Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Інфокомунікації_____
(повна назва)

Кафедра _____Інформаційно-мережної інженерії_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____перший (бакалаврський)_____
___Клієнт-серверний додаток для управління завданнями для ___ІТ-відділів___
_____
(тема)

Виконав:
здобувач ____4_____ року навчання,
групи _____ТРІМІ-21-2_____
_____Булгаков Роман_____
(власне ім'я, прізвище)

Спеціальність ___172 Телекомунікації та_____
___радіотехника_____
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна_____
Освітня програма _Інформаційно-мережна___
__інженерія_____
(повна назва освітньої програми)

Керівник ст. викл. Галина Ляшенко_____
_____
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри _____ _____ ____Валерій Безрук_____
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інфокомунікації

Кафедра Інформаційно-мережної інженерії

Рівень вищої освіти перший (бакалаврський)

Спеціальність 172 Телекомунікації та радіотехника
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформаційно-мережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Булгакову Роману Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Клієнт-серверний додаток для управління завданнями для IT-відділів

затверджена наказом університету від «23» 05 2025 р. № 410СТ

2. Термін подання здобувачем роботи до екзаменаційної комісії 13 06 2025 р.

3. Вихідні дані до роботи Система управління завданнями, HTML, CSS, PHP, вебсервер, база даних

4. Перелік питань, що потрібно опрацювати в роботі

1. Проаналізувати роботу клієнт-серверних моделей;
2. Проаналізувати існуючі системи управління завданнями;
3. Обрати технології для розробки клієнт-серверного додатку;
4. Створити інтерфейс користувача з урахуванням стандарту WCAG 2.2;
5. Розробити додаток з застосуванням обраних технологій.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Слайди

у форматі Power Point.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Ознайомлення із завданням. Уточнення ТЗ.	26.05-27.05.2025	виконано
2	Підбір літератури за темою роботи.	26.05-28.05.2025	виконано
3	Виконання розділу 1	29.05-31.05.2025	виконано
4	Виконання розділу 2	01.06-03.06.2025	виконано
5	Виконання розділу 3	04.06-05.06.2025	виконано
6	Виконання розділу 4	06.06-07.06.2025	виконано
7	Виконання розділу 5	08.06-09.06.2025	виконано
8	Оформлення пояснювальної записки	10.06-11.06.2025	виконано
9	Оформлення презентаційного матеріалу,	12.06-18.06.2025	виконано
	підготовка до захисту у ЕК		

Дата видачі завдання __26_ __червня_____2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _ст.викл. Галина Ляшенко_____
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка 73 с., 33 рис., 7 табл., 2 додатка, 30 джерел.

Об’єкт роботи – система управління завданнями.

Мета роботи – побудова системи управління завданнями для IT-відділу.

Проведено аналіз клієнт-серверної моделі, її переваг та недоліків. Детально проаналізовано типи архітектур, HTTP-протокол, вебсервери. Розглянуто поняття Task Managers, було проаналізовано сучасні мови програмування та засоби, які потрібні при створенні проекту. Відповідно до результатів аналізу було створено систему управління завдань для IT-відділів.

СИСТЕМА УПРАВЛІННЯ ЗАВДАНЬ, КЛІЄНТ-СЕРВЕР МОДЕЛЬ, СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ, HTML, CSS, PHP, WIREFRAME.

THE ABSTRACT

Explanatory slip 73 p., 33 fig., 7 tab., 2 app., 30 sources.

Object of the work is a task management system

Purpose of the work is to develop a task management system for an IT department.

An analysis of the client-server model, its advantages and disadvantages was conducted. Various architecture types, the HTTP protocol, and web servers were examined in detail. The concept of task managers was reviewed, and modern programming languages and tools required for the project development were analyzed. Based on the results of this analysis, a task management system for IT departments was developed.

TASK MANAGEMENT SYSTEM, CLIENT-SERVER MODEL, DATABASE MANAGEMENT SYSTEM, HTML, CSS, PHP, WIREFRAME.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

HTML – (HyperText Markup Language) мова розмітки гіпертексту;

UI – (User Interface) інтерфейс користувача;

UX – (User Experience) досвід користувача;

CSS – (Cascading Style Sheets) каскадні таблиці стилів;

JS – (JavaScript) джава скрипт;

PC – (personal computer) персональний комп’ютер;

DB – (database) база даних;

TCP – (Transmission Control Protocol) протокол управління передачею;

IP – (Internet Protocol) Інтернет протокол;

HTTP – (Hypertext Transfer Protocol) протокол передачі даних;

PHP – (Hypertext Preprocessor) гіпертекстовий препроцесор;

GUI – (Graphical User Interface) графічний інтерфейс користувача.

ВСТУП

ІТ-технології відіграють ключову роль у сучасному суспільстві, так як самі технології вже давно стали частинною нашого повсякденного життя та є потужним інструментом повідомлення інформації. Велика кількість українців зараз працює віддалено у ІТ. Такий великий попит до праці у сфері інтернет технологій обумовлений великою зарплатнею та гнучким графіком роботи. У будь-якій ІТ-компанії є спеціалізовані відділи, які відповідають за різні аспекти розробки проєктів. Під час роботи над проєктами актуальною проблемою є недостатнє розуміння задач, які потрібно виконати, особливо, коли у відділі велика кількість працівників. Для вирішення цієї проблеми потрібно створити систему управління задачами, яка дозволить призначати завдання до різних працівників та бачити на якому етапі воно знаходиться.

Метою роботи є створення системи управління задачами, яка допоможе оптимізувати час роботи між розробниками під час розгортання проєктів.

Поставлена мета обумовила необхідність вирішення наступних завдань:

– з’ясувати особливості систем управління задач та їх функціонал;

– дослідити можливі способи реалізації клієнт-серверних моделей;

– проаналізувати вже існуючі на ринку програми, які допоможуть визначити сучасні тенденції у цій сфері та зрозуміти, який функціонал має бути у програмі;

– дослідити тенденції розвитку ІТ-технологій в умовах цифрової трансформації;

– розглянути актуальні мови програмування, які мають змогу впровадити всі потрібні функції під час практичної реалізації.

– розробити систему управління завдань, яка складається з Frontend та Backend частин.

1 КЛІЄНТ-СЕРВЕРНА МОДЕЛЬ

1.1 Поняття клієнт-серверної моделі

Архітектура «клієнт-сервер» широко використовуються в сучасних застосунках та програмах. Ця модель широко використовується в середовищі Інтернет. Він є сервісом, який побудований на основі Інтернету з метою надійного та легкого обміну даними. Під терміном «дані» можуть бути: документи, текст та відео. Приклад простої клієнт-серверної моделлі зображено на рисунку 1.1.

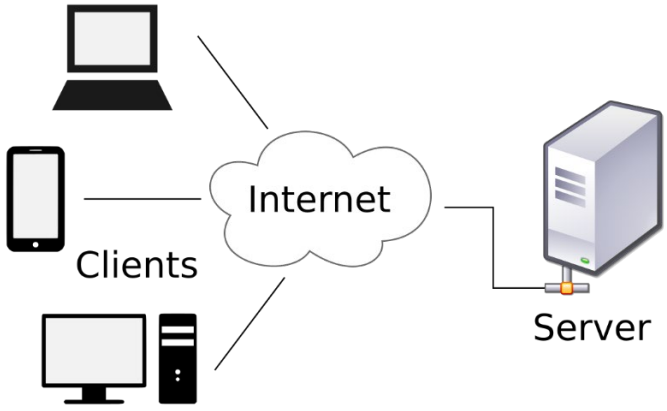

Рисунок 1.1 – Проста клієнт-сервер модель

Ідея цієї моделі полягає у тому, що існують клієнти – це програми або пристрої, під якими мається на увазі те, чим користується кінцевий користувач який запитує дані, це можна реалізувати через такі пристрої: комп’ютери, ноутбуки, смартфони, планшети тощо. З іншого боку знаходиться комп’ютер, який ці дані надає та обслуговує, тобто сервер. Зазвичай на цьому потужному комп’ютері працює програма, яка обробляє запити – серверна програма. Ця програма може обслуговувати велику кількість клієнтів одночасно та працює цілодобово. Існує можливість запуску декількох серверів на одному пристрої – це називається віртуальні сервери.

Сервер може розміщувати вебдодатки, містити вебресурси, зберігати дані користувачів. Також може обслуговувати велику кількість користувачів.

1.2 Переваги та недоліки клієнт-серверної моделі

Ця модель як і інші мають свої переваги та недоліки. У таблиці 1.1 показано їх та наведено значення характеристик [1].

Таблиця 1.1 – Переваги та недоліки клієнт-сервер моделі

Характеристика	Значення
1	2
Переваги	
Центральний доступ і контроль над даними	Вся потрібна інформація зберігається в одному місці, адміністратор мережі має повний контроль над управлінням і адмініструванням.
Адаптивність	Легке пристосування до змін без значного впливу на роботу системи.
Ефективна, масштабована та організована система	Ефективне оброблення зростаючого навантаження без втрати продуктивності, яке дозволяє серверу працювати без помилок.
Надійність і можливість відновлення	Надійне захищення даних та відновлення роботи після збоїв.
Операційна ефективність	Файли зберігаються на одному сервері, тому легко отримати доступ.
Продуктивність	Можливе розподілення завдань між клієнтами та серверами, що покращує загальну ефективність

Продовження таблиці 1.1

1	2
Недоліки	
Перенавантаження	Коли велика кількість клієнтів звертається до сервера, з'єднання може сповільнитися або перерватися.
Вартість	Висока вартість, тому що сервера потребують іншого обладнання ніж звичайні комп'ютери.
Складність	Складне налаштування та управління.

Ця архітектура надає можливість реалізовувати широкий спектр дій в інтернеті: від користування соціальними мережами до доступу до програмного забезпечення. При користуванні програмами за цією архітектурою дані користувачів зберігаються на сервері, це підвищує вимоги до безпеки системи, тому що сервер може використовуватися як ціль для злому.

1.3 Типи архітектур

Існує 3 основні типи архітектур: двокаскадана, трьохкаскадна та багатокаскадна. Використання певного виду залежить від потреб мережі.

1.3.1 Двокаскадна архітектура

Під час побудови вебінтерфейсів часто використовуються елементи моделі «клієнт-сервер». Двокаскадна архітектура (two-tier architecture) є основною клієнт-серверною архітектурою. Вона складається з таких частин [2]:

– Сервер, який відповідає за отримання запитів і відправку відповідей клієнту, використовуючи при цьому лише власні ресурси.

– Клієнт, який представляє користувацький інтерфейс.

– База даних, яка зберігає дані.

Схема двокаскадної архітектури позначена на рисунку 1.2.

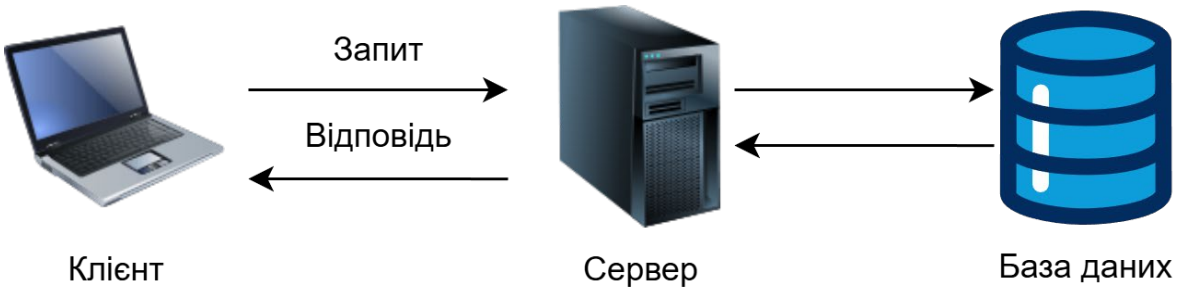

Рисунок 1.2 – Двокаскадна архітектура

Система управління базами даних (СУБД) є елементом зберігання та надання даних. Вона працює на серверному рівні, обробляючи запити від клієнтів.

У такій архітектурі користувачі запускають додатки на своїх PC, які підключаються до сервера через мережу. Клієнтський додаток виконує програмний код та бізнес-логіку, а потім відображає результат користувачеві. Такий тип клієнта також називають «товстим клієнтом» (thick client) [3].

Двокаскадна архітектура є оптимальним варіантом для маленьких та простих проєктів, так як дозволяє взаємодіяти з базою даних без додаткових рівнів, що робить структуру системи більш простою та зрозумілою.

1.3.2 Трьохкаскадна архітектура

Трьохкаскадна архітектура складається з таких елементів:

– Представлення даних у вигляді інтерфейсу.
– Серверу додатків.
– Серверу бази даних (рис. 1.3).

