовано її масштабованістю та підтримкою веб-доступу, а інструментальні засоби (PHP, MySQL, JavaScript, HTML, CSS) і вимоги до апаратного забезпечення обрано з урахуванням доступності, продуктивності та сумісності.

Другий розділ присвячено проєктуванню та реалізації системи. Визначення варіантів використання та об'єктно-орієнтованої структури дозволило чітко описати сценарії взаємодії користувачів із системою та основні сутності (користувачі, проєкти, завдання, коментарі). Розробка бази даних забезпечила створення реляційної моделі з нормалізованими таблицями та зовнішніми ключами для підтримки цілісності даних. Проєктування ключових алгоритмів охопило керування задачами, ресурсами та термінами з використанням асинхронних запитів (AJAX) і безпечних SQL-запитів через PDO. Реалізація веб-додатку включала створення клієнтської (HTML, CSS, JavaScript) та серверної (PHP, MySQL) частин, що забезпечують авторизацію, керування проєктами й завданнями та зручний інтерфейс.

Третій розділ зосереджено на практичних аспектах роботи із системою. Установка та налаштування параметрів детально описали процес розгортання системи на сервері, включаючи конфігурацію Apache, PHP, MySQL і бази даних. Опис інтерфейсу користувача висвітлив структуру сторінок, картковий дизайн, модальні форми та адаптивність, що забезпечують інтуїтивну взаємодію. Тестування підтвердило працездатність усіх функціональностей.

Розроблена система управління проєктами відповідає поставленим вимогам, забезпечуючи автоматизацію ключових процесів керування проєктами,

					ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

зручний інтерфейс і безпечну роботу з даними. Вона є гнучкою для подальшого розширення, зокрема шляхом додавання ролей користувачів, інтеграції з зовнішніми API або впровадження автоматизованого тестування. Отримані результати можуть бути використані в реальних проєктах для підвищення ефективності командної роботи та управління ресурсами.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Розробка веборієнтованої системи управління проєктами, представлена в даній дипломній роботі, є комплексним дослідженням, що охоплює аналіз, проєктування, реалізацію та тестування інформаційної системи, спрямованої на автоматизацію процесів керування проєктами. Робота структурована у три основні розділи, кожен з яких послідовно розкриває етапи створення системи, від теоретичних основ до практичної реалізації.

У першому розділі проведено аналіз сучасних інформаційних технологій у системах управління проєктами. Постановка задачі визначила ключові вимоги до системи, включаючи підтримку авторизації, керування проєктами, завданнями та ресурсами. Аналіз існуючих систем, таких як Trello, Jira та Asana, дозволив виявити їхні переваги й недоліки, що вплинуло на формування функціональних і нефункціональних вимог до розроблюваної системи. Вибір клієнт-серверної архітектури обґрунтовано її масштабованістю та підтримкою вебдоступу, а інструментальні засоби (PHP, MySQL, JavaScript, HTML, CSS) і вимоги до апаратного забезпечення обрано з урахуванням доступності, продуктивності та сумісності.

Другий розділ присвячено проєктуванню та реалізації системи. Визначення варіантів використання та об'єктно-орієнтованої структури дозволило чітко описати сценарії взаємодії користувачів із системою та основні сутності (користувачі, проєкти, завдання, коментарі). Розробка бази даних забезпечила створення реляційної моделі з нормалізованими таблицями та зовнішніми ключами для підтримки цілісності даних. Проєктування ключових алгоритмів охопило керування задачами, ресурсами та термінами з використанням асинхронних запитів (AJAX) і безпечних SQL-запитів через PDO. Реалізація вебдодатку включала створення клієнтської (HTML, CSS, JavaScript) та серверної (PHP, MySQL) частин, що забезпечують авторизацію, керування проєктами й завданнями та зручний інтерфейс.

					ІПЗ.КР.Б – 121 – 25 – ІЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Третій розділ зосереджено на практичних аспектах роботи із системою. Установка та налаштування параметрів детально описали процес розгортання системи на сервері, включаючи конфігурацію Apache, PHP, MySQL і бази даних. Опис інтерфейсу користувача висвітлив структуру сторінок, картковий дизайн, модальні форми та адаптивність, що забезпечують інтуїтивну взаємодію. Тестування підтвердило працездатність усіх функціональностей.

