ХЕРСОНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ
КАФЕДРА ПРОГРАМНИХ ЗАСОБІВ І ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему:

«ДОСЛІДЖЕННЯ ТА ПРОЄКТУВАННЯ КОРИСТУВАЦЬКОГО СЕРЕДОВИЩА ДЛЯ ПІДТРИМКИ УПРАВЛІННЯ ПРОЄКТНОЮ ДІЯЛЬНІСТЮ»

Виконав: здобувач 6 курсу, групи 6ПР
спеціальності 121 «Інженерія
програмного забезпечення»
Якимчук Ганна Богданівна
(прізвище та ініціали)

Керівник: к.т.н., доцентка Огнєва О.Є.
(прізвище та ініціали)

Рецензент к.т.н., доц. Григорова А.А.
(прізвище та ініціали)

Хмельницький - 2025 р.

Факультет Інформаційних технологій та дизайну
Кафедра Програмних засобів і технологій
Освітньо-кваліфікаційний рівень магістр
Галузі знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійної програми «Програмна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗіТ,
професор О.Є.Огнєва
«____» _____ 2025 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ ЗДОБУВАЧУ
Якимчук Ганні Богданівні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та проєктування користувацького середовища для підтримки управління проєктною діяльністю

керівник роботи Огнєва Оксана Євгенівна к.т.н., доцентка ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ХНТУ від «15» вересня 2025 року № 417-с.
2. Строк подання здобувачем роботи _ грудня 2025 року
3. Вихідні дані до роботи Матеріали навчально-виробничої практики –

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналітичний огляд джерел та методологій управління проєктами, системний аналіз і обґрунтування проблеми, методи та засоби проєктування прототипу, проєктування та реалізація прототипу.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) ____

6. Консультанти розділів роботи

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

7. Дата видачі завдання 1 жовтня 2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних тем кваліфікаційних робіт	01.10.2025 - 05.10.2025	Готово
2	Аналітичний огляд джерел та методологій управління проєктами	06.10.2025 - 20.10.2025	Готово
3	Системний аналіз і обґрунтування проблеми	21.10.2025 - 31.10.2025	Готово
4	Методи та засоби проєктування прототипу	01.11.2025 - 10.11.2025	Готово
5	Проєктування та реалізація прототипу системи	11.11.2025 - 21.11.2025	Готово
6	Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки	24.11.2025 - 30.11.2025	Готово
7	Перевірка на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів.	01.12.2025 - 12.12.2025	
8	Підготовка до захисту та захист роботи	15.12.2025	

здобувач _____ Якимчук Г.Б.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Огнєва О.Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Магістерська кваліфікаційна робота містить: 91с., 15 рис.,18 табл., 57 джерел.

Об’єкт дослідження - процес проектування та використання користувацьких середовищ в інформаційних системах управління проєктною діяльністю.

Предмет дослідження - принципи, методи та засоби проєктування користувацького середовища для систем управління проєктами.

Метою дослідження є розробка і обґрунтування моделі користувацького середовища для підтримки управління проєктною діяльністю, що сприятиме підвищенню ефективності планування, контролю та координації проєктів.

У роботі проведено аналітичний огляд сучасних методологій управління проєктами та існуючих програмних рішень. Виявлено проблему когнітивного навантаження користувачів у складних системах. На основі системного аналізу та методології людино-орієнтованого проєктування спроєктовано архітектуру та структуру бази даних програмного прототипу. Розроблено веб-застосунок з використанням технологій PHP, MySQL, JavaScript, який реалізує концепцію адаптивного роле-орієнтованого інтерфейсу. Проведено верифікацію та валідацію розробленого рішення, що підтвердили зниження візуального шуму та підвищення зручності роботи для різних ролей користувачів.

Ключові слова: УПРАВЛІННЯ ПРОЄКТАМИ, КОРИСТУВАЦЬКЕ СЕРЕДОВИЩЕ, АДАПТИВНИЙ ІНТЕРФЕЙС, WEB-ЗАСТОСУНОК, РОЛЕВИЙ ДОСТУП.

ANNOTATION

The Master’s qualification paper contains: 91 pp., 15 pic., 18 tabl., 57 sources.

The object of research is the process of designing and using user environments in project management information systems.

The subject of research is the principles, methods, and tools for designing a user environment for project management systems.

The aim of the research is to develop and substantiate a model of a user environment to support project management activities, which will contribute to increasing the efficiency of project planning, control, and coordination.

The paper provides an analytical review of modern project management methodologies and existing software solutions. The problem of cognitive load on users in complex systems was identified. Based on system analysis and User-Centered Design methodology, the architecture and database structure of the software prototype were designed. A web application was developed using PHP, MySQL, and JavaScript technologies, which implements the concept of an adaptive role-based interface. Verification and validation of the developed solution were carried out, confirming the reduction of visual noise and increased usability for different user roles.

Keywords: PROJECT MANAGEMENT, USER ENVIRONMENT, ADAPTIVE INTERFACE, WEB APPLICATION, ROLE-BASED ACCESS.

ЗМІСТ

ВСТУП

В теперішній час, коли інформаційні технології активно використовуються і розвиваються, концепція проєктів поширилась за рамки IT сфери і зараз спостерігається майже в кожній сфері. Виходячи з цього ефективне управління проєктами зараз не просто допоміжна функція, а один з ключових факторів конкурентоспроможності для більшості компаній. Але, відповідно, щоб успішно і ефективно реалізовувати проєкти також існують гнучкі методології (Agile, Scrum, Kanban), які все більше і більше використовуються поза IT сферою.

Для полегшення впровадження цих методологій та їх використання існують спеціалізовані програмні системи підтримки управління проєктною діяльністю. Зараз ці системи не просто інструмент для постановки задач, відслідковування прогресу і тд., а комплексне середовище, яке допомагає забезпечувати прозоре планування, ефективну комунікацію, моніторинг ресурсів та аналіз результатів.

Не зважаючи на велику кількість як надзвичайно поширених, так і більш локальних рішень на ринку, велика кількість команд та проєктних менеджерів досить часто стикається з проблемами в їх використанні. Існує фундаментальне протиріччя: інструмент, який створено щоб спростити роботу та впорядкувати хаос, сам створює проблеми і незручності через свою складність чи не зрозумілість для звичайного користувача.

Прикладом є великі корпоративні системи (як Jira), які мають досить багато «надлишкової» функціональності, та високий поріг входження, що може перевантажити не підготовленого користувача ще до початку роботи і вимагає досить багато часу на навчання і адміністрування таких систем.

З іншого боку, інструменти, які більше орієнтуються на візуальну простоту (як Trello), досить часто мають значно менший функціонал, якого достатньо для простих проєктів, але для складних багатоетапних проєктів цього вже не достатньо і може виникнути потреба в додаткових сервісах, чого максимально хочеться уникати як і з економічної точки зору так і з практичної.

Дана проблема включає в себе як програмну інженерію, так і людино-комп’ютерну взаємодію. Її досліджувала велика кількість науковців з обох цих сфер, включаючи Дональда Нормана та Якоба Нільсена, які наголошували що ефективність таких застосунків для управління проєктами вимірюється не тільки в кількості доступного функціоналу, а й в часі, який середньостатистичний користувач витрачає на виконання свого ключового завдання в системі. Тому останнім часом розробка таких застосунків орієнтується не лише на автоматизацію процесів, а й на адаптивність користувацьких середовищ.

Таким чином, **актуальність** магістерської роботи полягає в зростаючій потребі створення високоефективного програмного середовища для управління проєктами, яке б поєднувало в собі достатню кількість програмного функціоналу та зручність в ознайомленні та використанні користувачем.

Необхідність наукового дослідження, обґрунтування та проєктування такого середовища, яке б знижувало когнітивне навантаження користувачів та підвищувало продуктивність проєктних команд, визначає наукову та практичну значущість даної роботи.

Метою дослідження є розробка і обґрунтування моделі користувацького середовища для підтримки управління проєктною діяльністю, що сприятиме підвищенню ефективності планування, контролю та координації проєктів.

Досягнення поставленої мети передбачає аналіз існуючих програмних засобів управління проєктами, визначення вимог до користувацького середовища, розроблення його структури та створення прототипу для перевірки працездатності запропонованих рішень.

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

- проаналізувати теоретичні основи управління проєктною діяльністю та сучасні методології;
- дослідити підходи до проєктування програмних систем та проаналізувати існуючі програмні засоби управління проєктами, визначити їхні переваги та недоліки;

- провести системний аналіз предметної області та обґрунтувати проблему розробки;
- визначити вимоги до користувацького середовища системи підтримки управління проєктною діяльністю;
- вибрати та обґрунтувати методи та засоби проєктування прототипу, включаючи архітектуру та технологічний стек;
- розробити концептуальну модель та спроєктувати структуру бази даних прототипу;
- створити прототип програмного засобу на основі розробленої моделі та реалізувати його основні функціональні модулі;
- оцінити ефективність запропонованих рішень шляхом верифікації та валідації прототипу.

Об’єкт дослідження - процес проектування і використання користувацьких середовищ в інформаційних системах управління проєктною діяльністю.

Предмет дослідження - принципи, методи та засоби проєктування користувацького середовища для систем управління проєктами.

Наукова новизна одержаних результатів дослідження полягає в нижчевказаних пунктах:

- удосконалено підходи до відображення і візуалізації життєвого циклу робочих та персональних задач, що базується на комбінації Kanban-дошок та діаграм Ганта в єдиному інтегрованому інтерфейсі, що, на відміну від існуючих рішень, дозволить одночасно відображати як роботу як проєктної команди, так і загальний графік проєкту;
- набула подальшого розвитку модель адаптивного користувацького середовища, яка може брати за основу ролі користувача в команді (менеджер, розробник, тестувальник) та надає відповідний функціонал та персоналізований набір інструментів, що дозволяє зменшити навантаження співробітників зайвим або нерелевантним функціоналом та підвищує фокус на виконанні актуальних задач за оптимальний час;

- удосконалено архітектуру програмного прототипу для управління проєктами (на базі клієнт-серверного підходу) шляхом інтеграції модулів персоналізації інтерфейсу без втрати загальної продуктивності системи.

Таким чином, запропоновані наукові положення спрямовані на вирішення актуальної наукової задачі підвищення ефективності програмних систем управління проєктами шляхом оптимізації їх користувацьких середовищ.

Теоретичне значення одержаних результатів полягає в тому, що результати дослідження розширюють теорію людино-машинної взаємодії, систематизуючи принципи та патерни проєктування для специфічної предметної області - інструментів для управління проєктами.

Практичне значення одержаних результатів демонструє що проєктні рішення, блок-схеми і алгоритми, програмний прототип та сформульовані рекомендації можуть бути використані як ІТ-компаніями так і будь якими користувачами, які потребують управління проєктами для:

- розробки нових або адаптації існуючих комерційних або внутрішніх систем управління проєктами;
- покращення та оптимізації користувацького досвіду доступних програмних продуктів з метою зниження часу на підготовку до роботи з системою і, відповідно, підняття продуктивності команд;
- використання як методичний матеріал для навчання майбутніх проєктних менеджерів та розробників;

Апробація результатів. Концепція роботи, окремі її аспекти та одержані узагальнення і висновки були оприлюднені на XV Всеукраїнській науково-практичній конференції «Актуальні проблеми комп’ютерних наук АПКН-2023».

Публікації:

1. Якимчук Г.Б. Програмний застосунок для керування проектами // Збірник наукових праць за матеріалами XV Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН». Хмельницький - С. 329-332.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ДЖЕРЕЛ ТА МЕТОДОЛОГІЙ УПРАВЛІННЯ ПРОЄКТАМИ

1.1. Аналіз сучасних методологій управління проєктною діяльністю

Рішення про вибір методології управління проєктом на самому старті є одним з найвагоміших та впливових як на робочий процес, так і на кінцеві результати роботи команди над проєктом незалежно від галузі. Тобто від правильно вибраної методології може напряму залежати успіх проєкту.

Методологія використовується не тільки для того щоб спланувати послідовність робіт, але й для організації комунікації, керування ризиками, та, найважливіше для дослідження, допомагає сформувати вимоги до програмного середовища, яке буде використовуватись для відслідковування та підтримки проєкту. Як можна побачити в існуючих дослідженнях, інструменти які розроблено для слідування та підтримки однієї методології часто бувають не ефективними для використання інших[46].

Першою існуючою методологією, яку названо класичною є водоспадна модель (Waterfall). Підходи використані в цій методології в основному були запозичені з інженерних галузей, таких як будівництво, і характеризуються чітким порядком виконання задач, одна за одною: аналіз вимог, проєктування, реалізація, тестування та впровадження. Кожна наступна фаза може початись виключно після завершення попередньої.

Основними перевагами водоспадної моделі є чіткість, структурованість та простота контролю завдяки чітко визначеним етапам та очікуваним результатам. Але виходячи з цього з’явився її головний недолік - негнучкість, що ставало все більше і більше критичною проблемою для IT-галузі. Через постійні зміни вимог навіть в процесі розробки неможливість повернутись до попереднього етапу без втрат часу і ресурсів стала серйозним блокером для релізу актуального продукту, який би відповідав всім вимогам, і успіху проєкту загалом [6].

З ростом проблеми негнучкості класичних методологій на початку 2000-х років почали з’являтись гнучкі (Agile) методології. Основні принципи цих методологій було викладено в «Маніфесті гнучкої розробки програмного забезпечення» (Agile Manifesto).

Такий підхід змінив саму концепцію та пріоритети управління проєктами поставивши цінність людей та взаємодії вищою за процеси та інструменти; працюючий продукт - вищим за вичерпну документацію; співпрацю з замовником - вищою за узгодження контрактів; та готовність до змін - вищою за дотримання початкового плану [42]. Тобто зміщено основний фокус на короткі ітерації розробки, постійний зворотній зв'язок та визначення цінності для користувача.

Найструктурованішим та найпопулярнішим Agile- фреймворком є Scrum. Згідно з оновленим «The Scrum Guide» (2020)[48], це «полегшений фреймворк, який допомагає людям, командам та організаціям створювати цінність за допомогою адаптивних рішень для проблем». Його не можна назвати повноцінною методологією, але він надає правила і ролі для організації процесу розробки.

Основою даного фреймворку є Scrum-команда, яка включає в себе три основні ролі: Власник продукту (Product Owner), який відповідає за цінність продукту та вимоги до нього, Scrum-майстер, який по суті виконує завдання проєктного менеджера та Команда розробки (Developers) - команди спеціалістів, які займаються безпосередньою реалізацією продукту.

Концепція роботи такої команди полягає в створенні загального Беклогу продукту, тобто списку всіх завдань, та розділення роботи на спринти, де кожен спринт має свій власний міні-Беклог. Після завершення спринту команда отримує інкремент(готова частина проєкту) та проводить огляд та ретроспективу спринту для демонстрації результатів та аналізу і покращення процесів.

На основі цього, для дослідження важливо що методологія Scrum надає свої чіткі вимоги для програмного середовища: ведення двох беклогів (продукту та спринту), візуалізація робочого процесу(зазвичай через дошку задач) та фіксація результатів планування та завершення спринтів[40].

Також одним з популярних гнучких підходів на ряду з Scrum є Kanban. Головною їх відмінністю є те, що Kanban - це метод, який базується на потоках та орієнтований на візуалізацію робочих процесів та оптимізацію виконання задач. Його ключова та найвідоміша практика - це Kanban-дошка, тобто візуальне відображення всіх завдань та їх статусу. Також варто згадати про такі практики як обмеження незавершеної роботи (WIP Limits) для запобігання перевантаженню команди, та управління потоком [3]. Тобто якщо Scrum має за мету завершення фіксованого спринта, то Kanban - плавне і безперервне виконання задач.

Що стосується практичного застосування цих методологій в управлінні проєктами, багато компаній та організацій не використовують їх в чистому вигляді. На даний момент найбільш поширеними можна вважати так звані гібридні підходи.

Найпопулярнішим таким підходом можна вважати модель Scrumban. Вона використовує методику спринтів та розподілення на ролі з Scrum, але також додає візуалізацію процесів та гнучкість в виконанні завдань з Kanban[13]. Така методологія надає команді можливість адаптувати процеси під свої власні потреби з поєднанням і довгострокового планування і гнучкості виконання.

Отже проведений аналіз вищевказаних методологій демонструє що гнучкість в процесах управління проєктами грає дуже важливу роль і може виявитись ключовим фактором успіху проєкту. Однак, не зважаючи на велику кількість існуючих підходів та методологій не можна сказати що існує універсальний підхід, який підійде для будь якого проєкту чи команди. Як показує практика, найбільш використовуваними є гібридні моделі, які містять в собі елементи декількох методологій, але не є суворо прив’язаними до якоїсь однієї з них.

Саме на такий підхід орієнтовано дане дослідження та розробка прототипу. Базуючись на цьому дослідженні, така система повинна надавати можливості та інструменти, що дозволять команді використовувати переваги цих підходів та адаптувати їх під свої процеси та проєкти.

1.2. Дослідження підходів до проєктування програмних систем для підтримки управління проєктами

На відмінну від попереднього підрозділу, який демонструє аналіз методологій управління проєктами, що допомагають визначати роботу команди (наприклад, за Agile чи Scrum), даний підрозділ призначений для дослідження саме підходів проєктування відповідного програмного забезпечення. Вибір моделі життєвого циклу програмного забезпечення є одним з найважливіших етапів, так як на основі цього буде будуватись все проєктування і дослідження загалом, та це матиме вплив на його гнучкість та відповідність користувацьким потребам [51].

В концепції традиційної моделі розробки, як Waterfall (водоспадна), проєктування вважається окремим, незалежним від інших етапом. Тобто архітектура проєкту, структури даних та інтерфейс проєктуються не залежно від інших етапів і лише повне завершення проєктування дозволяє переходити до наступних етапів. Дана модель передбачає що всі вимоги вже відомі та не підлягатимуть змінам в процесі, що , як зазначено у 1.1, спостерігається досить рідко в сучасних реаліях, де вимоги часто змінюються [4].

Протилежно цим моделям, гнучкі (Agile) методології мають кардинально інший підхід до проєктування і значно більше підходять в рамках інноватівного дослідження, так як використовують ітеративні та інкрементальні моделі [21]. Замість одного великого проєкту, система може проєктуватись та розроблятись ітеративно по частинах даючи можливість для змін за потреби.

На основі цього з’явилась концепція «проєктування що розвивається» (Emergent Design). Архітектура такої системи не статична, а має можливість еволюціонувати та покращуватись з часом. Це дозволяє значно краще адаптуватись до нових вимог та реалій, та дає можливість приймати рішення постфактум, базуючись на актуальній інформації а не теоретичних даних [41]. Важливим в цьому процесі є рефакторинг, тобто реструктуризація існуючого коду без зміни роботи його функціоналу. Це дозволяє підтримувати систему і її код гнучким та адаптивним до нових вимог.

Одним з ключових елементів такого проєктування є оперативний та регулярний зворотній зв’язок. На основі демонстрації готової частини продукту в кінці кожного з спринтів, команда отримує фідбек, який буде враховано в подальшому плануванні. Таким чином можна гнучко коригувати напрямок проєктування та розробки, що забезпечить максимальну відповідність системи реальним потребам користувачів [32].

Хоча розглянуті вище ітеративні моделі і принципи гнучкого проєктування можуть забезпечити високу технічну адаптивність систем, але вони не завжди гарантуватимуть зручність та ефективність для користувача. В тому числі це стосується таких складних систем як програмне забезпечення для управління проєктами, де користувачі, які можуть мати різні ролі та різну технічну підготовку, повинні ефективно використовувати всі його можливості. Тому в сучасних реаліях проєктування програмного забезпечення все більше зміщується в бік потреб користувача, який буде користуватись системою [27].

Втілення цього зміщення можна спостерігати в парадигмі людино-орієнтованого проєктування (User-Centered Design, UCD), або Human-Centered Design (HCD). Вона включає в себе не просто набір технік чи інструментів, а фактично є філософією процесу проєктування, в основі якої лежать потреби, цілі можливості та обмеження кінцевих користувачів [26]. Основоположники цього підходу, такі як Дональд Норман [38] та Якоб Нільсен [36], виокремлювали, що перш за все, проєктована система повинна бути зрозумілою, легкою в навчанні та користуванні, а також підлаштовуватись під користувача замість того щоб користувач підлаштовувався під систему.

Отже, на основі вищевказаної інформації, при проєктуванні програмного забезпечення для управління проєктами повинні бути застосовані наступні принципи UCD [53]: ранній та постійний фокус на користувачах та їхніх завданнях(проєктування повинно починатись з дослідження цільової аудиторії, їх робочих процесів і тд.); емпіричне вимірювання (рішення повинні тестуватись на реальних або потенційних користувачах); ітеративне проєктування (на основі результатів тестування та зворотного зв'язку створюються покращені прототипи).

Базуючись на цій інформації, парадигма людино-орієнтованого проєктування (UCD) може вважатись загальною філософією, яка реалізується шляхом досліджень і проєктування компонентів системи. В рамках проєктування програмного забезпечення для управління проєктами найважливішими аспектами є проєктування взаємодії з користувачем та візуалізація даних.

Проєктування взаємодії (Interaction Design - IxD) займається дослідженням взаємодії кінцевого користувача з системою для досягнення його мети за найкоротший час. Сюди входить не тільки проєктування вигляду компонентів, а й реакція системи на дії користувача, надання йому зворотнього зв’язку та максимально ефективне введення користувача в систему [29]. Одним з найважливіших понять тут можна вважати «юзабіліті» (Usability), яке включає в себе легкість в освоєнні системи користувачем, ефективне використання та задоволення процесом роботи з системою[47]. Для проєктованої системи для управління проєктами це надзвичайно важливо, так як цільові користувачі(наприклад проєктні менеджери чи розробники) взаємодіятимуть з системою інтенсивно та практично щоденно.

Проєктування інформаційної архітектури (Information Architecture - IA) полягає в структуризації та організації інформації всередині системи, для того щоб користувач міг легко та безперешкодно знайти її та зрозуміти [34]. В контексті проєктованої системи це досить складне завдання, так як система включає в себе велику кількість різноманітних даних: проєкти включають в себе задачі, задачі можуть мати підзадачі, коментарі, вкладені файли, відповідальних виконавців, часові терміни, тощо. Ефективна IA передбачає що користувач буде інтуїтивно розуміти де він знаходиться, як створити задачу чи переглянути звіт.

Також важливим компонентом для користувацького середовища є візуалізація даних, призначення якої збирати та відображати дані проєкту в зрозумілому візуальному форматі.

Основними такими форматами вважаються: діаграми Ганта(візуалізація графіку проєкту, залежності задач в часі [55]), kanban-дошки(візуалізація потоку завдань та їх поточного статусу) та панелі моніторингу (Dashboards) (показують

ключові показники ефективності (KPI) проєкту: відсоток виконання, завантаженість команди, фінансові показники тощо [54]).

Також вадливою частиною людино-орієнтованого проєктування є використання прототипів. Тут прототип використовується не тільки як початкова версія системи, а й надзвичайно важливий дослідницький інструмент. Цей інструмент дозволяє створювати окремий функціонал, перевіряти гіпотези та отримувати актуальний фідбек навіть в процесі розробки [7]. В межах даної магістерської роботи створення прототипу системи позиціонується саме таким засобом для перевірки та валідації запропонованих рішень до значних вкладень.

Валідація створеного прототипу, тобто його відповідність очікуванням та потребам користувачів є критичною для загального успіху проєкту. На даний момент існує декілька методів для оцінки зручності користування та продуктивності середовища [30]. Серед них дня даного дослідження найрелевантнішими є експертна оцінка та тестування «юзабіліті».

Експертна оцінка полягає в тестуванні та аналізі функціоналу підготованими та досвідченими користувачами, які можуть порівняти його з набором загальновизнаних принципів хорошого дизайну, відомих як евристики (наприклад, 10 евристик юзабіліті Якоба Нільсена [35]). Цей підхід допомагає швидко та ефективно виявити основні проблеми програмного інтерфейсу.

З іншого боку, тестування юзабіліті - це емпіричний метод, що залучає кінцевих користувачів [45]. Це допомагає визначити ефективність спроєктованого середовища в реальних умовах використання.

Отже, дослідження підходів до проєктування програмних систем визначило необхідність комплексного підходу, який включатиме в себе не тільки технічну реалізацію, а й модель розробки за методологією людино-орієнтованого проєктування (UCD). Такий підхід містить в собі розуміння потреб користувачів, проєктування інформаційної архітектури та взаємодії, створення прототипу та його валідацію, та є методологічною основою для досягнення мети даного магістерського дослідження.

1.3. Порівняльний аналіз існуючих програмних рішень

Ефективне управління проєктами є ключовим фактором успіху у багатьох сферах діяльності, таких як бізнес, освіта, IT і т.д. Однак сучасні інструменти для управління проєктами часто виявляються недостатньо універсальними та не враховують всіх аспектів роботи різних команд, що створює труднощі у плануванні, виконанні та контролі проєктів.

Зараз на ринку існує досить велика кількість програмних рішень для управління проєктами, які відрізняються функціоналом, складністю та ціною. Більшість із них є комерційними продуктами, що вимагають значних фінансових витрат. У той же час, ці рішення часто мають обмежену гнучкість і не завжди повністю відповідають специфічним потребам користувачів. Порівняльний аналіз існуючих програмних рішень допомагає оцінити існуючі підходи до розробки таких систем, визначити їхні переваги та недоліки, а також виявити ключові тенденції та виклики в цій галузі для проєктування та розробки власного рішення.