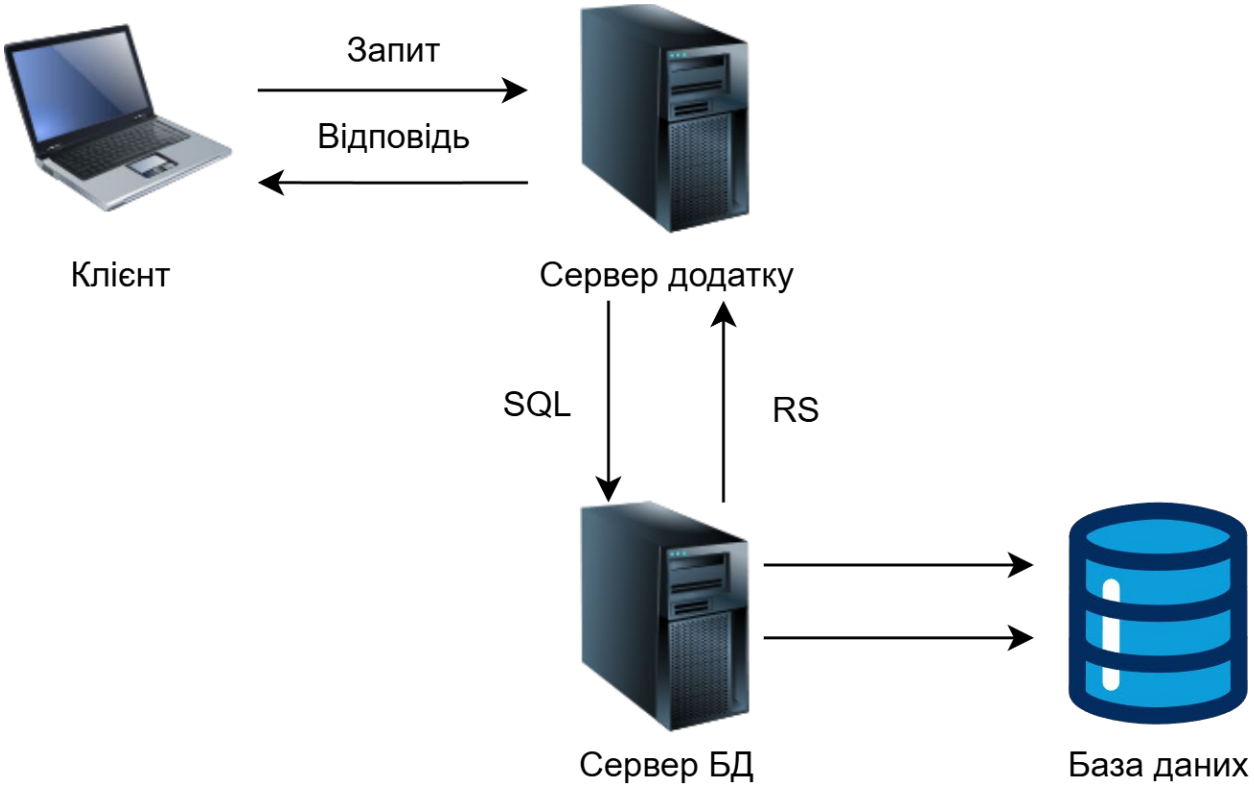

Рисунок 1.3 – Трьохкаскадна архітектура

У такій архітектурі декілька серверів обробляють запит клієнта, тобто є чіткий розподіл операцій, який зменшує навантаження на сервер. Через додавання додаткового рівня підвищується складність проектування та підтримки моделі.

Головна перевага такої архітектури це безпека даних, так як клієнт не має прямого доступу до БД. Це важливий аспект при побудові реєстрацій та інших систем, які потребують більшої уваги до захисту даних.

1.3.3 Багатокаскадна архітектура

Під час побудови складних систем, аналогічних до Amazon та Netflix з’являється необхідність використання додаткових серверів. Реалізація такої вимоги є розширення трьохкаскадної архітектури (3-tier) до багатокаскадної (multi-tier) (рис. 1.4). Ця архітектура дозволяє підвищити ефективність роботи системи та оптимізувати розподіл її ресурсів.

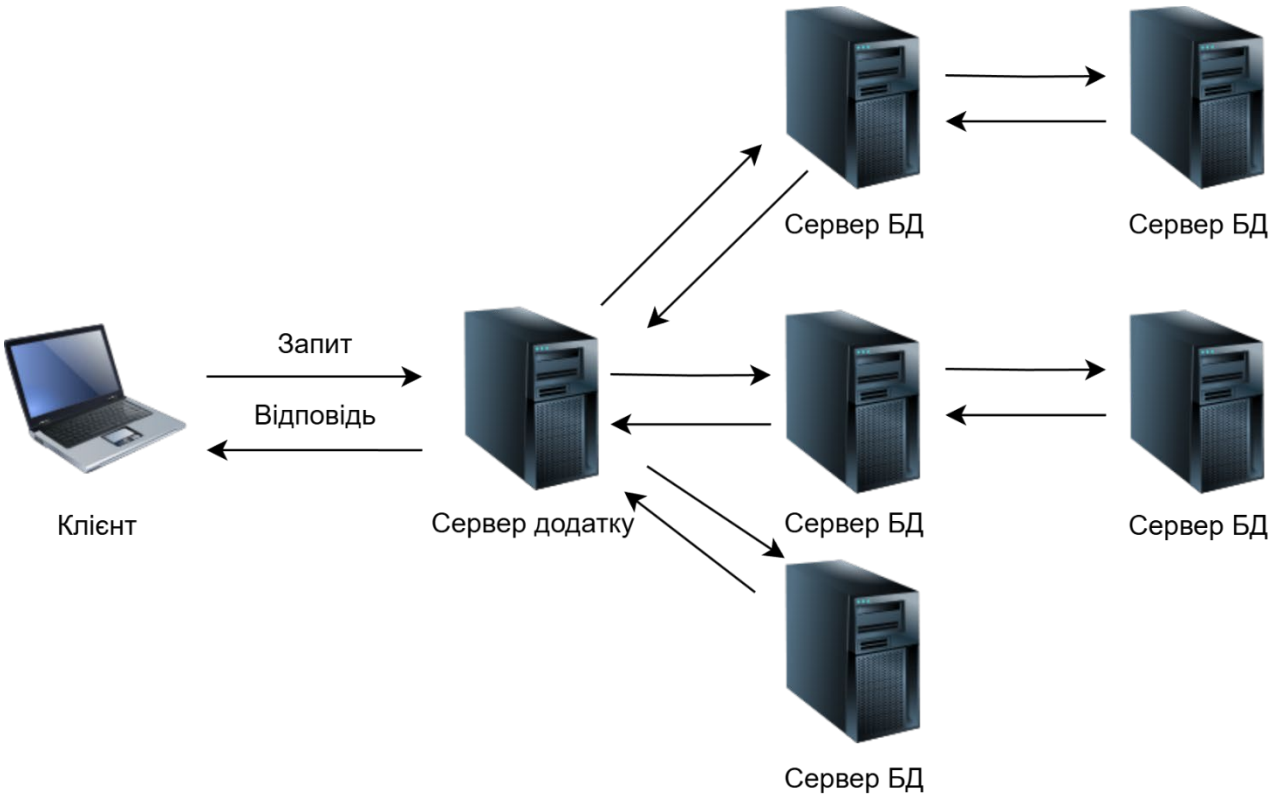

Рисунок 1.4 – Багатокаскадна архітектура

Така кількість серверів дозволяє розподілити дані більш ретельно, що дозволяє масштабувати окремі рівні без взаємодії з іншими. При виході з ладу окремого серверу який відповідає, наприклад, за реєстрацію, всі інші будуть працювати без змін. Таке розподілення даних зменшує ризики кібератак та підвищує безпеку. Багатокаскадна архітектура дозволяє легко взаємодіяти з окремими рівнями та обслуговувати їх.

Ця архітектура складна у реалізації та потребує ретельного проєктування.

1.4 HTTP-протокол

HTTP (HyperText Transfer Protocol) – це протокол прикладного рівня, який використовується для обміну інформацією між вебклієнтом і сервером. Він базується на TCP/IP, та використовується для передавання зображень, відео, аудіо та інших даних. Схема роботи HTTP запиту наведена на рисунку 1.5.

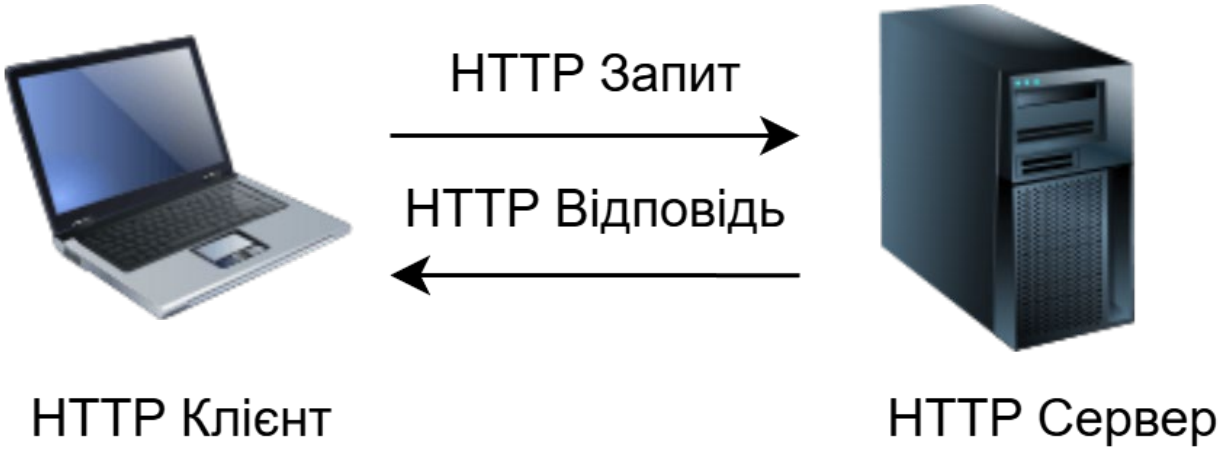

Рисунок 1.5 – Схема роботи HTTP запиту

HTTP має декілька важливих особливостей над іншими протоколами.

1) Відсутня потреба постійного з’єднання. Після надсилання запиту, комп’ютери роз'єднуються, а коли відповідь готова з’єднання знову встановлюється, щоб передати її. Потім з’єднання закривається.

2) HTTP має можливість передавати будь-які типи даних, якщо обидва комп’ютери здатні їх обробити.

3) HTTP є протоколом без збереження стану. Клієнт і сервер знають про один одного тільки під час поточного запиту, тобто після розриву з’єднання потрібно буде знову підключитися та обмінятися інформацією [4].

На сьогодні HTTP є простим та зручним інструментом для обміну інформацією та доступу до даних у мережі.

1.5 Вебсервери

Вебсервер є компь’ютером, який зберігає файли сайту (CSS, HTML, JavaScript, картинки та інше) та відправляє їх на пристрій кінцевого користувача (веббраузер). Він підключений до мережі Інтернет та може бути доступний через доменне ім’я. З точки зору програмного забезпечення вебсервер включає кілька компонентів, які контролюють доступ вебкористувачів до розміщених на сервері

файлів, а саме HTTP-сервер та HTTP-протокол, який браузер використовує для обміну вебсторінок (рис. 1.6) [5].

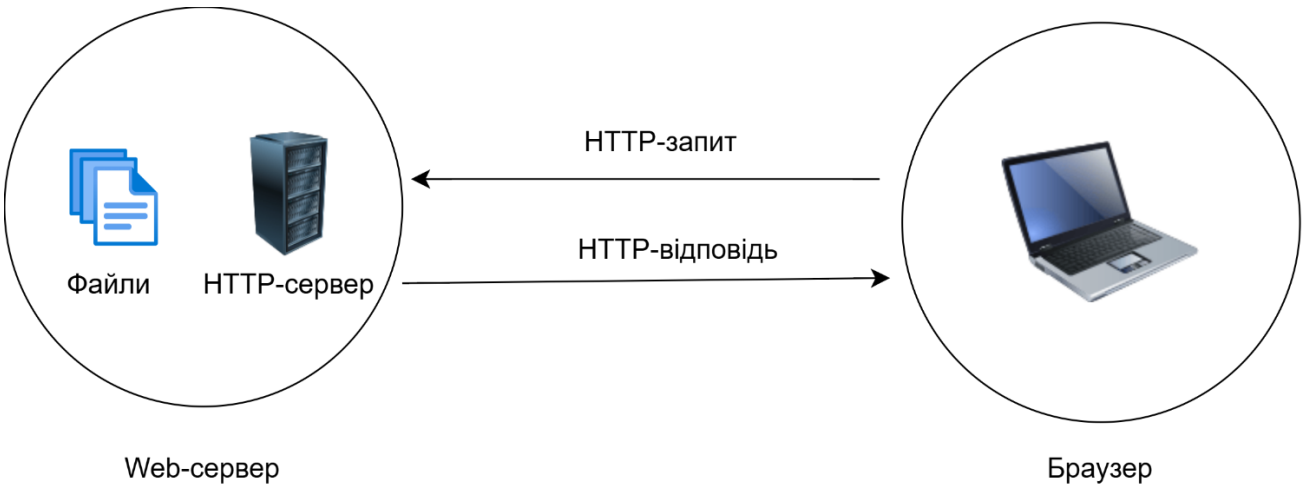

Рисунок 1.6 – Схема роботи вебсервера

Щоб отримати доступ до файлу, який розташований на вебсервері, браузер запрошує його через HTTP-протокол. Після отримання запиту на вебсервері, сервер приймає запит та виконує його, якщо його немає, то видає помилку та відправляє назад.

Кожен вебсервер має впроваджувати наступні функції, які покращать його роботу в умовах реального використання під час розгортання проєктів на серверах. У таблиці 1.2 було наведено основні аспекти, які мають бути присутніми у вебсервері [6].

Таблиця 1.2 – Характеристика вебсервера

Характеристика	Значення
Швидкодія	Можливість справлятися з великою кількістю запитів, використовуючи мінімум апаратних ресурсів.
Аутентифікація	Розподілення клієнтів за ролями та надавання різних прав.

Продовження таблиці 1.2

1	2
Багатозадачність	Обробляння декількох запитів одночасно та можливість підтримання даних без зупинки серверу.
Підтримка різних форматів	JPEG, GIF, PNG, TIFF.
Безпека	Захист від кібератак та шифрування даних.

Наявність таких характеристик у вебсервері дають можливість його простого використання та налаштування.

Існують два види вебсерверів: статичний та динамічний.

Вебсервери можуть надавати як статичний, так і динамічний контент. Статичний контент відображається без змін, а динамічний має можливість оновлюватися.

Статичний вебсервер надсилає заздалегідь підготовлені файли у незмінному вигляді до браузеру, тобто усі користувачі бачать однаковий контент. Також там не має обробки на стороні сервера, взаємодії з базою даних або генерації контенту у реальному часі.

Динамічний вебсервер включає у себе сервер додатків та базу даних. Він називається динамічним, тому що сервер додатків може змінювати файли перед тим, як вони будуть надіслані до браузера, також є можливість генерування контенту у реальному часі, звертаючись до бази [7].

