Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Якимчук Ганни Богданівни
Прізвище, ім'я, по батькові студента(ки)

на здобуття ступеня вищої освіти Бакалавра

Програмний застосунок для керування проєктами
Назва теми

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.200176.01.25.ПЗ

Виконав студент IV курсу група ІПЗ-20-1 _____ Ганна ЯКИМЧУК
Підпис Ім'я, ПРІЗВИЩЕ

Керівник канд. пед. наук, доцент _____ Наталія ПРАВОРСЬКА
Науковий ступінь, звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер канд. тех. наук, доцент _____ Оксана ЯШИНА
Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії
програмного забезпечення _____ Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

6 червня 2024 р.

Хмельницький 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Л. П. Бедратюк
02. 01 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Якимчук Ганні Богданівні
Прізвище, ім'я, по батькові студента

1. Тема роботи Програмний застосунок для керування проєктами

Керівник роботи Праворська Наталія Іванівна, канд. пед. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 08.01.2024 р. № 6
2. Строк подання студентом роботи на кафедру 01.06.2024 р.
3. Вихідні дані до роботи Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування програмного забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Три креслення:
1. Діаграма варіантів використання
2. Логічна модель БД
3. Структура Бази Даних

6. Консультанти розділів дипломного проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Яшина О. М. к.т.н. доцент	29.05.24	31.05.24
Антиплагіат	Форкун Ю. В., доцент	31.05.24	06.06.24

7. Дата видачі завдання « 02 » січня 2024р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12 – 31.12.2023	
2 Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	01.01 – 20.02.2024	
3 Проектування програмного забезпечення	21.02 – 20.03.2024	
4 Програмна реалізація з використанням відповідних засобів розробки. Тестування ПЗ	21.03 – 30.04.2024	
5 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог стандартів	01.05 – 25.05.2024	
6 Попередній захист КвР	Травень 2024	Згідно графіка
7 Перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2024	
8 Здача КвР на кафедру; підготовка КвР для розміщення у репозитарій ХНУ; підготовка до захисту та захист КвР	з 01.06.2024	

Студент _____ Ганна ЯКИМЧУК
Підпис Ім'я ПРІЗВИЩЕ

Керівник роботи _____ Наталія ПРАВОРСЬКА
Підпис Ім'я ПРІЗВИЩЕ

АНОТАЦІЯ

Тема кваліфікаційної роботи: Програмний застосунок для керування проєктами.

Автор проекту: Якимчук Ганна Богданівна.

Керівник проекту: Праворська Наталія Іванівна.

Пояснювальна записка: 92 с., 23 рис., 7 табл., 3 дод., 40 джерел.

Графічна частина: 13 слайдів, 3 креслення.

УПРАВЛІННЯ ПРОЄКТАМИ, ВЕБ-САЙТ, PHP, PHP-STORM, XAMPP

Метою проекту є розробка високоефективного застосунку, що дасть можливість вирішити завдання керування проєктами, і забезпечити зручний та зрозумілий інтерфейс для користувачів.

У кваліфікаційній роботі було проведено дослідження предметної області та постановку задач, спроєктовано програмне забезпечення, розроблено і протестовано застосунок та підведено підсумки.

Для розробки була використана мова PHP, та база даних MySQL.

В результаті було розроблено застосунок для керування проєктами, який має дружній інтерфейс, підтримує різноманітні функціональні можливості, такі як: створення проєктів, призначення завдань, відстеження прогресу, створення команд.

06.06.24
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ. 200176.01.25.ПЗ	Пояснювальна записка	92		
2	А4		Завдання на кваліфікаційну роботу	1		
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	13		
	А3		Креслення	3		

				КвРІПЗ. 200176.01.25.ПЗ	
Змн	Арк	№ докум.	Підпис	Дата	
Виконав		Якимчук Г.Б.		06.06	Програмний застосунок для керування проєктами
Керівник		ПраворськаН І.		06.06	
Н. Контр.		Яшина О. М.		06.06	Відомість документів
Зав. Каф.		Бевзпалюк П.П.		06.06	

Літ	Арк	Аркушів
	1	1

ХНУ, ІПЗ-20-1

ЗМІСТ

ВСТУП

В теперішньому інформаційному просторі, що досі стрімко розвивається, однією з важливих складових успіху є ефективне керування проєктами. Разом з зростанням обсягів інформації та появою нових технологій, з якими доводилось працювати, зросла потреба у програмних засобах, які спрямовані на спрощення та удосконалення процесів керування проєктами. Проте існуючі рішення не завжди відповідають теперішнім вимогам та можуть бути недостатньо ефективними у деяких аспектах.

Мета роботи – це розробка високоефективного застосунку, що дасть можливість вирішити завдання керування проєктами, і забезпечити зручний та зрозумілий інтерфейс для користувачів.

Щоб її досягти треба дослідити, розробити та впровадити інноваційні інструменти, які будуть сприяти оптимізації процесів планування, контролю та виконання проєктів.

Актуальність цього дослідження визначається необхідністю створення нового застосунку для керування проєктами, який буде враховувати сучасні тенденції розвитку та особливості предметної області.

Однією з ключових проблем, на яку націлена дана розробка, є покращення ефективності керування проєктами через використання нових доступних інформаційних технологій та оптимізацію наявних процесів взаємодії між учасниками проєкту.

Враховуючи те, що інформаційні технології стають невід'ємною частиною багатьох галузей діяльності, розроблення застосунку для керування проєктами потрібна не тільки для IT-проектів, а й для бізнесу, науки, освіти та багатьох інших сфер.

Тому розробка застосунку має велику значущість в контексті розвитку інформаційних технологій та їх впливу на сучасне керування проєктами. У цьому контексті особливо важливим стає використання інноваційних технологій

щоб автоматизувати процеси керування та вдосконалення комунікації між учасниками проєкту.

Сучасний застосунок для керування проєктами має враховувати не тільки планування та розподіл задач, але й забезпечувати аналіз даних, прогнозування та оперативний обмін інформацією.

Для досягнення поставленої мети передбачено вирішення всіх наступних задач проєктування:

– аналіз потреб користувачів: вивчення потреб та вимог потенційних користувачів майбутнього програмного продукту для визначення основних функціональних можливостей;

– проєктування інтерфейсу: розробка зручного та інтуїтивно зрозумілого інтерфейсу, що дозволить користувачам зручно та ефективно взаємодіяти з програмним засобом;

– розробка та тестування: створення програмного застосунку відповідно до встановлених вимог та проведення його випробувань для виявлення та усунення помилок;

– впровадження та підтримка: впровадження розробленого програмного продукту в робоче середовище користувачів та подальша підтримка і супровід.

Такий систематизований план дій дозволить крок за кроком реалізувати поставлену мету та враховує ключові аспекти керування проєктом. Розроблення програмного застосунку для керування проєктами має великий потенціал у забезпеченні ефективного та результативного керування проєктами в сучасному бізнес-середовищі, а розроблений програмний продукт має потенціал стати цінним інструментом для різноманітних галузей та компаній.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

На сьогоднішній день управління проєктами є все більш і більш важливим та складним завданням для різноманітних сфер бізнесу та промисловості. Предметна область керування проєктами тепер охоплює все широкий спектр діяльностей, включаючи планування, виконання та контроль над проєктами з метою досягнення конкретних цілей та результатів.