Для аналізу існуючих рішень у сфері управління проєктами було обрано популярні в міжнародній спільноті Jira та Trello та подібний за специфікою та фунціоналом застосунок українського виробництва Worksection.

Jira Software на даний момент один з найпопулярніших та найпотужніших інструментів в сфері управління проєктами, і найчастіше зустрічається саме в сфері розробки програмного забезпечення[15]. Сам застосунок позиціонується як інструмент для Agile-команд і надає досить великий набір інструментів та функцій що відповідають гнучким методологіям. Система реалізована компанією Atlassian і має вбудовану можливість інтеграції з іншими її продуктами, такими як Confluence (для документації) та Bitbucket (для репозиторіїв коду), що дає можливість створити комплекс систем якщо є така потреба.

Основні функціональні можливості Jira [9] включають в себе управління завданнями (Issues), підтримка Agile-фреймворків Scrum та Kanban, кастомізовані робочі процеси (Workflows), звіти та аналітика (Burndown charts, Velocity charts), Дошки (Boards) і багато іншого.

Центральним елементом середовища, з якого починається робота з системою, є Jira Dashboard (рисунок 1.1). На дешборді можна переглянути усі проекти команди, відображаючи список завдань, їх статус, пріоритет та термін виконання.

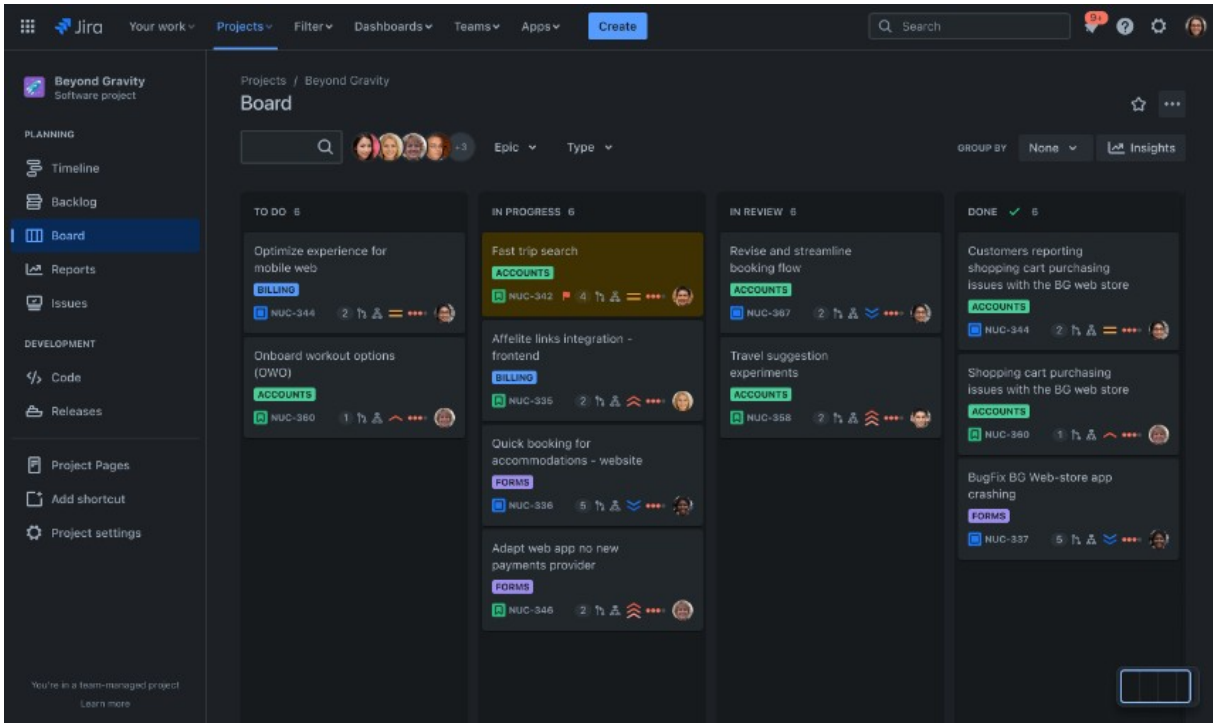

Рисунок 1.1 - Jira Dashboard

Головною перевагою Jira є дуже висока гнучкість та широкі можливості налаштування. Платформа дає можливість адаптації робочих процесів, типів завдань та полів під специфічні потреби практично будь-якої команди чи проєкту.

З іншого боку, ця ж потужність спричинила і головний недолік - високий поріг входження та складність для нових користувачів, що дуже часто призводить до відчуття перевантаженості інтерфейсу. Також одним з основних недоліків можна виділити вартість ліцензій, особливо для невеликих команд.

З точки зору Людино-Орієнтованого Проєктування (UCD) та юзабіліті, Jira можна вважати компромісом між функціоналом та адаптивністю. Хоча на початкових етапах, система все ще створює значне навантаження користувачів

через велику кількість опцій на екрані, необхідність вивчення специфічної термінології та мови запитів JQL для повноцінного використання аналітики.

На відмінну від Jira, яка являє собою дуже потужний, але складний в освоєнні інструмент, Trello (також розроблений компанією Atlassian) є практично цілковитою його протилежністю. Тобто Trello перш за все орієнтований на простоту та візуалізацію [33]. Отже це досить простий та інтуїтивно зрозумілий інструмент, який взяв за свою основу концепцію Канбан-дошки.

Користувацьке середовище Trello формується навколо дошки (board), яка представляє проєкт (рисунок 1.2). Дошка містить списки (lists), що зазвичай відображають етапи робочого процесу (наприклад, «Заплановано», «В роботі», «Перевіряється», «Готово»). Задачі представлені у вигляді карток (cards), які можна вільно перетягувати між списками в різні статуси. Кожна картка може містити опис, чек-листи, коментарі, файли, мітки та терміни виконання [39].

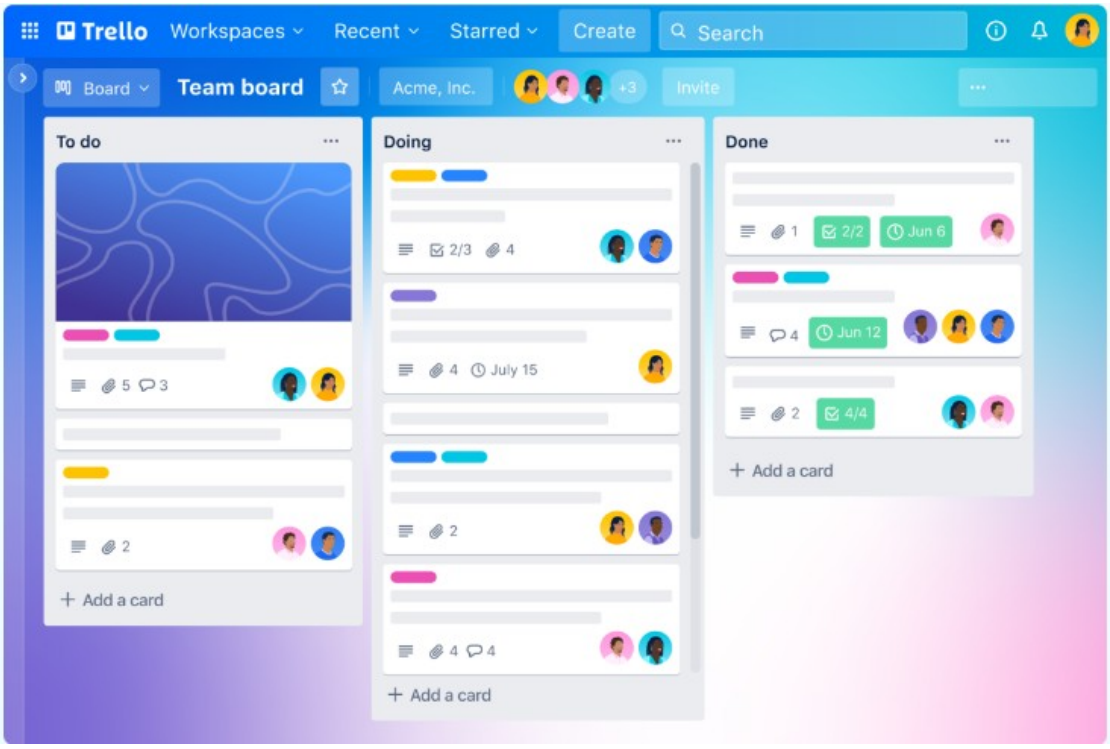

Рисунок 1.2 - Trello Board

Звідси можна дуже легко визначити основну перевагу Trello - це максимальна простота використання та мінімальні затрати часу на навчання.

Новий користувач може почати працювати з системою практично миттєво. Висока візуалізація процесів допомагає користувачу відстежувати та керувати завданнями легко та інтуїтивно. Концепція таких дошок є досить універсальною і може використовуватись не лише для розробки програмного забезпечення, а й для маркетингу, HR, особистого планування тощо [57].

Але ця ж перевага може бути й основним недоліком Trello. Інструмент має досить обмежену функціональність і не підходить для управляння складними та комплексними проєктами. Також при великій кількості задач дошка може стати занадто завантаженою та некерованою.

З точки зору Людино-Орієнтованого Проєктування (UCD) та юзабіліті, Trello отримує найвищі оцінки за легкістю вивчення, ефективністю для простих завдань та задоволеністю користувачів [50]. Але з іншого боку його інструментарій не задовольняє потреби користувачів, які стикаються зі складними проєктами, що вимагають глибокої аналітики та планування ресурсів.

На відміну від вищероглянутих популярних на широку аудиторію по всьому світу Jira та Trello, Worksection є яскравим представником українського ринку програмного забезпечення, який позиціонується, як орієнтований на потреби локального бізнесу та креативних індустрій [2]. Worksection - онлайн-платформа для управління проєктами, що дозволяє створювати та розподіляти задачі, планувати терміни, вести комунікацію через чати та коментарі, інтегрувати з іншими сервісами (Google Calendar, Slack тощо), моніторити прогрес через звіти та графіки, а також працювати з файлами і документами. Інтерфейс простий та зручний для користувачів, доступний на мобільних пристроях[1].

Як основний інтерфейс тут можна визначити окремий дешборд ля кожного конкретного проєкту (рис. 1.3). Тут відображено загальний стан роботи команди над проєктом. У верхньому лівому куті показано керівника, а також кількість учасників команди. Поруч знаходиться прогрес виконання проєкту візуалізоване на часовій шкалі. У центральній частині знаходяться основні метрики проєкту, фінансова статистика, графік активності команди. Також тут можна побачити

відкриті задачі за статусами та по людях. Ця панель дає візуальне уявлення про прогрес, ефективність роботи та активність команди.

Кожне з завдань проєкту можна відкрити в окремому вікні де також є досить великий інструментарій для роботи з конкретним завданням проєкту. Тут застосунок дає можливість детально описати завдання, призначити відповідального, вказати статус та запланований час. Для відповідального за завдання є можливість трекінгу витраченого часу на завдання.

Рисунок 1.3 - Інтерфейс головної сторінки(дешборда) Worksection

Загалом, як переваги, Worksection має інтуїтивно зрозумілий інтерфейс, зручне управління задачами і планування, та багато інших корисних функцій.

Мінусами Worksection можна зазначити обмежену кількість функцій у безкоштовній версії, може бути надмірно складним для малих команд з базовими потребами, деякі інтеграції потребують додаткових налаштувань.

З точки зору UCD та юзабіліті, Worksection надає багатофункціональну комбінацію інструментів. Його дешборди можуть бути дуже ефективними для впевнених користувачів, але важкими для сприйняття в процесі адаптації до

системи. Система забезпечує своїх користувачів великою кількістю функціоналу, але за рахунок певної складності в навігації та налаштуваннях.

Щоб підсумувати проведений аналіз існуючих рішень, їх ключових функцій, переваг та недоліків і тд. представлено їх узагальнення та порівняння(таблиця 1.1).

Таблиця 1.1 - Порівняльний аналіз існуючих програмних рішень

Критерій	Jira	Trello	Worksection
Основне призначення	Комплексне управління Agile-проєктами (Scrum, Kanban)	Візуалізація завдань, просте управління потоком (Kanban)	Управління проєктами, задачами та фінансами компанії
Ключові функції	Беклоги, Спринти, Кастомні Workflows, Звіти	Дошки, Списки, Картки, Чек-листи	Діаграма Ганта, Тайм-трекінг, Звіти, Фінанси
Оцінка UCD/Юзабіліті	Потужна, але складна. Високе навантаження нових користувачів.	Дуже проста. Низький поріг входження.	Функціональна, але потенційно перевантажена для новачків
Основні переваги	Гнучкість, потужність, інтеграції, підтримка Agile.	Простота, візуалізація, гнучкість для різних сфер.	Комплексність (задачі + час + фінанси), укр. підтримка.
Основні недоліки	Складність налаштування, високий поріг входження, ціна	Обмежена функціональність для складних проєктів, відсутність звітів	Перевантаженість для простих завдань, ціна.

На основі отриманих даних та порівняння Jira, Trello та Worksection можна визначити їх основні переваги та потреби сучасного ринку в програмному забезпеченні такого типу.

В підсумку маємо, що потужні системи як Jira надають великий інструментарій для професійного менеджменту але мають дуже високий поріг входження в систему. Прості системи (Trello) є простими в сприйнятті і використанні, але часто їм бракує якоїсь частини функціоналу. На противагу ним, Worksection - це спроба створити збалансоване програмне забезпечення, але фокус все одно зміщений більше в бік саме проєктного менеджменту та відповідних інструментів, що додає труднощів решті команди

Отже, як результат аналізу існуючих рішень можна побачити що відсутнє рішення, яке б дійсно було збалансованим і зручним для використання, підтримувало базові Agile-процеси і візуальну простоту сприйняття. На вирішення цієї проблеми і спрямоване дане дослідження.

Для підкріплення результатів аналізу існуючих програмних рішень під час проходження науково-дослідної практики на підприємстві було проведено коротке опитування спеціалістів, які працюють з подібним програмним забезпеченням щоденно та мають свої проблеми і побажання. Було опитано близько 10 співробітників, включаючи проєктних менеджерів, розробників, менеджерів з продажу і HR-спеціалістів.

Результати опитування чітко виявили проблему подібного програмного забезпечення, а саме не достатня гнучкість функціоналу відповідно до їх ролі.

Проєктні менеджери підтвердили, що такі системи орієнтовані саме на них і відповідно вони мають максимальні вимоги до функціоналу, тобто звітність, інструменти планування, спілкування в рамках задачі і тд. Основним пріоритетом чітко визначено функціональність та контроль процесів.

Для розробників навпаки важливіше мінімалізувати «шум» щоб мати змогу сконцентруватись саме на їх задачі. Переходи між екранами, складна звітність та велика кількість сповіщень - є основними проблемами в розглянутих системах. Основною потребою визначено чітко поставлену задачу з всією необхідною інформаціє в одному місці.

Менеджери з продажу, на противагу своїм колегам, хотіли б бачити в системі зовсім протилежне - загальну інформацію по кожному проєкту: поточний статус, ключові дедлайни, загальний бюджет або витрачений час, щоб оперативно комунікувати з клієнтом.

Для HR-спеціалістів такі системи не є основним інструментом роботи, тому для обліку робочого часу або відпусток вони потребують максимальної простоти та мінімуму зайвих елементів.

Отже на основі опитування реальних юзерів підтвердило актуальність мети даного дослідження - проєктування адаптивного користувацького середовища,

яке здатне підлаштовувати функціонал та інтерфейс під конкретну роль користувача.

1.4. Висновки до розділу 1

Отже, в рамках розділу 1, було проведено комплексний аналітичний огляд джерел та предметної області, що є теоретичним підґрунтям для подальшого дослідження та проєктування програмного прототипу.

Аналіз теоретичних основ управління проєктною діяльністю демонструє перехід сучасних тенденцій від класичних моделей до гнучких методологій, які перекривають більше процесів та проєктів. Також було виявлено, що на практиці найчастіше застосовуються гібридні моделі, які включають в себе елементи різних підходів та методологій.

Дослідження підходів до проєктування програмних систем показує, що для створення ефективного програмного забезпечення в сучасних реаліях не достатньо лише якісної технічної реалізації. Тому, виходячи з цього, було обґрунтовано доцільність застосування парадигми людино-орієнтованого проєктування для створення прототипу.

Порівняльний аналіз існуючих програмних рішень, таких як Jira , Trello та Worksection, показав що на ринку існує потреба саме в універсальному програмного забезпеченні щоб задовільнити потреби всіх учасників команди. Це ж підтвердило і проведене опитування різних IT-спеціалістів, які відзначили головним мінусом відсутність адаптації системи під їх роль в команді.

Таким чином, підсумовуючи результати теоретичного дослідження в рамках розділу 1, було обґрунтовано актуальність дослідження та практичну необхідність розробки програмного прототипу, мета якого - забезпечити всіх учасників команди адаптивним функціоналом, поєднуючи достатню функціональність з простотою та інтуїтивним розумінням інтерфейсу.

РОЗДІЛ 2. СИСТЕМНИЙ АНАЛІЗ І ОБҐРУНТУВАННЯ ПРОБЛЕМИ

2.1. Системний аналіз предметної області управління проєктною діяльністю

На основі аналізу, проведеного в попередньому розділі, як результат отримано чітке розуміння що сучасний ринок програмного забезпечення для управління проєктами є досить насиченим, але в той же час поляризованим. Доступні рішення часто ставлять перед користувачем вибір або висока функціональність, або простота використання. Тобто є потреба в рішенні, яке задовільнить обидві ці вимоги для команди з різними ролями та потребами. Щоб обґрунтувати проєктування прототипу, необхідно провести системний аналіз предметної області управління проєктною діяльністю.

Головною метою цього аналізу можна вважати підвищення ступеня обґрунтованості проєктних рішень шляхом структуризації предметної області, виявлення компонентів системи, їхніх взаємозв'язків та потоків інформації [18]. Цей процес є фундаментальним і допомагає перетворити проблеми бізнесу та користувачів в чіткий план розробки ефективного рішення.

В якості об’єкту дослідження дана система визначена як «Програмне середовище для підтримки управління проєктною діяльністю». Відповідно до класифікації, дана система буде належати до типу складних систем [44].

Складність даної системи полягає в наявності великої кількості компонентів та їх взаємодій між собою, в великій кількості сутностей та взаємопов’язаних даних, і не менш важливо, динамічності.

З метою описати декомпозицію такої системи прийнято рішення використовувати методи структурного аналізу, а саме діаграми потоків даних (DFD) та діаграми варіантів використання (Use Case). Це дозволить визначити та візуалізувати функціональну структуру системи та взаємодію системи з її зовнішніми акторами.

Для декомпозиції системи створено контекстну DFD діаграму (рис. 2.1), яка визначає та описує межі системи та взаємодію системи з зовнішнім оточенням.

Рисунок 2.1 - Контекстна DFD діаграма

Центральний процес «0. Програмне середовище УП» представляє собою всю систему як єдине ціле. Він взаємодіє з трьома ключовими сутностями (акторами):

- адміністратор: Сутність, що відповідає за загальне налаштування системи та управління доступом користувачів;
- проєктний Менеджер (PM): Сутність, відповідальна за планування, створення проєктів, постановку завдань та моніторинг їх виконання;
- розробник (Учасник команди): Сутність, що є виконавцем завдань та звітує про прогрес своєї роботи.

Між сутностями та центральним процесом визначено наступні основні потоки даних:

Вхідні потоки (до системи):

- «Дані нового користувача/Налаштування»: включають в себе інформацію, яку Адміністратор надсилає для створення або оновлення записів;

- «Дані нового проєкту/Нове завдання/Запит на звіт»: інформація, яку РМ надсилає для ініціалізації проєктів, створення завдань та запиту даних;

- «Оновлення статусу завдання/Коментар/Витрачений час»: оперативні дані, які Розробник надсилає до системи в ході виконання роботи.

Вихідні потоки (від системи):

- «Підтвердження/Статус системи»: зворотний зв'язок, який Адміністратор отримує від системи;

- «Звіт по проєкту/Дешборд»: агрегована інформація про стан проєкту, яку система надає Проєктному Менеджеру;

- «Список персональних завдань/Сповіщення»: інформація, яку система надає Розробнику для організації його роботи.

На основі цих даних створено діаграму потоків даних DFD Рівня 1(рис. 2.1), яка деталізує центральний процес «0. Програмне середовище УП», зображений на контекстній діаграмі (рис. 2.1). В ході декомпозиції було виділено чотири основні підпроцеси та три ключові сховища даних.

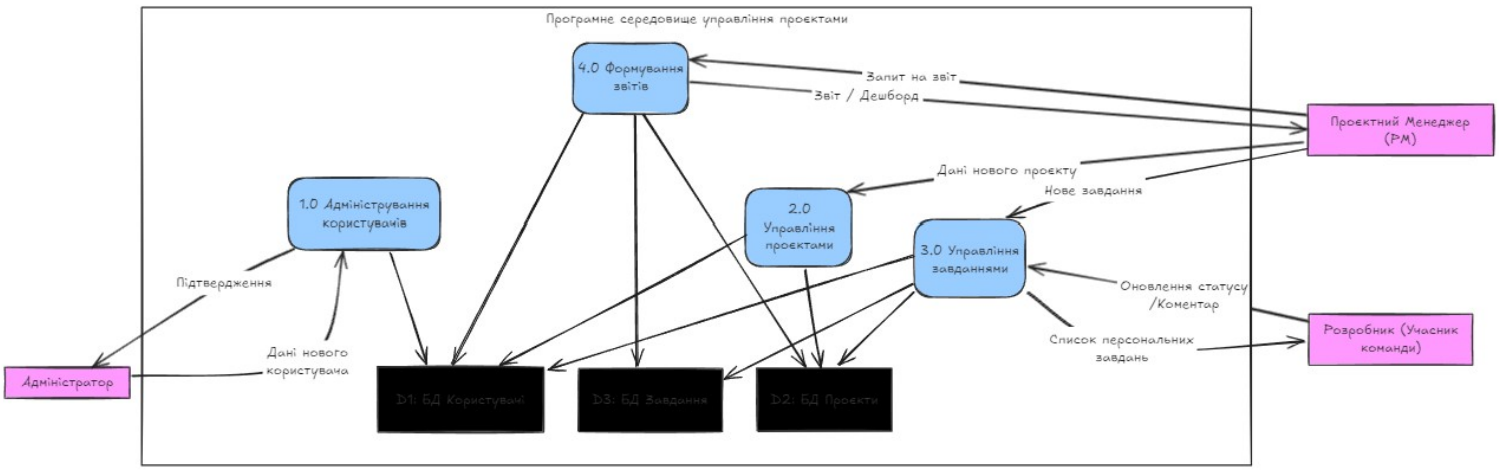

Рисунок 2.2 - Декомпозиція (DFD Рівень 1))

Перший процес в системі - «Адміністрування користувачів». Він відповідає за керування життєвим циклом облікових записів користувачів. Він

отримує потік даних «Дані нового користувача» від сутності «Адміністратор». Після обробки (створення, редагування, видалення користувача) процес записує зміни у сховище даних «D1: БД Користувачі» та повертає «Адміністратору» підтвердження.

Наступний процес «Управління проєктами» відповідає за створення, налаштування та керування проєктами. Процес запускається «Проєктним Менеджером» через потік «Дані нового проєкту». Для правильного створення проєкту та його команди процес отримує дані зі сховища «D1: БД Користувачі». Результат роботи записується у сховище «D2: БД Проєкти».

Третій процес - «Управління завданнями». Це буде найбільш завантажений процес, який описує щоденну операційну роботу. Він отримує вхідні потоки: «Нове завдання» від «PM» та «Оновлення статусу / Коментар» від «Розробника». Для коректної обробки процес отримує дані зі сховищ «D1: БД Користувачі» та «D2: БД Проєкти». Результати (нові завдання, зміни статусів) записуються у сховище «D3: БД Завдання». Також цей процес формує вихідний потік «Список персональних завдань» для «Розробника».

Останній процес «Формування звітів», цей процес відповідає за обробку даних та надання аналітики. Він активується «Запитом на звіт» від «PM». Для побудови звіту чи дешборду процес отримує актуальні дані з усіх трьох сховищ: «D1: БД Користувачі» (інформація про виконавців), «D2: БД Проєкти» (загальний статус проєктів) та «D3: БД Завдання» (прогрес по задачах). Результат повертається «PM» у вигляді потоку «Звіт / Дешборд».

Також на основі вищевказаної інформації маємо сховища даних:

- D1: БД Користувачі: інформація про облікові записи, ролі та права;
- D2: БД Проєкти: інформація про сутності проєктів, їхні налаштування та команди;
- D3: БД Завдання: оперативна інформація про завдання, їхні статуси, коментарі та витрачений час.

Отримана декомпозиція дозволяє розпочати наступний етап аналізу - тобто моделювання варіантів використання (Use Case), яке деталізує вимоги користувачів до описаних процесів.

Якщо продемонстровані вище діаграми DFD демонструють моделювання даних, які рухаються через систему, то щоб коректно описати функціональні вимоги до системи з точки зору користувача доцільно застосувати моделювання варіантів використання (Use Case). Діаграма варіантів використання (UML) - це візуальне представлення, що ілюструє взаємодію між зовнішніми сутностями (акторами) та системою, фіксуючи її функціональні вимоги та те, як актори досягають своїх цілей [19].

Ключовим елементом моделювання вважається Актор - це роль, яку виконує користувач в системі, або будь яка інша зовнішня система, яка взаємодіятиме з проєктованим програмним середовищем.