Під час створення вебдодатку було використано динамічну модель вебсервера, тому що було створено систему вирішення задач, яка має включати у себе інтерактивні функції, такі як форма входу, додавання та моніторинг задач, які у свою чергу можна реалізувати тільки з використанням динамічної моделі та підключенням бази даних. Такі функції у проєкті покращують простоту та легкість використання додатка, але ускладнює розробку та дизайн.

1.5.1 Вебсервер Apache

Apache був створений Apache Software Foundation та є популярним і поширеним вебсервером, який використовували протягом останнього десятиліття. Apache є кросплатформним, легким, надійним і використовується в малих компаніях та великих корпораціях. Apache також є безкоштовним і має відкритий код [8].

Через відкритий код Apache має велику кількість переваг, так як код може бути відредагованим іншими розробниками. Apache використовує технологію створення модулів, через це у будь-якого розробника є можливість впровадження нових функцій. При знаходженні помилок у роботі є можливість одразу виправити їх, це в свою чергу підвищує надійність.

1.5.2 Комплекс XAMPP

Для розробки коду локально на комп’ютері, необхідно встановити вебсервер який підтримує PHP, для можливості запуску коду на сервері. Існує багато різновидів програмних пакетів, які вже містять у собі потрібні інструменти та працюють на різних операційних системах. Одним з лідерів є XAMPP, який є доступним на будь-якій платформі.

XAMPP – це безкоштовний та простий в установці Apache дистрибутив, що містить MariaDB, PHP і Perl. XAMPP створений з відкритим вихідним кодом, щоб робить його простим у встановленні та використанні [9]. Він дозволяє використовувати Apache, MariaDB, PHP та інші компоненти. Також XAMPP встановлює вебінтерфейс phpMyAdmin через який можна працювати з базою даних.

2 СИСТЕМИ УПРАВЛІННЯ ЗАВДАННЯМИ

2.1 Поняття Task Managers

Швидкий розвиток IT-технологій обумовлює використання різних інструментів та програм, які допомагають під час створення проєктів. Одним з таких інструментів є системи управління завданнями. Основною метою таких систем є оптимізація робочого процесу шляхом ефективного розподілу завдань і контролю за їх виконанням.

Важливим аспектом під час побудови такої системи є проведення ретельного дослідження користувачів та бізнесу, розуміння того ким є користувач. Серед користувачів систем управління завданнями – менеджери проєктів, IT-відділи, а також представники малого, середнього та великого бізнесу.

Було проведено дослідження та виявлено популярність різних систем серед IT-команд (рис. 2.1).

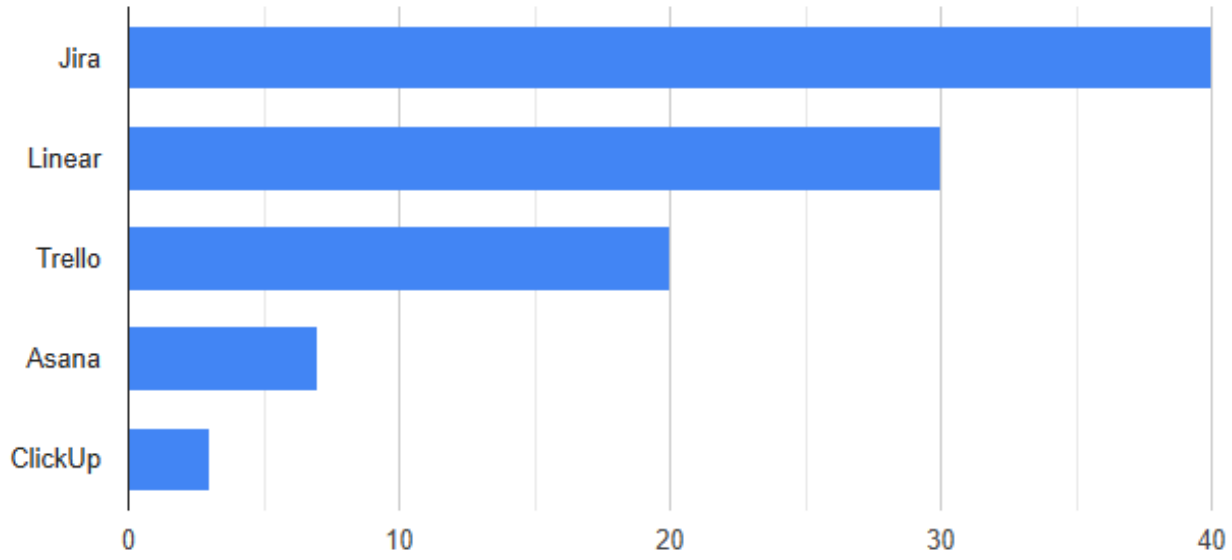

Рисунок 2.1 – Популярність систем управління завдань серед IT-команд [10]

Виходячи з статистики було виявлено, що перевагу віддають Jira та Linear, але навіть серед таких систем IT-відділи досі користуються базовими Asana та ClickUp.

Системи управління завданнями відіграють ключову роль в організації роботи IT-команд. Їх використання дозволяє оптимізувати робочі процеси, підвищити продуктивність і забезпечити прозорість у виконанні завдань. Для оцінки ефективності та розробки плану майбутньої програми було розглянуто лідерів у цій сфері: Jira та Linear.

2.2 Система управління проєктами Jira

Jira була розроблена австралійською компанією Atlassian. Ця система є комплексним рішенням для гнучкого управління проєктами, який використовують команди для планування, відстеження, випуску та підтримки програмного забезпечення [11].

Під час користування цією програмою було зареєстровано обліковий запис і створено тестовий проєкт (рис. 2.2).

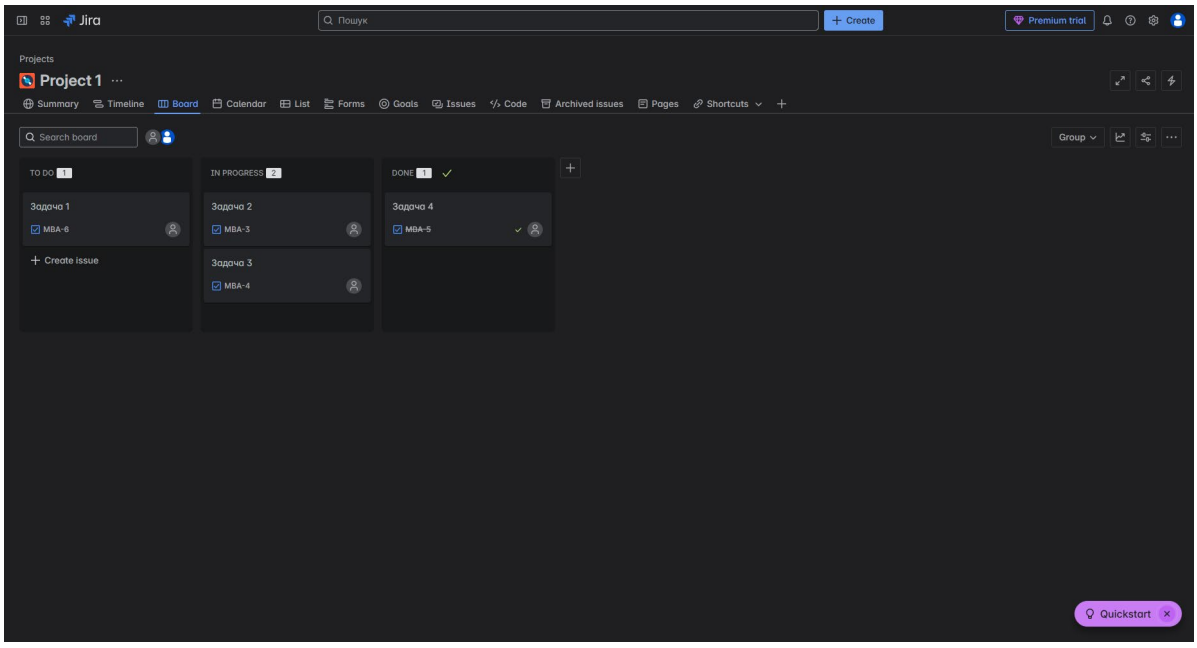

Рисунок 2.2 – Сторінка проєкту

Система управління завданнями Jira має широкий функціонал, який вирішує проблеми у сфері розподілення задач між членами команди. Порівняльна характеристика функцій Jira та Linear була зроблена у вигляді таблиці (табл. 2).

Таблиця 2 – Порівняльна характеристика функцій Jira та Linear

Функція	Jira	Linear
Налаштування різних типів «issues» для проєктів	+	+
Призначення поля обов'язковим	+	-
Видалення поля від перегляду	+	-
Вибір іншого рендера для поля (наприклад, markdown/text)	+	+
Створення нової конфігурації поля	+	-
Робота з плагінами (додатками, інтеграціями через маркетплейс чи API)	+	+
Додавання підказок (help text/tooltips) до користувацьких полів	+	-
Використання JavaScript з користувацькими полями (наприклад, валідація, обчислення значень)	+	-
Налаштування Agile-дошок проєкту (спринти, swimlanes, стани, колонки)	+	-
Інтуїтивний інтерфейс	-	+
Поради для користувача	-	+

Під час дослідження функцій було виявлено перевагу у програмі Jira, але більшість функцій зосереджені на вузькому колі задач, які знадобляться при створені масштабного проєкту для великої команди з задачами широкого спектру.

Jira є умовно безкоштовною. До 10 людей можуть користуватися цією програмою, але цього може бути недостатньо як для команди розробників так і для IT-відділу.

Головним недоліком є не зовсім інтерфейс. Отже, Jira демонструє широкий функціонал, орієнтований на великі команди та складні проєкти. Водночас її складність та перевантажений інтерфейс можуть ускладнити освоєння новачками.

2.3 Система управління проєктами Linear

Linear створений для допомоги командам розробників бути організованими та ефективними. Він зосереджений на керуванні проєктами, відстеженням «issues» та оптимізації роботи у команді. Спочатку він був створений, як простий трекер для відстеження завдань, але з часом трансформувався у потужну систему, яка допомагає на усьому етапі розробки продукту [12].

Після реєстрації було одразу запропоновано краткий перегляд основних функцій і можливостей, це є важливим аспектом у розумінні програми та її функцій. Домашня сторінка проєкту виглядає зрозумілою та не має нічого зайвого (рис. 2.3).

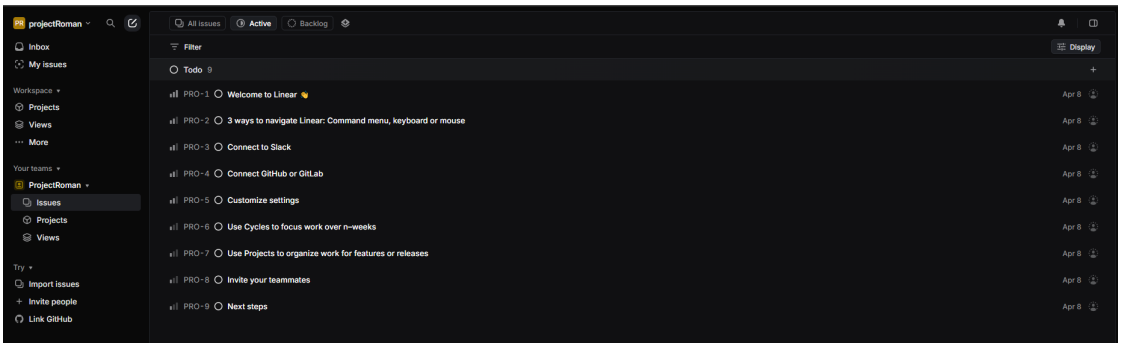

Рисунок 2.3 – Домашня сторінка Linear

На рисунку 2.3 можна побачити відмінність у інтерфейсі між цими двома програмами. У Linear є більш зрозумілий та простий інтерфейс, він не перенавантажений. Після реєстрації створено 9 «issues», які є не просто

шаблонними написами, як було у Jira, а повноцінними маленькими статтями, які додатково розповідають про функціонал цього застосунку (рис. 2.4).

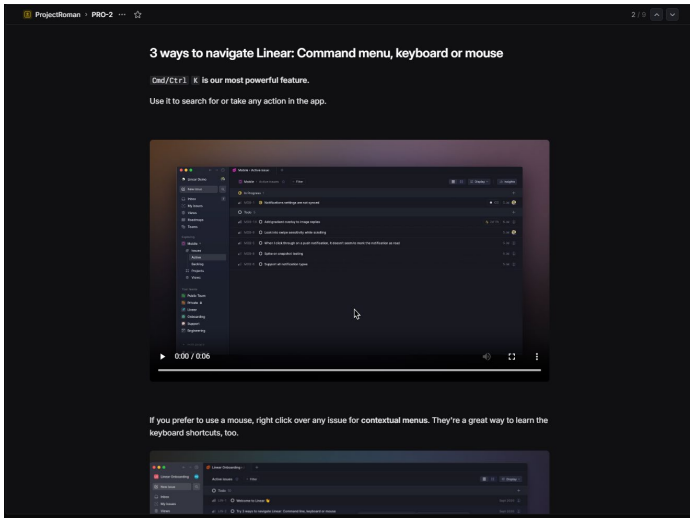

Рисунок 2.4 – Наповнення «issue» 3 ways to navigate Linear

На рисунку 2.4 показано приклад створеного запиту. Він розповідає про навігацію у Linear. У цьому ««issue» присутнє інформативне відео та його опис і команди, які використовуються для навігації. Додатково є інформація про підключення до Slack, GitHub та інші функції.

Jira та Linear успішно виконують спільне завдання: підвищення ефективності роботи команди. Ці програми дають можливість невеликому IT-відділу до 10 працівників ознайомитися з більшістю функцій безкоштовно. Але ці програми все ж таки відрізняються між собою: Jira більш функціональна та складна у опануванні, Linear має не таку велику кількість функцій та має мінімалістичний інтерфейс.

Linear вирізняється простим, інтуїтивно зрозумілим інтерфейсом та зручним стартом для користувача, що робить його доцільним вибором для невеликих команд або користувачів без досвіду роботи з подібними системами.