Розроблена система управління проєктами відповідає поставленим вимогам, забезпечуючи автоматизацію ключових процесів керування проєктами, зручний інтерфейс і безпечну роботу з даними. Вона є гнучкою для подальшого розширення, зокрема шляхом додавання ролей користувачів, інтеграції з зовнішніми API або впровадження автоматизованого тестування. Отримані результати можуть бути використані в реальних проєктах для підвищення ефективності командної роботи та управління ресурсами.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Загальна характеристика управління проєктами. – Режим доступу: <http://dspace.wunu.edu.ua/retrieve/19690/>
2. Управління проєктами: навчальний посібник до вивчення дисципліни для магістрів галузі знань 07 «Управління та адміністрування» спеціальності 073 «Менеджмент» спеціалізації: «Менеджмент і бізнес-адміністрування», «Менеджмент міжнародних проєктів», «Менеджмент інновацій», «Логістика»/ Уклад.: Л.Є. Довгань, Г.А.Мохонько, І.П.Малик. Київ: КПІ ім. Ігоря Сікорського, 2017. 420 с.
3. Косенко Н. В. Проєктний менеджмент : слайд-конспект для здобувачів першого (бакалаврського) рівня вищої освіти всіх форм навчання спеціальності 073– Менеджмент; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. Харків : ХНУМГ ім. О. М. Бекетова, 2023. 130 с.
4. Документація Trello. – Режим доступу: <https://trello.com>
5. Документація Trello: що це таке та як ним користуватися. – Режим доступу: <https://hostiq.ua/blog/ukr/what-is-trello-2/>
6. Документація Asana. – Режим доступу: <https://asana.com>
7. Що таке Asana project management — як спростити ваш робочий процес. – Режим доступу: <https://cloudfresh.com/ua/cloud-blog/shho-take-asana-project-management-yak-sprostyty-vash-robochyj-protses/>
8. Документація Jira. – Режим доступу: <https://www.atlassian.com/software/jira>
9. Що таке Jira і як з нею працювати. – Режим доступу: <https://iampm.club.ua/blog/shho-take-jira-i-yak-z-neyu-praczuuvati/>
10. Документація Microsoft Project. – Режим доступу: <https://www.microsoft.com/en-us/microsoft-365/planner/microsoft-planner>
11. Система автоматизації управління проєктами Microsoft Project. – Режим доступу: <https://buklib.net/books/28938/>
12. Документація Monday.com. – Режим доступу: <https://monday.com>

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

13. Asana vs Monday: Яка платформа управління проєктами підійде вам більше. – Режим доступу: <https://cloudfresh.com/ua/cloud-blog/asana-vs-monday-yaka-platforma-upravlinnya-proektamy-pidijde-vam-bilshe/>

14. Клієнт-серверна архітектура. – Режим доступу: <https://training.gatestlab.com/blog/technical-articles/client-server-architecture/>

15. model-view-controller (MVC). – Режим доступу: <https://www.techtarget.com/whatis/definition/model-view-controller-MVC>

16. Kevin Wilson. The Absolute Beginner's Guide to HTML and CSS: A Step-by-Step Guide with Examples and Lab Exercises First Edition. 2023. с. 256.

17. Marijn Haverbeke. Eloquent JavaScript, 4th Edition 4th Edition. 2024. с. 456.

18. Jon Duckett. PHP & MySQL: Server-side Web Development 1st Edition. Wiley. 2022. с. 672.

19. Daniel Nichter. Efficient MySQL Performance: Best Practices and Techniques 1st Edition. O'Reilly Media. 2022. с. 335.

20. Bruce Johnson. Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers 1st Edition. Wiley. 2019. с. 192.

21. SuperSummary. Study Guide: Ajax by Sophocles (SuperSummary) Kindle Edition. 2022. 49 с.

22. PHP Data Objects. – Режим доступу: <https://www.php.net/manual/en/book.pdo.php>

23. Kameron Hussain, Frahaan Hussain. Mastering Bootstrap 5: From Basics to Expert Projects. 2023. 274 с.

24. Zak Ruvalcaba, Mary Delamater. Murach's JavaScript and jQuery. 4th Edition. 2020. 752 с.

25. Костішко С. О. Інформаційні системи і технології в управлінні проєктами: навч. посіб. – Київ: Центр учбової літератури, 2020. – 304 с.