На основі проведеного системного аналізу (DFD) та результатів опитування цільової аудиторії (див. 1.3), для проєктованого прототипу визначено ключових акторів(табл. 2.1).

Таблиця 2.1 - Актори системи та їх опис

Актор	Опис
Неавторизований користувач	Будь-яка особа, що має доступ до сторінки входу в систему, але ще не пройшла автентифікацію.
Розробник (Учасник команди)	Авторизований користувач, основна мета якого виконувати завдання. Потребує чіткого списку власних завдань та мінімуму відволікаючих факторів.
Проєктний Менеджер (PM)	Авторизований користувач, основна мета якого - це планування, контроль та аналіз проєкту. Потребує доступу до всіх завдань, звітів, дешбордів та налаштувань проєкту.
Адміністратор	Привілейований користувач, який має повні права на адміністрування системи: управління всіма користувачами та загальними налаштуваннями системи.

Таке визначення акторів в системі та, зокрема, чітке розподілення «Розробника» як базового користувача системи та «Проєктного Менеджера»,

який потребує розширеного функціоналу та доступу до інструментів, які не потрібні решті користувачів є ключовим для подальшого проєктування адаптивного користувацького середовища.

На основі визначених вище ключових акторів системи представлено загальну діаграму варіантів використання (Use Case), яка візуалізує взаємодію акторів з основними функціями системи(рис. 2.3).

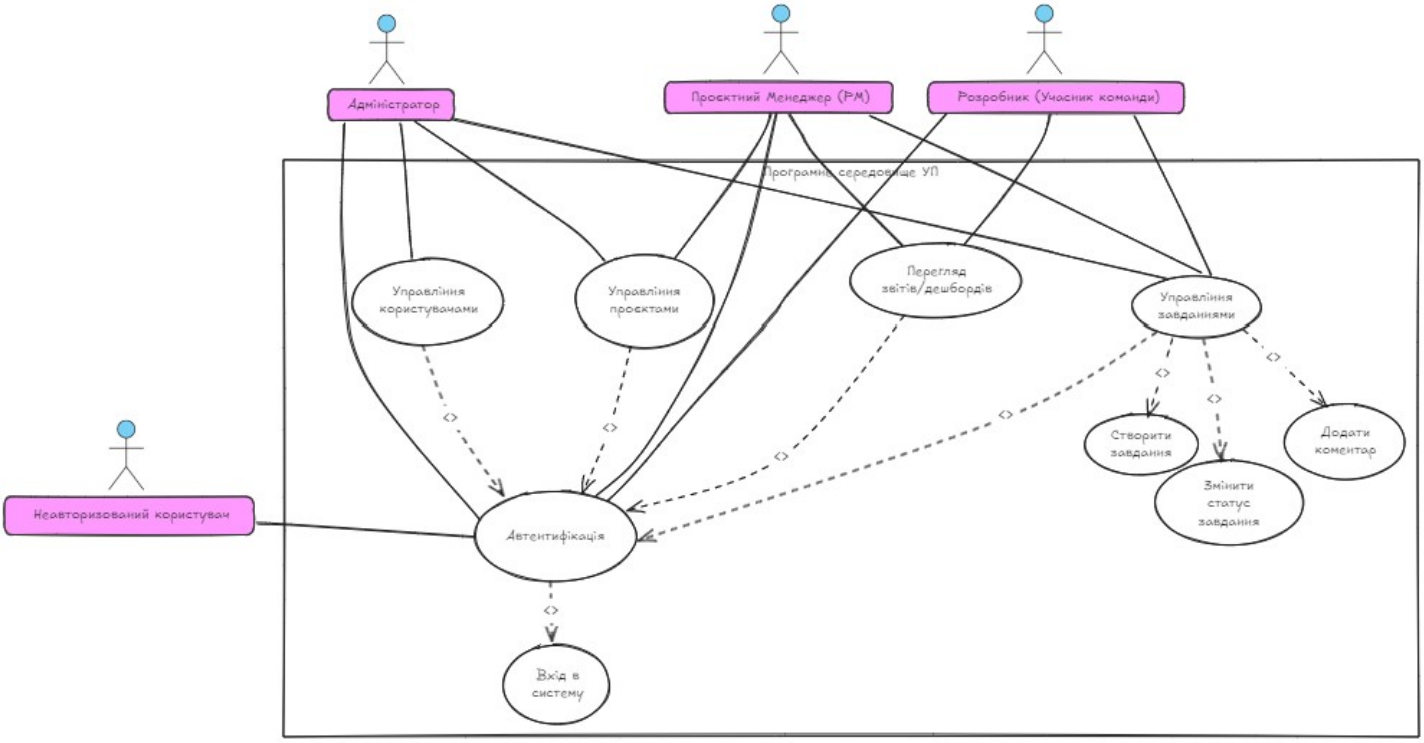

Рисунок 2.3 - Загальна діаграма варіантів використання)

Ключові взаємодії продемонстровані на діаграмі:

- «Неавторизований користувач» може взаємодіяти тільки з варіантом використання (ВВ) «Автентифікація», який, у свою чергу, включає (<<include>>) ВВ «Вхід в систему»;
- «Розробник (Учасник команди)» має доступ до «Управління завданнями» та «Перегляду звітів/дешбордів»;
- «Проєктний Менеджер (PM)» має ширші повноваження: «Управління проєктами», «Управління завданнями» та «Перегляд звітів/дешбордів»;

- «Адміністратор» має найвищі права, що включають в себе унікальну функцію «Управління користувачами» та абсолютно весь функціонал доступний кожному з типів користувачів.

Як можна побачити на діаграмі, всі основні варіанти використання вимагають проходження автентифікації в системі.

Деталізація ВВ «Управління завданнями» показує, що він включає такі ключові під-сценарії, як «Створити завдання», «Змінити статус завдання» та «Додати коментар».

На основі загальної діаграми варіантів використання та всієї вищезгаданої інформації, для кращого розуміння функціональних вимог та логіки взаємодії акторів із системою, проведено деталізацію ключових сценаріїв.

Вибір сценаріїв для деталізації базується на результатах аналізу та опитування. Для демонстрації основних моментів взаємодії кожного з основних акторів системи створено відповідний опис варіанту використання.

Першим та найбільш базовим сценарієм є «Вхід в систему», який є обов'язковим для доступу до будь-яких функцій системи (табл. 2.2).

Таблиця 2.2 - Опис варіанту використання «Вхід в систему»

Характеристика	Опис
Назва Use Case	Вхід в систему (є частиною <<include>> ВВ «Автентифікація»).
Актори	Неавторизований користувач, Адміністратор, PM, Розробник.
Передумови	Користувач знаходиться на сторінці входу в систему.
Основний потік	1. Користувач вводить логін та пароль. 2. Користувач натискає кнопку «Увійти». 3. 3.Система перевіряє валідність даних у «D1: БД Користувачі». 4. Система автентифікує користувача, створює сесію. 5. Система перенаправляє користувача на головний Дешборд відповідно до його ролі (PM, Розробник тощо).
Альтернативні потоки	5а. Система не знаходить користувача або пароль невірний. 5б. Система виводить повідомлення «Помилка входу. Невірний логін або пароль».

Постумови	Користувач автентифікований в системі та має доступ до функцій своєї ролі.

Наступний сценарій описує одну з ключових функцій, яка завжди буде доступна «Проєктному Менеджеру» (PM), а саме створення нового проєкту, що є основою для всієї подальшої роботи(табл. 2.3). Цей сценарій демонструє взаємодію з кількома сховищами даних (читання користувачів для формування команди та запис у базу даних проєктів).

Таблиця 2.3 - Опис варіанту використання «Створення нового проєкту»

Характеристика	Опис
Назва Use Case	Створення нового проєкту (є частиною ВВ «Управління проєктами»).
Актори	Проєктний Менеджер (PM), Адміністратор.
Передумови	Актор автентифікований в системі та має права на створення проєктів.
Основний потік	1. Актор обирає опцію «Створити проєкт». 2. Система відображає форму створення проєкту. 3. Актор вводить дані проєкту (назва, опис, дедлайн). 4. Актор обирає учасників команди (зчитування з «D1: БД Користувачі»). 5. Актор натискає «Зберегти». 6. Система валідує дані. 7. Система створює новий запис у «D2: БД Проєкти». 8. Система перенаправляє актора на сторінку нового проєкту.
Альтернативні потоки	6а. Актор ввів неповні або некоректні дані (наприклад, назва проєкту вже існує). 6б. Система виводить повідомлення про помилку валідації.
Постумови	У системі створено новий проєкт, запис про нього збережено в «D2: БД Проєкти».

Останнім описаним сценарієм є зміна статусу завдання, що є досить частою операцією, яку виконують практично всі користувачі системи(а найчастіше юзер з роллю «Розробника»). Цей сценарій є дуже важливим в контексті підтримки гнучких методологій та актуалізації даних по проєкту для проєктного менеджера.

Також він напряму демонструє взаємодію користувача з візуалом системи для зміни даних(табл. 2.4).

Таблиця 2.4 - Опис варіанту використання «Змінити статус завдання»

Характеристика	Опис
Назва Use Case	Змінити статус завдання (є частиною <<include>> ВВ «Управління завданнями»).
Актори	Розробник (Учасник команди), PM.
Передумови	Актор автентифікований та знаходиться на сторінці проєкту (наприклад, Канбан-дошці).
Основний потік	1. Актор візуально ідентифікує потрібне завдання (картку) на дошці. 2. Актор натискає та перетягує картку завдання з однієї колонки (статусу) в іншу (напр., з «В роботі» в «Готово»). 3. Система фіксує нову позицію (статус). 4. Система оновлює запис про завдання в «D3: БД Завдання». 5. Система миттєво оновлює вигляд дошки для всіх користувачів.
Альтернативні потоки	4а. Система перевіряє права (наприклад, чи має право цей Розробник переміщати завдання в «Готово»). 4б. Система не дозволяє змінити статус та повертає картку на місце.
Постумови	Завдання має новий статус, зміни збережені в «D3: БД Завдання».

Ці деталізовані сценарії визначили конкретну логіку, яку має реалізувати програмний прототип, та слугуватимуть основою для визначення функціональних вимог та для написання тест-кейсів під час верифікації системи.

Разом з функціональними вимогами, які було визначено за допомогою DFD та Use Case діаграм, для успішного проєктування також необхідно врахувати і нефункціональні вимоги до системи. Вони включають в себе здатність системи коректно та ефективно виконувати свої функції та визначають критерії якості, які впливатимуть безпосередньо на досвід користувача[20].

Для проєктованого прототипу визначено наступні нефункціональні вимоги:

1. Вимоги до юзабіліті (Зручності використання):

- адаптивність: інтерфейс системи має адаптуватися до ролі користувача (PM, Розробник і тд.), щоб приховати зайвий функціонал і зменшити навантаження та підвищити зосередженість користувача;
- інтуїтивність: ключові сценарії, як зміна статусу завдання, повинні виконуватися за мінімальну кількість кліків та бути інтуїтивно зрозумілими відразу без тривалого навчання;
- консистентність: всі елементи інтерфейсу мають бути однаковими.

2. Вимоги до продуктивності:

- час завантаження: завантаження головного дешборду або дошки проєкту не має перевищувати 3 секунди при стабільному інтернет-з'єднанні;
- час відгуку: час реакції системи на основні дії користувача не повинен перевищувати 1 секунду [20].

3. Вимоги до надійності:

- доступність: система має бути доступна для користувачів 99.9% часу;
- цілісність даних: система має забезпечити коректне зберігання даних, і запобігати їх втраті в разі збоїв системи.

4. Вимоги до безпеки:

- авторизація на основі ролей: система має розмежовувати доступ до даних. Юзер повинен мати доступ до інформації виключно відповідної йому ролі;
- захист даних: всі дані користувачів, особливо паролі, мають зберігатися у зашифрованому вигляді [28].

Отже, визначені нефункціональні вимоги вважаються невід'ємною частиною системного аналізу та будуть використовуватись як критерії для оцінки якості розробленого прототипу в подальшому.

2.2. Постановка і обґрунтування проблеми розробки програмного середовища

Проведений в попередньому підрозділі системний аналіз мав за мету декомпозиціонувати предметну область, визначити ключових акторів, потоки

даних та основні функціональні блоки системи. Також було формалізовано взаємодію користувача та проєктованого програмного середовища. Враховуючи отримані результати цього аналізу та висновки, зроблені як підсумок розділу 1, подальшими діями є чітке визначення та дослідження проблеми, для вирішення якої і призначений проєктований прототип.

Першим кроком в цьому буде визначення мети проєктування та розробки. Відповідно до вимог, визначених вище, метою даної роботи є проєктування та розробка дієздатного програмного прототипу для перевірки та підтвердження дослідницької гіпотези.

Ця дослідницька гіпотеза полягає в тому, що можна суттєво підвищити ефективність програмного середовища для управління проєктами через впровадження адаптивного користувацького інтерфейсу (UI), який б підлаштовувався під конкретну роль користувача, що є ключовою відмінністю від досліджуваних існуючих рішень на сучасному ринку.

Також для дослідження важливо визначити призначення розробленого програмного продукту. Це призначення напряму випливає з його мети, а саме - продукт призначений для вирішення проблем, які було виявлено в аналітичному огляді системи. Сюди входить:

- вирішення проблеми високого когнітивного навантаження на користувача: метою прототипу є демонстрація того, як можна знизити когнітивне навантаження на користувачів, які не потребують повної функціональності системи, шляхом приховування надлишкового та нерелевантного для їх ролі програмного функціоналу;
- підтримка гібридних проєктних команд: система дозволить забезпечити баланс між необхідною функціональністю для проєктних менеджерів, та простотою, яка більше необхідна для розробників та нетехнічних спеціалістів;
- підвищення продуктивності: необхідність забезпечення швидкого та оперативного доступу користувачам з всіма ролями саме до того інструментарію,

який призначено для цієї ролі. Це дозволить мінімізувати час на пошук необхідної інформації та на навігацію по складних меню;

- апробація проєктних рішень: тестування і валідація конкретних UI/UX рішень на основі розробленого прототипу, які в подальшому можна доопрацювати та імплементувати в комерційний продукт.

На основі цього, визначено місце застосування розробленого програмного продукту. Воно визначається тією «прогалиною» ринку, яку він призначений заповнити та зайняти.

Цільовою аудиторією можна вважати:

1. Малі та середні IT-компанії та стартапи: сюди входять організації, яким вже не достатньо по функціоналу використовувати додатки по типу Trello, але поки вважають застосунки типу Jira занадто складними, дорогими або занадто надлишковими для їх процесів.

2. Віддалені та гібридні команди: в цю категорію можна включити команди, в яких присутні учасники з різним рівнем технічної підготовки (розробники, дизайнери, маркетологи, менеджери), що потребує гнучкого та інтуїтивно зрозумілого інструменту.

3. Малий та середній бізнес: такі застосунки можуть застосовуватись і поза IT сектором для керування різноманітними бізнес-проєктами і потребують простого та інтуїтивно зрозумілого інтерфейсу для максимально не підготованих користувачів що цінують свій час.

4. Освітні заклади: прототип можна використовувати як навчальний інструмент на кафедрах інженерії програмного забезпечення для демонстрації принципів UCD та роле-орієнтованої архітектури.

На основі всієї цієї інформації на даному етапі доцільно провести обґрунтування розроблення програмного продукту.

Необхідність проєктування та розробки програмного прототипу системи для управління проєктами базується на наявності чітко визначеної проблеми, яку виявлено та деталізовано в рамках аналітичного огляду в попередньому розділі. Отже, проблемою можна вважати відсутність на ринку відповідних програмних

застосунків, які могли б запропонувати користувачу систему з ефективним балансом між наповненим та доцільним функціоналом і простоту та інтуїтивно зрозуміле користування системою без довготривалої підготовки, а також додатково мінімізує конфлікти потреб для різних ролей в команді, забезпечуюючи їх тільки необхідним функціоналом.

На початку варто розглянути наявність проблеми визначеної вище. Як її загальне визначення можна вважати Компроміс «Складність vs Простота», тобто, як показав аналіз існуючих рішень (підрозділ 1.3), ринок програмного забезпечення для управління проєктами чітко поляризований.

Тобто складні високофункціональні системи (як Jira) пропонують більш ніж достатній функціонал та інструментарій для Agile-методологій (Scrum, Kanban) та забезпечують деталізовану звітність і глибоку кастомізацію. Однак це ж породжує головну проблему таких систем - їх надзвичайну складність в сприйнятті, високий поріг входження в систему та високе когнітивне навантаження на користувача. Часто інтерфейс таких систем визначається як надлишковий для простих проєктів або нетехнічних користувачів чи ролей, яким не він не потрібен.

На відмінну від них, прості системи(напр., Trello) навпаки відзначаються високим юзабіліті, візуальною простотою та практично нульовими потребами в підготовці до використання системи. Але дуже часто вони є обмеженими функціонально для професійного управління проєктами, не включаючи в свій інструментарій важливих для проєктних менеджерів вбудованих діаграм Ганта, залежностей між завданнями чи просунутих звітів. Це робить подібні системи не валідними для використання великою часткою цільової аудиторії.

Сучасні дослідження ринку програмного забезпечення для управління проєктами підтверджують цю гіпотезу. Малі та середні компанії досить часто стикаються з так званою «Jira complexity cliff» (скелею складності Jira), коли затрати на адміністрування і навчання персоналу перевищують загальну користь від використання обраної системи. З іншого боку компанії, які вибирають простіші інструменти досить часто досягають «Trello functional ceiling»

(функціональної стелі Trello), коли проєкт стає некерованим через брак структури [16]. Тобто ця проблема змушує компанії або миритись з неефективністю програмного забезпечення, або вкладати кошти в додаткові плагіни та адміністрування системи.

Отже, наступною проблемою, яку варто розглянути є неможливість вирішення проблеми наявними засобами. Тобто можна зробити висновки, що існуючі інструменти доступні на ринку не вирішують дану проблему повністю. До прикладу є можливість кастомізації Jira для спрощення її інтерфейсу, але це вимагає значних витрат на адміністрування та покупку додаткових плагінів, як опцію. Trello також може бути модифікованим, але такі модифікації досить швидко можуть перетворити звичайну дошку на аналогічний до першого варіанту складний та перевантажений інструмент.

Проте найвагоміші результати аналізу було отримано на основі результатів опитування цільової аудиторії (підрозділ 1.3). Там було емпірично підтверджено існування конфлікту потреб користувачів в залежності від їх ролі в системі та команді загалом:

Проєктний менеджер потребує максимального фунціоналу: формування розширеної звітності, графіків проєкту та загального контролю.

Розробник(або звичайний юзер) має потребу в чіткому списку завдань з інформацією водному місці, та мінімумом відволікаючих факторів.

Sales/HR (Нетехнічні ролі) потребують мати тільки загальну картину проєкту та максимальну простоту системи.

Універсальний інтерфейс, який зараз може запропонувати більшість систем, зазвичай не може задовільнити всі ці потреби для кожної ролі одночасно, так як деякі з них є діаметрально протилежними за своєю суттю. Дослідження в галузі людино-комп’ютерної взаємодії підтверджують, що ефективність корпоративного ПЗ напряму залежить від того, наскільки добре інтерфейс відповідає конкретній ролі та завданням користувача, а не лише загальним функціям системи [31].

Далі визначено новизну та переваги пропонованого прототипу. На відмінну від вищезгаданих систем, які пропонують статичний універсальний інтерфейс, даний проєкт базується на гіпотезі адаптивного, роле-орієнтованого середовища.

Обґрунтуванням доцільності розробки можна вважати те, що проєктований програмний прототип буде демонструвати наступні переваги для вирішення конфлікту, згаданого вище:

Для Розробника або стандартного користувача системи буде доступний тільки мінімальний інтерфейс, що значно знизить навантаження та підвищить концентрацію на виконанні безпосередніх завдань користувача, що в свою чергу, підніме його продуктивність.

Для Проєктного Менеджера буде доступний максимальний набір інструментів та функціоналу (звіти, діаграми Ганта, створення та адміністрування проєктів і тд.), який буде прихованим або недоступним для інших ролей.

Для Sales/HR: можна додати окрему роль з спрощеним дешбордом, який містить лише загальну інформацію та базовий функціонал.

Отже, використання таких адаптивних користувацьких інтерфейсів в системі є обґрунтованим методом зменшення когнітивного навантаження на користувачів системи, пришвидшення виконання задач та зниження кількості помилок, так як користувач отримує тільки релевантну для нього інформацію [5].

Оцінюючи часові і фінансові затрати на проєктування та розробку даного програмного прототипу для валідації вищезгаданої гіпотези є економічно та практично обґрунтованими. Затрати на розробку прототипу, базованого на вже наявних напрацюваннях, буде значно нижчою ніж витрати на купівлю, впровадження та адміністрування повнофункціональних комерційних систем (як Jira) для всієї компанії, включаючи більшість працівників, які не потребують повного функціоналу.

Таким чином, проєктування та розробка розглянутого програмного прототипу є обґрунтованою, так як вона призначена для вирішення науково-практичного завдання усунення конфлікту між потребою в функціональній

повноті та простотою використання в програмному забезпеченні для управління проєктами шляхом застосування інтерфейсу орієнтованого на ролі користувачів.

Наступним кроком буде визначення очікуваних ефектів від розробленого програмного продукту. Впровадження спроєктованого та розробленого програмного прототипу в випадку його подальшого розвитку в комерційний проєкт, може привести до комплексного позитивного ефекту. Цей ефект можна конкретизувати в декількох напрямках:

1. Прогнозований економічний ефект: економічний ефект від розробки та впровадження системи з адаптивним інтерфейсом може формуватись на основі прямої та опосередкованої економії фінансів компаній.

Джерело 1 - пряма економія на ліцензуванні. Аналіз поточного ринку в попередніх розділах досить чітко продемонстрував, що вартість потужних та багатофункціональних систем як Jira, є досить високою та не завжди підходить малим та середнім компаніям. Зазвичай ця вартість розраховується по кількості користувачів, які будуть заведені в систему. Таким чином компанія з 50 користувачів повинна оплачувати повний функціонал, яким по факту користуватимуться 5 проєктних менеджерів та адміністратор. Враховуючи цей нюанс проєктований прототип, орієнтований на розподілення по ролях, дозволить створити гнучку модель ціноутворення на різну кількість специфічних ролей. Таким чином команда з великою кількістю спеціалістів, які потребують мінімуму від системи, отримає економічну вигоду від покупки такої системи.

Джерело 2 - опосередкована економія на навчанні персоналу. Часто приховані витрати на налаштування та впровадження складного програмного забезпечення, особливо на навчання співробітників, часто можуть перевищувати вартість безпосередньо ліцензії на програмне забезпечення [17]. Адаптивний інтерфейс проєктованого продукту матиме спрощений інтерфейс для ролей, які не потребують максимального функціоналу, що значно скоротить їх час на навчання та входження в систему і практично прибере проблему цих витрат. Також це значно скоротить навчання нових працівників та дозволить їм швидше ввійти в

робочий процес та почати виконувати свої безпосередні завдання, що можна вважати прямою економією зарплатного фонду компанії.

Наступним визначено організаційний ефект проєктованої системи. Впровадження адаптивного програмного середовища в компанії крім прямої та опосередкованої економічної вигоди несе за собою не менш важливий організаційний ефект, який виражено в зростанні продуктивності та оптимізації внутрішніх процесів в компанії. Сюди відноситься:

- підвищення продуктивності працівників: за результатами опитування, одним з найбільш відволікаючих факторів для більшості звичайних працівників(в даному випадку - розробників) є взаємодія з занадто навантаженим інтерфейсом. Проєктований прототип має за мету надати таким співробітникам лише той мінімум інструментів, який потребує його роль (як Канбан-дошка та список його завдань). Це дозволить значно зменшити його когнітивне навантаження, що знизить кількість помилок через неуважність та підвищить загальну швидкість виконання задач, так як розробник збереже концентрацію над кодом замість пошуку кнопки таймеру в системі до прикладу;
- прискорення комунікації і прийняття рішень в команді: адаптовані дешборди напряму вирішують проблему асинхронної комунікації між членами команди. Для менеджера замість формування складних JQL-запитів у Jira або очікування ручного звіту від команди, він може отримати необхідну аналітику шляхом натискання декількох кнопок. Так само Sales менеджер може легко отримати базову необхідну інформацію про проєкт самостійно, не відволікаючи проєктного менеджера або розробників. Це прискорює цикл прийняття рішень та покращує клієнтський сервіс;
- покращення прозорості робочих процесів: суцільний та гнучкий візуальний інструмент для відображення проєктів та їх статусу надає покращення загального розуміння стану речей для всієї команди. Це може бути вадливим для безпосередніх керівників чи до прикладу HR, якому потрібно розуміти «хто над чим працює» і дає можливість оперативно виявляти та усувати «вузькі місця» в робочих процесах, що є ключовим для Agile-методологій.

Наступним розглянуто технологічний ефект проєктованої системи. Він полягає не в розробці принципово нових алгоритмів, а в апробації та практичній реалізації пропонованої моделі адаптивного користувацького середовища.

Проєктований прототип буде використаний як технологічна база (proof-of-concept), яка має за мету довести технічну можливість реалізації системи, яка орієнтована на ролі користувачів. Також буде продемонстровано архітектуру(детальніше в Розділі 3), яка побудована для динамічної зміни інтерфейсу користувача залежно від його встановлених попередньо прав.

Даний прототип може вважатись не просто доказом концепції, а й моделлю для симуляції взаємодії користувача з продуктом такого типу [11]. Таким чином технологічний ефект буде полягати в створені готового прототипу програмного рішення, який можна використати в якості основи для подальшого розвитку та масштабування в повноцінний комерційний продукт.