Аналіз існуючих систем управління завданнями дозволяє визначити ключові функції, які варто реалізувати в майбутньому застосунку, а також уникнути типових недоліків, зокрема перевантаженості інтерфейсу та відсутності зрозумілого навчання для користувачів-початківців.

3 ВИБІР ІНСТРУМЕНТІВ ДЛЯ ВИРІШЕННЯ ЗАДАЧ

3.1 Вибір мов програмування

Перед початком розробки проєкту важливо визначити, які мови програмування доцільно використовувати. Це залежить від специфіки завдань, що стоять перед розробником. Оскільки в межах даного проєкту необхідно створити вебінтерфейс, було обрано такі мови веброзробки: HTML5, CSS3, JavaScript та PHP.

Типовий клієнт-серверний застосунок складається з двох основних частин:

Frontend (клієнтська частина) – відповідає за інтерфейс користувача, тобто все, що бачить і з чим взаємодіє користувач на вебсторінці: кнопки, банери, форми, анімації тощо;

Backend (серверна частина) – забезпечує обробку запитів, взаємодію з базою даних, виконання бізнес-логіки та генерацію відповідей для клієнта [13].

3.1.1 Мова розмітки HTML

HTML (HyperText Markup Language) розроблений консорціумами W3C та WHATWG на чолі з Тімом Бернерсом у 1991 році. Увесь цей час команда розробників продовжує вдосконалювати та впроваджувати нові функції. Останньою версією є HTML5 [14]. Ця мова гіпертекстової розмітки є основою усіх сайтів. Кожен HTML документ буде складатися з деякої групи елементів, де кожен елемент буде визначатися певним тегом [15]. У Frontend частині він завжди присутній, але без додаткових засобів, як: CSS та JavaScript сайт не буде мати ніякого дизайну та стилю.

HTML використовується для розміщення тексту, створення структури, розташування контенту та для побудови візуального фундаменту, який буде відображатися на екрані через браузер.

3.1.2 Стилізація за допомогою CSS

Каскадні таблиці стилів (CSS) – це мова стилів, яка використовується для опису зовнішнього вигляду документа, написаного мовою HTML. CSS визначає, як елементи мають відображатися на екрані, на папері, у мовному форматі або на інших носіях [16].

Ця мова стилів була розроблена компанією W3C (World Wide Web Consortium) в 1996 році з метою можливості вебдизайнерам впроваджувати стилі на вебсторінки. CSS активно підтримується розробниками та впроваджується новими модулями такими як CSS Grid, Variables та Flexbox.

Під час створення проєкту було застосовано CSS для оформлення HTML-структури відповідно до дизайну.

3.1.3 Мова JavaScript

JavaScript – мова веббраузера, яка дозволяє розширити функціонал вебсторінок. Вона стала одною з перших мов програмування, з підтримкою функціонального підходу. JavaScript має багато переваг, основні з яких:

– Динамічна типізація. Більшість мов використовують сильну типізацію, яка дозволяє виявляти обширну кількість помилок під час компіляції. У випадку з JavaScript вона не виявляє помилки типів, це може бути спочатку незвично, але динамічна типізація не вирішує проблему уважного тестування.

– Потужна нотація літералів об’єктів Об’єкти створюються шляхом перерахування їхніх компонентів.

– Повністю взаємодіє з верстkою та серверною частиною вебдодатка. Доказом популярності на ринку мов програмування є те що на базі JavaScript створено велику кількість фреймворків, які розширюють існуючий функціонал мови. Серед лідерів можу зазначити: React JS, Vue JS та Angular [17].

Під час розробки проекту JS було використано у створенні інтерактивних елементів та реагування програми на дії користувача.

3.1.4 Скриптова мова PHP

PHP – це скриптова мова програмування з відкритим початковим кодом, яка була створена для генерації HTML-сторінок на стороні вебсервера. Головне завдання цієї мови це ведення веб розробок, код якого можна впроваджувати безпосередньо в HTML, це допоможе запобігти використання багатьох команд для виведення того ж самого коду (рис. 3.1).

```
<!DOCTYPE html>
<html>
    <head>
        <title>Приклад</title>
    </head>
    <body>

        <?php
            echo "Приклад!";
        ?>

    </body>
</html>
```

Рисунок 3.1 – Приклад генерації HTML-коду з використанням PHP

Щоб вивести код HTML, потрібно вказати спеціальні початкові та кінцеві теги «<? Php» та «?>», які дозволяють перейти в режим PHP та вийти з нього.

Основними сферами застосування PHP є [18]:

– Робота на стороні сервера (створення динамічного вебконтенту, такого як: HTML, графіка, PDF-файли та інше).

– Написання сценаріїв у командному рядку (запуск скриптів з командного рядка, наприклад для системного адміністрування).

– Створення клієнтських програм з графічним інтерфейсом (можливість написання GUI – додатків на PHP.

PHP має ознаки об'єктно-орієнтованої мови програмування, а саме створення: класів та об’єктів, інтерфейсів, інкапсуляції, наслідування та поліморфізму.

Під час аналізу та розробки вебдодатка з допомоги цієї мови було виявлено її переваги та недоліки (табл. 3).

Таблиця 3 – Основні переваги та недоліки мови PHP

Переваги	Недоліки
Відкритий код та легкість початку роботи. PHP просто встановити та почати роботу.	Завдяки відкритому коду та доступу до файлу ASCII, PHP-додатки вразливі до атак.
Кросплатформеність.	Слабка типізація може призвести до некоректності даних та ускладнення налагодження.
Стабільність та впровадження нових версій зі сторони розробників.	Недостатня безпека
Простота підключення різних видів баз даних.	Під час створення великих вебдодатків мова PHP не є кращим вибором через менш ефективну підтримку модульності.
Гнучкість PHP впроваджує можливість комбінування з іншими мовами програмування.	Обмежена підтримка багатопотоковості (працює в однопотоковому режимі)

Ця мова є актуальною у сфері розробки нескладних вебдодатків, що робить її доцільною для реалізації вебінтерфейсу системи управління проєктами.

3.2 Система управління базами даних

База даних (БД) – це організований за певними правилами набір інформації, який можна легко редагувати та доповнювати. Класичні БД організовані у вигляді таблиці: кожен рядок виступає як окремий запис, а стовпець – його атрибут. Для виконання запитів використовується мова SQL. А зручне керування всією БД забезпечує система управління базами даних (СУБД) [19].

3.2.1 Робота з базами даних через PhpMyAdmin

Для зручності під час роботи з базою даних було обрано phpMyAdmin. Одним з його переваг є надання зручного графічного інтерфейсу, а не просто командного рядка (рис. 3.2).

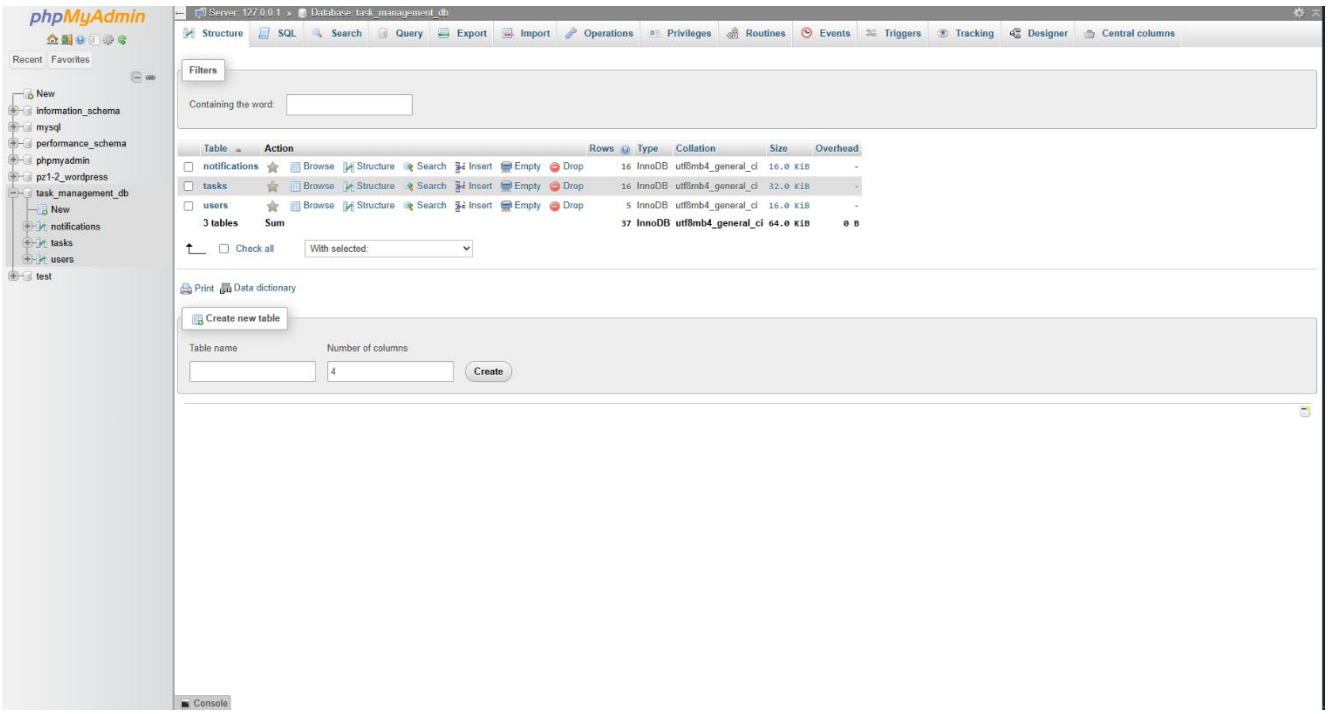

Рисунок 3.2 – Інтерфейс phpMyAdmin

PhpMyAdmin має відкритий вихідний код для адміністрування БД. Він може виконувати різні завдання, такі як створення, модифікація або видалення

баз даних, таблиць, стовпців або рядків. Він також працює з операторами SQL та може керувати користувачами та їхніми дозволами [20].

3.2.2 База даних MariaDB

MariaDB – це розширена сумісна заміна популярного MySQL, яка базується на його відкритому коді та була створена кількома колишніми основними розробниками MySQL.

Під час використання MariaDB було виявлено, що мова SQL і команди такі самі, як у MySQL. Файли конфігурації, порти та сокети, клієнтські API та протоколи також ідентичні. Тобто користувач, який працював з MySQL може відразу почати працювати з MariaDB.

Вона зазвичай використовується у вебдодатках, банківських та фінансових системах.

Незважаючи на велику схожість з MySQL, MariaDB має декілька важливих переваг, серед яких: покращена продуктивність, краща система тестування, додавання нових команд та виправлення помилок [21].

Саме через ці переваги під час створення проєкту було обрано MariaDB, а не MySQL.

3.3 Вибір та використання фреймворків

Фреймворк (framework) – це набір інструментів, бібліотек та правил, який використовується для створення програмних додатків. Він зазвичай являє собою структуру, яка визначає, як компоненти програми повинні взаємодіяти між собою, які шаблони використовувати для створення інтерфейсів і які методи використовувати для роботи з базами даних та іншими ресурсами [22].

Як було зазначено вони допомагають при написані коду, часто надаючи вже готові структури, компоненти та шаблони, підвищуючи ефективність розробки вебзастосунку.

Bootstrap – це безкоштовний фреймворк для веброзробки з відкритим вихідним кодом. Він розроблений для полегшення процесу веброзробки адаптивних, мобільних вебсайтів, надаючи колекцію синтаксису для дизайну шаблонів [23].

Використання Bootstrap дозволяє розробникам швидко створювати елементи вебдодатків, такі як навігаційні панелі, форми, кнопки, таблиці та інші. Фреймворк базується на використанні HTML, CSS та JavaScript для реалізації різноманітних компонентів і функцій.

3.4 Вибір засобів для створення UI/UX

UI/UX дизайн відіграє вирішальну роль у розробці цифрових продуктів, оскільки він безпосередньо впливає на те, як користувачі взаємодіють з продуктом. Добре продуманий інтерфейс може зробити цифровий продукт візуально привабливим, інтуїтивно зрозумілим і зручним для навігації, а добре продуманий UX може гарантувати, що користувачі отримають позитивний досвід протягом усього періоду використання. Успішний UI/UX дизайн може допомогти підвищити задоволеність, лояльність та залученість користувачів до цифрового продуктом, що призводить до вищих показників конверсії та доходу для бізнесу [24].

Дизайн інтерфейсу є одним із визначальних факторів при створенні будь-якого проекту. Щоб інтерфейс виглядав привабливо, а робочий процес був достатньо зрозумілим, використовується додаток для допомоги в дизайні інтерфейсу, а саме Figma [25]. Цей застосунок був створений Ділан Філдом та Еваном Воллесом у 2012 році та вийшов у реліз у 2016 році. Майже одразу став лідером у сфері дизайну UI/UX. Зараз його використовують такі компанії, як Google та Microsoft. Цей дизайн інструмент дуже добре оптимізований для роботи у команді, є можливість робити разом дизайн, бачити курсори та коментарі своїх працівників. Figma може працювати в браузері на різних платформах, також є застосунок, який встановлюється на комп’ютер.

Figma має наступні корисні функції: Auto Layout, Smart Select, Pen tool, Styles, components, and shared libraries, Variable fonts, Branch and merging, Annotations, Powerful plugins [26].

Під час створення проекту цей інструмент використовується для того, щоб зробити каркас сайту та зрозуміти розташування ключових елементів програми. У результаті було отримано дизайн, який полегшує подальшу розробку.