26. Храмов Ю. В. Управління ІТ-проєктами: навч. посіб. – Харків: ХНЕУ ім. С. Кузнеця, 2021. – 280 с.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

27. Laudon K., Laudon J. Management Information Systems: Managing the Digital Firm. 17th Edition. Pearson, 2020. – 648 с.

28. Schwalbe K. Information Technology Project Management. 9th Edition. Cengage Learning, 2018. – 672 с.

29. Ковальчук К. А. Менеджмент інформаційних технологій: навч. посіб. – Львів: Новий Світ-2000, 2022. – 212 с.

30. ISO/IEC 27001 — Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. – Режим доступу: <https://www.iso.org/isoiec-27001-information-security.html>

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						64
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг створення бази даних

```
CREATE DATABASE project_management;  
USE project_management;
```

```
CREATE TABLE users (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    username VARCHAR(255) NOT NULL,  
    email VARCHAR(255) UNIQUE NOT NULL,  
    password VARCHAR(255) NOT NULL  
);
```

```
CREATE TABLE projects (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    name VARCHAR(255) NOT NULL,  
    description TEXT,  
    deadline DATE NOT NULL,  
    status ENUM('Не розпочато', 'В процесі', 'Завершено', 'Заморожено') NOT NULL,  
    user_id INT,  
    FOREIGN KEY (user_id) REFERENCES users(id)  
);
```

```
CREATE TABLE tasks (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    project_id INT,  
    title VARCHAR(255) NOT NULL,  
    description TEXT,  
    assignee_id INT,  
    status ENUM('Нове', 'В процесі', 'Завершено', 'Заморожено') NOT NULL,  
    FOREIGN KEY (project_id) REFERENCES projects(id),  
    FOREIGN KEY (assignee_id) REFERENCES users(id)  
);
```

Лістинг коду керування задачами:

```

<?php
session_start();
require_once 'config.php';
header('Content-Type: application/json');

if (!isset($_SESSION['user_id'])) {
    echo json_encode(['success' => false, 'message' => 'Необхідна авторизація']);
    exit;
}

$action = $_POST['action'] ?? $_GET['action'] ?? '';

if ($action === 'create') {
    $project_id = filter_input(INPUT_POST, 'project_id',
FILTER_SANITIZE_NUMBER_INT);
    $title = filter_input(INPUT_POST, 'title', FILTER_SANITIZE_STRING);
    $description = filter_input(INPUT_POST, 'description', FILTER_SANITIZE_STRING);
    $assignee_id = filter_input(INPUT_POST, 'assignee_id',
FILTER_SANITIZE_NUMBER_INT);
    $status = filter_input(INPUT_POST, 'status', FILTER_SANITIZE_STRING);

    $stmt = $pdo->prepare("INSERT INTO tasks (project_id, title, description, assignee_id,
status) VALUES (?, ?, ?, ?, ?)");
    if ($stmt->execute([$project_id, $title, $description, $assignee_id, $status])) {
        echo json_encode(['success' => true]);
    } else {
        echo json_encode(['success' => false, 'message' => 'Помилка створення завдання']);
    }
} elseif ($action === 'get') {
    $project_id = filter_input(INPUT_GET, 'project_id',
FILTER_SANITIZE_NUMBER_INT);
    $assignee_id = filter_input(INPUT_GET, 'assignee_id',
FILTER_SANITIZE_NUMBER_INT);
    $status = filter_input(INPUT_GET, 'status', FILTER_SANITIZE_STRING);

    $query = "SELECT t.*, p.name AS project_name, u.username AS assignee_name
FROM tasks t
LEFT JOIN projects p ON t.project_id = p.id
LEFT JOIN users u ON t.assignee_id = u.id
WHERE 1=1";
    $params = [];

    if ($project_id && $project_id !== 'all') {
        $query .= " AND t.project_id = ?";
        $params[] = $project_id;
    }
    if ($assignee_id && $assignee_id !== 'all') {
        $query .= " AND t.assignee_id = ?";