Ергономічний та соціальний ефекти проєктованого прототипу напряму пов'язані з людино-орієнтованим проєктуванням, яке було визначено як методологічна основа дослідження в Розділі 1. Ергономіка проєктованого програмного забезпечення фокусується на проєктуванні інтерфейсів, що відповідають когнітивним та фізичним можливостям користувачів, мінімізуючи дискомфорт та максимізуючи ефективність [12].

Отже зниження когнітивного навантаження га користувача є ключовим ергономічним ефектом. Як показує дослідження, перевантажені функціоналом інтерфейси змушують багатьох учасників команди витрачати час та зусилля на пошук потрібної функції замість прямого виконання задачі. Якраз таки адаптивний інтерфейс призначений для мінімізації даної проблеми.

Як основний соціальний ефект варто відзначити зниження стресу та підвищення задоволеності роботою. Тобто інструмент, який не вважається інтуїтивно зрозумілим часто є щоденним джерелом стресу та фрустрації для співробітників. Запровадження простого, ергономічного і адаптивного середовища може підвищити загальну задоволеність співробітника робочим процесом і підвищити лояльність до компанії загалом.

Додатково також можна визначити деякі інші ефекти (соціальні та психологічні), які впливають на робоче середовище.

Одним з таких ефектів є покращення взаємодії між відділами та між конкретними користувачами між собою. Як показало дослідження, досить часто може виникати комунікаційний розрив між технічними та нетехнічними співробітниками та відділами. Проєктоване адаптивне середовище, яке отримує і збирає інформацію від кожного члена команди та кожної ролі в системі на зрозумілій їм мові (технічні деталі для одних, загальний статус для інших), може слугувати сполученням в комунікації та передачі даних, що значно покращує внутрішню взаємодію команди.

Загалом, підводячи підсумки постановки і обґрунтування проблеми розробки програмного середовища, було виконано формальну постановку та обґрунтовано проблеми розробки програмного прототипу, за вимогами методології. Також визначено мету, призначення та місця застосування проєктованої розробки.

Детально обґрунтовано потребу в створенні прототипу й готового продукту в подальшому, базуючись на виявленій проблематиці з попереднього розділу і емпірично підтвердженому конфлікті в потребах для різних користувачів системи та їх ролей відповідно.

Як завершення блоку обґрунтування проблеми проведено аналіз очікуваних економічних, організаційних, технологічних та ергономічних ефектів від впровадження. Даний аналіз підтвердив, що розробка прототипу доцільна, так як вона націлена на вирішення актуальних проблем реальних користувачів, які стосуються продуктивності роботи, комунікації між відділами і конкретними співробітниками між собою загалом, та зниження когнітивного навантаження співробітників, що знизить стрес під час виконання робочих задач та підвищить задоволення робочими процесами загалом.

2.3. Визначення вимог до програмного прототипу

Беручи за основу проведений вище системний аналіз та детальне обґрунтування проблеми, можна чітко визначити і зафіксувати всі деталізовані вимоги до проєктованого програмного прототипу. На основі цих вимог виконуватиметься подальше проєктування архітектури системи та безпосередньо практична реалізація прототипу.

За стандартною практикою в інженерії програмного забезпечення [52], буде визначено загальні вимоги, функціональні (що система повинна робити) і нефункціональні(як саме система має це робити) вимоги.

Як загальні системні вимоги визначено наступні:

- реалізація прототипу має бути в форматі веб-застосунку, який доступний через стандартні браузери (Google Chrome, Mozilla Firefox);
- за основу має братись клієнт-серверна архітектура, де frontend(тобто інтерфейс) логічно відокремлено від backend(серверна логіка та база даних);
- система має забезпечувати безперебійну роботу як мінімум 50 користувачів одночасно.

Наступними визначено функціональні вимоги до проєктованої системи. Ці вимоги описують визначені дії, які може виконувати користувач в системі. Функціональні вимоги базуються на Use Case діаграмі (рис. 2.3) та визначених для неї сценаріях (табл. 2.2-2.4). Щоб проєкт максимально відповідав головній меті дослідження, тобто адаптивне середовище, всі вимоги згруповано за ролями попередньо визначених акторів.

Вимоги до автентифікації (ВВ «Автентифікація»):

1. Система повинна мати сторінку входу в систему;
2. Система повинна валідувати логіни та паролі відповідно до бази даних;
3. Створювати сесію після авторизації та ідентифікувати роль користувача для адаптації інтерфейсу;
4. Виводити повідомлення про помилку при невдалій автентифікації.

Вимоги до ролі «Адміністратор» (ВВ «Управління користувачами»):

1. адміністратор повинен мати функціонал для перегляду списку всіх користувачів зареєстрованих в системі;

2. адміністратор має мати можливість додавати нових користувачів в базу даних системи, вказуючи щонайменше логін, тимчасовий пароль та роль цього нового користувача;

3. адміністратор має мати доступ до редагування даних існуючих користувачів системи (наприклад, змінювати роль);

4. адміністратор повинен мати можливість деактивувати (видаляти чи блокувати) облікові записи користувачів, таким чином забороняючи їм вхід в систему. Подібна система управління доступом на основі ролей є стандартом для корпоративних систем [10].

Вимоги до ролі «Проєктний Менеджер» для ВВ «Управління проєктами»:

1. проєктний менеджер повинен мати функціонал для створення нових проєктів, вказуюючи його назву, опис, дедлайн (як описано в табл. 2.3), титульне зображення за потреби(для кращої візуалізації дешборда);

2. проєктний менеджер повинен мати можливість змінювати інформацію та редагувати атрибути існуючого проєкту;

3. проєктний менеджер повинен мати можливість додавати та/або видаляти учасників проєкту шляхом вибору співробітника з списку існуючих користувачів системи(бази даних користувачів);

4. проєктний менеджер має мати змогу архівувати або закривати закінчені чи неактивні проєкти, щоб приховати їх з активних дешбордів.

Вимоги до ролі «Проєктний Менеджер» для ВВ «Управління завданнями»:

1. проєктний менеджер повинен мати функціонал для створення нових задач в рамках існуючого проєкту, вказуючи назву, детальний опис, пріоритет, терміни виконання та орієнтовний обсяг годин;

2. проєктний менеджер має призначати відповідального за нову задачу або змінювати відповідального за вже існуючу задачу, обираючи будь якого з всіх учасників проєкту;

3. проєктний менеджер повинен мати можливість встановити залежність між конкретними задачами, визначати ключову задачу, без якої неможливий початок чи завершення інших задач;

4. проєктний менеджер мусить мати швидкий доступ до візуального дешборда для відстеження активних проєктів та слідкування за статусом всіх відкритих задач;

5. проєктний менеджер має мати доступ до візуалізації часових рамок та залежностей в рамках проєкту.

Вимоги до ролі «Проєктний Менеджер» для ВВ «Перегляд звітів/дешбордів»:

1. система повинна забезпечити проєктного менеджера доступом до головного дешборду проєкту, який відображає його агреговані дані: загальний прогрес проєкту у відсотках, кількість відкритих/закритих задач, список учасників проєкту, звіт по завантаженості команди.

Вимоги до ролі «Розробник (Учасник команди)» в контексті ВВ «Управління завданнями» наступні:

1. учасник команди має бачити тільки спрощену версію дешбордів з проєктами та задачами, який відображає лише проєкти до яких він залучений та задачі які йому призначено;

2. учасник команди повинен мати можливість змінювати статус задач де він відповідальний;

3. учасник команди повинен мати можливість додавати коментарі до задач;

4. учасник команди має мати можливість додавати свій, та відстежувати вже доданий час, витрачений на задачу.

Вимоги до ролі «Розробник (Учасник команди)» в контексті ВВ «Перегляд звітів/дешбордів»:

1. учасник команди не має мати можливість переглядати чи змінювати загальні звіти по проєкту чи фінансах;

2. учасник команди може мати доступ тільки до власного персонального звіту, в який входить його затрачений на виконання задач час, та кількість цих виконаних задач.

Центральним блоком вимог до прототипу для дослідження, який випливає з безпосереднього обґрунтування проблеми є вимоги до інтерфейсу.

Ключові вимоги до Адаптивного UI:

1. система повинна автоматично ідентифікувати роль користувача;

2. загальний дешборд з проектами має бути орієнтованим на роль конкретного користувача і відображати інформацію та проєкти відповідно. Для ролі Менеджер: відображати повний дешборд з усіма проєктами, звітами, прогресом і тд., для ролі Розробник(Учасник команди) - спрощена варіація дешборда яка містить тільки його проєкти та задачі. Адміністратор відповідно повинен мати доступ до абсолютно всіх даних;

3. меню для навігації також повинне бути динамічним, тобто користувачі повинні бачити тільки ті пункти меню, до яких вони мають доступ відповідно до їх ролі та функціонал, який вони можуть використовувати.

Окремо також відокремлено нефункціональні вимоги, які описують критерії якості проєктованої системи. До них належать:

Вимоги до юзабіліті (зручності використання):

1. інтуїтивна зрозумілість - основні сценарії роботи з системою повинні виконуватись з мінімальною кількістю кліків;

2. простота освоєння - нові користувачі системи повинні виконувати свої базові задачі без попередньої підготовки та навчання, весь функціонал системи має бути легким в освоєні;

3. консистентність - всі елементи інтерфейсу (кнопки, меню, форми) повинні бути узгодженими по всій системі.

Вимоги до продуктивності:

1. час завантаження: час повного завантаження головної сторінки, основного дешборду або дошки конкретного проєкту не повинен перевищувати три секунди при стабільному інтернет-з'єднанні;

2. час відгуку: час реакції системи на основні дії користувача (наприклад, перетягування картки завдання, відкриття модального вікна) не повинен перевищувати 1 секунду.

Вимоги до безпеки:

1. авторизація на основі ролей - система має чітко обмежувати доступ до даних системи. Співробітник повинен мати доступ до інформації та функціоналу яка виключно відповідає його ролі. Доступ до редагування загальних даних про проєкти та користувачів системи повинна мати тільки визначена група користувачів Адміністратори;

2. захист даних: всі автентифікаційні дані користувачів(в тому числі паролі) повинні зберігатись в зашифрованому вигляді в базі даних, та обмежуватись в доступі до них;

3. система повинна мати базовий захист від вразливостей (наприклад, XSS-атак та SQL-ін'єкцій).

Вимоги до масштабованості (для прототипу):

1. архітектура розроблюваного прототипу повинна бути достатньо гнучкою щоб підтримувати розширення в майбутньому;

2. на етапі прототипу система має забезпечувати безперебійну роботу до 50 одночасно працюючих користувачів, та ефективне управління до 10 активних проєктів в системі.

Підсумовуючи визначення функціональних та нефункціональних вимог для проєктування та розробки прототипу системи для управління проєктами, отримано попереднє технічне завдання, яке буде використано в подальшому дослідженні та розробці. Функціональні вимоги до системи визначено на основі ролей акторів, які було ідентифіковано під час виконання системного аналізу. Як основу дослідження визначено ключові вимоги до системи, які зосереджено на реалізації адаптивного користувацького середовища. Нефункціональні вимоги визначили критерії якості, такі як юзабіліті, продуктивність та безпеку майбутньої системи. На даному етапі цих вимог достатньо щоб перейти до наступних етапів

дослідження - проєктування архітектури системи та безпосередньо реалізації прототипу.

2.4. Висновки до розділу 2

Підсумовуючи проведений системний аналіз та обґрунтування проблеми, як результат було сформовано повний набір вимог до проєктованого програмного прототипу системи для управління проєктами.

На першому етапі було проведено системний аналіз предметної області. За допомогою діаграм потоків даних (DFD) було визначено межі системи та її основні підпроцеси, такі як «Адміністрування користувачів», «Управління проєктами», «Управління завданнями» і «Формування звітів». Використовуючи UML-діаграму варіантів використання (Use Case) ідентифіковано акторів (Адміністратор, PM, Розробник) та візуалізовано їхню взаємодію з системою. Також проведено деталізацію ключових сценаріїв та визначено базові нефункціональні вимоги.

На другому етапі виконано постановку й обґрунтування проблеми. Доведено, що необхідність розробки такого прототипу базується на виявленій прогалині в ринку доступного програмного забезпечення та конфлікті потреб користувачів різних ролей. Також проаналізовано очікувані економічні, організаційні та ергономічні ефекти від впровадження пропонованого рішення.

Як завершальний етап сформульовано деталізовані вимоги до проєктованого прототипу. Сюди входять функціональні вимоги до системи відповідно до кожної з ролей користувачів та нефункціональні вимоги до юзабіліті, продуктивності, безпеки і масштабованості.

Отже даний розділ включає в себе повну аналітику та визначені вимоги, необхідні для переходу до наступного етапу - проєктування архітектури та вибору засобів реалізації прототипу.

РОЗДІЛ 3. МЕТОДИ ТА ЗАСОБИ ПРОЄКТУВАННЯ ПРОТОТИПУ

3.1. Вибір та обґрунтування методів проєктування системи

Проведений системний аналіз та отримані на його основі формалізовані вимоги до проєктованого програмного забезпечення можна вважати фундаментом для виконання наступного етапу - вибору методів проєктування та реалізації. В попередньому розділі було визначено що саме має бути спроєктовано та розроблено, то цей розділ призначений для визначення того, як саме це все буде відбуватись. Коректний вибір методів є одним з найкритичніших етапів щоб досягти поставленої мети дослідження та вирішити виявлені проблеми.

Обґрунтування методології проєктування:

Основна проблема, яку було виявлено та проаналізовано раніше має більше ергономічний ніж технічний характер. Вона визначається як високе когнітивне навантаження і конфлікт потреб для користувачів різних ролей в вже доступних на даним момент системах. Класичні методи проєктування, які орієнтовані в основному на функціонал не можуть вирішити дану проблему.

Враховуючи вище описане, в контексті даного дослідження, як основний метод проєктування користувацького середовища обрано людино-орієнтоване проєктування, яке було досліджено в підрозділі 1.2.

Вибір цього методу базується на тому, що дана методологія ставить в центр всього процесу розробки потреби, цілі та обмеження кінцевого користувача [25]. Це дає можливість спроєктувати систему, яка буде вважатись не «від технології до користувача» а «від користувача до технології», що можна вважати набагато ефективнішим способом для вирішення виявлених проблем юзабіліті користувачів та високого когнітивного навантаження.

Застосування людино-орієнтованого методу в даному дослідженні включає в себе наступні підходи до проєктування та реалізації прототипу системи для управління проєктами:

- ролеорієнтоване проєктування: вимоги сформовано не загальні для всіх користувачів системи, а для конкретних акторів;
- ітеративне прототипування: перед розробкою повноцінного продукту створюється прототип системи для перевірки проєктних гіпотез;
- емпірична валідація: аналіз успішності прототипу оцінюватиметься шляхом тестування на відповідність попередньовиявленим вимогам.

Щоб ефективно реалізувати всі виявлені раніше вимоги, необхідно використовувати чіткі методи структурування наявної проблеми.

Обґрунтування методів структурного аналізу:

Раніше, на етапі системного аналізу для проєктування було прийнято рішення використовувати методи DFD (діаграми потоків даних) і UML Use Case (діаграми варіантів використання). Їх було обрано завдяки їх здатності взаємодоповнююче описати проєктовану систему.

За допомогою діаграми потоків даних було візуалізовано систему з точки зору процесів та чітко продемонстровано рух даних між акторами, внутрішніми процесами та сховищами даних.

Діаграми варіантів використання дали можливість змоделювати систему з точки зору функціональності користувача шляхом чіткої фіксації того, що повинна робити система в контексті кожного з акторів.

Використання цих двох методів забезпечило створення повної картини вимог, що є критичним для подальшого проєктування.

Обґрунтування методу моделювання даних:

Наступний етап - вибір методу для організації сховищ даних визначених у DFD. Для даного прототипу найоптимальнішим рішенням обрано реляційний метод моделювання даних. Вибір базується на тому, що предметна область управління проєктами є високоструктурованою. Дані системи мають чіткі, довготривалі зв'язки: проєкт включає в себе багато задач; користувач має багато задачі; задача має багато коментарів. Реляційна модель найкраще забезпечує цілісність даних (Data Integrity) та консистентність (Consistency) цих зв'язків [49].

Наприклад, вона може на рівні бази даних заборонити видалення проєкту, поки в ньому є активні задачі, що критично важливо для надійності системи.

Обґрунтування методів реалізації та валідації:

Після визначення методології проєктування та моделювання даних наступним кроком необхідно обрати метод для практичної реалізації та перевірки запропонованих рішень.

Як метод реалізації та основну форму готового програмного рішення в рамках дослідження обрано прототипування. Прототип дає можливість відповісти на питання як саме можна використовувати цей продукт і на скільки це зручно для звичайного середньостатистичного користувача. Так як головною метою дослідження є не перевірка можливості технічної реалізації чи запуск повноформатного продукту, а підтвердження гіпотези про ефективність роле-орієнтованого підходу для таких систем, то метод прототипування є найоптимальніший в цьому випадку. Його головна перевага це можливість швидкої перевірки основних проєктних рішень та збір зворотнього зв'язку від користувачів з мінімальними затратами часу та ресурсів.

Для верифікації розробленого прототипу доцільно використовувати метод функціонального тестування. Він дасть можливість перевірити чи реалізовані функції прототипу відповідають попередньо визначеним вимогам.

Для валідації обрано метод тестування юзабіліті, який дозволяє перевірити прототип на реальних користувачах щоб оцінити, чи спроєктований адаптивний інтерфейс буде зручнішим та чи знизить він когнітивне навантаження на нового непідготовленого користувача.

Отже підсумовуючи, було обрано та обґрунтовано ключові методи для вирішення виявленої попередньо проблеми. Як основну методологію проєктування обрано людино-орієнтоване проєктування; структурний аналіз (DFD, Use Case) для моделювання системи; реляційну модель бази даних; прототип як формат реалізації та методи його тестування. Цей набір методів є оптимальним для досягнення основної мети в рамках дослідження.

3.2. Обґрунтування архітектури програмного прототипу

На основі попередньо визначених вимог та вибраних ключових методів проєктування і моделювання наступним логічним кроком дослідження буде вибір і обґрунтування архітектури проєктованого прототипу. Основна вимога до архітектури, яка буде використовуватись надалі - це ефективно підтримувати реалізацію всіх визначених вимог.

Для реалізації прототипу було обрано клієнт-серверну архітектуру. Такий вибір базується на попередньо визначених вимогах. Так як майбутній прототип є багатокористувацьким веб-додатком, де користувачі з різними ролями (проєктний менеджер, розробник, адміністратор) мають мати одночасний, спільний та консистентний доступ до загального набору даних системи (проєкти, задачі, користувачі, тощо.).

Відповідність клієнт-серверної архітектури визначеним вимогам полягає в:

- централізації даних: весь обсяг інформації в системі зберігається на сервері. Це дозволяє використовувати обрану раніше важливу для дослідження реляційну модель даних, яка гарантує їх цілісність та робить сервер єдиним «джерелом правди»;
- розподілення відповідальності: це підхід, який дозволяє чітко відокремити логіку презентації (клієнт), яка працює в браузері користувача, від бізнес-логіки та логіки даних (сервер) [24];
- підтримці роле-орієнтованості: таке розділення є критично важливим для підтримання головної мети дослідження - розділення інтерфейсу та повноти функціоналу відповідно до ролей користувачів. Архітектура в цьому випадку дозволяє зосередитись на розробці гнучкого клієнту інтерфейсу, який буде мати можливість адаптуватись до різних ролей, в той час як сервер буде обробляти основну бізнес-логіку та забезпечуватиме безпеку системи шляхом перевірки прав доступу для кожної ролі і для кожного запиту до бази даних.

Отже після визначення клієнт-серверної архітектури як основи для розробки проєктованої системи для досягнення головної мети дослідження, потрібно

вибрати метод для комунікації між цими двома компонентами. Для цього прийнято рішення використовувати архітектурний стиль RESTful API (Representational State Transfer). Це стандартизований, гнучкий та легкий підхід до взаємодії, що базується на протоколі HTTP.

Головна перевага REST API в контексті даного дослідження - це можливість реалізовувати роле-орієнтовану безпеку (NFR 3.0). Таким чином можливо захистити окремі «ендпоінти» (наприклад, DELETE /api/projects/{id} або POST /api/users) на рівні сервера.

Тоді, навіть якщо буде здійснюватись модифікація клієнтського інтерфейсу, сервер не дозволятиме користувачу виконувати таку дію, якщо автентифікований користувач не буде мати відповідну для цієї дії роль. Крім того, REST є «stateless» (без стану), що означає, що кожен запит від клієнта містить всю необхідну для виконання інформацію. Це спрощує масштабування системи та ізолює логіку клієнта від сервера [43].

Щоб організувати код на стороні сервера, який оброблятиме всі ці REST-запити, було прийнято рішення використовувати класичний, але ефективний паттерн Model-View-Controller (MVC). Він дуже добре взаємодіє та доповнює клієнт-серверну архітектуру, так як він чітко розділяє відповідальність всередині серверного застосунку:

- Model (модель): повністю ізолює бізнес-логіку і роботу з базою даних. Відповідає за дотримання цілісності даних.
- Controller (контролер): приймає вхідні HTTP-запити від REST API, валідує дані, перевіряє права доступу і викликає відповідні методи Моделі.
- View (вид): в випадку REST API, вид не генерує HTML. Тут, він відповідає за форматування даних, отриманих від моделі, у стандартизований JSON-формат для відправки назад клієнту.

Отже, об’єднуючи всі обрані вище методи (клієнт-серверна архітектура), паттерни взаємодії (REST API) і організації серверної логіки (MVC) сформовано єдину архітектурну схему проєктованої системи(рис. 3.1).

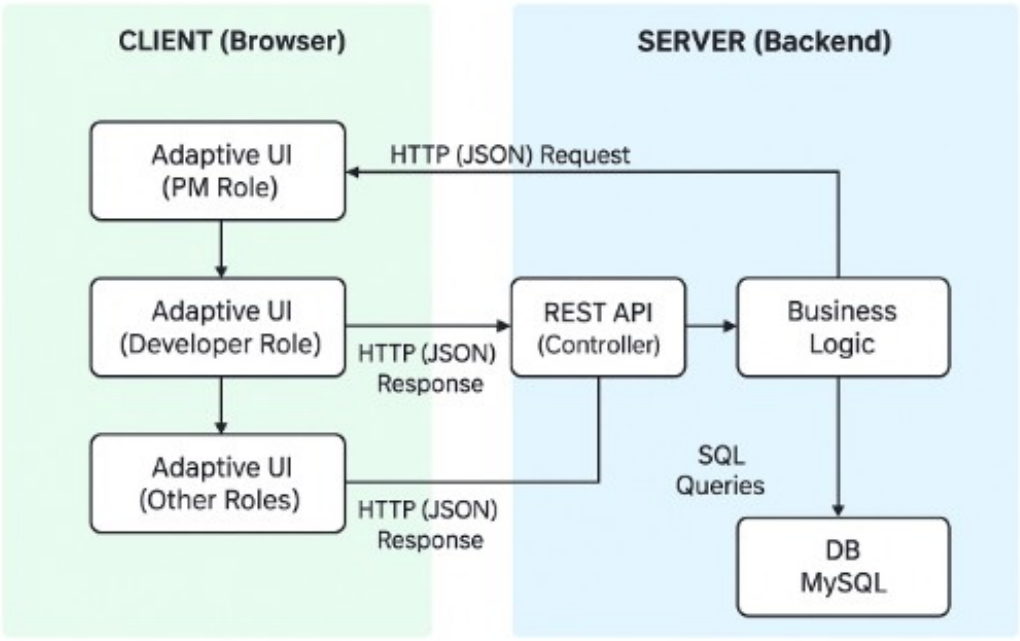

Рисунок 3.1 - Загальна архітектура програмного прототипу

Отже отримано наступну логіку роботи системи:

- Клієнт (браузер користувача) завантажує адаптивний UI, що відповідає ролі користувача (PM, Розробник тощо);
- При будь-якій дії (наприклад, Розробник перетягує картку завдання) Клієнт формує та відправляє HTTP-запит на сервер через REST API;
- Controller (API) на сервері приймає цей запит, валідує його та перевіряє права доступу (безпеку);
- Controller викликає відповідний метод Model (бізнес-логіка);
- Model виконує необхідні операції та надсилає SQL-запит до БД;
- Дані повертаються по ланцюжку назад: БД -> Model -> Controller;
- Controller формує JSON-відповідь та відправляє її клієнту.

3.3. Вибір та обґрунтування засобів розробки

На основі обраних раніше методів проєктування та визначеного та сформованого архітектурного каркасу тепер, як останній крок перед реалізацією,

необхідно визначити конкретні програмні засоби(технології), які будуть використовуватись безпосередньо для реалізації прототипу.

Для реалізації серверної частини (Backend) та коду загалом, що включає в себе REST API і бізнес-логіку системи, було прийнято рішення використовувати мову програмування PHP (Hypertext Preprocessor).

Таке рішення було прийнято по наступних причинах:

1. Відповідність архітектурі - на сьогоднішній день, мова PHP вважається однією з провідних мов в контексті веб-розробки і має хорошу сумісність з клієнт-серверною архітектурою. Вона добре працює у зв'язці з веб-сервером Apache (для обробки HTTP-запитів) і має багато різних бібліотек і фреймворків (як Laravel, Symfony), що побудовані саме на паттерні MVC. Це дозволяє чітко реалізувати розділення на контролери (для REST API) та моделі (для бізнес-логіки).