Управління проєктами включає розподіл ресурсів, координацію робіт, контроль за бюджетом та термінами, а також забезпечення ефективного комунікаційного процесу між учасниками проєкту[1]. Вимоги до програмних засобів для управління проєктами постійно зростають через збільшення складності та потреби у більш детальному аналізі сучасних проєктів[2].

Що стосується цільової аудиторії програмного засобу для керування проєктами, вона включає в себе проєктних менеджерів, команду, виконавців задач, а також керівників підприємств та власників бізнесу. Кожен з учасників проектного процесу має свої особливі потреби та очікування від ПЗ для керування проєктами. Тому, таке програмне забезпечення повинно бути універсальним та гнучким. Основні вимоги до подібного програмного забезпечення включають в себе зручний користувацький інтерфейс, можливість ведення проєктів у реальному часі, гнучкість в налаштуванні процесів керування, а також надійність та захист всіх користувацьких даних.

Предметна область керування проєктами постійно розвивається, впроваджуються нові методи та технології, такі як Agile, Lean, Scrum[3], Kanban тощо. Важливу роль у цьому грають інноваційні технології, такі як: штучний інтелект, аналіз даних та хмарні технології, що допомагають покращити ефективність та результативність керування проєктами.

Загальний огляд та опис предметної області керування проєктами є важливою передумовою для розробки програмного засобу в цій галузі. Це дозволяє отримати базове розуміння основних принципів, процесів та вимог, які стосуються керування проєктами. Такий огляд забезпечує команду розробників необхідною інформацією для створення ефективного та корисного програмного продукту, який відповідає потребам користувачів[4].

В сучасному керуванні проєктами основною проблемою стає неефективність традиційних методів керування та обмежена гнучкість у реагуванні на зміни в проєктному середовищі. Класичні методи, такі як водоспадна модель керування проєктами, не завжди відповідають потребам в сучасних умовах, де вимагається швидке реагування на зміни в умовах ринку.

Однією з основних проблем є недостатня ясність та доступність інформації про проєкт для всіх членів команди. Це може призводити до непорозумінь, затримок у виконанні задач та неспроможності реагувати на зміни в умовах проєкту своєчасно.

Також важливою проблемою є складність управління ресурсами та виконанням задач у рамках проєкту. Недостатня прозорість, щодо навантаження ресурсів та неефективний розподіл задач, можуть викликати перевантаження певних учасників команди та призвести до затримок у реалізації проєкту.

Сучасні проєкти потребують гнучких та динамічних методів керування. Традиційні підходи не завжди ефективні, адже не враховують мінливість умов та потреб. Зміни в ринку чи вимогах клієнтів можуть змусити швидко переглядати стратегію, а ось традиційні методи не завжди дають змогу зробити це оперативно. Отже, розробка ПЗ для керування проєктами має на меті створити гнучкий інструмент, що підвищить ефективність керування сучасними динамічними проєктами. Аналіз інформаційних потреб користувачів – це невід'ємний етап розробки програмного забезпечення для керування проєктами.

Існує декілька ключових напрямків, які допоможуть у визначенні інформаційних потреб для побудови якісного плану для ефективного керування будь-якими проєктами:

– вимоги до проєкту: чітке розуміння задач є ключовим фактором успішного проєкту. Це включає в себе опис обсягу робіт, термінів виконання, бюджету, технічних вимог та інших ключових параметрів;

– ресурси та навантаження: важливо знати, як саме будуть розподілені ресурси (людські, матеріальні, фінансові) між різними задачами проєкту, а також, яке навантаження очікує на членів команди;

– статус та прогрес проєкту: користувачі повинні мати можливість бачити, на якому етапі знаходиться проєкт, як проходить виконання задач, чи дотримуються графіки та терміни;

– ризики та проблеми: знання про потенційні ризики та проблеми, які можуть виникнути під час реалізації проєкту, допоможе заздалегідь спланувати дії для їх усунення або мінімізації негативного впливу;

– комунікація та співпраця: користувачі повинні мати доступ до інструментів, які допоможуть їм спілкуватися з іншими учасниками проєкту (членами команди, замовниками, зацікавленими сторонами).

Розуміння інформаційних потреб різних учасників проєктного процесу[5] дає чітке уявлення про те, яким має бути програмний продукт. Це робить програмне забезпечення не просто набором інструментів, а й дійсно корисним та ефективним помічником у керуванні задачами та проєктами.

Розробка була висвітлена в тезах Всеукраїнської науково-практичної конференції «Актуальні проблеми компютерних наук АПКН-2023»[6].

Отже, зосередженість на потребах користувачів робить програмне забезпечення зручним, ефективним, гнучким, надійним та безпечним інструментом для успішного керування проектами.

Таким чином, розробка програмного забезпечення з урахуванням інформаційних потреб користувачів, робить його цінним інструментом для будь-якої команди, що прагне досягти успіху в реалізації своїх проєктів.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

У рамках дослідження проблеми, пов’язаної з керування проєктами, важливим етапом є аналіз наявного програмно-технічного забезпечення, яке застосовується в цій сфері. Для проектування корисного та ефективного програмного забезпечення доцільно зробити детальний огляд та оцінку існуючих програмних засобів та технічних рішень, які використовуються для керування проєктами у різних організаціях та галузях.

Аналіз наявного програмно-технічного забезпечення дозволяє зрозуміти, які підходи та інструменти вже використовуються на практиці, їх переваги та недоліки, і, також, виявити можливості для інновацій та покращень у даній предметній області.

Для наочного розуміння переваг та недоліків наведено комплексний аналіз для різних технічних рішень та програмних засобів, які часто використовуються в сфері керування проєктами.

Для аналізу було обрано три популярних застосунки для керування проєктами, а саме Jira, Trello та MS Project.

Jira[7] – це популярний інструмент для керування проєктами, який використовують команди щоб відстежувати завдання, планувати sprintи, вести backlog-списки та спілкуватися.

Перевагами Jira є:

– гнучкість: Jira підходить для різних типів проєктів, будь то Agile розробка, Kanban, Scrum чи інші методології;

– налаштування: можна гнучко налаштувати середовище під потреби команди, додаючи поля, типи завдань, workflows та інтеграції;

– Agile-функціонал: підтримує методології Scrum, Kanban, Scrumban;

– спілкування: Jira містить в собі вбудовані інструменти для комунікації, такі як чат, коментарі до завдань та форуми;

– інтеграції: Jira може легко інтегруватись з іншими інструментами, наприклад Slack, Bitbucket , Confluenc, та GitHub.

Однак Jira має також ряд недоліків:

– складність: зазвичай Jira складна для новачків, і потребує певного часу для налаштування;

– вартість: рішення може бути дорогим для невеликих команд;

– не підходить для не-Agile проєктів: Jira не оптимальний вибір для проєктів, які не використовують Agile-методології.

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

Dashboard використовується для візуалізації завдань та їх статусу. Користувачі можуть переміщати картки завдань між колонками, щоб відстежувати їх прогрес. Jira Dashboard – це потужний інструмент, який має багато можливостей і функціоналу та допомагає командам бути більш організованими та продуктивними.