4 СТВОРЕННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

4.1 Створення мапи додатку

Мапа додатку є візуальним інструментом, який допомагає організувати контент і його структуру, а саме показує, як вебсторінки пов’язані між собою. Ціль цього інструменту – показати функціонал майбутньої програми. Під час розробки було створено мапу додатку (рис. 4.1)

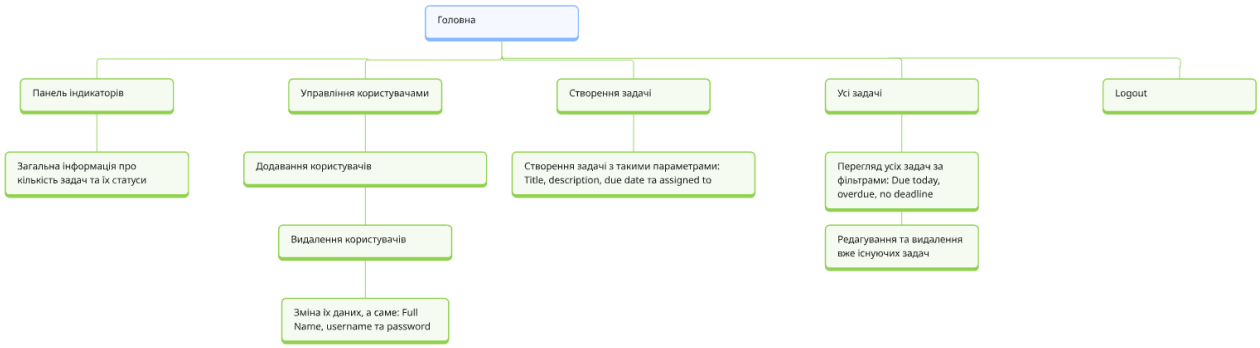

Рисунок 4.1 – Sitemap проекту

На мапі відображені всі функції системи управління завданнями.. Основних сторінок налічується п’ять:, а саме: панель індикаторів, управління користувачами, створення задачі, усі задачі та logout. Такий функціонал є достатнім для поставлених задач.

4.2 Проведення дослідження UX

Додаток проектується для команд розробників IT-відділу. В Україні є попит на такі системи, тому що зараз велика кількість українців працює у сфері IT (рис. 4.2) [27].

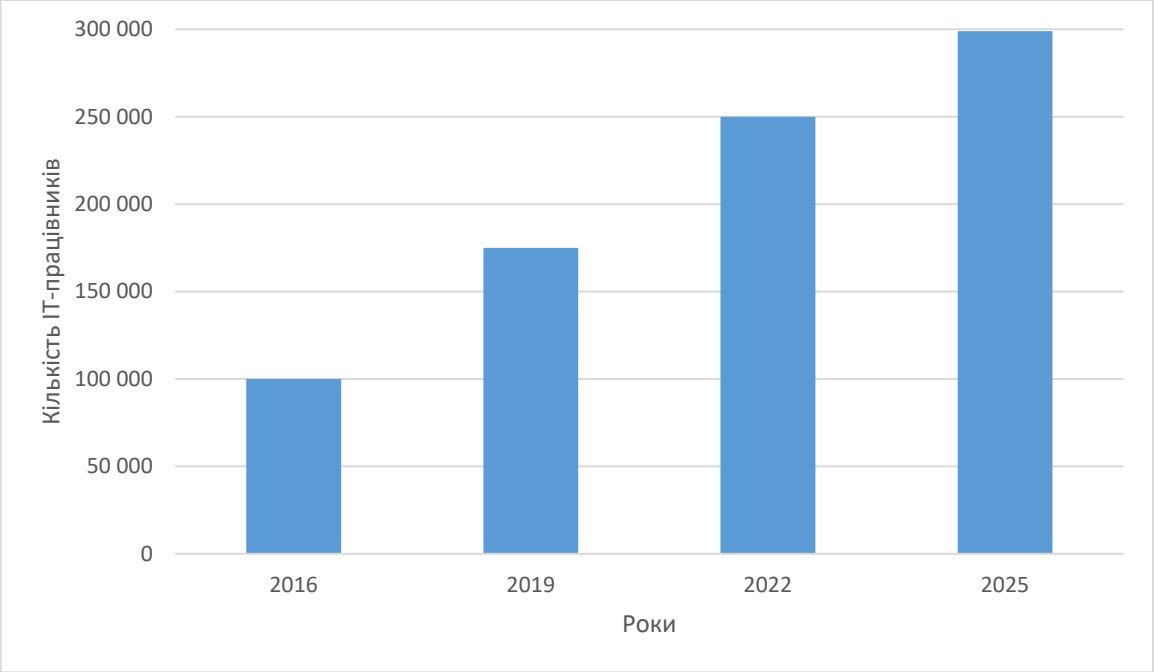

Рисунок 4.2 – Статистика кількості IT-працівників в Україні

Починаючи з 2016 року кількість таких працівників зросла у три рази.

Розробка такої системи є актуальною та важливою, так як для коректної роботи у команді потрібно правильне розподілення задач та їх моніторинг, а це можливе просто та легко зробити при використанні Task Managers.

Система, що розроблюється має вирішувати такі проблеми:

– Ефективність планування та централізація (так як усі задачі для працівників знаходяться у одній програмі це підвищує ефективність планування).

– Комунікація (можливість відстеження задач працівників та їх редагування, що покращує комунікацію під час створення проектів).

– Правильне розподілення праці (відстеження завантаженості кожного працівника та їх зміна, наприклад: у одного користувача 10 задач, а у іншого 5, у програмі це буде видно і можливо буде віддати декілька задач до іншого користувача задля оптимізації праці).

– Економія часу (функціонал таких систем дозволяє тратити менше часу на розподілення задач та їх моніторинг).

Функціонал мапи додатку, який наведений на рисунку 4.1 дозволяє вирішити всі зазначені проблеми.

4.3 Створення wireframe

Wireframe – це двовимірний каркас вебсторінки або додатку. Структурні каркаси дають чітке уявлення про структуру сторінки, макет, інформаційну архітектуру, функціональність і передбачувану поведінку. Основними задачами каркасів сайту є чітке розуміння функцій та створення концепції, орієнтованої на користувача [28]. Wireframe часто використовують, тому що їх вартість відносно мала та це не займає багато часу, а функціонал, який він виконує є важливим під час розробки додатків.

Створенням каркасу додатку зазвичай займаються UX-дизайнери. Цей інструмент допомагає розібратися, де буде розміщена інформація, перш ніж розробники почнуть писати код.

Існує три види каркасів за ступенем деталізації: каркаси низької, середньої та високої точності (табл. 4).

Таблиця 4 – Типи Wireframes за ступенем деталізації

Вид	Характеристика
Каркаси низької точності	Зосереджені на навігації та інформаційній архітектурі.
Каркаси середньої точності	Формування остаточного дизайну, можливе додавання анотацій та вмісту.
Каркаси високої точності	Рання версія макету з інтерактивними та візуальними елементами дизайну.

Каркас може бути намальованим від руки або створений у спеціальній програмі, наприклад Figma, вибір засобу створення залежить від складності додатку та кількості інформації у ньому.

Під час створення Wireframe було обрано варіант розробки каркасу низької точності у програмі Figma, тому що вона має широкий спектр інструментів для дизайнерів (рис. 4.3).

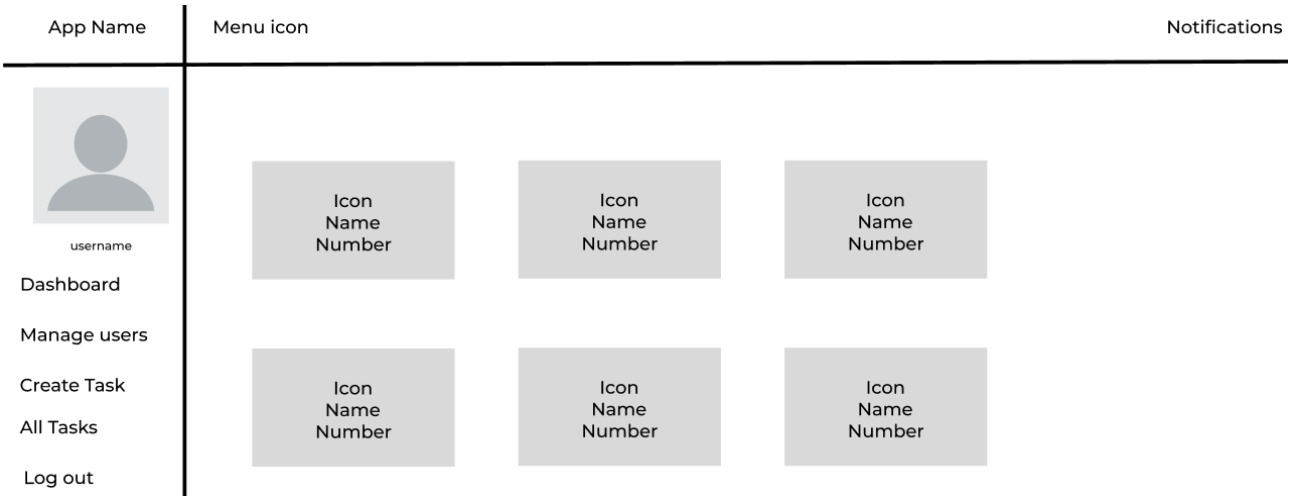

Рисунок 4.3 – Wireframe меню «Dashboard» системи управління завдань

Створений Wireframe показує конструкцію та розташування візуальних елементів відносно один одного. Він не містить ніякого візуального дизайну, анімацій, меню та змісту. Це дозволяє зосередитися на основній структурі і функціональності додатку без відволікання на деталі оформлення. Завдяки такому підходу можна швидко оцінити зручність навігації, логіку розміщення елементів і додати необхідні корективи на ранньому етапі розробки. Наступним кроком є створення більш детальних прототипів та візуального дизайну, що забезпечить комфортне й інтуїтивне використання системи для кінцевих користувачів.

4.4 Доступність контенту згідно з WCAG

WCAG 2.2 (Web Content Accessibility Guidelines) – це міжнародні рекомендації зі створення онлайн-контенту, доступнішого для користувачів. Наприклад, в настановах зібрані правила перекладу жестовою мовою, особливості читабельності тексту, розміщення картинок тощо. Їх мають використовувати веброзробники при створенні онлайн-ресурсів [29].

Існує три види доступності: A, AA, AAA , їх основні вимоги були зображені та таблиці 4.1 [30].

Таблиця 4.1 – Види доступності WCAG

Стандарт	Харектиристика
A (мінімальний рівень)	доступ до контенту за допомогою клавіатури; чітке маркування форм із вказівками; сумісність контенту з допоміжними технологіями; надання інформації різними способами.
AA (середній рівень)	належний контраст між текстом і фоном (мінімум 4.5:1); логічна структура заголовків та контенту.
AAA (найвищий рівень)	контраст між текстом і фоном не менше ніж 7:1; переклад жестовою мовою; розширені аудіоописи.

У таблиці 4.1 наведено основні характеристики кожного стандарту. Стандарти A та AA, які є більш загальні та акцентують увагу на звичайних користувачах, а стандарт AAA зосереджений на людей з обмеженими можливостями.

Під час розробки проєкту було проведено дослідження створеної системи на відповідність деяким критеріям WCAG 2.2, а саме контрастність та шрифт.

Перевірка цих критеріїв була зроблена за допомогою сайта: https://accessibleweb.com/color-contrast-checker/, результати були зроблені у вигляді таблиці 4.2.

Таблиця 4.2 – Відповідність критеріям WCAG за кольором

Назва кольору	Контрастність	Колір фона
#009D22	3.56:1	#ffffee
#006ce0	4.93:1	#ffffee
#e00051	4.85:1	#ffffee
#000000	20.79:1	#ffffee
#ffffff	14.54:1	#262931
#eeeeee	12.54:1	#262931

Таким чином було проведено дослідження контрастності UI-компонентів та тексту. У таблиці 4.2 показано, що один з обраних кольорів має контрастність 3.56:1, це є малим показником для тексту, але це колір UI-компонента, а саме кнопки «Add user» та «Edit», для таких компонентів така контрастність є нормальною (рис. 4.4).

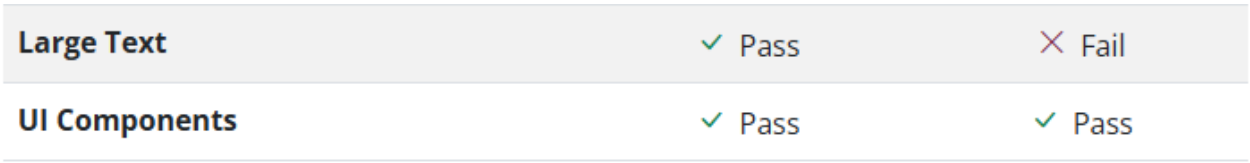

Рисунок 4.4 – Перевірка на відповідність контрастності для UI-компонентів

Контрастність тексту є 20.79:1 та 14.54:1, які підходять за усіма параметрами.

У результаті було виявлено, що усі кольори збігаються з критеріями WCAG 2.2. Це у свою чергує покращує видимість програми для людей з плохим зором.

5 ПРАКТИЧНА РЕАЛІЗАЦІЯ

5.1 Розмежування прав доступу

Розмежування прав доступу є основним рішенням для впровадження безпеки у програмі. Такий інструмент дозволяє вирішити питання помилок між користувачами. Система управління завдання має містити можливість додавання нових користувачів та надання їм паролю та імені, щоб вони могли авторизуватися у систему. Для уникнення помилок між користувачами було прийнято рішення надання цієї функції тільки одному користувачу. Тобто інші не будуть бачити цю конфіденційну інформацію та не зможуть викрасти дані користувачів.

У системі управління завданнями було реалізовано дві категорії користувачів: адміністратор та звичайний користувач. Для розуміння прав адміністратора було створено схему варіантів дій адміністратора (рис. 5.1).