2. Загальна популярність, широка спільнота - технологія PHP має відкритий вихідний код, і дає можливість розробки прототипу без вкладень. Мова вже існує досить довгий часі на даний момент є популярною та стабільною і включає в себе одну з найбільших спільнот розробників у світі[56]. Це в свою чергу забезпечує великою кількістю доступної документації, готових рішень і навчальних матеріалів, що знижує ризики та прискорює розробку.

3. Ефективність для розробки прототипу - використання мови PHP дає можливість досить ефективно перевикористовувати і модернізовувати вже існуючі програмні напрацювання, які могли бути реалізовані раніше. Це дає змогу не витрачати багато часу на реалізацію прототипу для валідації теорій дослідження та адаптовувати його відповідно до результатів тестування. Це ж дозволить в майбутньому доопрацювати прототип до повноцінного продукту.

Щоб забезпечити швидку та стабільну роботу системи, в контексті обраної мови програмування і її взаємодії з базою даних, потрібно обрати відповідне до вимог серверне оточення. Щоб успішно забезпечити всі цілі розробки і валідації проєктованого прототипу було обрано програмний пакет XAMPP.

Такий вибір базується на його комплексності та орієнтованості на швидке прототипування. XAMPP - це вільний, крос-платформовий дистрибутив, який включає в себе весь необхідний в рамках дослідження стек: Apache (веб-сервер для обробки HTTP-запитів), PHP (мова логіки) і MySQL/MariaDB (сервер бази даних).

Головною перевагою даного програмного засобу є уникнення складного та часозатратного процесу окремого встановлення та конфігурування кожного з цих компонентів. Це дає можливість зекономити час на системному адмініструванні сервера та присвятити більше часу безпосередньо проєктуванню та розробці, включаючи бізнес-логіку і користувацьке середовище прототипу що забезпечить швидке і ефективне досягнення мети дослідження[8].

Наступним кроком після вибору і аналізу серверного середовища є вибір та обґрунтування системи управління базами даних. Для реалізації попередньо визначених у DFD-аналізі сховищ даних (D1, D2, D3), та відповідно до обраного раніше реляційного методу, прийнято рішення використовувати СУБД MySQL.

Вибір MySQL обґрунтовується наступними причинами:

1. Відповідність меті дослідження: MySQL вважається практично однією з найпотужніших, надійних та поширених реляційних СУБД, що гарантує високу продуктивність при роботі зі структурованими даними, забезпечує їх цілісність (Data Integrity) та безпеку, що надзвичайно важливо для предметної області управління проєктами.

2. Технологічна сумісність: MySQL є нативним та де-факто стандартом для роботи у зв'язці з PHP. Їхня глибока інтеграція (як у пакеті XAMPP) забезпечує стабільну та безпроблемну взаємодію між бізнес-логікою (сервером) і базою даних згідно визначеної раніше архітектури.

Щоб мати змогу об’єднати та використовувати всі попередні засоби обрані вище необхідно вибрати інтегроване середовище розробки (IDE) для написання, відлагодження і керування кодом спроєктованого прототипу. На основі аналізу, для цього прийнято рішення використовувати IDE PhpStorm.

На відміну від простих текстових редакторів, вибір професійної IDE для розробки прототипу обґрунтовується необхідністю забезпечення високої якості коду та прискорення процесу реалізації. PhpStorm надає критично важливі для досягнення мети дослідження інструменти:

1. Інтелектуальне автодоповнення коду: це значно прискорює написання коду на PHP та JavaScript, а також дозволяє зручно модернізувати (проводити рефакторинг) код після тестування.

2. Інструменти для відладки (Debugger): вбудований дебаггер може бути ключовою перевагою для швидкого пошуку та виправлення помилок у складній бізнес-логіці системи.

3. Статичний аналіз коду: IDE може автоматично аналізувати код під час його написання і попереджати користувача про можливі помилки, невідповідності стандартам кодування або проблеми з продуктивністю ще до етапу запуску програмного забезпечення та його тестування.

Отже, на даному етапі було обрано та обґрунтовано конкретні програмні засоби (технологічний стек) для безпосередньої програмної реалізації прототипу.

Підсумовуючи вищепроведений аналіз прийнято рішення використовувати мову програмування PHP, СУБД MySQL, локальне середовище XAMPP та IDE PhpStorm. Така комбінація програмних засобів є технічно обґрунтованою та збалансованою, так як вона досить часто зустрічається в розробці програмного забезпечення і є перевіреною та надійною.

Також всі ці програмні засоби можна використовувати в вільному доступі або безкоштовному тарифі, що дозволяє розробити прототип без фінансових витрат.

3.4. Висновки до розділу 3

В результаті проведення огляду методів та засобів проєктування прототипу було обрано і теоретично обґрунтовано методології, архітектуру та засоби для реалізації програмного прототипу, що відповідають попередньо визначеним вимогам до систему для управління проєктами.

Першочергово було обрано методи для вирішення виявленої проблеми. Як ключову методологію обрано людино-орієнтоване проєктування, так як воно найкраще вирішує виявлену раніше проблему конфлікту ролей і когнітивного навантаження на користувача. Для моделювання даних обрано реляційний метод, а як метод реалізації - прототип.

Наступним кроком було обґрунтування архітектури прототипу. В результаті було обрано клієнт-серверну архітектуру, яка доповнюватиметься паттерном взаємодії RESTful API та серверним паттерном MVC. Така комбінація забезпечить розподілення відповідальності та реалізує роле-орієнтовану безпеку.

На завершення огляду методів та засобів було обрано конкретні засоби для програмної реалізації прототипу. Для цього прийнято рішення використовувати мову програмування PHP, СУБД MySQL, локальне середовище XAMPP і IDE PhpStorm. Це поширена комбінація стеку для подібних проєктів що дозволить реалізувати прототип швидко, ефективно та фінансово не затратно.

Отже підсумовуючи, після огляду методів та засобів було сформовано повний методологічний, архітектурний та інструментальний базис, необхідний для переходу до практичної реалізації програмного прототипу.

РОЗДІЛ 4. ПРОЄКТУВАННЯ І РЕАЛІЗАЦІЯ ПРОТОТИПУ СИСТЕМИ

4.1. Проєктування структури бази даних прототипу

Реалізація спроєктованого прототипу програмного забезпечення на практиці починається з формування його бази даних. Для цього будуть використовуватись засоби, теоретично обрані в попередніх розділах, а саме застосування реляційного методу та СУБД MySQL.

Головна мета правильно спроєктованої бази даних полягає в створенні такої логічної і фізичної структури даних, яка буде ефективно зберігати інформацію і надійно підтримувати виконання всіх функціональних та нефункціональних вимог визначених в дослідженні. Найбільша увага в процесі проєктування бази даних повинна бути приділена роле-орієнтованому доступу до системи, що реалізуватиме головну мету дослідження.

Проєктування бази даних для системи управління проєктами повинно дотримуватись ключових принципів реляційної теорії [23]:

- нормалізація: Структура таблиць проєктується з метою уникнення надлишковості даних (Redundancy) та аномалій оновлення. Для проєктування бази даних використовуються щонайменше перша, друга та третя нормальні форми (1NF, 2NF, 3NF);
- цілісність даних (Data Integrity): Забезпечується шляхом чіткого визначення первинних ключів (PK) для ідентифікації сутностей та зовнішніх ключів (FK) для підтримки логічних зв'язків між таблицями (наприклад, зв'язок між проєктом та його завданнями);
- атомарність: Поля таблиць проєктуються так, щоб зберігати атомарні (неподільні) значення, що спрощує запити та підтримку бази даних.

Базуючись на цих принципах та попередньо проведеному системному аналізу, було визначено ключові сутності, які необхідно реалізувати в базі даних: Користувачі, Ролі, Проєкти, Задачі і допоміжні сутності(статуси, коментарі і тд.).

Враховуючи попередньо визначені вимоги до прототипу з попередніх розділів, для того щоб реалізувати ключову гіпотезу про адаптивне середовище, ядро проєктованої бази даних повинно мати гнучку систему щоб керувати ролями. Для цього буде створено три пов'язані таблиці.

Таблиця Users(табл. 4.1) зберігатиме основні облікові дані для автентифікації та персональну інформацію користувачів.

Таблиця 4.1 - Структура таблиці Users

Назва поля	Тип даних	Призначення
user_id	INT (PK, AI)	Унікальний ідентифікатор користувача
username	VARCHAR(100)	Логін для входу в систему
password_hash	VARCHAR(255)	Хешований пароль
email	VARCHAR(255)	Електронна пошта
full_name	VARCHAR(255)	Повне ім’я користувача

Таблиця Roles(табл. 4.2) буде зберігати визначені в системі ролі(як додані по дефолту так і новостворені), що дозволить легко додавати нові ролі в майбутньому без зміни структури БД.

Таблиця 4.2 - Структура таблиці Roles

Назва поля	Тип даних	Призначення
role_id	INT (PK, AI)	Унікальний ідентифікатор ролі
role_name	VARCHAR(50)	Назва ролі («Адміністратор», «PM», «Розробник»)

Таблиця User_Roles(табл. 4.3) потрібна для того, щоб зв'язувати попередні таблиці між собою, та реалізувати відношення «багато-до-багатьох» (N:M) між користувачами та ролями.

Таблиця 4.3 - Структура таблиці User_Roles

Назва поля	Тип даних	Призначення
user_id	INT (FK)	Посилання на Users(user_id)
role_id	INT (FK)	Посилання на Roles(role_id)

Така структура є гнучкою та масштабованою реалізацією контролю доступу до системи на основі ролей, що є стандартом для сучасних корпоративних систем та ключовою вимогою прототипу.

Наступним визначеним блоком таблиць, які описують систему є проєкт. Ці таблиці призначені для реалізації функціональних вимог системи, які були визначені в попередніх розділах.

Таблиця Projects(табл. 4.4) зберігає основну інформацію про кожен проєкт, створений у системі. Вона вважатиметься центральною сутністю для групування задач і проєктних команд.

Таблиця 4.4 - Структура таблиці Projects

Назва поля	Тип даних	Призначення
project_id	INT (PK, AI)	Унікальний ідентифікатор проєкту
project_name	VARCHAR(255)	Назва проєкту (вимога FR 3.1.1)
description	TEXT	Детальний опис проєкту
start_date	DATE	Запланована дата початку
deadline	DATE	Запланована дата завершення (дедлайн)
owner_id	INT (FK)	Посилання на Users(user_id), вказує хто (PM) створив проєкт
project_status	VARCHAR(50)	Поточний статус («Активний», «Архівований»)

Таблиця Project_Members(табл. 4.5) зв'язуюча таблиця, яка реалізуватиме відношення «багато-до-багатьох» (N:M) між проєктами та користувачами. Вона є ключовою для реалізації вимоги додавання учасників до проєкту і для визначення, хто з розробників має бачити завдання проєкту.

Таблиця 4.5 - Структура таблиці Project_Members

Назва поля	Тип даних	Призначення
project_id	INT (FK)	Зовнішній ключ, посилається на Projects(project_id)
user_id	INT (FK)	Зовнішній ключ, посилається на Users(user_id)

Таким чином, система може ідентифікувати склад команди для будь-якого проєкту та, навпаки, показати користувачу всі проєкти, в яких він бере участь.

Найбільш динамічною частиною системи для управління проєктами буде блок управління задачами. Тут буде відбуватись основна взаємодія користувачів з системою (додавання нових ,зміна статусів, коментування і тд.). Для реалізації цього функціоналу необхідно мати дві ключові таблиці:

Таблиця Tasks(табл. 4.6) - це основна операційна таблиця, яка зберігатиме всі задачі в системі. Тут повинні бути зв’язки (Foreign Keys) майже з усіма іншими сутностями (проєктами, користувачами, статусами).

Таблиця 4.6 - Структура таблиці Tasks

Назва поля	Тип даних	Призначення
task_id	INT (PK, AI)	Унікальний ідентифікатор задачі.
project_id	INT (FK)	Посилання на Projects(project_id). Вказує, до якого проєкту належить задача.
title	VARCHAR(255)	Короткий заголовок задачі.
description	TEXT	Детальний опис технічного завдання.
status_id	INT (FK)	Посилання на Statuses(status_id). Визначає поточний етап задачі.
priority_id	INT (FK)	Посилання на Priorities(priority_id). Рівень пріоритету.
creator_id	INT (FK)	Посилання на Users(user_id). Хто створив задачу.
due_date	DATE	Дедлайн виконання задачі.
created_at	TIMESTAMP	Дата й час створення (автоматично).

Таблиця Task_Assignees(табл. 6.7) призначена для зберігання даних про призначеного виконавця задачі. Для підтримки гнучких методологій та реальних сценаріїв командної роботи (наприклад, парне програмування або спільна робота над задачею), прототип має підтримувати призначення кількох виконавців на одне завдання. Для цього використовується таблиця зв'язку «багато-до-багатьох».

Таблиця 4.7 - Структура таблиці Task_Assignees

Назва поля	Тип даних	Призначення
task_id	INT (FK)	Посилання на Tasks(task_id).
user_id	INT (FK)	Посилання на Users(user_id).

Для забезпечення повноцінного функціонування системи та реалізації специфічних вимог (як-от Канбан-дошка або облік часу) необхідно спроєктувати ряд допоміжних таблиць.

Таблиця Statuses(табл. 4.8) є критичною для візуалізації Канбан-дошки. Сюди відносяться колонки, по яких будуть рухатись задачі.

Таблиця 4.8 - Структура таблиці Statuses

Назва поля	Тип даних	Призначення
status_id	INT (PK, AI)	Унікальний ідентифікатор статусу.
status_name	VARCHAR(50)	Назва статусу (напр., *«To Do», «In Progress», «Testing», «Done»).*
project_id	INT (FK) *(Nullable)*	Прив’язка до конкретного проєкту для кастомних воркфлоу.
sort_order	INT	Порядок відображення статусу на Канбан-дошці (зліва направо).

Таблиця Comments(табл. 4.9) дає змогу реалізувати вимогу до можливості коментування задач. Таблиця призначена щоб зберігтиє історію спілкування в контексті конкретного завдання.

Таблиця 4.9 - Структура таблиці Comments

Назва поля	Тип даних	Призначення
comment_id	INT (PK, AI)	Унікальний ідентифікатор коментаря.
task_id	INT (FK)	Посилання на Tasks(task_id). Вказує, до якої задачі належить коментар.
user_id	INT (FK)	Посилання на Users(user_id). Хто залишив коментар.
comment_text	TEXT	Текст повідомлення.
created_at	TIMESTAMP	Дата та час створення коментаря.

Таблиця Time_Entries(табл. 4.10) буде реалізовувати вимогу по створенню персональних звітів для будь якого з менеджерів чи співробітників і дозволить фіксувати витрати часу.

Таблиця 4.10 - Структура таблиці Time_Entries

Назва поля	Тип даних	Призначення
entry_id	INT (PK, AI)	Унікальний ідентифікатор запису.
task_id	INT (FK)	Посилання на Tasks(task_id), де конкретно виконувалась робота.
user_id	INT (FK)	Посилання на Users(user_id). Хто звітує про виконану роботу.
hours_spent	DECIMAL(5,2)	Кількість витрачених годин (наприклад, 1.5).
log_date	DATE	Дата виконання роботи.

Отже, всі описані вище сутності та їхні зв'язки можна об’єднати у єдину фізичну модель. Для візуалізації структури бази даних прототипу сформовано діаграму «сутність-зв'язок»(рис. 4.1).

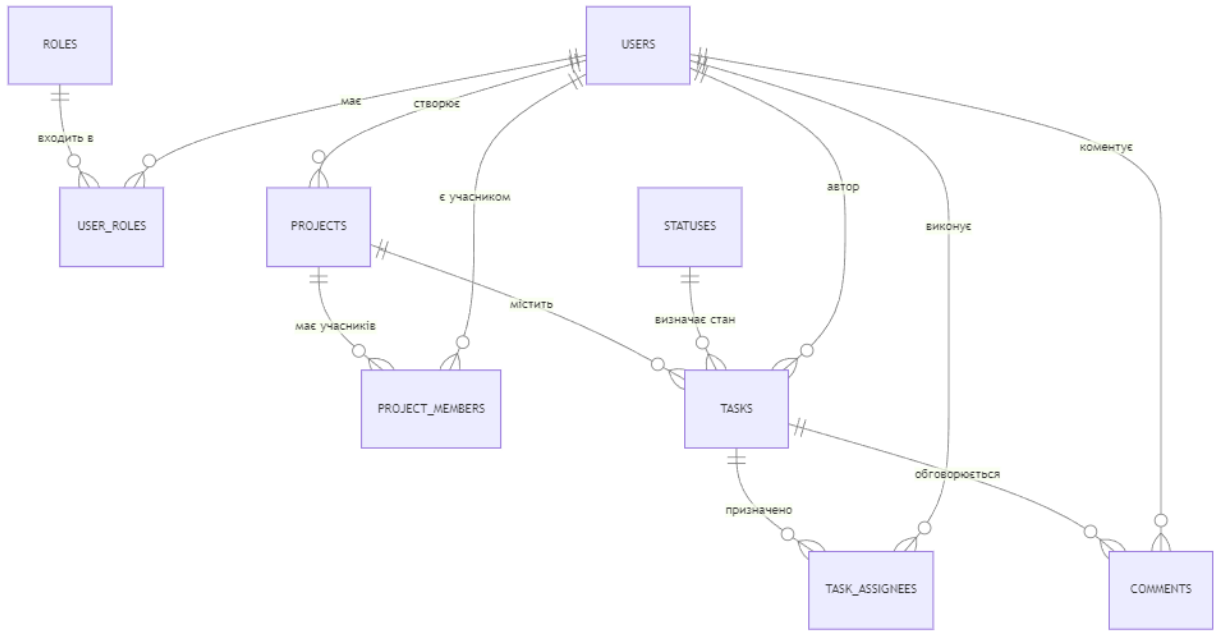

Рисунок 4.1 - Фізична модель бази даних (ER-діаграма)

Діаграма демонструє, що спроєктована база даних знаходиться у третій нормальній формі: усі неключові атрибути залежать лише від первинного ключа, що виключає надлишковість даних. Зв'язки між таблицями забезпечено механізмом зовнішніх ключів, що гарантує посилальну цілісність на рівні СУБД MySQL.

4.2. Опис реалізації ключових функціональних модулів прототипу

На основі проведеного дослідження та вибору засобів, було практично реалізовано програмний прототип, з використанням мови PHP, СУБД MySQL і технологій HTML/CSS/JavaScript у середовищі XAMPP. Проєкт побудовано за модульною архітектурою, що дозволяє чітко розділити логіку підключення до даних, обробку запитів користувача та відображення інтерфейсу.

Структура файлової системи проєкту організована таким чином, щоб відображати логічну структуру застосунку. Коренева директорія проєкту містить наступні ключові папки та файли:

- includes/: містить системні файли і повторювані елементи інтерфейсу;
- db.php: файл конфігурації та підключення до бази даних;
- auth.php, login.php: модулі для автентифікації і перевірки сесій;
- header.php, sidebar.php, topbar.php: шаблони шапки, бокового меню та верхньої панелі, які використовуються для забезпечення єдиного стилю сторінок;
- pages/: папка містить файли відображення (Views), тобто окремі PHP-скрипти для кожної сторінки інтерфейсу (наприклад, home.php для дешборду, project_list.php для списку проєктів);
- forms/: скрипти-обробники (add_forms.php, update_forms.php), потрібні для прийому даних з HTML-форм методом POST, їх валідації і виконують CRUD-операції (Create, Read, Update, Delete) над базою даних;
- assets/, css/, js/, fonts/: папки для зберігання статичних ресурсів: стилів Bootstrap, користувацьких скриптів JavaScript, бібліотек та шрифтів.

Одним з найважливіших елементів проєкту є реалізація підключення системи до бази даних. Взаємодія з базою даних була реалізована через єдиний файл конфігурації includes/db.php.

Такий підхід до підключення дає можливість централізувати налаштування доступу (хост, ім'я користувача, пароль, назва бази даних). Файл використовує бібліотеку mysqli для встановлення з'єднання.

```
<?php
$servername = "localhost";
$username = "root";
$password = "";
$dbname = "construction_pms_db";

$conn = mysqli_connect($servername, $username, $password, $dbname);
if (!$conn) {
    die("Connection failed: " . mysqli_connect_error());
}
?>
```

Цей файл (db.php) підключається на початку кожного скрипта, який потребує доступу до даних (у папках pages/, forms/ та includes/), що забезпечує стабільне з'єднання протягом усього сеансу роботи користувача.

Наступний етап - це реалізація модуля автентифікації. Захист доступу до системи та ідентифікація користувачів реалізовані у файлі includes/login.php. Цей скрипт виступає контролером, який приймає дані з форми входу (через AJAX-запит), перевіряє їх валідність та ініціалізує сесію користувача.

```
<?php
/* includes/login.php */
session_start();
include('db.php');
$user = $_POST['user'];
$pass = $_POST['pass'];

$query = mysqli_query($conn,"SELECT * FROM users natural join employee where
username='$user' and password='$pass' and io='1'");
$count = mysqli_num_rows($query);

while($row = mysqli_fetch_assoc($query)){
    if($count > 0) {
        // Оновлення ID сесії для безпеки
        session_regenerate_id();

        $name = $row['firstname'] . ' ' . $row['lastname'];
        $_SESSION['ID'] = $row['eid'];      // ID співробітника
        $_SESSION['UID'] = $row['uid'];     // ID користувача
        $_SESSION['TYPE'] = $row['user_type']; // Роль (1=Admin, 2=Staff)
        $_SESSION['NAME'] = $name;          // Повне ім'я для відображення

        echo "true";
 }?>
```

Цей алгоритм роботи модуля призначений для виконання декількох основних та критичних для системи функцій, таких як перевірка облікових даних

користувачів, захист активної сесії, ініціалізація ролі користувача та взаємодія системи з клієнтом.

Для захисту внутрішніх сторінок від несанкціонованого доступу використовується модуль includes/auth.php, який перевіряє наявність встановленої змінної $_SESSION['ID']. Якщо змінна відсутня, скрипт примусово перериває виконання та перенаправляє користувача на сторінку входу.

Основна реалізація вимоги про адаптивний інтерфейс знаходиться в файлі includes/sidebar.php. Цей модуль відповідає за генерацію бокового навігаційного меню. Замість того, щоб створювати окремі файли меню для кожної ролі, використовується підхід динамічного рендерингу на основі значення змінної сесії $_SESSION['TYPE'], яка була ініціалізована на етапі автентифікації.

```
      <div class="collapse navbar-collapse navbar-ex1-collapse">
          <ul class="nav navbar-nav side-nav styled-sidebar">
              <li>
                  <a href="index.php?page=home">
                      <i class="fa fa-fw fa-dashboard"></i> <span>Dashboard</span>
                  </a>
              </li>
              <li>
                  <a href="index.php?page=employee&io=1">
                      <i class="fa fa-fw fa-users"></i> <span>Employee List</span>
                  </a>
              </li>
              <li>
                  <a href="index.php?page=project_list&io=1">
                              <i class="fa fa-fw fa-files-o"></i> <span>Project
List</span>
                  </a>
              </li>

              <?php if($_SESSION['TYPE'] == 1): ?>
                  <li>
                      <a href="index.php?page=user_list&io=1">
                          <i class="fa fa-fw fa-user"></i> <span>Users</span>
                      </a>
                  </li>
              <?php endif; ?>


              <li>
                     <a id="dem3" href="javascript:;" data-toggle="collapse" data-
target="#demo3">
                      <i class="fa fa-fw fa-cogs"></i> <span>Maintenance</span>
                      <i class="fa fa-fw fa-caret-down pull-right"></i>
                  </a>
                  <ul id="demo3" class="collapse">
                      <li><a href="index.php?page=position">Position</a></li>
                                  <li><a  href="index.php?page=division">Project
Sections</a></li>
```

```
                                <li><a  href="index.php?page=project_team">Project
Team</a></li>
                    </ul>
                </li>
            </ul>
        </div>
```

Тут конструкція if($_SESSION['TYPE'] == 1) перевіряє права користувача. Якщо користувач має роль «Staff» (TYPE != 1), HTML-код посилання на сторінку «Users» навіть не генерується сервером і не відправляється у браузер.

Таким чином вирішуються дві задачі: безпека системи та юзабіліті користувачів з різними правами доступу.

Організація навігації в системі побудована за принципом Front Controller. Всі запити користувача обробляються одним головним файлом index.php, який виконує функцію маршрутизатора в системі. Це дозволяє підключати спільні елементи інтерфейсу (шапку, меню, підвал) в одному місці, уникаючи дублювання коду.

У файлі index.php реалізовано логіку, яка перевіряє GET-параметр page і на його основі підключає відповідний файл з папки pages/.

```
<?php
session_start();

if(!isset($_SESSION['login_id']))
    header('location:login.php');

include 'db_connect.php';
include 'includes/header.php';

echo '<body class="hold-transition sidebar-mini layout-fixed">';
    include 'includes/topbar.php';
    include 'includes/sidebar.php';

    $page = isset($_GET['page']) ? $_GET['page'] : 'home';
    if(file_exists('pages/'.$page.'.php')){
        include 'pages/'.$page.'.php';
    }else{
        include '404.html';
    }
    include 'includes/footer.php';
echo '</body>';
?>
```

Тут можна побачити, як працює навігація прописана в sidebar.php. Після кліку користувача на посилання index.php?page=project_list, контролер перехоплює цей запит і підключає файл pages/project_list.php всередину основного

макету. Для забезпечення єдиного візуального стилю та роботи інтерактивних елементів, всі CSS-стилі та бібліотеки підключаються у файлі includes/header.php.