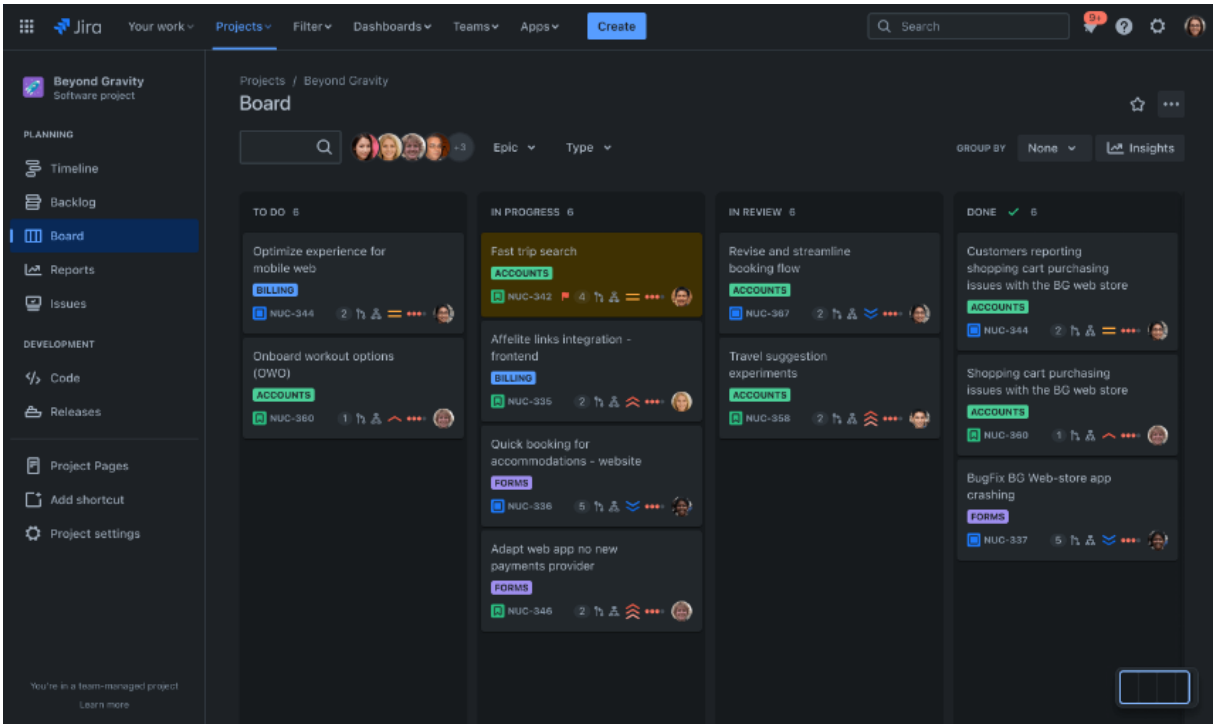


Рисунок 1.1 – Jira Dashboard

Функція відстеження часу (рисунок 2.2) – це інструмент, який допомагає командам орієнтуватися в прогресі виконання проєкту, розставляти пріоритети, обговорювати роботу команди та забезпечувати видимість на кожному рівні.

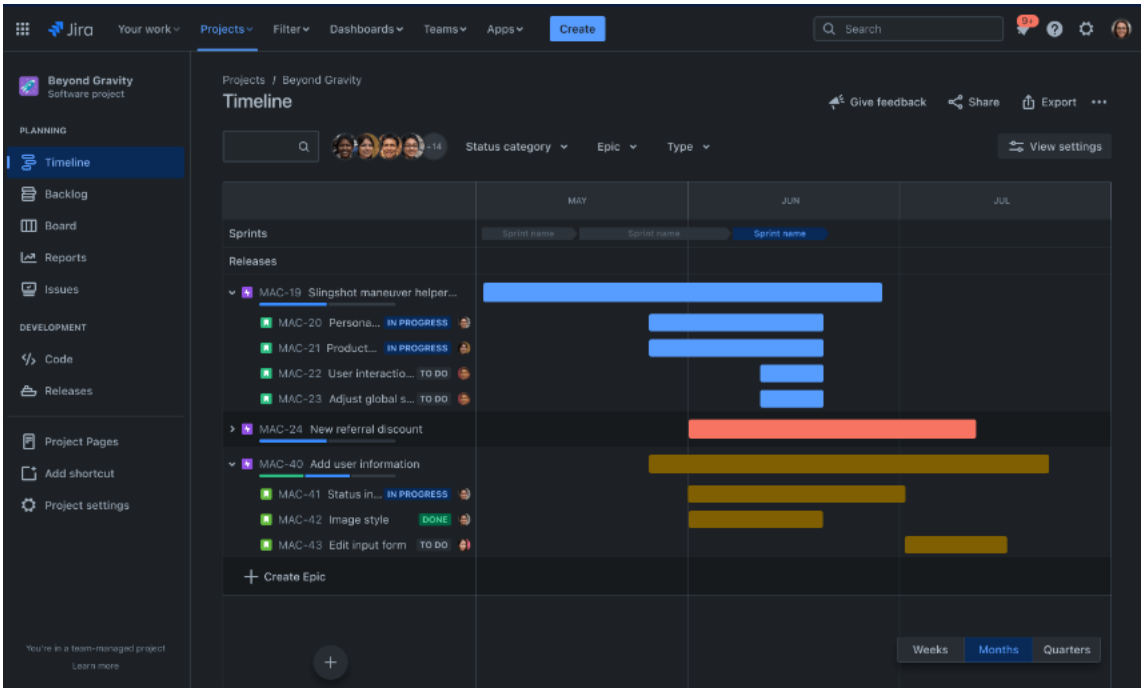


Рисунок 2.2 – Jira Timeline

Отже, Jira – це потужний інструмент, який може зробити команду більш продуктивною та організованою. Jira надає команді можливість: планувати та організовувати свою роботу, відстежувати прогрес та результати, спілкуватися та співпрацювати між собою.

Наступний застосунок для аналізу – Trello.

Trello[8] – це відомий інструмент для візуального керування проєктами, який використовується командами для структурування завдань, спільного доступу до інформації та комунікації.

Перевагами Trello є:

– простота: Trello легкий у використанні, навіть для початківців;

– візуалізація: використовує дошки, картки та списки для візуального представлення задач, щоб зробити їх зрозумілими та зручними для відстеження;

– гнучкість: інструмент можна використовувати для різних типів проєктів, включаючи agile-розробку, Scrum, Kanban та інші методології;

– співпраця: є вбудовані інструменти для комунікації, такі як коментарі до карток та історія дій на дошці;

– доступність: є безкоштовний план, що підійде для невеликих команд.

Недоліки Trello:

– обмежена функціональність: у порівнянні з Jira, Trello не має такого ж широкого спектру функцій;

– не підходить для складних проектів: не оптимальний вибір для великих проєктів з великою кількістю завдань;

– відсутність інтеграцій: Trello не надає інтеграцію з популярними інструментами, такими як Slack та GitHub.

Центральним елементом є дошка Trello (рисунок 1.3), призначена для візуалізації завдань та їх стану. Користувачі можуть переміщати картки з завданнями між колонками, щоб відслідковувати їх прогрес. Завдяки дошці Trello завдання організовані, і робота продовжується.