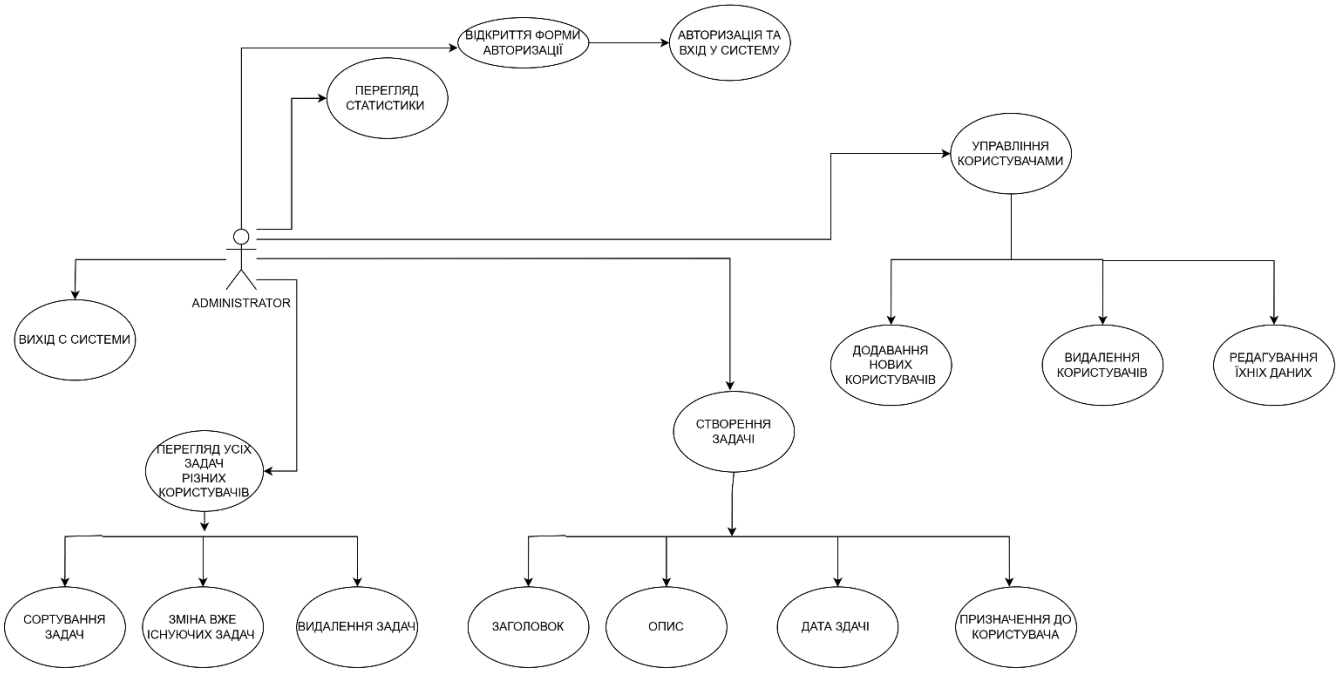

Рисунок 5.1 – Схема варіантів дій адміністратора у системі

Адміністратору відкрито повний функціонал програми, його головною задачею є: додавання нових користувачів та призначення їм задач. Тільки адміністратор має доступ до цих функцій.

Другою категорією користувачів є звичайний користувач, це може бути рядовий розробник у IT-відділу. Варіанти дій, які йому доступні було зображено на схемі варіантів дій (рис. 5.2).

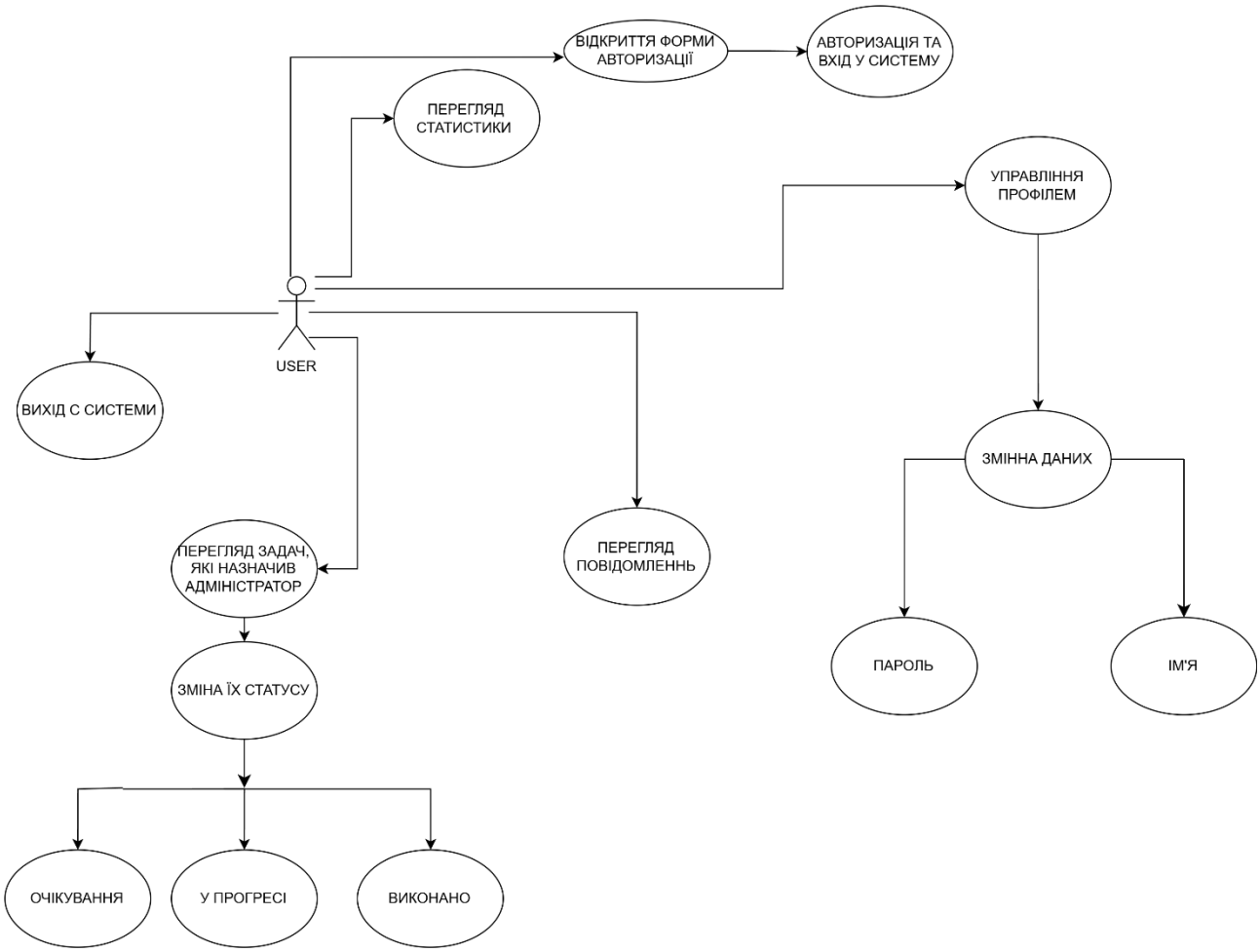

Рисунок 5.2 – Схема варіантів дій звичайного користувача у системі

Задачею користувача є виконання наданих задач адміністратором та зміна їх статусу у процесі для того, щоб було зрозуміло на якому етапі знаходиться кожний працівник IT-відділу. Головна особливість цієї категорії користувачів це неможливість змінювати дані інших користувачів та система повідомлень, яка сповіщає про нові задачі.

Таке розподілення на категорії робить програму зрозумілою та безпечною, а її використання простим, так як є чітке розподілення задач та обов'язків між адміністратором та іншими користувачами.

5.2 Процес авторизації у системі

Для підвищення захисту даних користувачів та розмежування прав доступу було реалізовано авторизацію у системі (рис. 5.3).

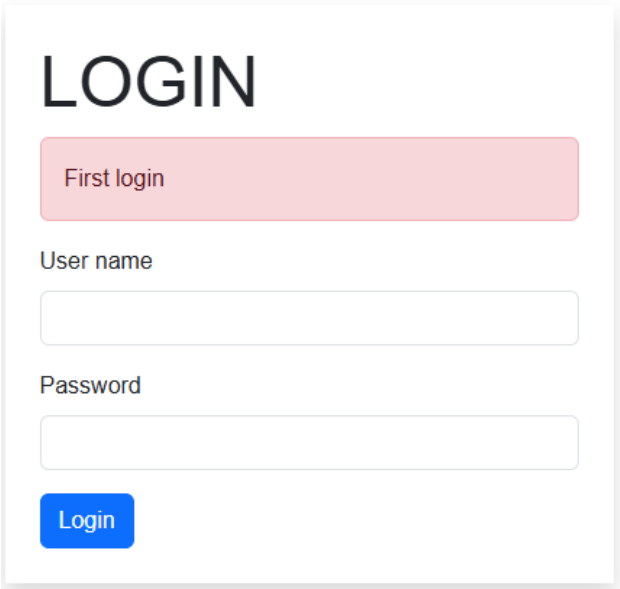

Рисунок 5.3 – Форма авторизації

Для вдалої авторизації потрібно ввести правильне ім'я користувача та пароль, який адміністратор видає своїй команді. При некоректно ведених даних користувач отримує попередження про це (рис. 5.4)

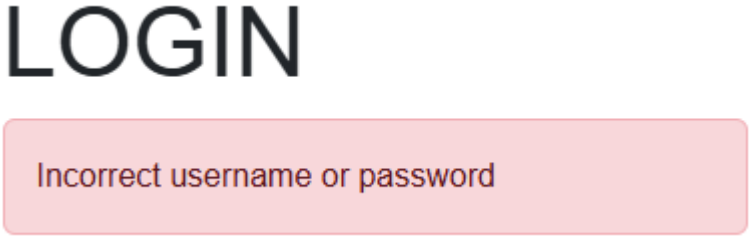

Рисунок 5.4 – Невдала спроба авторизації

Фрагмент коду, який показує процес авторизації у систему (лістинг 5.1).

Лістинг 5.1 – Login.php

```
$sql = "SELECT * FROM users WHERE username = ?";
    $stmt = $conn->prepare($sql);
    $stmt->execute([$user_name]);

    if ($stmt->rowCount() == 1) {
      $user = $stmt->fetch();
      $usernameDb = $user['username'];
      $passwordDb = $user['password'];
      $role = $user['role'];
      $id = $user['id'];

      if ($user_name === $usernameDb) {
         if (password_verify($password, $passwordDb)) {
           if ($role == "admin") {
            $_SESSION['role'] = $role;
            $_SESSION['id'] = $id;
            $_SESSION['username'] = $usernameDb;
            header("Location: ../index.php");
           }else if ($role == 'employee') {
            $_SESSION['role'] = $role;
            $_SESSION['id'] = $id;
            $_SESSION['username'] = $usernameDb;
            header("Location: ../index.php");
           }else {
            $em = "Unknown error occurred ";
            header("Location: ../login.php?error=$em");
            exit();
           }
         }else {
           $em = "Incorrect username or password ";
           header("Location: ../login.php?error=$em");
           exit();
```

Продовження лістингу 5.1

```
            }
          }else {
              $em = "Incorrect username or password ";
          header("Location: ../login.php?error=$em");
          exit();
          }
        }
    }
}else {
    $em = "Unknown error occurred";
    header("Location: ../login.php?error=$em");
    exit();
}
```

Код працює наступним чином. Спочатку виконується пошук користувача за введеним ім'ям у базі даних. Якщо користувач знайдений, відбувається перевірка пароля за допомогою *«password_verify()»*. Якщо пароль правильний, створюються сесійні змінні з інформацією про роль, ID та ім'я користувача. Залежно від ролі користувача (адміністратор чи звичайний користувач) він перенаправляється на головну сторінку. Якщо виникають помилки (неправильний логін або пароль, або інші), користувач отримує повідомлення про помилку (рис. 5.3-5.4).

5.3 Функція меню «Dashboard»

Після успішного входу до системи одразу відкривається меню dashboard. Така опція присутня у адміністратора та у звичайних користувачів.

Dashboard є інформативним, так як містить чисельну інформаційну про задачі та користувачів, надання такої інформації прискорює навігацію у програмі. Також присутня можливість відслідковування усіх можливих статусів

задач, що також полегшує подальше розподілення задач. Ця опція працює у режимі реального часу та оновлюється при зміні параметрів (рис. 5.5).

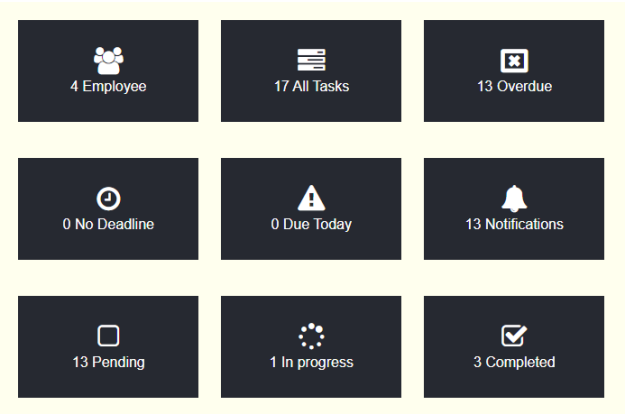

Рисунок 5.5 – Меню dashboard у адміністратора

Крім того така функція доступна для звичайних користувачів, але з іншою кількістю параметрів (рис. 5.6).

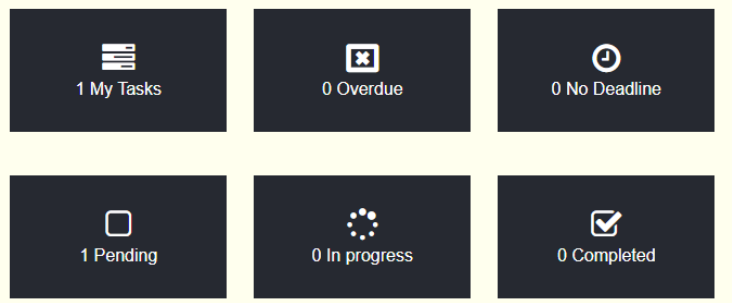

Рисунок 5.6 –Меню dashboard для звичайних користувачів

Для звичайних користувачів програми ця функція є теж важливою, тому що дозволяє одразу зрозуміти обсяг подальшої роботи та у якому статусі вона знаходиться. Зручно використовувати її для моніторингу кількості задач наданих адміністратором.

Фрагмент коду, який показує процес реалізації цієї функції (лістинг 5.2).