У прототипі використано стек фронтенд-технологій: Font Awesome, Google Fonts, AdminLTE / Bootstrap, DataTables.

Після успішної авторизації та обробки маршруту в index.php, користувач потрапляє на головну сторінку системи - дешборд (файл pages/home.php). Цей модуль фактично є головною сторінкою системи(рис. 4.2).

Інтерфейс дешборду розділений на три логічні зони: статистичні віджети, активні проєкти і сповіщення.

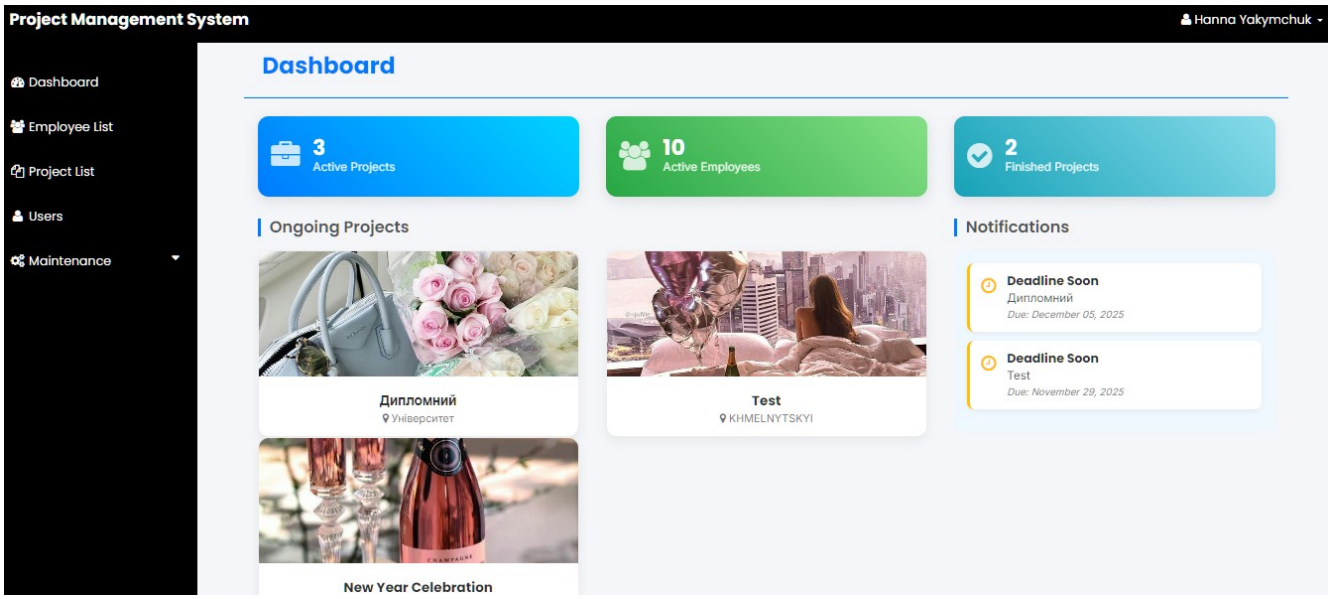

Рисунок 4.2 - Головний дешборд системи

Реалізація статистичних віджетів картки з статистикою реалізовані шляхом виконання прямих SQL-запитів до бази даних. Використання градієнтних фонів (bg-primary-gradient) та іконок дозволяє візуально розділити типи даних, що покращує сприйняття інформації.

Для проактивного управління ризиками реалізовано алгоритм перевірки дедлайнів. Система автоматично сканує всі активні проєкти та порівнює їхню дату завершення з поточною датою.

```
<?php
```

```
$d1 = date("Ymd", strtotime($row['deadline'].' -15 days'));
$d2 = date("Ymd");

if($d2 >= $d1 && $deadline_date >= $d2){
    echo '<div class="alert-card warning">...</div>';
} elseif($deadline_date < $d2){
    echo '<div class="alert-card danger">...</div>';
}
?>
```

Цей функціонал гарантує, що менеджер не пропустить критичні терміни, оскільки система сама підсвічує проблемні проєкти червоним кольором.

Однак найважливішим елементом на дешборді залишається саме список активних проєктів. Ці проєкти потрапляють на дешборд з детального списку проєктів. Цей функціонал реалізовано у файлі pages/project_list.php.

На відміну від карток на дешборді, тут використано табличне подання даних (рис. 4.3), що дозволяє відобразити більше деталей в компактному вигляді.

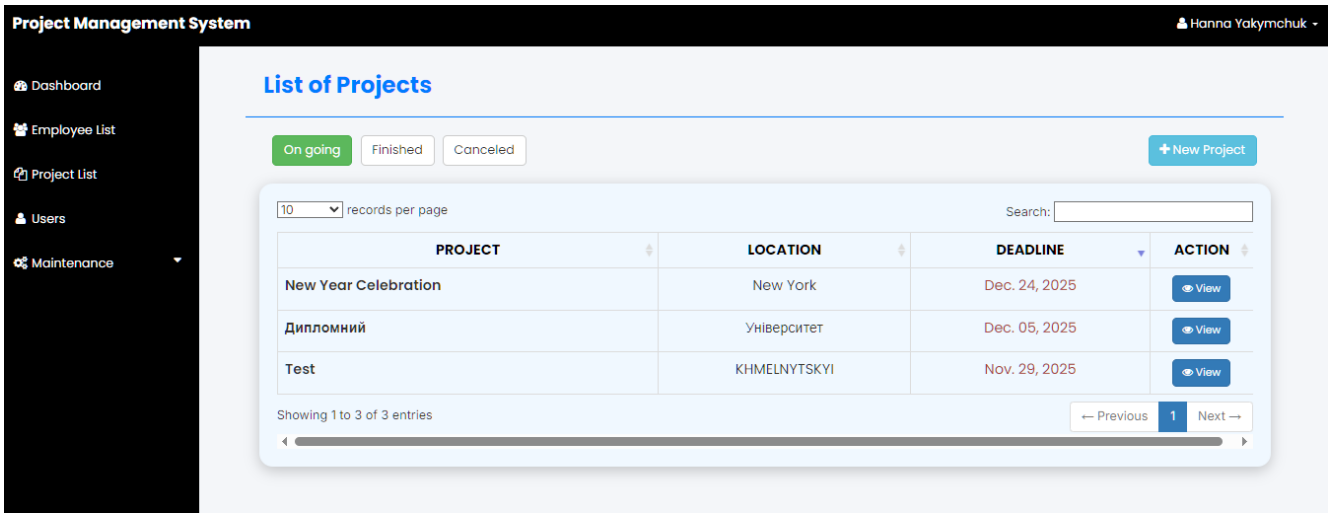

Рисунок 4.3 - Інтерфейс списку проєктів

Для забезпечення вимог юзабіліті та зручної роботи з великими масивами даних, таблиця інтегрована з бібліотекою DataTables. Це плагін для jQuery, який автоматично додає до стандартної HTML-таблиці розширений функціонал, такий як живий пошук (Search), що дозволяє миттєво фільтрувати проєкти за назвою або локацією без перезавантаження сторінки, пагінація і сортування.

Серверна частина отримує дані з таблиці projects і формує рядки в циклі.

```
<tbody>
```

```
          <?php
             $qry = $conn->query("SELECT * FROM projects ORDER BY date_created
DESC");

          while($row = $qry->fetch_assoc()):
              $deadline = date("M d, Y", strtotime($row['deadline']));
          ?>
          <tr>
              <td>
                  <b><?php echo ucwords($row['name']) ?></b><br>
                  <small><?php echo $row['description'] ?></small>
              </td>
              <td><?php echo $row['location'] ?></td>
              <td><?php echo $deadline ?></td>
              <td>
                              <a class="btn btn-sm btn-primary" href="index.php?
page=view_project&id=<?php echo $row['id'] ?>">
                          View
                  </a>
              </td>
          </tr>
          <?php endwhile; ?>
      </tbody>
```

Також над таблицею реалізовані вкладки-фільтри («On going», «Finished», «Canceled»). На рівні коду це реалізовано через передачу GET-параметра або через фільтрацію на стороні клієнта, що дозволяє швидко перемикатися між активними та архівними проєктами.

При виборі конкретного проєкту зі списку, користувач перенаправляється на сторінку pages/project_detail.php. Цей модуль є найбільш інформаційно насиченим, оскільки агрегує дані з кількох таблиць.

Для формування сторінки виконується складний SQL-запит, який об'єднує таблиці projects, project_team та employee для отримання повної інформації про проєкт та відповідального менеджера.

```
      $id = $_GET['id'];
      $emp_query = mysqli_query($conn,"
              SELECT *, CONCAT(lastname,', ',firstname, ' ',midname) as name,
projects.io as stats
          FROM projects
          LEFT JOIN project_team ON projects.tid = project_team.tid
          LEFT JOIN employee ON project_team.eid = employee.eid
          WHERE project_id = '$id'
      ");
      $row = mysqli_fetch_assoc($emp_query);
```

Інтерфейс детального перегляду проєкту (рис. 4.4) розділено на інформаційні блоки: профіль (зображення), ключові метрики (бюджет, дедлайн, тип) та графік прогресу. Для візуального розрізнення статусів використано кольорові «бейджі»: On Going (зелений) - активна робота; Finished (сірий) - проєкт завершено; Canceled (червоний) - проєкт скасовано.

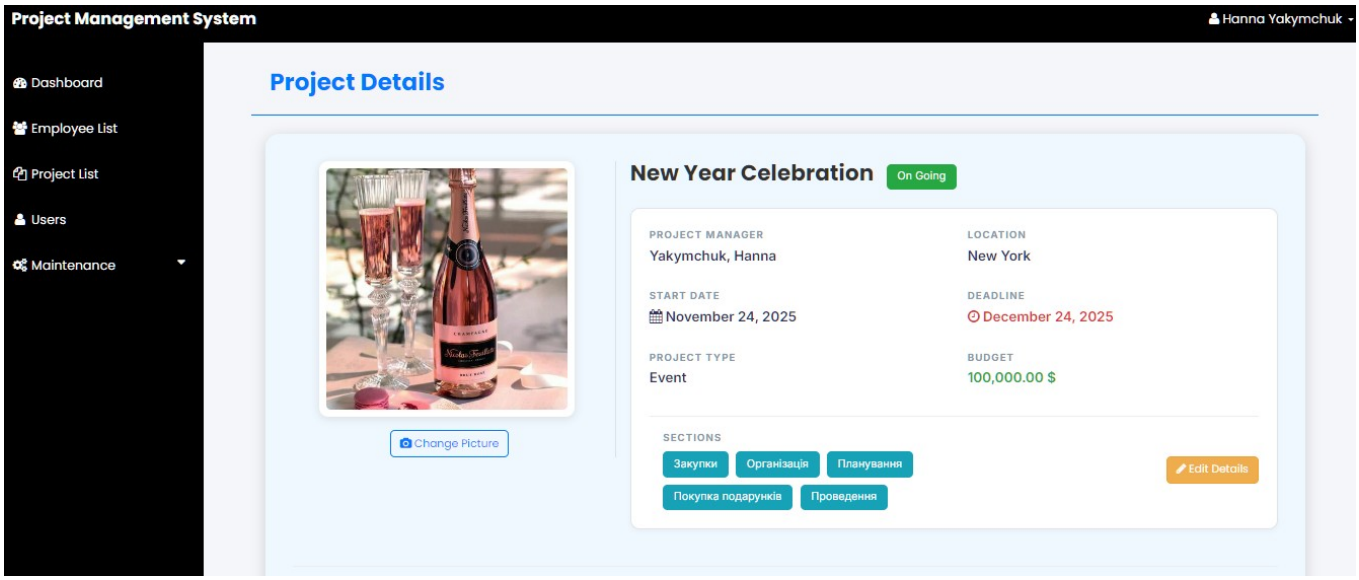

Рисунок 4.4 - Інтерфейс детального перегляду проєкту

Однією з вимог юзабіліті була мінімізація переходів між сторінками. Тому редагування властивостей проєкту реалізовано через модальне вікно, яке працює через виклик кнопкою «Edit Details». Обробка форми редагування відбувається асинхронно за допомогою технології AJAX та бібліотеки jQuery.

```
jQuery("#proj_form_local").submit(function(e){
    e.preventDefault(); // Блокування стандартної відправки форми
    var formData = jQuery(this).serialize();
    $.ajax({
        type: "POST",
        url: "../forms/update_forms.php?action=project",
        data: formData,
        success: function(html){
            $('#retCode_local').html(html);
            // Автоматичне оновлення сторінки через 1 секунду
            setTimeout(function(){ location.reload(); }, 1000);
        }
    });
    return false;
});
```

Цей скрипт перехоплює подію submit, збирає дані з полів форми (serialize()) і відправляє їх на серверний скрипт update_forms.php. Це забезпечує швидкий відгук інтерфейсу та покращує користувацький досвід.

Для забезпечення наочності даних та зниження когнітивного навантаження на менеджера, система використовує графічну візуалізацію прогресу виконання проєкту(рис. 4.5). Цей функціонал реалізовано у файлі pages/progress_chart.php, який підключається до сторінки деталізації.

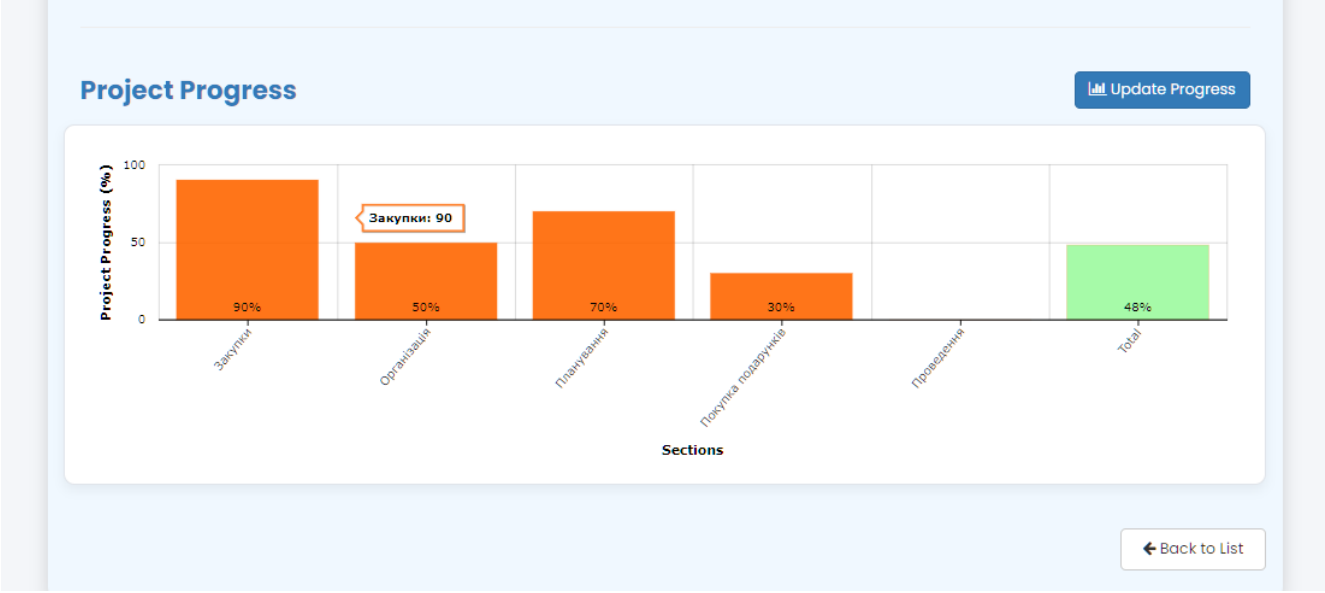

Рисунок 4.5 - Графік прогресу виконання проєкту

Перед відображенням графіка сервер повинен агрегувати дані. Проєкт у системі розділений на секції (partitions/divisions). Прогрес фіксується окремо для кожної секції.

```
      /* progress_chart.php - Підготовка даних */
      $prog = $conn->query("SELECT * FROM project_partition NATURAL JOIN
project_division WHERE project_id = '$id'");

      while($progress = $prog->fetch_assoc()){
          $pid = $progress['pp_id'];
          $name = $progress['division'];

              $prog3 = $conn->query("SELECT SUM(progress) as total_prog FROM
project_progress WHERE pp_id = '$pid'");
          $row_prog = $prog3->fetch_assoc();

                                          $array[$id][] ='{"progress":"'.
$row_prog['total_prog'].'","name":"'.ucfirst($name).'"}';
```

```
}
$data = implode(',', $array[$id]);
```

Для побудови інтерактивного графіка використовується бібліотека AmCharts. Вона приймає підготовлений PHP-скриптом рядок $data і будує стовпчикову діаграму (Serial/Column Chart).

Такий підхід дозволяє динамічно будувати графіки будь-якої складності, базуючись на актуальних даних з БД, без необхідності перезавантаження сторінки для оновлення картинки.

Сторінка «Управління Користувачами» доступна тільки для користувачів з роллю «Адміністратор». Тут знаходиться функціонал для перегляду списку усіх користувачів системи, створення нових облікових записів і призначення їм ролей.

Сторінка pages/user_list.php (рис. 4.6) відображає таблицю користувачів і дає можливість фільтрації за статусом («Active»/«Inactive»).

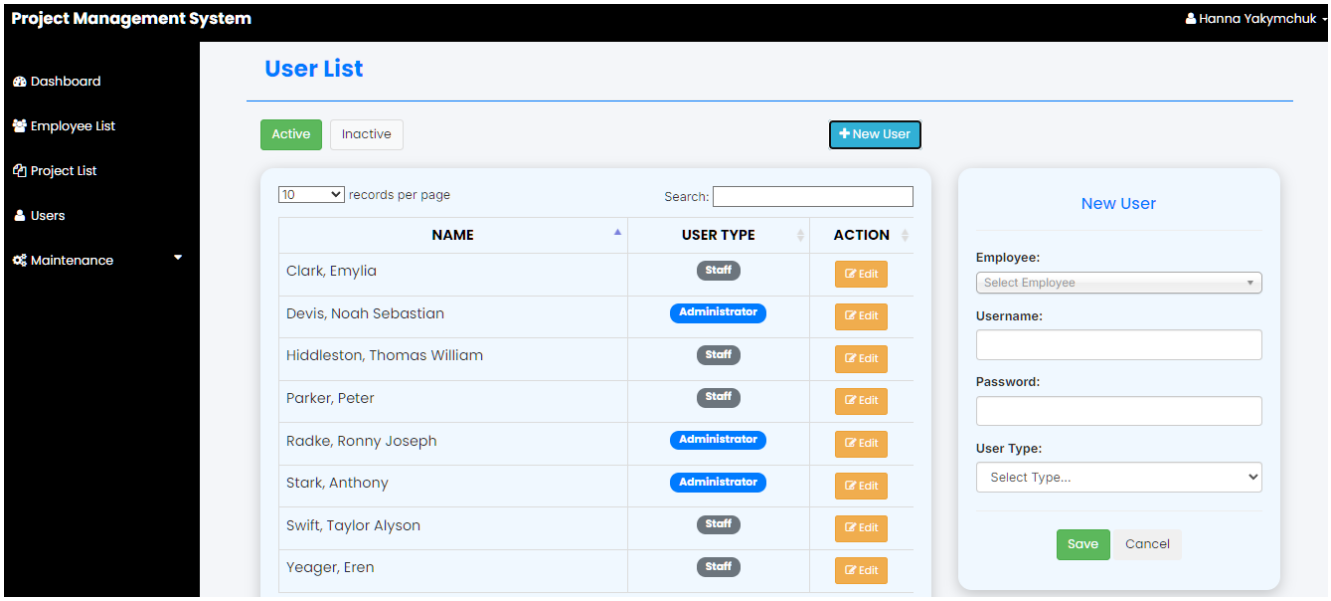

Рисунок 4.6 - Інтерфейс управління користувачами та ролями

Форма створення нового користувача реалізована як динамічний блок, що з'являється при натисканні кнопки «New User». Ключовим елементом форми є вибір User Type (ролі), який визначає рівень доступу: Administrator (повний доступ) або Staff(обмежений доступ).

Обробка форми відбувається через AJAX-запит до контролера ../forms/add_forms.php?action=user.

Використання бібліотеки Chosen покращує юзабіліті випадаючого списку співробітників, додаючи можливість пошуку, що особливо корисно при великій кількості персоналу.

Для коректного функціонування системи обліку персоналу реалізовано модуль управління посадами (Positions). Він дозволяє стандартизувати назви посад та відповідні їм ставки оплати (Daily Rate).

Сторінка pages/position.php реалізована за схемою «Master-Detail» на одній сторінці. Зліва відображається таблиця існуючих посад, а справа (приховано за замовчуванням) - форма додавання нових.

```
$query = mysqli_query($conn, "SELECT * FROM position ORDER BY position");
while($row = $qry->fetch_assoc()):
?>
<tr>
    <td><?php echo $row['position'] ?></td>
    <td><?php echo $row['daily_rate'].' $' ?></td>
    <td><a href="#edit_modal" class="btn-edit">Edit</a></td>
</tr>
<?php endwhile; ?>
```

Для покращення юзабіліті, форма створення нової посади прихована. Вона з'являється лише при натисканні кнопки «Add Position», яка при цьому ховається.

```
function add_form(){
    $('#add_pos_btn').fadeOut(); // Ховаємо кнопку "Додати"
    $('#add_form_container').slideDown(); // Плавно показуємо форму
}

function can_pos(){
    $('#add_form_container').slideUp(function(){
        $('#add_pos_btn').fadeIn(); // Повертаємо кнопку
    });
    $('#pos_form')[0].reset(); // Очищуємо поля
}
```

Ця проста анімація робить інтерфейс менш навантаженим, показуючи елементи керування лише тоді, коли вони потрібні користувачу.

Для класифікації проєктів та розподілу відповідальності у системі впроваджено поняття «Секцій». Цей функціонал дає можливість створювати категорії (наприклад, «Розробка», «Дизайн», «Підтримка») та прив'язувати їх до типів проєктів(рис. 4.7). Сторінка pages/division.php реалізована аналогічно до модуля посад: таблиця зліва та динамічна форма справа.

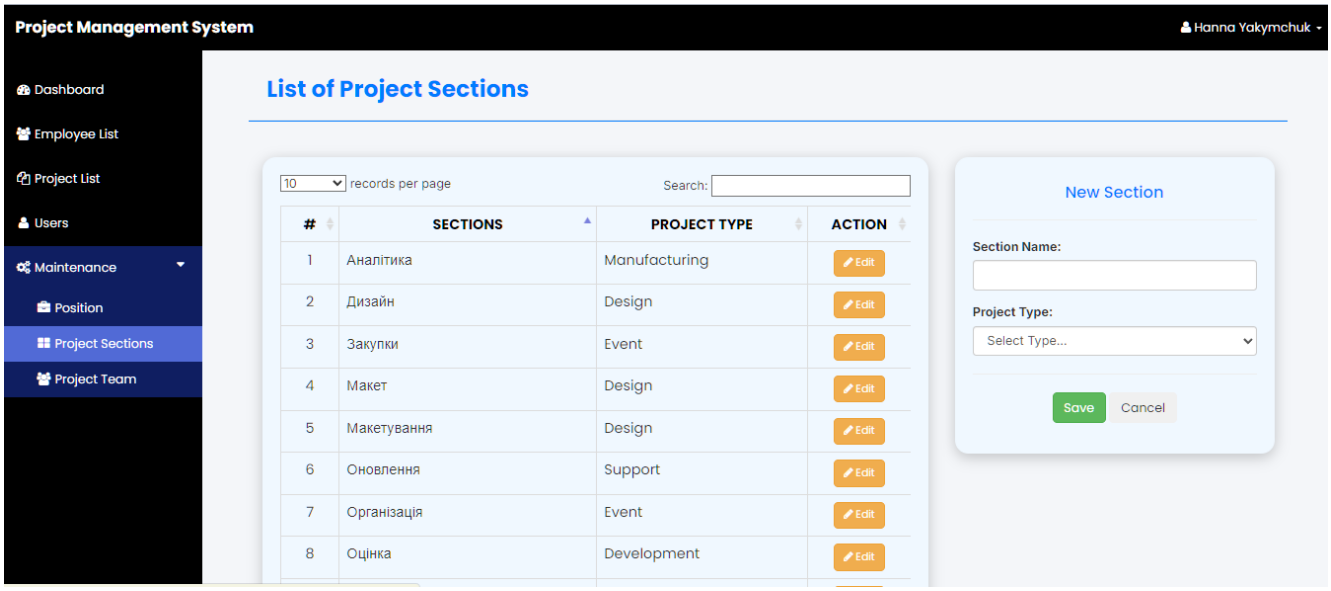

Рисунок 4.7 - Інтерфейс управління секціями проєктів

На відміну від попередніх модулів, тут реалізовано єдину форму для додавання та редагування. JavaScript-функція update_div(i) заповнює поля форми існуючими даними та змінює логіку відправки.

```
function add_form(){jQuery("#pos_form").submit(function(e){
    e.preventDefault();
    var formData = jQuery(this).serialize();
    var id = $('#id_hidden').val(); // Перевірка ID

      var action = (id == '') ? '../forms/add_forms.php?action=division' :
'../forms/update_forms.php?action=division';
```

```
        $.ajax({
            type: "POST",
            url: action,
            data: formData,
            success: function(html){
            }
        });
    });
```

Останньою сторінкою в меню є управління командами. Цей функціонал дозволяє групувати співробітників, призначати керівника команди та сформувати стійкі робочі групи для призначення на проєкти(рис. 4.8).