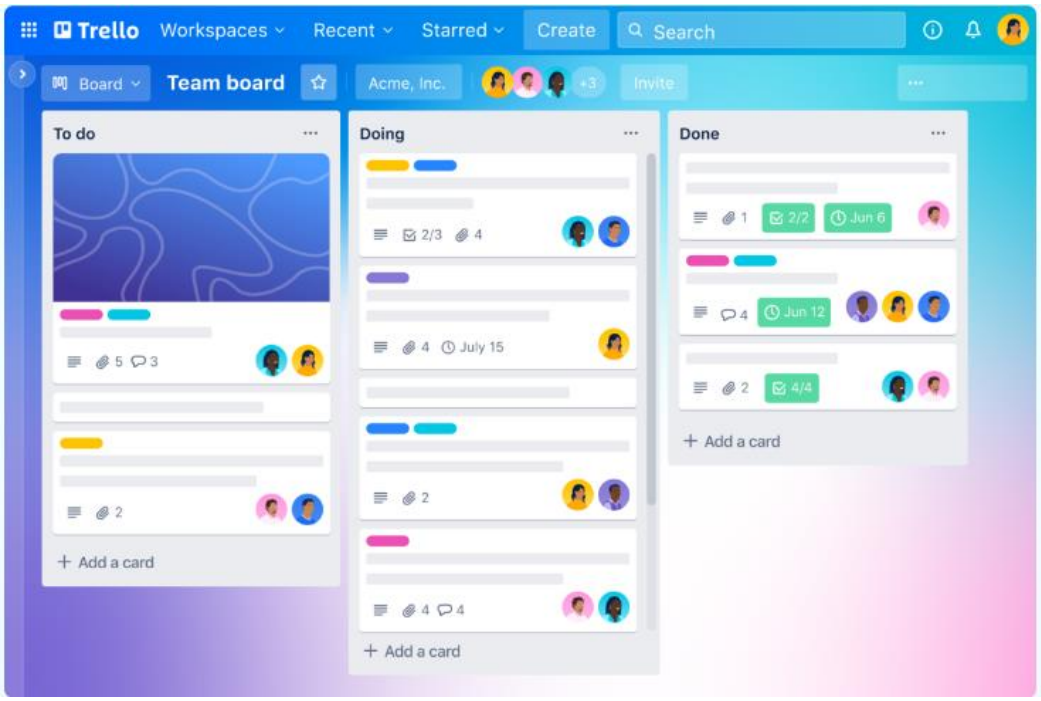


Рисунок 1.3 – Trello Board

Сторінка Календаря в Trello (рисунок 1.4) надає користувачеві огляд завдань, запланованих на найближчі дні, тижні та місяці. Якщо команда оперує датами виконання та строками, календар допомагає чітко уявити загальну роботу і сфокусуватися на конкретних завданнях, які потрібно виконати прямо зараз, а які можна відкласти на даний момент.

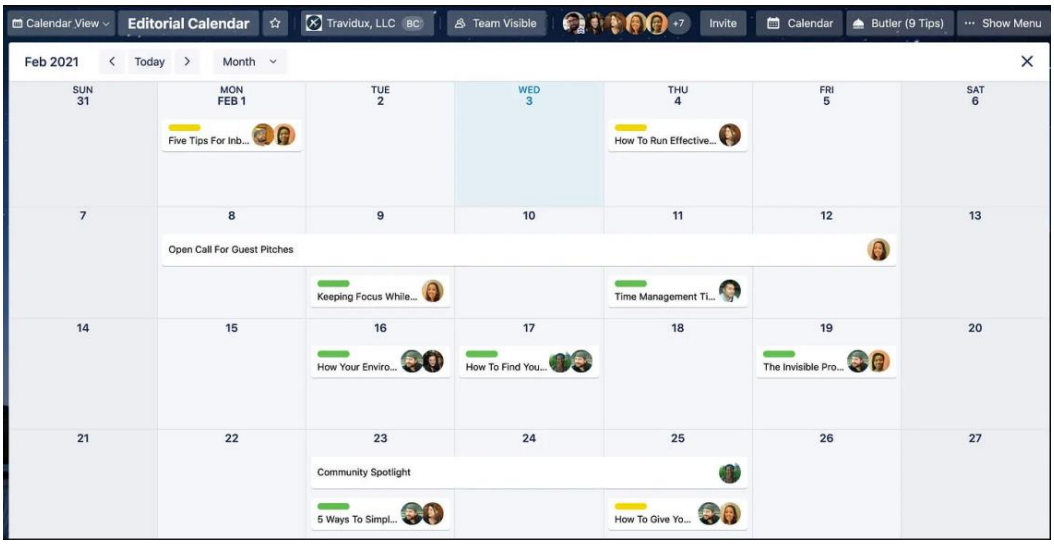


Рисунок 1.4 – Trello Calendar

Отже, Trello є простим та зручним інструментом для візуального керування проєктами, для організованості та продуктивності команд та її членів.

Ще один огляд наявного рішення – MS Project.

MS Project[9] – це високопотужний інструмент для керування проєктами, який використовується командами для планування, створення Gantt-діаграм, відстеження задач, складання звітів та багатьох інших завдань.

До переваг MS Project відносяться:

– потужні функції: має широкий спектр інструментів для управління проєктами, для планування, Gantt-діаграм, ресурсів, бюджетування, звітності;

– гнучкість: може бути використаний для будь-яких проектів, незалежно від того, чи це agile-розробка, Kanban, Scrum або інші методології;

– інтеграція з Office: MS Project працює з іншими продуктами Microsoft Office, такими як Excel, Word та PowerPoint;

– широке використання: застосунок зараз один з найпоширеніших інструментів для управління проєктами у світі.

Недоліками MS Project можна назвати:

– складність: складний для новачків, і потребує часу для навчання;

– вартість: дороге рішення, особливо для невеликої команди;

– не підходить для Agile-команд: MS Project не оптимальний вибір для команд, які працюють з Agile-методологіям.

MS Project має великий функціонал, який включає можливість планування завдань проєкту, управління ресурсами, відстеження ходу виконання проєкту та інші можливості. Програма надає різноманітні представлення (рисунок 1.5), які відображають інформацію про проєкт у різних форматах і з різних точок зору. Ці представлення можна використовувати за потребами проєкту.

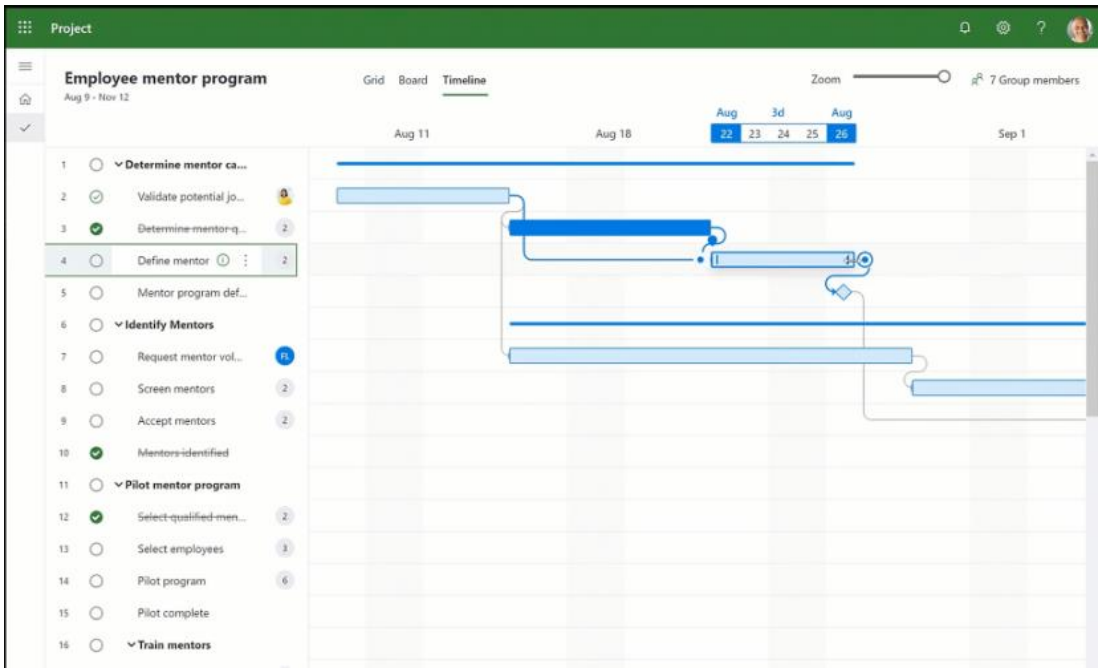


Рисунок 3.5 – MS Project Board

MS Project пропонує величезний спектр налаштувань (рисунок 1.6), що дозволяє адаптувати програму під будь-який робочий процес та будь-яку команду. Ці налаштування допомагають автоматизувати робочі процеси для досягнення ефективної роботи.