```

```

$params[] = $assignee_id;
}
if ($status && $status !== 'all') {
    $query .= " AND t.status = ?";
    $params[] = $status;
}

$stmt = $pdo->prepare($query);
$stmt->execute($params);
$tasks = $stmt->fetchAll(PDO::FETCH_ASSOC);
echo json_encode(['success' => true, 'tasks' => $tasks]);
}
?>
Файл: script.js (фрагмент)
window.filterTasks = async function() {
    const projectId = document.getElementById('project-filter').value;
    const assigneeId = document.getElementById('assignee-filter').value;
    const status = document.getElementById('status-filter').value;
    let url = 'tasks.php?action=get';
    if (projectId !== 'all') {
        url += '&project_id=${projectId}';
    }
    if (assigneeId !== 'all') {
        url += '&assignee_id=${assigneeId}';
    }
    if (status !== 'all') {
        url += '&status=${encodeURIComponent(status)}';
    }

    const response = await fetch(url);
    const result = await response.json();
    const taskList = document.getElementById('task-list');
    taskList.innerHTML = '';
    if (result.success) {
        result.tasks.forEach(task => {
            const taskItem = document.createElement('div');
            taskItem.className = 'task-item';
            taskItem.dataset.taskId = task.id;
            taskItem.innerHTML = `
                <h4><a href="task_details.php?task_id=${task.id}">${task.title}</a></h4>
                <p>Проект: ${task.project_name}</p>
                <p>Опис: ${task.description}</p>
                <p>Виконавець: ${task.assignee_name || 'Не призначено'}</p>
                <p>Статус: ${task.status}</p>
                <form class="delete-task-form" method="POST" style="display:inline;">
                    <input type="hidden" name="action" value="delete">
                    <input type="hidden" name="id" value="${task.id}">
                    <button type="submit" class="delete-btn">Видалити</button>
                </form>
            `;
            taskList.appendChild(taskItem);
        });
    }
}

```

```

});
taskList.querySelectorAll('.delete-task-form').forEach(form => {
  form.addEventListener('submit', async (e) => {
    e.preventDefault();
    const formData = new FormData(form);
    const response = await fetch('tasks.php', {
      method: 'POST',
      body: formData
    });
    const result = await response.json();
    if (result.success) {
      const taskItem = form.closest('.task-item');
      taskItem.remove();
    } else {
      alert(result.message);
    }
  });
});
}
};

```

					ПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

Фрагмент коду для відображення та призначення виконавців (з файлу dashboard.php):

```
<?php
$stmt = $pdo->prepare("SELECT * FROM users");
$stmt->execute();
$users = $stmt->fetchAll(PDO::FETCH_ASSOC);
?>
<!-- Форма створення завдання -->
<div id="task-modal" class="modal">
    <div class="modal-content">
        <span class="close-btn" onclick="closeModal('task-modal')>×</span>
        <h2>Створити завдання</h2>
        <form id="task-form" action="tasks.php" method="POST">
            <input type="hidden" name="action" value="create">
            <label for="task-project-id">Проект:</label>
            <select id="task-project-id" name="project_id" required>
                <?php foreach ($projects as $project): ?>
                    <option value="<?php echo $project['id']; ?>"><?php echo
htmlspecialchars($project['name']); ?></option>
                <?php endforeach; ?>
            </select>
            <label for="task-title">Назва:</label>
            <input type="text" id="task-title" name="title" required>
            <label for="task-description">Опис:</label>
            <textarea id="task-description" name="description" required></textarea>
            <label for="task-assignee">Виконавець:</label>
            <select id="task-assignee" name="assignee_id" required>
                <?php foreach ($users as $user): ?>
                    <option value="<?php echo $user['id']; ?>"><?php echo
htmlspecialchars($user['username']); ?></option>
                <?php endforeach; ?>
            </select>
            <label for="task-status">Статус:</label>
            <select id="task-status" name="status" required>
                <option value="Нове">Нове</option>
                <option value="В процесі">В процесі</option>
                <option value="Завершено">Завершено</option>
                <option value="Заморожено">Заморожено</option>
            </select>
            <button type="submit">Створити завдання</button>
        </form>
    </div>
</div>
```