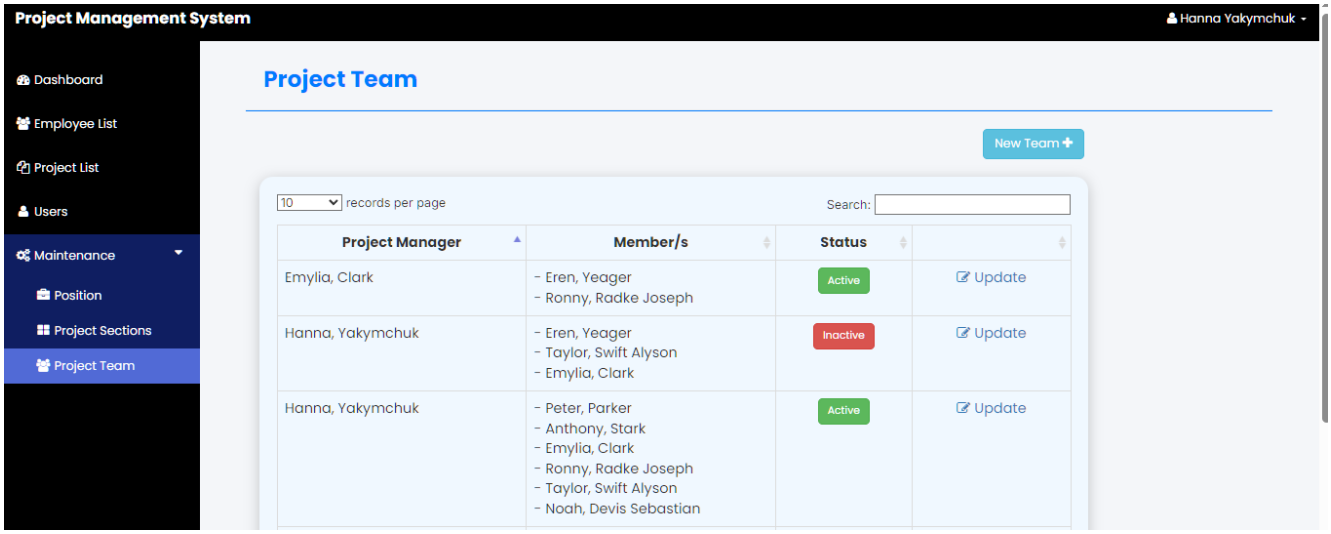

Рисунок 4.8 - Інтерфейс управління командами

Файл pages/project_team.php реалізує контейнерну логіку. За замовчуванням у блоці підключається файл team.php, який відображає таблицю існуючих команд.

Особливістю реалізації є відсутність окремої сторінки для додавання команди. При натисканні кнопки «New Team», замість переходу за посиланням, спрацьовує JavaScript-функція add_team().

Такий підхід значно покращує чуйність інтерфейсу, оскільки користувач не чекає повного перезавантаження сторінки, а лише оновлення робочої області.

Отже, підсумовуючи результати розробки, маємо функціонально дієздатний прототип системи з базовим функціоналом.

Створений програмний прототип можна вважати готовим до наступного етапу дослідження, а саме, проведення його верифікації та валідації.

4.3. Тестування та аналіз отриманих результатів

Фінальним етапом дослідження можна вважати тестування розробленого програмного прототипу і аналіз його результатів. Основна мета цього етапу - перевірка відповідності функціональним вимогам і чи вирішує прототип поставлену проблему високого когнітивного навантаження користувачів.

Тестування проведено з використанням методу «чорної скриньки», тобто взаємодією тільки з інтерфейсом системи без прямого втручання в код. Тестування проводилось на персональному комп’ютері через веб-браузер Google Chrome.

Однією з головних задач тестування була верифікація підсистеми безпеки та ролей, так як це є основою загальної логіки системи. Для цього розроблено і виконано декілька тест-кейсів (табл. 4.11).

Таблиця 4.11 - Тестування модуля безпеки та авторизації

№	Сценарій тестування	Вхідні дані	Очікуваний результат	Фактичний результат	Статус
1	Спроба входу з невірним паролем	Логін: admin Пароль: wrong123	Система не створює сесію. Виводиться повідомлення «Incorrect username or password».	Повідомлення виведено. Доступ заборонено.	Passed
2	Спроба прямого доступу (без входу)	URL: ../index.php?page=user_list	Примусовий редірект на сторінку login.php.	Редірект відбувся миттєво.	Passed
3	Вхід з роллю «Адміністратор»	Логін: Admin Пароль: admin123	Успішний вхід. У меню відображаються всі пункти.	Меню повне. Доступні всі розділи	Passed
4	Вхід з роллю	Логін: Dev1	Успішний вхід. У	Меню	Passed

	«Розробник»	Пароль: 12345	меню відсутній пункт «Users».	спрощене. Адмін-розділ приховано.	

Підсумовуючи, результати тестування підтверджують, що програмний код файлів login.php та sidebar.php працює коректно: система динамічно верифікує користувача, ідентифікує його роль та адаптує інтерфейс, фізично приховуючи недоступні функції з меню.

Далі було перевірено коректність роботи основного функціоналу системи: управління проєктом та зміни його прогресу.

Метою цього тестування було підтвердити, що система коректно обробляє дані та відображає їх у відповідних інтерфейсах(табл. 4.12).

Таблиця 4.12 - Тестування функціоналу системи

№	Сценарій тестування	Очікуваний результат	Фактичний результат	Статус
1	Створення нового проєкту	Запис з'являється в БД. Проєкт відображається у списку «Project List» та на Дешборді.	Проєкт створено. Картка з'явилася на дешборді миттєво.	Passed
2	Фільтрація списку проєктів	При натисканні на вкладку «Finished» відображаються лише проєкти зі статусом 2.	Список оновлено коректно. Активні проєкти приховано.	Passed
3	Розрахунок прогресу	При закритті завдання відсоток виконання проєкту має автоматично перерахуватися.	Після зміни статусу завдання графік на сторінці деталізації оновився.	Passed
4	Редагування через AJAX	При зміні даних у модальному вікні сторінка не повинна перезавантажуватися до моменту збереження.	Форма відправлена асинхронно. Дані оновлено без повного перезавантаження сторінки.	Passed

Ці тести підтверджують, що реалізований функціонал взаємодії користувача з модулем проєктів прототипу системи працює стабільно і відповідає попередньо визначеним вимогам.

Для перевірки нефункціональних вимог, які стосуються швидкодії системи було проведено вимірювання часу відгуку інтерфейсу. Вимірювання проводилися за допомогою інструменту Google Chrome DevTools (вкладка Network) [22].

Об'єктом вимірювання був показник DOMContentLoad (час до повної готовності HTML-структури сторінки) та час виконання AJAX-запитів.

Результати вимірювань:

- Завантаження Дешборду (PM): 0.45 с (вимога: < 3 с).
- Завантаження Списку проєктів: 0.38 с (вимога: < 3 с).
- Зміна статусу (AJAX): 0.12 с (вимога: < 1 с).

Отримані показники перевищують мінімальні вимоги. Зі збільшенням кількості даних в системі швидкість може трохи збільшитись, але в рамках допустимого відповідно до вимог.

Щоб протестувати гіпотезу про зменшення когнітивного навантаження на користувача було проведено порівняльний аналіз інтерфейсу.

Як метрику для порівняння обрано «візуальний шум», тобто кількість елементів навігації та управління, які відображаються користувачу, але не потрібні для виконання його поточної роботи. Об'єктом тестування обрано сценарій роботи «Розробника», якому потрібно просто переглянути список проєктів і перевірити його дедлайни.

Порівняння навігації для складних систем (представником обрано Jira) та розробленого програмного прототипу(табл. 4.13):

Jira Software: для звичайного виконавця інтерфейс містить глобальне меню (Your work, Projects, Filters, Dashboards, People, Apps), кнопку «Create», меню налаштувань профілю, а також бокове меню проєкту (Backlog, Board, Reports, Issues, Components, etc.). Це створює ситуацію, коли з 20+ пунктів меню розробнику реально потрібні лише 2-3.

Розроблений прототип: завдяки реалізації адаптивного меню (sidebar.php), користувач з роллю Staff бачить лише 3 пункти: «Dashboard», «Employee List», «Project List». Адміністративні розділи фізично вилучені з HTML-коду сторінки.

Таблиця 4.13 - Порівняння навантаження інтерфейсу (роль: розробник)

Критерій	Jira / Складні системи	Розроблений Прототип
Кількість	> 15 пунктів (глобальне + локальне)	3 пункти (тільки

пунктів меню		необхідне)
Ризик помилки	Високий (можна випадково змінити налаштування фільтру або зайти в чужий звіт)	Нульовий (адмін-функції недоступні)
Фокус уваги	Розсіюється на навігацію	Сфокусований на контенті (списку)

Таким чином прототип підтверджує що така реалізація дозволяє спростити інтерфейс для деяких типів користувачів і допомагає збільшити увагу на виконанні безпосередніх задач не відволікаючись на зайвий функціонал.

Отже, тестування програмного прототипу показало, що система успішно пройшла функціональні тести. Ключові механізми працюють коректно та без збоїв. Порівняльний аналіз інтерфейсу підтвердив, що реалізація адаптивного меню та розмежування прав доступу на рівні сесій ефективно вирішує проблему «візуального шуму» та високого когнітивного навантаження на користувача.

На основі цього можна визначити основні перспективи та напрямки розвитку розробленого прототипу:

- деталізація та подальша розробка функціоналу задач, додавання інтерактивного дешборда для інтерактивності, детальної сторінки конкретної задачі, коментарів до неї, тайм-трекера і тд.;
- інтеграції: створення API для зв'язку з зовнішніми сервісами (Google Calendar для дедлайнів, Slack/Telegram для сповіщень);
- хмарне розгортання: перенос локації розробленої системи з локального середовища XAMPP на хмарний хостинг (AWS або Azure) для забезпечення доступності 24/7;
- посилення безпеки: додавання двофакторної автентифікації для ролі адміністратора, перехід на більш сучасні алгоритми хешування паролів.

Таким чином можна здійснювати доопрацювання прототипу в ще декілька ітерацій, включаючи таке тестування щоб створити повнофункціональний та конкуренто-здатний продукт, який можна буде запустити на ринок.

Перша варіація прототипу показала потенціал проєктованого програмного забезпечення та можливості для його покращення, що дає підстави для

продовження дослідження та поступовий перехід з формату прототипу в повноцінний комерційний продукт.

4.4. Висновки до розділу 4

В результаті проєктування і реалізації прототипу системи для управління проєктами було здійснено практичну реалізацію працюючого прототипу відповідно до вимог, визначених в попередніх розділах.

На етапі проєктування структури даних розроблено реляційну модель бази даних, що відповідає третій нормальній формі (3NF). Фізична реалізація бази даних виконана в середовищі СУБД MySQL.

Як результат програмної реалізації створено дієздатний прототип системи з впровадженням механізму адаптивного користувацького інтерфейсу. Реалізовано ключові функціональні модулі: інтерактивний дешборд з візуалізацією статистики, систему управління проєктами з підтримкою статусів та графіком прогресу, а також модуль управління командами.

На завершення проведено комплексне тестування прототипу, а саме функціональне тестування, оцінка ефективності роботи прототипу та аналіз продуктивності розробленої системи.

Таким чином розроблений програмний прототип успішно вирішує поставлене завдання. Результати його тестування показують технічну можливість та загальну доцільність розробки повноцінного програмного забезпечення такого типу з створенням роле-орієнтованих середовищ управління проєктами, які можуть використовуватись в багатьох сферах та великою кількістю різних команд.

На даному етапі прототип повністю готовий до демонстрації та подальшого розвитку в повноцінний комерційний продукт.

ВИСНОВКИ

Отже, під час проведення дослідження для магістерської роботи було проаналізовано та вирішено актуальне науково-практичне завдання підвищення ефективності управління проєктною діяльністю шляхом розроблення та дослідження адаптивного користувацького середовища.

Відповідно до поставлених завдань отримано наступні результати:

Проаналізовано теоретичні основи управління проєктною діяльністю та сучасні методології. Встановлено, що перехід індустрії до гнучких та гібридних методологій вимагає зміни підходів до організації робочого процесу. Виявлено, що ключовими факторами успіху сучасного проєкту є не лише дотримання планів, а й забезпечення ефективної комунікації та швидкості реакції на зміни, що безпосередньо залежить від якості використовуваного інструментарію.

Досліджено підходи до проєктування програмних систем, зокрема методологію людино-орієнтованого дизайну. Проведено порівняльний аналіз існуючих програмних засобів та визначено їхні переваги й недоліки. Виявлено ключову проблему ринку - наявність компромісу між функціональною потужністю та простотою використання, а також відсутність адаптації інтерфейсу під специфічні потреби різних ролей.

Проведено системний аналіз предметної області з використанням методів структурного та об’єктно-орієнтованого моделювання. На основі побудованих моделей інформаційних потоків та варіантів використання обґрунтовано проблему дослідження: когнітивне перевантаження користувачів у складних системах. Запропоновано вирішення цієї проблеми шляхом створення адаптивного середовища, яке змінює набір інструментів залежно від ролі користувача.

Визначено вимоги до користувацького середовища системи підтримки управління проєктною діяльністю. Сформульовано перелік функціональних вимог, згрупованих за ролями акторів та нефункціональних вимог. Ключовою

вимогою визначено автоматичну адаптацію навігації та приховування недоступного функціоналу для зниження візуального шуму.

Вибрано та обґрунтовано методи та засоби проєктування прототипу. В якості архітектурного базису обрано клієнт-серверну архітектуру з використанням патерну MVC та RESTful API, що забезпечує надійність та масштабованість. Обґрунтовано вибір технологічного стеку: мова PHP для серверної частини, СУБД MySQL для зберігання даних, середовище XAMPP для розгортання та бібліотеки HTML/CSS/JS для реалізації клієнтського інтерфейсу.

Розроблено концептуальну модель та спроєктовано структуру бази даних прототипу. Створено реляційну модель бази даних у третій нормальній формі. Розроблена структура таблиць забезпечує гнучку реалізацію ролевої моделі доступу та цілісність даних при управлінні проєктами, завданнями та командами.

Створено прототип програмного засобу на основі розробленої моделі та реалізовано його основні функціональні модулі. Розроблений веб-застосунок включає модуль безпечної автентифікації, механізм динамічної генерації адаптивного меню, інтерактивний дешборд з аналітикою, а також модулі управління проєктами, завданнями та користувачами.

Оцінено ефективність запропонованих рішень шляхом верифікації та валідації прототипу. Функціональне тестування підтвердило повну відповідність системи сформульованим вимогам. Порівняльний аналіз сценаріїв роботи (User Journey) довів, що запропонований адаптивний інтерфейс дозволяє скоротити кількість елементів навігації для деяких ролей.

Загалом, підсумовуючи результати проведеного дослідження можна зробити висновок, що мета роботи була успішно досягнута. В ході роботи було розроблено та обґрунтовано модель користувацького середовища, реалізовану у вигляді програмного прототипу.

Результати дослідження мають практичну цінність для компаній та команд, оскільки пропонують перевірене рішення для підвищення ефективності процесів управління проєктами. Створений прототип готовий до подальшого розвитку, масштабування та впровадження в реальні виробничі процеси.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Блог Worksection. Як вести облік фінансів по проєктах [Електронний ресурс] // Блог Worksection. - 2023. - Режим доступу: https://worksection.com/blog/
2. Про Worksection: Управління проєктами та задачами [Електронний ресурс] - Режим доступу: https://worksection.com/
3. Ahmed M. A. Kanban vs Scrum: A Comparative Analysis in Software Development / M. A. Ahmed, G. Almashaqbeh // International Journal of Computer Science and Information Security. - 2023. - Vol. 21, № 1.
4. Ahmeti A. Challenges of the Waterfall Model in Modern Software Development Projects / A. Ahmeti, V. Vokrri // 2nd International Conference on Business, Management and Economics. - 2022.
5. Akiki P. A. Benefits of Adaptive User Interfaces on Cognitive Load and Task Performance in Enterprise Systems / P. A. Akiki, R. Shami // Journal of Organizational Computing and Electronic Commerce. - 2023. - Vol. 33, № 1. - P. 45-67.
6. Al-Saqqa S. A Comparative Analysis of Project Management Methodologies: Waterfall vs. Agile / S. Al-Saqqa, M. Al-Nofal // Journal of Applied Computer Science & Technology. - 2023. - Vol. 3, № 1. - P. 22-30.
7. Ambler S. W. User Experience (UX) Design [Електронний ресурс] / S. W. Ambler // Disciplined Agile Consortium / PMI. - 2022.
8. Apache Friends. XAMPP FAQs [Електронний ресурс] // Apache Friends. - 2024. - Режим доступу: https://www.apachefriends.org/faq_linux.html
9. Atlassian. Jira Software Overview [Електронний ресурс] - Режим доступу: https://www.atlassian.com/software/jira
10. Atlassian. Permissions for Jira Software [Електронний ресурс] // Atlassian. - 2024. - Режим доступу: https://support.atlassian.com/jira-cloud-administration/docs/
11. Codebridge. Proof of Concept vs Prototype: Which Fits Your Business? [Електронний ресурс] // Codebridge. - 2023.

12. Developing User-Centered Excellence in Software Design Using Ergonomic Elegance [Електронний ресурс] // Longdom Publishing. - 2024.

13. Digital.ai. 17th State of Agile Report [Електронний ресурс]. - 2023. - Режим доступу: https://stateofagile.com/

14. Forbes. The Role Of Project Management Software In Boosting Organizational Productivity [Електронний ресурс] // Forbes. - 2023.

15. García R. G. Comparative analysis of project management tools for software development: Jira, Redmine, and Trello / R. G. García et al. // IEEE Latin America Transactions. - 2022. - Vol. 20, № 3. - P. 567-575.

16. Gartner. Magic Quadrant for Enterprise Agile Planning Tools [Електронний ресурс] // Gartner. - 2023.

17. Gartner. TCO of Enterprise Software: Beyond the License Fee [Електронний ресурс] // Gartner. - 2023.

18. GeeksforGeeks. System Analysis | System Design [Електронний ресурс] // GeeksforGeeks. - 2024. - Режим доступу: https://www.geeksforgeeks.org/system-design-tutorial/

19. GeeksforGeeks. Use Case Diagram - Unified Modeling Language (UML) [Електронний ресурс] // GeeksforGeeks. - 2024. - Режим доступу: https://www.geeksforgeeks.org/unified-modeling-language-uml-use-case-diagrams/

20. Ghahrai A. Performance evaluation of web applications: A systematic review / A. Ghahrai et al. // Information and Software Technology. - 2022. - Vol. 147. - 106900.

21. Ghani I. A review of agile software development design practices / I. Ghani, A. Rauf // Journal of Software: Evolution and Process. - 2022. - Vol. 34, № 5.

22. Google Developers. Core Web Vitals: The metrics that matter [Електронний ресурс] // Google Developers. - 2024. - Режим доступу: https://web.dev/articles/vitals

23. Guru99. Database Normalization: 1NF, 2NF, 3NF, BCNF with Examples [Електронний ресурс] // Guru99. - 2024. - Режим доступу: https://www.guru99.com/database-normalization.html

24. IBM Technology. What is Client-Server Architecture? [Електронний ресурс] // IBM Technology. - 2023. - Режим доступу: https://www.ibm.com/topics/client-server

25. Interaction Design Foundation (IxDF). What is User-Centered Design? [Електронний ресурс] // Interaction Design Foundation (IxDF). - 2024. - Режим доступу: https://www.interaction-design.org/literature/topics/user-centered-design

26. ISO 9241-210:2019. Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems. - International Organization for Standardization, 2019.

27. Karvonen K. Human-centered design: origins, components, and concepts / K. Karvonen // The SAGE Handbook of Human-Machine Communication. - SAGE Publications, 2020.

28. Kaspersky. Role-Based Access Control (RBAC): Principles and Best Practices [Електронний ресурс] // Kaspersky. - 2023. - Режим доступу: https://www.kaspersky.com/

29. Lallemand C. Methods for evaluating interaction design: An updated survey / C. Lallemand, G. Gronier // Design Studies. - 2020. - Vol. 68.

30. Lallemand C. Methods for evaluating interaction design: An updated survey / C. Lallemand, G. Gronier // Design Studies. - 2020. - Vol. 68.

31. Lee S. Role-Based User Experience Design for Collaborative Enterprise Software / S. Lee, J. Kim // International Journal of Human-Computer Interaction. - 2022. - Vol. 38, № 10. - P. 914-928.

32. Lucassen G. Aligning requirements and architecture in agile development: A functional perspective / G. Lucassen et al. // Journal of Systems and Software. - 2021. - Vol. 171. - 110825.

33. Miro J. A Comparative Study of Trello and Jira for Agile Project Management in Startups / J. Miro, S. Dabi // International Journal of Advanced Computer Science and Applications. - 2023. - Vol. 14, № 5.

34. Morville P. Information Architecture: For the Web and Beyond (4th Edition) / P. Morville, L. Rosenfeld. - O'Reilly Media, 2021.

35. Nielsen J. 10 Usability Heuristics for User Interface Design [Електронний ресурс] / J. Nielsen // Nielsen Norman Group. - 2020. - Режим доступу: https://www.nngroup.com/articles/ten-usability-heuristics/

36. Nielsen Norman Group. Articles and Videos on User Experience Research, Training, and Consulting [Електронний ресурс]. - 2024. - Режим доступу: https://www.nngroup.com/

37. Nielsen Norman Group. Cognitive Load in UX Design: Definition and Principles [Електронний ресурс] // Nielsen Norman Group. - 2023. - Режим доступу: https://www.nngroup.com/articles/minimize-cognitive-load/

38. Norman D. A. Design for a Better World: Meaningful, Sustainable, Humanity Centered / D. A. Norman. - MIT Press, 2023.

39. PCMag. Trello Review [Електронний ресурс] // PCMag. - 2024. - Режим доступу: https://www.pcmag.com/reviews/trello

40. Pino F. J. Requirements for software tools supporting the Scrum framework: A systematic literature review / F. J. Pino, L. Romero // Information and Software Technology. - 2022. - Vol. 148. - 106927.

41. PMI. A Primer on Emergent Design / Project Management Institute. - 2020.

42. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK® Guide) - Seventh Edition / Project Management Institute. - PMI, 2021.

43. Red Hat. What is a REST API? [Електронний ресурс] // Red Hat. - 2023. - Режим доступу: https://www.redhat.com/en/topics/api/what-is-a-rest-api

44. ResearchGate. Towards a Definition of Complex Software System [Електронний ресурс] // ResearchGate. - 2023.

45. Rubin J. Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests (3rd Edition) / J. Rubin, D. Chisnell. - Wiley, 2023.

46. Salameh A. The Impact of Project Management Methodologies on Project Success: A Comparative Study / A. Salameh // International Journal of Project Management and Productivity. - 2024. - Vol. 2, № 1. - P. 1-15.

47. Sauro J. Benchmarking the User Experience / J. Sauro. - Springer Nature, 2021.

48. Schwaber K. The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game / K. Schwaber, J. Sutherland. - 2020.

49. Sharma N. A Comparative Study of Relational and NoSQL Databases for Modern Web Applications / N. Sharma et al. // International Journal of Computer Science and Network Security. - 2022. - Vol. 22, № 6. - P. 114-120.

50. Smith J. A. Role-Based Interfaces in Enterprise Software: Balancing Functionality and Usability / J. A. Smith, R. Johnson // ACM Transactions on Computer-Human Interaction (TOCHI). - 2022. - Vol. 29, № 4. - P. 1-25.

51. Sommerville I. Software Engineering (11th Edition) / I. Sommerville. - Pearson, 2021.

52. Sommerville I. Software Engineering (11th Edition) / I. Sommerville. - Pearson, 2021.

53. Still B. Fundamentals of User-Centered Design: A Practical Approach / B. Still, K. Crane. - CRC Press, 2021.

54. Telea A. C. Data Visualization: Principles and Practice (3rd Edition) / A. C. Telea. - CRC Press, 2023.

55. Vanhoucke M. Project Management with Dynamic Scheduling: Gantt Charts, PERT/CPM, and EVM / M. Vanhoucke. - Springer, 2023.

56. W3Techs. Usage statistics of server-side programming languages for websites [Електронний ресурс] // W3Techs. - 2024. - Режим доступу: https://w3techs.com/

57. What is Trello? [Електронний ресурс] - Режим доступу: https://trello.com/guide

58.