Лістинг 5.2 – Index.php

```
if (isset($_SESSION['role']) && isset($_SESSION['id']) ) {
  include "DB_connection.php";
```

Продовження лістингу 5.2

```
include "app/Model/Task.php";
include "app/Model/User.php";
```

Після перевірки, що користувач авторизований відбувається підключення файлів за параметрами бази даних та підключення файлів моделі (лістинг 5.3).

Лістинг 5.3 – Index.php

```
if ($_SESSION['role'] == "admin") {
  $todaydue_task = count_tasks_due_today($conn);
  $overdue_task = count_tasks_overdue($conn);
  $nodeadline_task = count_tasks_NoDeadline($conn);
  $num_task = count_tasks($conn);
  $num_users = count_users($conn);
  $pending = count_pending_tasks($conn);
  $in_progress = count_in_progress_tasks($conn);
  $completed = count_completed_tasks($conn);
}else {
  $num_my_task = count_my_tasks($conn, $_SESSION['id']);
  $overdue_task = count_my_tasks_overdue($conn,
$_SESSION['id']);
  $nodeadline_task = count_my_tasks_NoDeadline($conn,
$_SESSION['id']);
  $pending = count_my_pending_tasks($conn, $_SESSION['id']);
  $in_progress = count_my_in_progress_tasks($conn,
$_SESSION['id']);
  $completed = count_my_completed_tasks($conn, $_SESSION['id']);
}
```

У цьому фрагменті було реалізовано розподілення за ролями, так як ця функція відображає різну статистику. Якщо користувач – адміністратор, система збирає статистику по всіх задачах та користувачах (наприклад, скільки задач з дедлайном на сьогодні, скільки прострочених і т.д.). Якщо це звичайний користувач, то збирається статистика лише по його власних задачах. Результати

передаються у змінних типу: *«$overdue_task»* для кожного виду користувача. Це дозволяє відобразити відповідну інформацію у дашборді з урахуванням прав доступу.

5.4 Функція меню «Manage Users»

Було створено меню, яке дозволяє додавати нових користувачів та управляти їх даними. Доступ до цього меню є тільки у адміністратора. Такі функції є важливими, так як дають можливість управління користувачами та їх даними.

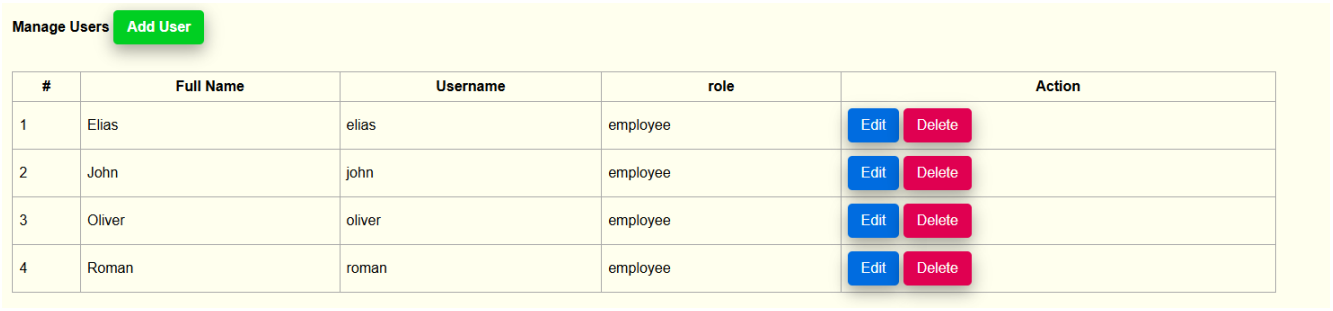

Рисунок 5.7 –Загальний вигляд меню Manage Users

У вкладці *«Manage Users»* знаходиться вся інформація про користувачів та їх дані. При натисканні на кнопку «Edit» відкривається форма з інформацією про користувача, а саме його ім’я та пароль. Основною функцією цього меню є додавання нових користувачів за допомогою кнопки *«Add User»* (рис. 5.8).

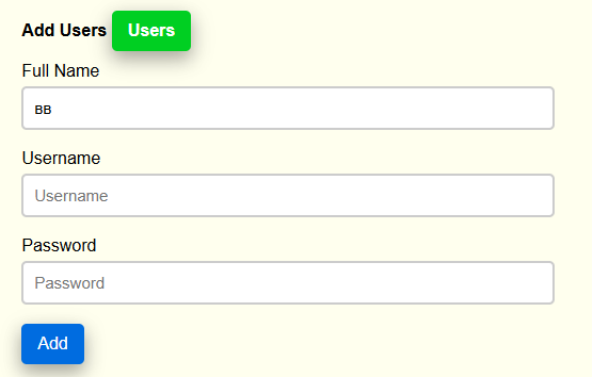

Рисунок 5.8 – Додавання нового користувача

Фрагмент коду, яким було реалізовано такі функції (лістинг 5.4).

Лістинг 5.4 – Add-user.php

```
$user_name = validate_input($_POST['user_name']);
$password = validate_input($_POST['password']);
$full_name = validate_input($_POST['full_name']);

if (empty($user_name)) {
 $em = "User name is required";
  header("Location: ../add-user.php?error=$em");
  exit();
}else if (empty($password)) {
 $em = "Password is required";
  header("Location: ../add-user.php?error=$em");
  exit();
}else if (empty($full_name)) {
 $em = "Full name is required";
  header("Location: ../add-user.php?error=$em");
  exit();
}else {

  include "Model/User.php";
  $password = password_hash($password, PASSWORD_DEFAULT);

  $data = array($full_name, $user_name, $password, "employee");
  insert_user($conn, $data);

  $em = "User created successfully";
  header("Location: ../add-user.php?success=$em");
  exit();
}
```

Цей фрагмент відповідає за додавання нового користувача до системи. Спочатку виконується перевірка наявності обов’язкових полів форми – імені

користувача, пароля та повного імені. Якщо одне з полів не заповнено, спрацьовує система обробки помилок: користувача перенаправляє назад на форму з відповідним повідомленням про відсутні дані (рис. 5.9). Якщо всі дані введено коректно, пароль хешується для забезпечення безпеки, після чого інформація записується до бази даних за допомогою функції *«insert_user»*. Після успішного додавання нового користувача виводиться повідомлення: *«User created successfully»*.

Рисунок 5.9 – Приклад помилки при не введеному паролі

5.5 Функція меню «Create Task»

Задачею цього меню є створення задачі та призначення її до відповідного працівника IT-відділу. Вона включає ключові функції, які дозволять створити задачу зрозумілою для користувачів, а саме: створення заголовка, опису задачі, призначення дедлайну та кому призначити задачу (рис. 5.10). Ця функція є тільки на стороні адміністратора.

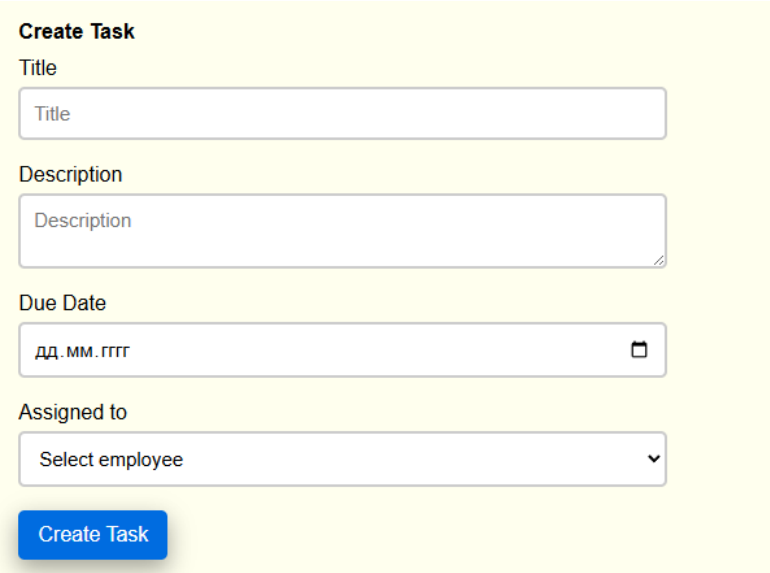

Рисунок 5.10 – Меню Create Task

Додавання задачі є основною функцією цієї програми, так як дозволяє прискорити вирішення різних IT – проектів. Правильно розподілені задачі між працівниками надають можливість відстеження на якому етапі знаходиться проект та будувати подальшу стратегію розподілення праці та її оптимізації.

Фрагмент коду через який було реалізовано цю функцію (лістинг 5.5).

Лістинг 5.5 – Task.php

```
function insert_task($conn, $data){
  $sql = "INSERT INTO tasks (title, description, assigned_to,
due_date) VALUES(?,?,?,?)";
  $stmt = $conn->prepare($sql);
  $stmt->execute($data);
}
```

Функція *«insert_task»* відповідає за додавання нової задачі до бази даних.. Вона приймає два параметри, які відповідають за з’єднання з базою та масиву, який містить значення для полів на рисунку 5.10. Тут було використано підготовлений запит для захисту від SQL-ін’єкцій. Для підстановки значень з масиву було обрано метод *«execute($data)»*.

5.6 Функція меню «All Tasks»

Під час роботи у команді виникає необхідність створення функції перегляду усіх задач, тому що це допомагає зрозуміти хто на якому етапі виконання знаходиться. Для вирішення цього питання було створено вкладку меню *«All Tasks»* (рис. 5.10).

#	Title	Description	Assigned To	Due Date	Status	Action
1	Програмування функціоналу (бекенд + фронтенд)	Реєстрація, авторизація, кошик, пошук, фільтри	Roman	2025-05-23	pending	Edit Delete
2	Верстка сторінок (HTML/CSS/JS)	Адаптивна верстка, кросбраузерна сумісність	Elias	2025-07-06	pending	Edit Delete
3	Інтеграція з базою даних	SQLite	John	2026-06-07	pending	Edit Delete
4	Підключення сторонніх сервісів	Платіжні системи, CRM, чат-боти, Google Analytics	Elias	2025-07-07	pending	Edit Delete
5	Тестування сайту	Юзабіліті, безпека, навантаження, перевірка помилок	John	2024-05-18	pending	Edit Delete
6	Розгортання сайту на сервері / хостингу	Налаштування домену, HTTPS, оптимізація	Elias	2024-09-06	pending	Edit Delete

Рисунок 5.10 – Вкладка меню All Tasks

На рисунку 5.10 зображено усі створені задачі з можливістю їх редагування. Часто виникає проблема коли задач багато і їх стає важко відстежувати та редагувати. Для вирішення цього питання було реалізовано сортування задач (рис. 5.11).

#	Title	Description	Assigned To	Due Date	Status	Action
1	Програмування функціоналу (бекенд + фронтенд)	Реєстрація, авторизація, кошик, пошук, фільтри	Roman	2025-05-23	pending	Edit Delete
2	Тестування сайту	Юзабіліті, безпека, навантаження, перевірка помилок	John	2024-05-18	pending	Edit Delete
3	Розгортання сайту на сервері / хостингу	Налаштування домену, HTTPS, оптимізація	Elias	2024-09-06	pending	Edit Delete

Рисунок 5.11 – Приклад роботи відображення задач за сортуванням

Наприклад, може виникнути ситуація, коли потрібно дізнатися у яких задач закінчився строк виконання і вони не були виконанні. За допомогою сортування це можливо зробити швидко та просто.

Фрагмент коду, яким було реалізовано сортування (лістинг 5.6).

Лістинг 5.6 – Task.php

```
function get_all_tasks_overdue($conn){
  $sql = "SELECT * FROM tasks WHERE due_date < CURDATE() AND
status != 'completed' ORDER BY id DESC";
  $stmt = $conn->prepare($sql);
 $stmt->execute([]);
```

Продовження лістингу 5.6

```
if($stmt->rowCount() > 0){
    $tasks = $stmt->fetchAll();
  }else $tasks = 0;

  return $tasks;
}
```

Було реалізовано SQL – запит з умовами, які вибирають задачі із датою дедлайну раніше поточної дати та дозволяють пропустити задачі, які вже були виконанні, потім було виконано сортування задач за спаданням ID. Якщо задачі з даним параметром знайдені за допомогою *«fetchAll»* вони будуть відображенні.

5.7 Вкладка меню «Notifications»

Під час роботи над проектами розробник може забути про деякі задачі або не побачити, що вони були додані, це в свою чергу призводить до проблем та непорозумінь під роботи над проектом. Система сповіщень є одним з варіантів вирішення цього питання. Така функція була реалізована у проекті(рис. 5.12).

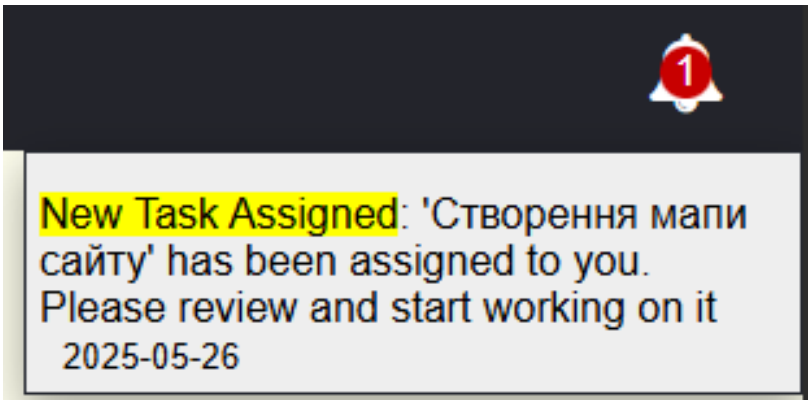

Рисунок 5.12 – Система сповіщень

Такий вигляд сповіщень привертає увагу до задач, які були додані адміністратором та мінімізує ризики бути залишеним без уваги. Після

натискання на поле, де написано про додану задачу відкривається меню *«Notifications»* (рис. 5.13).