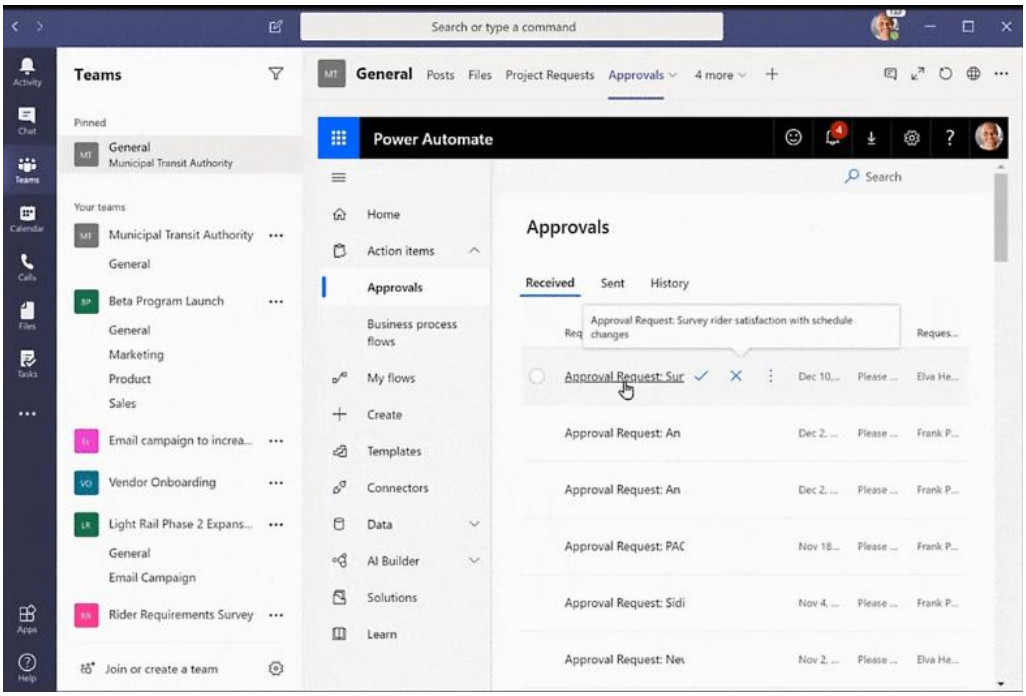


Рисунок 1.6 – MS Project Settings

Звідси видно, що MS Project є потужним інструментом для управління проєктами, який може допомогти командам стати більш організованими та продуктивними.

На основі короткого огляду цих трьох застосунків можна визначити їх порівняльні характеристики у вигляді таблиці (таблиця 1.1).

Таблиця 1.1 – Порівняльні характеристики

Критерій	Jira	Trello	MS Project
Призначення	Управління проектами, відстеження завдань, agile-розробка	Візуальне управління проектами, командна робота	Професійне управління проектами, планування, Gantt-діаграми
Фірма-розробник	Atlassian	Trello Enterprises	Microsoft
Інтерфейс	Дошки, списки, картки, календарі	Дошки, картки, списки	Gantt-діаграми, таблиці, календарі
Переваги	Гнучкість, налаштування, agile-методології, інтеграції	Простота, візуалізація, командна робота	Потужні інструменти, планування, Gantt-діаграми.
Недоліки	Складність, висока ціна	Обмежена функціональність, не для складних проектів	Складність, висока ціна, не підходить для agile-команд

Отже, порівнявши цю інформацію, можна зробити висновки щодо основних характеристик, функціоналу, переваг і недоліків проаналізованого програмного забезпечення:

– Jira – універсальний варіант для більшості команд, особливо тих, хто використовує Agile-методології;

– Trello – чудовий вибір для невеликих команд та простих проектів, де потрібен швидкий та візуальний підхід;

– MS Project – потужний інструмент для великих та складних проектів з чіткими етапами та залежностями.

Отриманий аналіз можна використати для подальшого проектування власного програмного застосунку для керування проєктами з метою максимізації його ефективності та привабливості для потенційних користувачів.

1.3 Визначення вимог до програмного забезпечення та технічне завдання

Етап аналізу вимог та розробка технічного завдання є важливим кроком у процесі розробки програмного забезпечення. Цей етап необхідний для того, щоб чітко визначити та описати вимоги до майбутнього програмного продукту на основі детального аналізу предметної області.

Використання діаграм потоків даних[10] може виявитися вельми корисним для наочного відображення основного функціоналу програмного застосунку та його взаємодії з базою даних. Ці діаграми дозволяють краще розуміти процеси, які відбуваються всередині системи, і показують, як дані переміщуються від одного етапу до іншого.

На діаграмі потоків даних[11] (рисунок 1.7), можна побачити основні характеристики системи керування проєктами, які переходять у функціонал програмного застосунку та взаємодіють між собою. Це допомагає сформувати загальну структуру програмного застосунку та визначити необхідний функціонал для його реалізації.

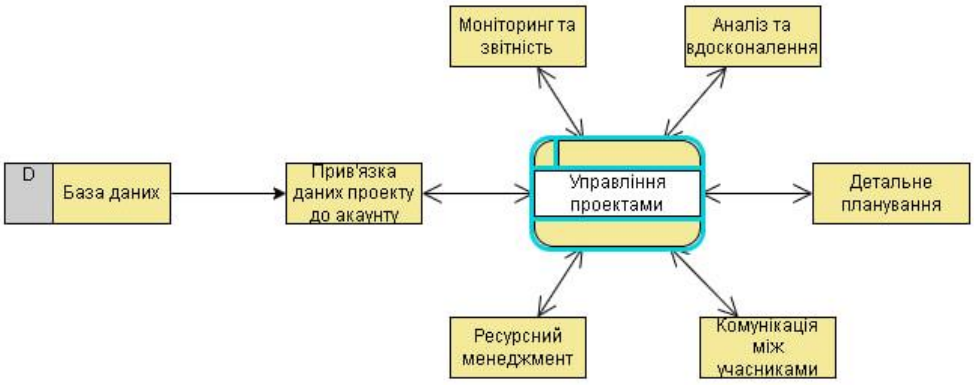


Рисунок 1.7 – DFD діаграма

Для кращого усвідомлення процесів та ролей, що забезпечуватиме розроблюване програмне забезпечення, цілком доцільним є використання діаграм, які візуально відображають систему. Ці візуальні зображення дозволяють легше сприймати складні системи та визначити взаємодію їх складових частин.