ДОДАТКИ

Програмний код

Home.php:

```
<?php include '../includes/db.php'; ?>

<div class="container mt-4">


    <div class="row mb-4 align-items-center">
        <div class="col-12">
            <h2 class="page-title">Dashboard</h2>
            <hr style="border-bottom:1px solid #007bff">
        </div>
    </div>

    <div class="row mb-4">
        <div class="col-md-4">
            <div class="stat-card bg-primary-gradient">
                <div class="stat-icon"><i class="fa fa-briefcase"></i></div>
                <div class="stat-info">
                    <?php
                    if($_SESSION['TYPE'] == 1){
                        $p_sql = "SELECT * FROM projects WHERE io = '1'";
                    } else {
                        $my_id = $_SESSION['ID'];
                        $p_sql = "SELECT p.* FROM projects p
                                  JOIN project_team t ON p.tid = t.tid
                                  JOIN team_member tm ON t.tid = tm.tid
                                  WHERE p.io = '1' AND tm.eid = '$my_id'";
                    }

                    $p_query = mysqli_query($conn, $p_sql);
                    $p_count = mysqli_num_rows($p_query);
                    ?>
                    <h3><?php echo $p_count; ?></h3>
                    <p>Active Projects</p>
                </div>
            </div>
        </div>

        <div class="col-md-4">
            <div class="stat-card bg-success-gradient">
                <div class="stat-icon"><i class="fa fa-users"></i></div>
                <div class="stat-info">
                    <?php
                      $e_query = mysqli_query($conn, "SELECT * FROM employee
WHERE io = '1'");
                    $e_count = mysqli_num_rows($e_query);
                    ?>
```

```
                        <h3><?php echo $e_count; ?></h3>
                        <p>Active Employees</p>
                    </div>
                </div>
            </div>

            <div class="col-md-4">
                <div class="stat-card bg-info-gradient">
                                            <div  class="stat-icon"><i  class="fa
fa-check-circle"></i></div>
                    <div class="stat-info">
                        <?php
                          $f_query = mysqli_query($conn, "SELECT * FROM projects
WHERE io = '2'");
                        $f_count = mysqli_num_rows($f_query);
                        ?>
                        <h3><?php echo $f_count; ?></h3>
                        <p>Finished Projects</p>
                    </div>
                </div>
            </div>
        </div>

        <div class="row">


            <div class="col-lg-8">
                <h4 class="section-title">Ongoing Projects</h4>
                <div class="row">
                    <?php
                    mysqli_data_seek($p_query, 0);
                    if($p_count > 0):
                        while($row = mysqli_fetch_assoc($p_query)){
                    ?>
                                <div  class="col-md-6  mb-3">  <a  href="index.php?
page=project_detail&id=<?php  echo  $row['project_id']  ?>&action=view  details"
style="text-decoration: none;">
                            <div class="project-card">
                                <div class="img-wrapper">
                                                <img  src="../images/<?php  echo
$row["site_pic"] ?>" alt="Project Image">
                                    <div class="overlay">
                                          <span class="btn btn-sm btn-light">View
Details</span>
                                    </div>
                                </div>
                                <div class="card-body">
                                            <h5 class="card-title"><?php echo
ucfirst($row['project']) ?></h5>
                                     <p class="card-text text-muted"><i class="fa
fa-map-marker"></i> <?php echo $row['location'] ? $row['location'] : 'Online'; ?
></p>
                                </div>
                            </div>
                        </a>
```

```
                    </div>
                    <?php } else: ?>
                            <div class="col-12"><p class="text-muted">No active
projects found.</p></div>
                    <?php endif; ?>
                </div>
            </div>

            <div class="col-lg-4">
                <h4 class="section-title">Notifications</h4>
                <div class="notification-area">
                    <?php
                      $query1 = mysqli_query($conn,"SELECT * FROM projects WHERE
io = '1'");
                    $has_notif = false;

                    while($row1 = mysqli_fetch_assoc($query1)){
                            $d1 = date("Ymd", strtotime($row1['deadline'].' -15
days'));
                        $d2 = date("Ymd");
                                                $deadline_date  =  date("Ymd",
strtotime($row1['deadline']));

                         if($d2 >= $d1 && $deadline_date >= $d2){
                            $has_notif = true;
                    ?>
                            <a href="index.php?page=project_detail&id=<?php echo
$row1['project_id'] ?>" class="notif-link">
                            <div class="alert-card warning">
                                                <div  class="icon"><i  class="fa
fa-clock-o"></i></div>
                                <div class="content">
                                    <h6>Deadline Soon</h6>
                                            <p class="project-name"><?php echo
ucfirst($row1['project']) ?></p>
                                                <small>Due: <?php echo date("F d, Y",
strtotime($row1['deadline'])) ?></small>
                                </div>
                            </div>
                        </a>

                    <?php
                        } elseif($deadline_date < $d2){
                            $has_notif = true;
                    ?>
                            <a href="index.php?page=project_detail&id=<?php echo
$row1['project_id'] ?>" class="notif-link">
                            <div class="alert-card danger">
                                  <div class="icon"><i class="fa fa-exclamation-
circle"></i></div>
                                <div class="content">
                                    <h6>Overdue!</h6>
                                            <p class="project-name"><?php echo
ucfirst($row1['project']) ?></p>
```

```
                                        <small>Was due: <?php echo date("F d, Y",
strtotime($row1['deadline'])) ?></small>
                                    </div>
                                </div>
                            </a>
                        <?php }} ?>

                        <?php if(!$has_notif): ?>
                            <div class="alert-card success">
                                <div class="icon"><i class="fa fa-check"></i></div>
                                <div class="content">
                                    <h6>All Good!</h6>
                                    <p>No upcoming deadlines.</p>
                                </div>
                            </div>
                        <?php endif; ?>
                    </div>
                </div>
            </div>
        </div>
```

Sidebare.php:

```
        <div class="collapse navbar-collapse navbar-ex1-collapse">
    <ul class="nav navbar-nav side-nav styled-sidebar">
            <li class="sidebar-brand">
                <span style="color: #6c757d; font-weight: bold; padding-left: 20px;">
</span>
            </li>
            <li>
                <a href="index.php?page=home">
                    <i class="fa fa-fw fa-dashboard"></i> <span>Dashboard</span>
                </a>
            </li>
            <li>
                <a href="index.php?page=employee&io=1">
                    <i class="fa fa-fw fa-users"></i> <span>Employee List</span>
                </a>
            </li>
            <li>
                <a href="index.php?page=project_list&io=1">
                    <i class="fa fa-fw fa-files-o"></i> <span>Project List</span>
                </a>
```

```
        </li>


        <?php if($_SESSION['TYPE'] == 1): ?>
            <li>
                <a href="index.php?page=user_list&io=1">
                    <i class="fa fa-fw fa-user"></i> <span>Users</span>
                </a>
            </li>
        <?php endif; ?>
        <?php if($_SESSION['TYPE'] == 1): ?>
    <li>
        <a id="dem3" href="javascript:;" data-toggle="collapse" data-
target="#demo3">
            <i class="fa fa-fw fa-cogs"></i> <span>Maintenance</span> <i class="fa
fa-fw fa-caret-down pull-right"></i>
        </a>
        <ul id="demo3" class="collapse">
            <li>
                <a href="index.php?page=position">
                    <i class="fa fa-fw fa-briefcase"></i> Position
                </a>
            </li>
            <li>
                <a href="index.php?page=division">
                    <i class="fa fa-fw fa-th-large"></i> Project Sections
                </a>
            </li>
            <li>
                <a href="index.php?page=project_team">
                    <i class="fa fa-fw fa-users"></i> Project Team
                </a>
            </li>
        </ul>
    </li>
<?php endif; ?>
    </ul>
```

```
</div>
```

Project-detail.php:

```
<?php <?php
include '../includes/db.php';

if(!isset($_GET['id'])){
    echo "No Project ID provided.";
    exit;
}
$id = $_GET['id'];
$emp_query = mysqli_query($conn,"
    SELECT *, CONCAT(lastname,', ',firstname, ' ',midname) as name,
projects.io as stats
    FROM projects
    LEFT JOIN project_team ON projects.tid = project_team.tid
    LEFT JOIN employee ON project_team.eid = employee.eid
    WHERE project_id = '$id'
");
$row = mysqli_fetch_assoc($emp_query);
$proj_name = isset($row['project']) ? htmlspecialchars($row['project']) :
'';
$proj_loc  = isset($row['location']) ? htmlspecialchars($row['location']) :
'';
$proj_cost = isset($row['overall_cost']) ? $row['overall_cost'] : '';
$proj_sdate = isset($row['start_date']) ? $row['start_date'] : '';
$proj_dead  = isset($row['deadline']) ? $row['deadline'] : '';
$proj_type  = isset($row['proposed_project']) ? $row['proposed_project'] :
'';
$proj_stat  = isset($row['stats']) ? $row['stats'] : '';
$proj_tid   = isset($row['tid']) ? $row['tid'] : '';
$proj_mngr  = isset($row['name']) ? $row['name'] : 'Select Manager...';
$current_eid = isset($row['eid']) ? $row['eid'] : 0;
?>

<div class="container mt-4">
```

```
        <div class="row mb-4 align-items-center">
            <div class="col-12">
                <h2 class="page-title">Project Details</h2>
                <hr style="border-bottom:1px solid #007bff">
            </div>
        </div>

        <div class="profile-card">
            <div class="row">

                <div class="col-md-4 text-center border-right-custom">
                    <div class="img-container" style="width: 100%; max-width:
280px; height: 280px; overflow: hidden; margin: 0 auto;">
                        <?php
                            $img_file = (isset($row['site_pic']) && !
empty($row['site_pic'])) ? $row['site_pic'] : "default.png";
                            $img_src = "../images/" . $img_file;
                        ?>
                        <img src="<?php echo $img_src ?>" alt="Project Image"
class="project-img" style="width: 100%; height: 100%; object-fit: cover;">
                    </div>
                    <br>
                    <a href="#local_change_pic" data-toggle="modal" class="btn
btn-outline-primary btn-sm mt-3" style="margin-top: 15px; display:inline-block;">
                        <i class="fa fa-camera"></i> Change Picture
                    </a>
                </div>
                <div class="col-md-8">

                    <div class="project-header" style="margin-bottom: 20px;">
                        <h3 class="project-name"><?php echo
ucfirst($proj_name) ?></h3>
                        <?php
                            if($proj_stat == '1') echo '<span class="badge
badge-success">On Going</span>';
```

```
                                elseif($proj_stat == '2') echo '<span class="badge
badge-secondary">Finished</span>';

                                elseif($proj_stat == '3') echo '<span class="badge
badge-danger">Canceled</span>';

                            ?>

                        </div>


                        <div class="project-info-box">

                            <div class="row">

                                <div class="col-sm-6 mb-3">

                                    <small class="text-uppercase">Project
Manager</small>

                                    <div class="text-dark"><?php echo !
empty($row['name']) ? $row['name'] : '<span class="text-muted">Not
Assigned</span>'; ?></div>

                                </div>

                                <div class="col-sm-6 mb-3">

                                    <small class="text-uppercase">Location</small>

                                    <div class="text-dark"><?php echo $proj_loc ?
></div>

                                </div>

                            </div>


                            <div class="row">

                                <div class="col-sm-6 mb-3">

                                    <small class="text-uppercase">Start Date</small>

                                    <div class="text-dark"><i class="fa fa-
calendar"></i> <?php echo $proj_sdate ? date("F d, Y", strtotime($proj_sdate)) :
'-'; ?></div>

                                </div>

                                <div class="col-sm-6 mb-3">

                                    <small class="text-uppercase">Deadline</small>

                                    <div class="text-dark" style="color:
#dc3545;"><i class="fa fa-clock-o"></i> <?php echo $proj_dead ? date("F d, Y",
strtotime($proj_dead)) : '-'; ?></div>

                                </div>

                            </div>


                            <div class="row">
```

```
                            <div class="col-sm-6 mb-3">
                                <small class="text-uppercase">Project
Type</small>
                                <div class="text-dark">
                                    <?php
                                    $types = [
                                        '1' => 'Development', '3' => 'Support',
'4' => 'Design',
                                        '5' => 'Event', '6' => 'Sales', '7' =>
'Manufacturing', '8' => 'Custom Project'
                                    ];
                                    echo isset($types[$proj_type]) ?
$types[$proj_type] : 'Other';
                                    ?>
                                </div>
                            </div>
                            <div class="col-sm-6 mb-3">
                                <small class="text-uppercase">Budget</small>
                                <div class="text-dark" style="color:
#28a745;"><?php echo number_format((float)$proj_cost, 2) . ' $' ?></div>
                            </div>
                        </div>

                        <hr style="margin-top: 5px; margin-bottom: 15px; border-
top: 1px solid #eee;">

                        <div class="row align-items-center" style="display:
flex; align-items: center;">
                            <div class="col-md-8 col-sm-12">
                                <small class="text-uppercase" style="padding-
left: 15px;">Sections</small>
                                <div class="section-container">
                                    <?php
                                        $sql = mysqli_query($conn,"SELECT * from
project_partition natural join project_division where project_id = '$id' order by
division ");
                                        if($sql && mysqli_num_rows($sql) > 0){
                                            while($row2 =
mysqli_fetch_assoc($sql)){
```

```
                                                    echo '<span class="badge badge-
info">'.$row2['division'].'</span>';
                                                }
                                            } else {
                                                echo '<span class="text-muted"
style="font-size:13px; padding-left: 15px;">No sections assigned.</span>';
                                            }
                                        ?>
                                    </div>
                                </div>

                                <div class="col-md-4 col-sm-12 text-right"
style="text-align: right;">
                                    <?php if(!isset($_GET['dattyp'])){ ?>
                                    <a href="#edit_project_modal" data-
toggle="modal" class="btn btn-warning text-white btn-sm">
                                        <i class="fa fa-pencil"></i> Edit Details
                                    </a>
                                    <?php } ?>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>

                <hr style="margin: 40px 0;">

                <div class="row">
                    <div class="col-12">
                        <div style="display: flex; justify-content: space-between;
align-items: center; margin-bottom: 15px; padding: 0 15px;">
                            <h3 class="text-primary" style="font-weight: bold;
margin: 0;">Project Progress</h3>
                            <?php if(!isset($_GET['dattyp'])){ ?>
                                <a href="index.php?page=update_progress&id=<?php
echo $_GET['id'] ?>" class="btn btn-primary btn-md">
                                    <i class="fa fa-bar-chart"></i> Update Progress
                                </a>
```

```
                        <?php } ?>
                    </div>
                    <div class="progress-container" style="background: #fff;
padding: 15px; border-radius: 10px; border: 1px solid #e1e5eb; box-shadow: 0 4px
10px rgba(0,0,0,0.05);">
                        <?php include 'progress_chart.php' ?>
                    </div>
                </div>
            </div>
            <br>
            <div class="row" style="margin-top: 20px;">
                <div class="col-12 text-right" style="text-align: right;">
                    <?php if(isset($_GET['dattyp'])){ ?>
                        <a href="index.php?page=project_list&io=1" class="btn
btn-success"><i class="fa fa-check"></i> Done </a>
                        <a href="cancel_proj.php?id=<?php echo $id ?>"
class="btn btn-danger"><i class="fa fa-times"></i> Cancel Project</a>
                    <?php } else { ?>
                        <a href="index.php?page=project_list&io=1" class="btn
btn-default" style="border: 1px solid #ccc; padding: 8px 20px;">
                            <i class="fa fa-arrow-left"></i> Back to List
                        </a>
                    <?php } ?>
                </div>
            </div>


        </div>
    </div>
    <div id="edit_project_modal" class="modal fade" tabindex="-1" role="dialog"
aria-hidden="true">
        <div class="modal-dialog modal-lg">
            <div class="modal-content">
                <div class="modal-header" style="background-color: #007bff;
color: white;">
                    <h4 class="modal-title" style="margin:0;">Edit Project</h4>
                    <button type="button" class="close" data-dismiss="modal"
style="color:white;">&times;</button>
                </div>
```

```
                <form method="POST" id="proj_form_local">
                    <div class="modal-body" style="max-height: 65vh; overflow-y:
auto; padding: 20px;">
                        <div class="form-horizontal">
                            <input type="hidden" name="id" value="<?php echo $id
?>">
                            <div class="col-sm-12"><div
id="retCode_local"></div></div>


                            <div class="form-group">
                                <label class="col-sm-3 control-label">Project
Name:</label>
                                <div class="col-sm-9">
                                    <input class="form-control" style="text-
transform:capitalize" name="pname" type="text" value="<?php echo $proj_name; ?>"
required>
                                </div>
                            </div>


                            <div class="form-group">
                                <label class="col-sm-3
control-label">Type:</label>
                                <div class="col-sm-9">
                                    <select class="form-control" name="p_type"
required>
                                        <option value="<?php echo $proj_type; ?
>" selected hidden>Current</option>
                                        <option value="1">Development</option>
                                        <option value="3">Support</option>
                                        <option value="4">Design</option>
                                        <option value="5">Event</option>
                                        <option value="6">Sales</option>
                                        <option value="7">Manufacturing</option>
                                        <option value="8">Custom
Project</option>
                                    </select>
                                </div>
                            </div>
```

```
                                <div class="form-group">
                                    <label class="col-sm-3
control-label">Status:</label>
                                    <div class="col-sm-9">
                                        <select class="form-control" name="stats"
required>
                                            <option value="1" <?php echo ($proj_stat
== 1) ? 'selected' : ''; ?>>On Going</option>
                                            <option value="2" <?php echo ($proj_stat
== 2) ? 'selected' : ''; ?>>Finished</option>
                                            <option value="3" <?php echo ($proj_stat
== 3) ? 'selected' : ''; ?>>Canceled</option>
                                        </select>
                                    </div>
                                </div>


                                <div class="form-group">
                                    <label class="col-sm-3 control-
label">Location:</label>
                                    <div class="col-sm-9">
                                        <textarea class="form-control" style="text-
transform:capitalize" name="location" required><?php echo $proj_loc; ?></textarea>
                                    </div>
                                </div>


                                <div class="form-group">
                                    <label class="col-sm-3 control-label">Cost
($):</label>
                                    <div class="col-sm-9">
                                        <input class="form-control" style="text-
align:right" name="cost" type="text" value="<?php echo $proj_cost; ?>">
                                    </div>
                                </div>
                                <div class="form-group">
                                    <label class="col-sm-3 control-
label">Manager:</label>
                                    <div class="col-sm-9">
                                        <select class="form-control" name="tid"
required>
```

```
                                        <?php if(!empty($proj_tid)): ?>
                                            <option value="<?php echo $proj_tid;
?>" selected><?php echo $proj_mngr; ?></option>
                                        <?php else: ?>
                                            <option value="" disabled
selected>Select Manager...</option>
                                        <?php endif; ?>


                                        <?php
                                        $managers = mysqli_query($conn,"SELECT
*, Concat(lastname,', ',firstname) as fullname FROM project_team LEFT JOIN
employee ON project_team.eid = employee.eid WHERE project_team.eid !=
'$current_eid'");
                                        if($managers){
                                            while($m =
mysqli_fetch_assoc($managers)){
                                                echo '<option value="'.
$m['tid'].'">'.$m['fullname'].'</option>';
                                            }
                                        }
                                        ?>
                                    </select>
                                </div>
                            </div>


                            <div class="form-group">
                                <label class="col-sm-3 control-label">Start
Date:</label>
                                <div class="col-sm-9">
                                    <input class="form-control" name="sdate"
type="date" value="<?php echo $proj_sdate; ?>" required>
                                </div>
                            </div>
                            <div class="form-group">
                                <label class="col-sm-3 control-
label">Deadline:</label>
                                <div class="col-sm-9">
                                    <input class="form-control" name="deadline"
type="date" value="<?php echo $proj_dead; ?>" required>
```

```
                                </div>
                            </div>
                        </div>
                    </div>


                    <div class="modal-footer" style="background-color: #f9f9f9;
border-top: 1px solid #ddd;">
                        <button type="submit" class="btn btn-info">Save
Changes</button>
                        <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
                    </div>
                </form>
            </div>
        </div>
    </div>


    <div id="local_change_pic" class="modal fade" tabindex="-1" role="dialog"
aria-hidden="true">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header" style="background-color: #007bff;
color: white;">
                    <h4 class="modal-title" style="margin:0;">Change
Picture</h4>
                    <button type="button" class="close" data-dismiss="modal"
style="color:white;">&times;</button>
                </div>
                <form method="POST" action="../forms/update_forms.php?
action=change_pic2&id=<?php echo $id ?>" enctype="multipart/form-data">
                    <div class="modal-body">
                        <div class="form-horizontal">
                            <div class="form-group" style="display: flex; align-
items: center; margin-bottom: 0;">
                                <label class="col-sm-4 control-label"
style="text-align: right;">Upload File:</label>
                                <div class="col-sm-8">
                                    <input class="form-control" name="file"
type="file" required style="padding-top: 5px;">
```

```
                                </div>
                            </div>
                        </div>
                        <div class="modal-footer" style="background-color: #f9f9f9;
border-top: 1px solid #ddd;">
                            <button type="submit" class="btn btn-info">Save</button>
                            <button type="button" class="btn btn-default" data-
dismiss="modal">Close</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>

        <?php include '../includes/update_modals.php' ?>
        <div id="retCode1"></div>
        <script>
        jQuery(document).ready(function(){
            jQuery("#proj_form_local").submit(function(e){
                e.preventDefault();
                var formData = jQuery(this).serialize();
                $.ajax({
                    type: "POST",
                    url: "../forms/update_forms.php?action=project",
                    data: formData,
                    success: function(html){
                        $('#retCode_local').html(html);
                        setTimeout(function(){ location.reload(); }, 1000);
                    },
                    error: function(){
                        alert("Error saving data. Please check required fields.");
                    }
                });
                return false;
            });
```

```
});
</script>
```

Додаток 2

Апробація результатів кваліфікаційної роботи магістра на XV Всеукраїнській науково-практичній конференції «Актуальні проблеми комп’ютерних наук АПКН»

Тези

ПРОГРАМНИЙ ЗАСТОСУНОК ДЛЯ КЕРУВАННЯ ПРОЕКТАМИ

Управління проектами у сучасному світі є критично важливим для успіху організацій у різних галузях. Зміни в суспільстві та технологічний прогрес потребують постійного розвитку систем управління проектами. Ця робота є дослідженням та розробкою інноваційної системи управління проектами, яка враховує сучасні тенденції та потреби. Аналіз показує зростання популярності Agile методологій, розвиток інструментів та перехід до віддаленої роботи. Розробка системи спрямована на поліпшення планування, ресурсного менеджменту, моніторингу та комунікації в управлінні проектами, враховуючи потреби сучасного світу.

Project management in today's world is critical to the success of organizations in various industries. Changes in society and technological progress require continuous development of project management systems. This work is research and development of an innovative project management system that takes into account modern trends and needs. The analysis shows the growing popularity of Agile methodologies, the development of tools and the transition to remote work. The development of the system is aimed at improving planning, resource management, monitoring and communication in project management, taking into account the needs of the modern world.

Управління проектами в сучасному світі є важливим складовим елементом ефективного функціонування організацій у різних галузях, від бізнесу та науки до громадських ініціатив. Ефективне планування, виконання та контроль проектів стає стратегічним завданням для досягнення успіху та конкурентоспроможності. Однак, у зв'язку з постійними змінами в суспільстві та технологічному прогресі, системи управління проектами також повинні постійно розвиватися та адаптуватися до цих змін.

Метою роботи є розгляд і дослідження питань, пов'язаних із розробкою і вдосконаленням систем управління проектами. Дослідження спрямоване на інтеграцію передових методологій та інноваційних технологій для створення більш ефективної системи управління проектами, яка відповідає сучасним потребам.

Аналіз сучасних тенденцій у галузі управління проектами відображає наступні ключові аспекти:

- Зростання популярності Agile методологій: Agile підходи до управління проектами, такі як Scrum та Kanban, все більше розповсюджуються завдяки їх гнучкості та спроможності швидко реагувати на зміни в умовах проекту. Agile дозволяє командам працювати більш ітеративно та відповідати на потреби клієнта.
- Розвиток багатофункціональних інструментів: Інструменти для управління проектами додають більше функцій, об'єднуючи в собі можливості для планування, моніторингу, комунікації та аналізу проектів. Це спрощує процес управління проектами та забезпечує доступ до всіх необхідних інструментів в одному місці.
- Зростання популярності віддаленої роботи в управлінні проектами: Віддалена робота стає стандартом у світі, що вимагає нових підходів до управління проектами, включаючи віддалену комунікацію та спільну роботу команд, що може бути викликом та одночасно перевагою для управління проектами.

На основі вищесказаного формується концепція розробки інноваційної системи управління проектами, яка орієнтована на вдосконалення та оптимізацію управління проектами.

Основні характеристики цієї системи включають:

1. Поліпшене планування: Система надає можливість створювати докладні та ефективні плани проектів, враховуючи ресурси, терміни та завдання.
2. Ефективний ресурсний менеджмент: Вона допомагає керувати ресурсами проектів, включаючи людські та матеріальні ресурси, з метою забезпечення їх ефективного використання.
3. Моніторинг та звітність: Система надає засоби для постійного моніторингу стану проектів та генерації звітів для аналізу продуктивності та досягнень.
4. Комунікація та співпраця: Вона сприяє ефективній комунікації та співпраці між членами проектних команд, забезпечуючи якісне обмін інформацією та робочими процесами.
5. Аналіз та вдосконалення процесів: Система надає інструменти для аналізу та оптимізації процесів управління проектами, що спрямовані на підвищення продуктивності та якості управління проектами.

В ході проектування програмного застосунку було розроблено діаграму потоків даних(рисунок 1) та flowchart(рисунок 2) для демонстрації загальної роботи з застосунком..

Ця розробка має потенціал стати важливим інструментом для покращення управління проектами, полегшуючи роботу організацій та сприяючи їхній конкурентоспроможності. Вона може служити базовою платформою для інновацій та змін в управлінні проектами, а також сприяти розвитку активної спільноти користувачів, що спільно вдосконалюють процеси управління проектами. Загалом, ця розробка відкриває широкі перспективи для подальшого росту та вдосконалення у сфері управління проектами.

Перелік посилань

1. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). Sixth Edition. USA. PMI, 2017. 66-67 p.
2. Пуліна Т. В. Можливості та перспективи управління проектами на основі методології ISO 21500 / Т. В. Пуліна, Т О. Фоміних, О. О. Соріна // Економічний простір. - 2017. - № 122. - С. 126-138.