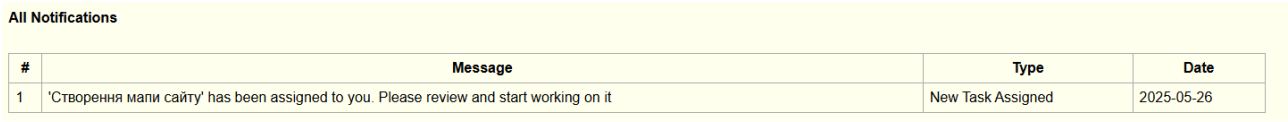

Рисунок 5.13 – Меню notifications

Це меню дублює вже існуючу інформацію у вигляді таблиці. Це потрібно для відстеження кількості задач та коли вони були додані.

Фрагмент коду, яким було реалізовано вставку нового повідомлення (лістинг 5.7).

Лістинг 5.7 – Notification.php

```
function insert_notification($conn, $data){
  $sql = "INSERT INTO notifications (message, recipient, type)
VALUES(?,?,?)";
  $stmt = $conn->prepare($sql);
  $stmt->execute($data);
}
```

В даному фрагменті коду відбувається робота з базою даних *«notifications»,* та створюється SQL-запит (лістинг 5.8).

У наступному фрагменті була реалізована можливість отримання всіх повідомлень користувача. Після створення SQL – запиту було створено умову, яка перевіряє чи є результати та виводить їх (лістинг 5.8).

Лістинг 5.8 – Notification.php

```
function get_all_my_notifications($conn, $id){
  $sql = "SELECT * FROM notifications WHERE recipient=?";

  $stmt = $conn->prepare($sql);
```

Продовження лістингу 5.8

```
  $stmt->execute([$id]);

  if($stmt->rowCount() > 0){
   $notifications = $stmt->fetchAll();
  }else $notifications = 0;
  return $notifications;
}
```

Функція, яка підраховує кількість непрочитаних сповіщень для користувача з певним ID та виконує вибірку повідомлень із статусом *«is_read=0»* була реалізована у наступному фрагменті (лістинг 5.9).

Лістинг 5.9 – Notification.php

```
function count_notification($conn, $id){
  $sql = "SELECT id FROM notifications WHERE recipient=? AND
is_read=0";
  $stmt = $conn->prepare($sql);
  $stmt->execute([$id]);

  return $stmt->rowCount();
}
```

5.8 Вкладка меню «My task»

Меню *«My Task»* є основним інтерфейсом для розробника IT-відділу. Він надає інформацію про задачі та їх опис (рис. 5.14).


My Tasks

#	Title	Description	Status	Due Date	Action
1	Програмування функціоналу (бекенд + фронтенд)	Реєстрація, авторизація, кошик, пошук, фільтри	pending	2025-05-23	Edit
2	Створення мапи сайту	За допомогою програми Miro	pending	2025-05-30	Edit

Рисунок 5.14 – Меню My task

Розробник бачить свої задачі та може редагувати їх статус по мірі виконання, що дозволяє зрозуміти адміністратору на якому етапі знаходиться реалізація той чи іншої задачі.

Фрагмент коду, яким було реалізовано зміну статусу користувачем (лістинг 5.10).

Лістинг 5.10 – Update-task-employee

```
  if (empty($status)) {
   $em = "status is required";
    header("Location: ../edit-task-
employee.php?error=$em&id=$id");
    exit();
  }else {

    include "Model/Task.php";

    $data = array($status, $id);
    update_task_status($conn, $data);

    $em = "Task updated successfully";
    header("Location: ../edit-task-
employee.php?success=$em&id=$id");
    exit();


  }
```

Використовуючи «*(empty($status))»* йде перевірка на введення нового значення статусу, якщо це поле не заповнене, користувач буде повідомлений про помилку. Якщо поле заповнене то буде підключений відповідний файл з логікою задач та викликана функція *«update_task_status»,* яка оновлює статус задачі у базі даних. Після успішного оновлення користувач буде повідомлений про це.

5.9 Вкладка меню «Profile»

Під час користування програмою у користувачів може виникнути бажання змінити свої дані, наприклад, коли пароль, який був використано для входу був викрадений через виток даних. Також можливий сценарій, коли у розробника було змінено ім’я. Щоб уникнути ці проблеми було реалізовано меню з відповідним функціоналом (рис. 5.15).

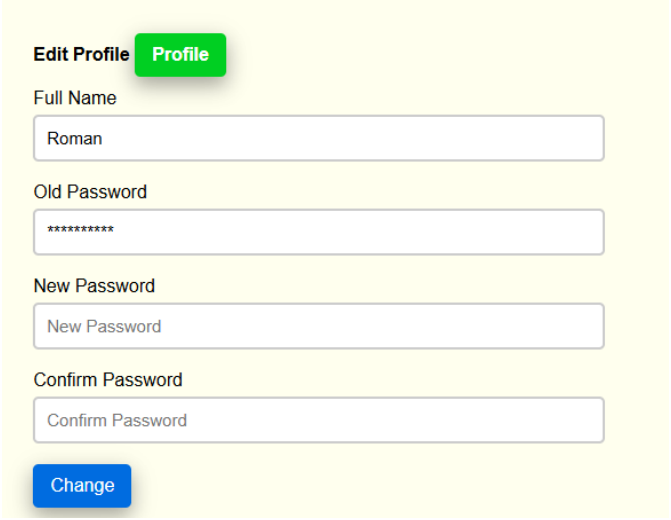

Рисунок 5.15 – Меню Profile

У цьому меню можливо змінити свої дані, а саме пароль та ім’я. Такий функціонал покращує захист програми та запобігає злому даних користувача.

Фрагмент коду, яким було реалізовано оновлення даних профілю (лістинг 5.11).

Лістинг 5.11 – Update-user

```
if (empty($user_name)) {
 $em = "User name is required";
  header("Location: ../edit-user.php?error=$em&id=$id");
  exit();
}else if (empty($password)) {
```

Продовження лістингу 5.11

```
$em = "Password is required";
    header("Location: ../edit-user.php?error=$em&id=$id");
    exit();
  }else if (empty($full_name)) {
   $em = "Full name is required";
    header("Location: ../edit-user.php?error=$em&id=$id");
    exit();
  }else {
    include "Model/User.php";
    $password = password_hash($password, PASSWORD_DEFAULT);

    $data = array($full_name, $user_name, $password, "employee",
$id, "employee");
    update_user($conn, $data);

    $em = "User created successfully";
    header("Location: ../edit-user.php?success=$em&id=$id");
    exit();

  }
```

У цьому фрагменті йде перевірка заповнення полів, якщо одне з полів не було заповнено, користувач буде повідомлений про це. Після правильного заповнення полів виконується хешування пароля за допомогою *«password_hash»* та формується масив, який передається у функцію *«update_user»* , що оновлює дані користувача у базі даних.

Таким чином було створено систему, яка виконує такі функції як: додавання задач, зміна їх статусу, призначення задач до користувачів, можливість моніторингу роботи працівників через систему статусів, зміна даних користувачів з боку адміністратора та користувача, створення систему повідомлення. Усі наведені функції покращують комунікацію під час розгортання проєктів та оптимізують час роботи.

ВИСНОВКИ

В результаті проведеної роботи було проаналізовано особливості систем управління задач та визначено їх основний функціонал до якого належать додавання задач, зміна їх статусу та моніторинг.

Системи управління завданнями є важливими інструментами для забезпечення ефективної праці та оптимізації робочого процесу під час розробки проектів. Такі системі можливо використовувати у будь-яких напрямках програмування, де є необхідність моніторингу задач та їх вчасного виконання.

У межах дослідження було розглянуто можливі підходи до реалізації клієнт-серверних архітектур зокрема двокаскадної, трьохкаскадної та багатокаскадної, а також принципи роботи вебсерверів.

Проведено аналіз популярних систем управління завданнями, що використовуються в IT-сфері та виявлено основні функції таких систем.

Досліджено тенденції розвитку IT-технологій в умовах цифрової трансформації та виявлено, що за останні роки в Україні активно зросла кількість IT-фахівців та IT-відділів, що підтверджує актуальність та важливість створення систем управління завданнями.

Розглянуто актуальні мови програмування, що дозволяють зробити систему довготривалою з можливістю подальшого вдосконалення та реалізації нових функцій.

Опрацьовано важливий аспект розробки додатків, а саме UI/UX, який безпосередньо впливає на те, як користувачі взаємодіють з продуктом. У результаті було створено мапу додатку та Wireframe, які допомогли зробити інтерфейс програми інтуїтивно простим та зрозумілим.

У результаті роботи розроблено клієнт-серверний додаток, який складається з клієнтської частини, яка відповідає за інтерфейс користувача та серверної частини, яка відповідає за обробку запитів. Цей додаток покращує комунікацію під час розгортання проєктів та оптимізує час роботи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Client-Server Architecture - Advantages and Disadvantages - KITRUM. KITRUM. URL: https://kitrum.com/blog/client-server-architecture-advantages-and-disadvantages/ (дата звернення: 03.06.2025).

2. Клієнт-серверна архітектура. Онлайн-курси від компанії QATestLab | Головна сторінка. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/ (дата звернення: 03.06.2025).

3. Sulyman, S. Client-Server Model [Електронний ресурс]. – URL: https://www.researchgate.net/profile/ShakiratSulyman/publication/271295146_Client Server_Model/links/5864e11308ae8fce490c1b01/Client-Server-Model.pdf (дата звернення: 03.06.2025).

4. HTTP: The Definitive Guide [Електронний ресурс]. – URL: https://books.google.com.ua/books?hl=ru&lr=&id=3EybAgAAQBAJ&oi=fnd&pg=PR5 (дата звернення: 03.06.2025).

5. What is a web server? [Електронний ресурс] // MDN Web Docs. – URL: https://developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Web_mechanics/What_is_a_web_server (дата звернення: 03.06.2025).

6. Laurie, B., & Laurie, P. Apache: The Definitive Guide. – O'Reilly Media, 2003.

7. Web server definition [Електронний ресурс] // TechTarget. – URL: https://www.techtarget.com/whatis/definition/Web-server (дата звернення: 03.06.2025).

8. Apache web server [Електронний ресурс]. – URL: https://httpd.apache.org/ (дата звернення: 03.06.2025).

9. Apache Friends – XAMPP [Електронний ресурс]. – URL: https://www.apachefriends.org/index.html (дата звернення: 03.06.2025).

10. Stack Overflow Developer Survey 2023 [Електронний ресурс]. – URL: https://survey.stackoverflow.co/2023/ (дата звернення: 03.06.2025).

11. Atlassian Jira [Електронний ресурс] // Atlassian Software. – URL: https://www.atlassian.com/software/jira (дата звернення: 03.06.2025).

12. Linear – The issue tracking tool you'll enjoy using [Електронний ресурс] // Linear. – URL: https://linear.app/ (дата звернення: 03.06.2025).

13. Back-End Developer: What It Is, and How to Become One [Електронний ресурс]. – URL: https://www.coursera.org/articles/back-end-developer (дата звернення: 03.06.2025).

14. WHATWG. HTML Living Standard [Електронний ресурс]. – URL: https://html.spec.whatwg.org/multipage/ (дата звернення: 03.06.2025).

15. Що таке HTML? [Електронний ресурс] // CSS.in.ua. – URL: https://css.in.ua/article/shcho-take-css_3 (дата звернення: 03.06.2025).

16. CSS: Cascading Style Sheets [Електронний ресурс]. – URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 16.04.2025).

17. Flanagan, D. JavaScript: The Definitive Guide [Електронний ресурс] / David Flanagan. – 6th ed. – Sebastopol: O'Reilly Media, 2011. – 1096 p. – URL: https://books.google.com.ua/... (дата звернення: 17.04.2025).

18. PHP Programming Language [Електронний ресурс]. – URL: https://books.google.com.ua/books?hl=ru&lr=&id=h-E1lVko-skC&oi=fnd&pg=PR5 (дата звернення: 03.06.2025).

19. Manger, M. Mastering phpMyAdmin 3.4 for Effective MySQL Management. – Packt Publishing, 2011.

20. Що таке база даних та для чого вона потрібна [Електронний ресурс] // Cityhost.ua. – URL: https://cityhost.ua/uk/blog/scho-take-baza-danih-ta-dlya-chogo-vona-potribna.html (дата звернення: 03.06.2025).

21. MariaDB vs MySQL [Електронний ресурс]. – URL: https://changestar.co.uk/app/uploads/2014/06/mariadb-vs-mysql.pdf (дата звернення: 03.06.2025).

22. Frameworks in Programming Languages [Електронний ресурс] // ITStep. – URL: https://cloud.itstep.org/blog/frameworks-in-programming-languages-what-are-they-for-and-how-to-choose-them#5 (дата звернення: 03.06.2025).

23. What is Bootstrap? [Електронний ресурс] // Hostinger Tutorials. – URL: https://www.hostinger.com/tutorials/what-is-bootstrap (дата звернення: 03.06.2025).

24. Norman, Donald A. *The Design of Everyday Things*. Revised and expanded ed., Basic Books, 2013.

25. Wijaya, E. Y., Arif, M., Aini, N., & Putri, Y. N. UI/UX Web Based Learning Design with UCD Approach to Basic Programming using FIGMA. // bit-Tech, 2024, 6(3), 412–420. – URL: https://doi.org/10.32877/bt.v6i3.1534 (дата звернення: 03.06.2025).

26. Figma: The Collaborative Interface Design Tool [Електронний ресурс] // Figma. – URL: https://www.figma.com/ (дата звернення: 03.06.2025).

27. Скільки розробників в Україні у 2025 році? [Електронний ресурс] // DOU.ua. – URL: https://dou.ua/lenta/articles/how-many-devs-in-ukraine-2025/ (дата звернення: 03.06.2025).

28. What is a wireframe? [Електронний ресурс] // CareerFoundry. – URL: https://careerfoundry.com/en/blog/ux-design/what-is-a-wireframe-guide/#what-is-a-wireframe (дата звернення: 03.06.2025).

29. Web Content Accessibility Guidelines (WCAG) 2.2 [Електронний ресурс] // W3C. – URL: https://www.w3.org/TR/WCAG22/ (дата звернення: 03.06.2025).

30. What is WCAG? [Електронний ресурс] // WCAG. – URL: https://www.wcag.com/resource/what-is-wcag/ (дата звернення: 03.06.2025).