Щоб визначити вимоги до програмного забезпечення потрібно створити діаграму варіантів використання[12]. Для цієї діаграми описано всіх акторів, що беруть участь у системі (таблиця 1.2). Опис акторів у системі надає додатковий контекст для розуміння взаємодії між користувачами та програмним забезпеченням. Це допомагає уникнути непорозумінь та забезпечує чіткість у визначенні вимог до системи.

Таблиця 1.2 – Опис акторів розроблюваного ПЗ

Актор	Короткий опис
Неавторизований користувач	Не має доступу безпосередньо до додатку, може тільки авторизуватись
Користувач	Може створювати та працювати вже з існуючими проектами, задачами, додавати коментарі, редагувати інформацію про себе
Адміністратор	Має усі права, має доступ до всіх користувачів, змінювати їх інформацію та додавати нових користувачів

Наступним кроком необхідно описати всі варіанти використання, які доступні для кожного з цих акторів (таблиця 1.3).

Таблиця 1.3 – Опис варіантів використання

Актор	Варіант використання	Опис
Неавторизований користувач	Вхід	Користувач вводить логін і пароль для входу в систему
Авторизований користувач	Управління проєктами	Користувач змінює інформацію про проєкт, додає нові задачі, закриває старі.
	Створення задач	Користувач додає нові задачі в проект
	Управління задачами	Користувач змінює існуючі задачі, закриває вже виконані
	Призначення відповідального	Користувач може призначати і змінювати відповідального за задачу
	Зміна статусу	Користувач може змінювати статус задачі або проекту
	Додавання коментарів	Користувач може залишати коментарі до задачі
Адміністратор	Додавання нових користувачів	Адміністратор може додавати нових користувачів в систему
	Створення проєкту	Адміністратор може створити новий проєкт
	Управління користувачами	Адміністратор може переглядати і змінювати дані користувачів
	Управління всім застосунком	Адміністратор може налаштовувати робочий простір в системі

На основі цих описів акторів та варіантів використання для застосунку для керування проєктами можна побачити діаграми варіантів використання, які демонструють всі процеси взаємодії[13] (рисунок 1.8, рисунок 1.9). Дані діаграми спрощують подальше проєктування та демонструють взаємодію між акторами та структуру майбутнього застосунку.

Діаграма варіантів використання (рисунок 1.8) демонструє взаємодію між акторами і додатком беручи за основу статус користувача в системі[14].

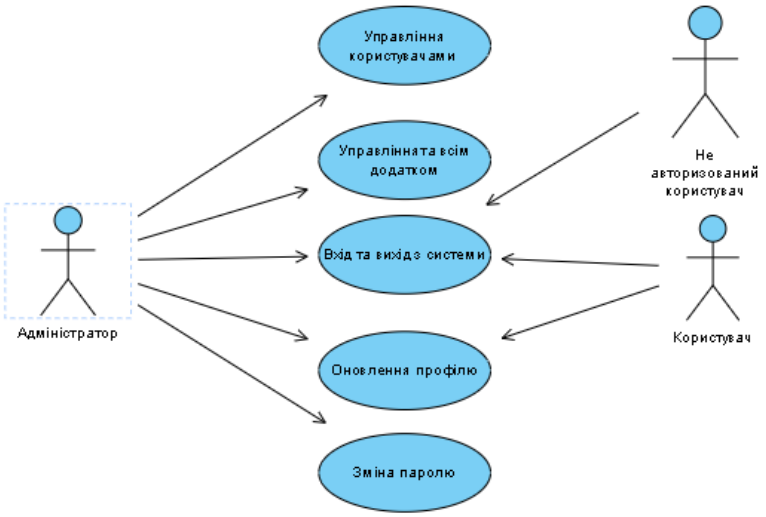


Рисунок 1.8 – Діаграма варіантів використання

Інша діаграма використання(рисунок 1.9) демонструє взаємодію акторів безпосередньо з доступним функціоналом застосунку[15].

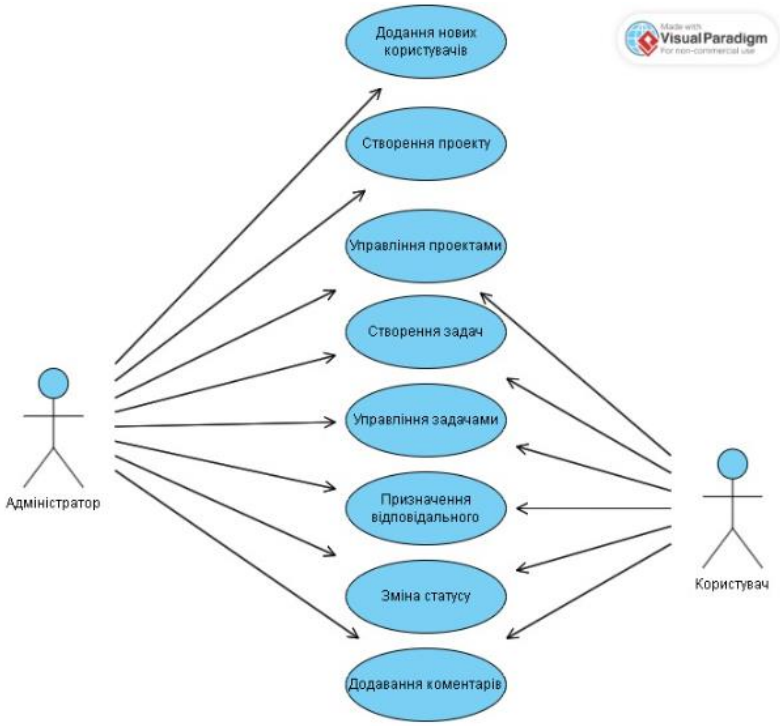


Рисунок 1.9 – Діаграма варіантів використання

На основі проведеного аналізу вимог до ПЗ було розроблене технічне завдання[16], яке подано у додатку А.

1.4 Висновки дослідження предметної області та постановки задачі

Отже, у даному розділі проведено аналіз сучасних методологій керування проєктами, встановлено особливості предметної області та проаналізовано існуючі рішення, які користуються попитом на даний момент, та вимоги користувачів до програмного забезпечення.

Аналіз показав, що керування проєктами є складним та багатогранним процесом, для якого існують різні методології. Потреби користувачів є різноманітними та включають в себе не лише планування та контроль, а й використання ресурсів, аналіз ризиків та звітність.

Вимоги користувачів до програмного забезпечення показують, що необхідно розробити інструмент, який буде гнучким, зручним у використанні та забезпечить повний цикл управління проєктами від планування до аналізу результатів.

Отже, на основі аналізу визначено важливі аспекти, які необхідно врахувати під час подальшої розробки програмного застосунку для керування проєктами, які є основою для подальшого проєктування та реалізації.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Проєктування архітектури та структури системи

Архітектура системи – це не просто набір технічних деталей, це фундамент, на якому ґрунтується вся система. Вона визначає не лише структуру програми, але й її здатність до масштабування, підтримки та розвитку в майбутньому[17]. Якщо порівняти систему з будівлею, то архітектура – це її фундаментальний каркас, на якому базуються всі інші складові.