Фрагмент коду (dashboard.php):

```

<div class="container">
  <h1>Панель керування</h1>
  <nav>
    <a href="dashboard.php">Проекти</a>
    <a href="tasks.php">Завдання</a>
    <a href="logout.php">Вихід</a>
  </nav>
  <h2>Проекти</h2>
  <button onclick="openModal('project-modal')">Створити проект</button>
  <div id="project-list">
    <?php foreach ($projects as $project): ?>
      <div class="project-item" data-project-id="<?php echo $project['id']; ?>">
        <h3><?php echo htmlspecialchars($project['name']); ?></h3>
        <p><?php echo htmlspecialchars($project['description']); ?></p>
        <p>Дедлайн: <?php echo $project['deadline']; ?></p>
      </div>
    <?php endforeach; ?>
  </div>
  <h2>Завдання</h2>
  <button onclick="openModal('task-modal')">Створити завдання</button>
  <div id="task-list">
    <?php foreach ($tasks as $task): ?>
      <div class="task-item" data-task-id="<?php echo $task['id']; ?>">
        <h4><?php echo htmlspecialchars($task['title']); ?></h4>
        <p>Проект: <?php echo htmlspecialchars($task['project_name']); ?></p>
        <p>Опис: <?php echo htmlspecialchars($task['description']); ?></p>
        <p>Виконавець: <?php echo htmlspecialchars($task['assignee_name'] ?? 'Не
призначено'); ?></p>
        <p>Статус: <?php echo $task['status']; ?></p>
      </div>
    <?php endforeach; ?>
  </div>
<!-- Модальна форма для створення завдання -->
<div id="task-modal" class="modal">
  <div class="modal-content">
    <span class="close-btn" onclick="closeModal('task-modal')">×</span>
    <h2>Створити завдання</h2>
    <form id="task-form" action="tasks.php" method="POST">
      <input type="hidden" name="action" value="create">
      <label for="task-project-id">Проект:</label>
      <select id="task-project-id" name="project_id" required>
        <?php foreach ($projects as $project): ?>
          <option value="<?php echo $project['id']; ?>"><?php echo
htmlspecialchars($project['name']); ?></option>
        <?php endforeach; ?>
      </select>
      <label for="task-title">Назва:</label>

```

```

<input type="text" id="task-title" name="title" required>
<label for="task-description">Опис:</label>
<textarea id="task-description" name="description" required></textarea>
<label for="task-assignee">Виконавець:</label>
<select id="task-assignee" name="assignee_id" required>
  <?php foreach ($users as $user): ?>
    <option value="<?php echo $user['id']; ?>"><?php echo
htmlspecialchars($user['username']); ?></option>
  <?php endforeach; ?>
</select>
<label for="task-status">Статус:</label>
<select id="task-status" name="status" required>
  <option value="Нове">Нове</option>
  <option value="В процесі">В процесі</option>
  <option value="Завершено">Завершено</option>
  <option value="Заморожено">Заморожено</option>
</select>
<button type="submit">Створити завдання</button>
</form>
</div>
</div>

```

Лістинг файлу auth.php

```

<?php
session_start();
require_once 'config.php';
header('Content-Type: application/json');

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $action = $_POST['action'] ?? '';

    if ($action === 'register') {
        $username = filter_input(INPUT_POST, 'username', FILTER_SANITIZE_STRING);
        $email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_EMAIL);
        $password = password_hash($_POST['password'], PASSWORD_DEFAULT);

        $stmt = $pdo->prepare("SELECT * FROM users WHERE email = ?");
        $stmt->execute([$email]);
        if ($stmt->rowCount() > 0) {
            echo json_encode(['success' => false, 'message' => 'Email вже зареєстровано']);
            exit;
        }

        $stmt = $pdo->prepare("INSERT INTO users (username, email, password) VALUES (?, ?, ?)");
        if ($stmt->execute([$username, $email, $password])) {
            $_SESSION['user_id'] = $pdo->lastInsertId();
            echo json_encode(['success' => true]);
        } else {
            echo json_encode(['success' => false, 'message' => 'Помилка реєстрації']);
        }
    } elseif ($action === 'login') {
        $email = filter_input(INPUT_POST, 'email', FILTER_SANITIZE_EMAIL);
        $password = $_POST['password'];

        $stmt = $pdo->prepare("SELECT * FROM users WHERE email = ?");
        $stmt->execute([$email]);
        $user = $stmt->fetch();

        if ($user && password_verify($password, $user['password'])) {
            $_SESSION['user_id'] = $user['id'];
            echo json_encode(['success' => true]);
        } else {
            echo json_encode(['success' => false, 'message' => 'Невірний email або пароль']);
        }
    }
}
?>

```