Проєктування програмного забезпечення – це процес, важливий не менше, ніж будівництво самої системи. Продумана архітектура відіграє критичну роль у спрощенні розробки, впровадженні нових функцій та управлінні змінами. Ефективне проєктування передбачає вивчення та аналіз вимог до системи, розробку архітектурних концепцій та планування стратегій реалізації[18].

Варто зауважити, що немає однієї «правильної» архітектури, яка б підходила для всіх випадків. Кожен проєкт має свої унікальні вимоги, і ефективність архітектури завжди залежить від контексту.

Щоб спростити процес створення архітектури та вирішити типові проблеми, застосовуються патерни проєктування. Вони допомагають ідентифікувати та реалізувати основні компоненти та процеси застосунку, забезпечуючи при цьому оптимальний рівень ефективності та масштабованості.

Існує велика кількість підходів до проєктування архітектури веб-застосунків, які допомагають забезпечити їх ефективність, масштабованість та надійність. Розглянемо деякі з них, які можуть бути підходящими для розробки застосунку для керування проєктами.

Ось кілька з них:

- архітектура клієнт-сервер;
- MVC (Model-View-Controller);
- RESTful архітектура;
- мікросервісна архітектура.

Для вибору найоптимальнішої архітектури для застосунку для керування проєктами розглянемо ці патерни детальніше.

Перша на черзі клієнт-серверна архітектура[19]. Дана архітектура передбачає розділення системи на дві основні частини – клієнтську (браузер або мобільний додаток) і серверну (сервер надає доступ до даних та логіки застосунку). Це дасть змогу застосунку розділити відповідальності між клієнтом і сервером і забезпечити гнучкість та масштабованість системи. Ця архітектура часто використовується саме в веб-застосунках і є дуже поширеною загалом.

Клієнт-серверна архітектура – це модель взаємодії, де клієнт (наприклад, веб-браузер) звертається до сервера (наприклад, веб-сервера) для отримання ресурсів або послуг. Сервер надає відповідь на запити, надаючи необхідну інформацію або виконуючи операції (рисунок 2.1).

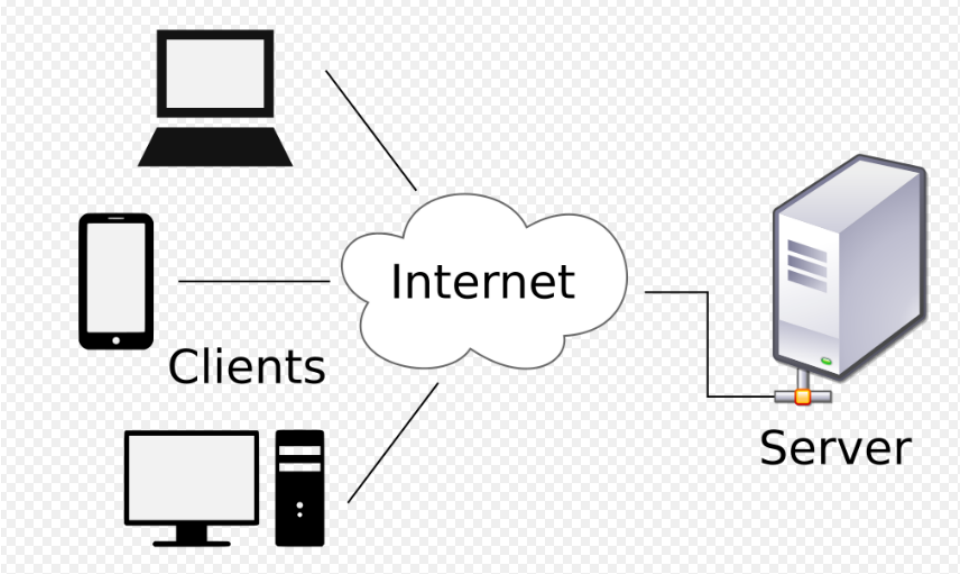


Рисунок 2.1 – Клієнт-серверна архітектура

Переваги клієнт-серверної архітектури полягають у зменшенні навантаження на клієнтські пристрої, централізації управління та полегшенні розширення системи через розділення логіки між клієнтом і сервером. Однак недоліками є потенційна залежність від доступності сервера, передача даних по мережі, що може призвести до затримок, а також потреба в постійному підтриманні інфраструктури сервера.

Далі розглянемо шаблон MVC[20] (Model-View-Controller). Цей шаблон дозволяє розділити архітектуру застосунку на три основні компоненти: модель (дані), вид (представлення) і контролер (логіка). Це полегшує розробку та підтримку коду, оскільки розділяє логіку застосунку на модулі.

Модель відповідає за логіку застосунку і роботу з даними та забезпечує доступ і виконує операції з ними, не залежно від того, як вони відображаються чи взаємодіють з користувачем. Представлення отримує дані від моделі та відображає їх у відповідному форматі. Контролер обробляє вхідні запити від користувача, виконує необхідні дії (наприклад, викликає методи моделі) і обирає відповідне представлення для відображення результатів, спілкуючись з моделлю та представленням (рисунок 2.2).

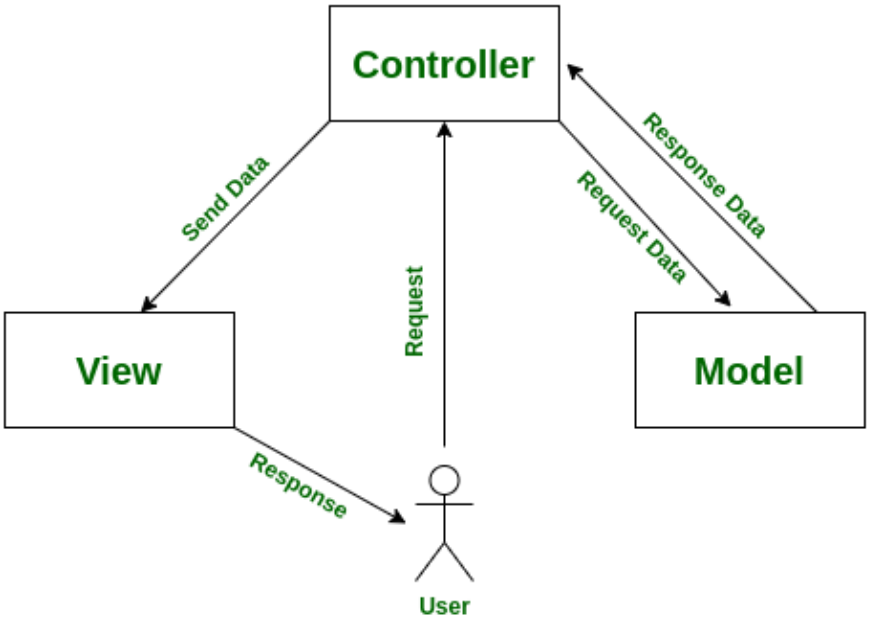


Рисунок 2.2 – Архітектура MVC

Переваги підходу MVC включають чітке розділення логіки застосунку, спрощення підтримки і розширення коду, а також полегшення роботи в команді. Недоліки можуть включати збільшення складності структури додатку та можливість виникнення надмірної кількості шарів абстракції, що ускладнює розуміння застосунку.

RESTful (Representational State Transfer) архітектура[21] – це архітектурний стиль програмного забезпечення для веб-сервісів, що базується на принципах протоколу HTTP. Основні принципи REST включають в себе використання стандартних методів HTTP (GET, POST, PUT, DELETE) для взаємодії з ресурсами, розділення клієнт-серверної архітектури, а також використання безстатевої комунікації між клієнтом і сервером. RESTful архітектура спрощує розробку та підтримку веб-застосунків, полегшуючи їх масштабування та інтеграцію з іншими системами.

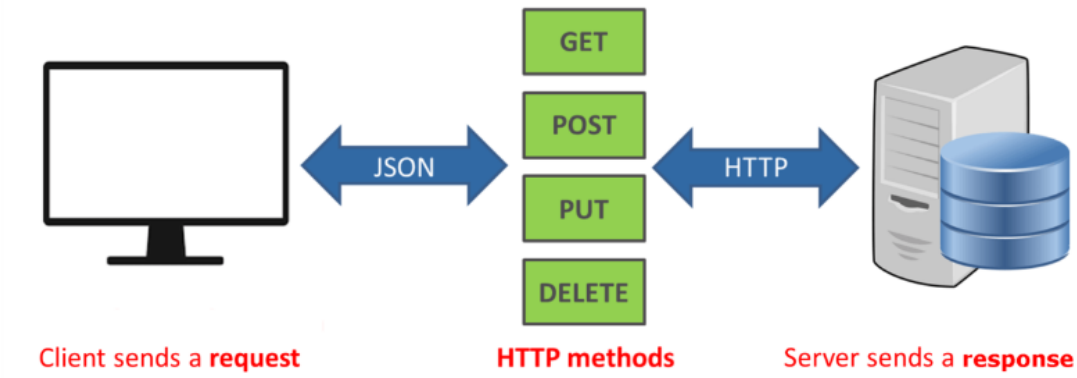


Рисунок 2.3 – RESTful архітектура

Переваги RESTful архітектури включають простоту, чітку структуру та легкість розуміння, використання стандартних протоколів із відкритою архітектурою, можливість масштабування та підтримки кешування, а також незалежність від мови програмування. Недоліки можуть включати складність при реалізації операцій, що вимагають зберігання стану, а також обмеження з відображенням навігаційної структури застосунку.

Що стосується мікросервісної архітектури[22] – це підхід до розробки програмного забезпечення, в якому додаток складається з невеликих, незалежних компонентів, які називаються мікросервісами. Кожен мікросервіс відповідає за виконання певної обмеженої функціональності і взаємодіє з іншими мікросервісами через API.

Незалежні сервіси можуть функціонувати і розвиватися окремо. Кожен сервіс відповідає за конкретну функціональність, що сприяє гнучкості та масштабованості системи (рисунок 2.4).

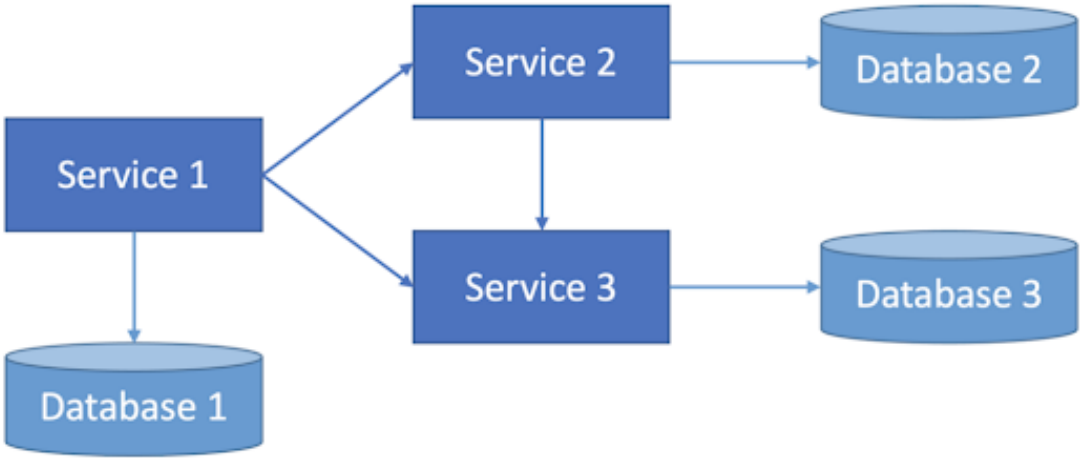


Рисунок 2.4 – Мікросервісна архітектура

Переваги мікросервісної архітектури включають високу гнучкість, масштабованість та можливість незалежного розвитку та розгортання кожного мікросервісу. Однак недоліками можуть бути складність управління багатьма мікросервісами, збільшення складності тестування та моніторингу, а також можливість виникнення проблем з консистентністю даних в такому середовищі.

Ці підходи можна комбінувати, оскільки, кожен з них має свої переваги та недоліки, тому важливо обрати той, який найбільш підходить для проєкту.

Для цього визначимо основні характеристики застосунку, які мають значну вагу при виборі архітектурного шаблону:

– система повинна мати можливість легко змінюватися та адаптуватися до змін в вимогах та потребах користувачів;

– система повинна бути здатна масштабуватися в залежності від обсягу та складності проєктів, які вона обробляє;

– система має забезпечити конфіденційність, для захисту інформації цілісність та доступність даних.

Для більш наочного аналізу архітектурних шаблонів наведено їх узагальнений огляд (таблиця 2.1), який включає в себе короткий опис, переваги, недоліки і декілька відомих прикладів.

Таблиця 2.1 – Порівняння архітектурних шаблонів

Критерій	Клієнт-Сервер	MVC	RESTful	Мікросервісна
Опис	Розподіляє логіку між клієнтом (інтерфейс) і сервером (зберігання даних, обробка)	Розділяє логіку на модель (дані), представлення (інтерфейс) і контролер (зв'язок)	Набір ресурсів, доступних через HTTP-методи	Розбиває систему на незалежні служби
Переваги	Проста, масштабована, безпечна	Гнучка, модульна, легко тестується	Легко кешувати, гнучка, масштабована	Швидка розробка, гнучка, масштабована
Недоліки	Складність реалізації складних систем	Не підходить для динамічних інтерфейсів	Складність забезпечення безпеки	Складність координації служб
Приклади	Google Chrome, Dropbox	WordPress, Django	Amazon Product Advertising API, Spotify API	Netflix, Amazon

Отже на основі вищеописаних даних було проведено аналіз цих архітектурних шаблонів, проаналізовано їх недоліки та переваги. І на основі всієї отриманої інформації та спостережень було прийнято рішення використовувати клієнт-серверну архітектуру з елементами RESTful, так як в даному випадку це є найоптимальнішим рішенням для розробки швидкого та зручного застосунку.

Розробка застосунку для керування проєктами на основі клієнт-серверної архітектури з елементами RESTful передбачає чітке розділення відповідальності між клієнтом і сервером. Клієнтська частина відповідає за інтерфейс користувача та відображення даних, тоді як серверна частина відповідає за бізнес-логіку, зберігання та обробку даних. Цей підхід забезпечує оптимальний обмін даними, забезпечення безпеки та масштабованість системи, а також зручність у використанні для розробників.

2.2 Проєктування логічної моделі бази даних

Реалізація системи керування проєктами передбачає не лише розробку архітектури, але й глибоке проєктування структури бази даних, яка забезпечить зберігання та обробку необхідної інформації. Визначення даних є першим етапом у цьому процесі. Структуру треба продумати, враховуючи всі аспекти функціонування системи, щоб забезпечити її ефективність та надійність[22].

Для цього ідентифіковано та систематизовано дані, що включають в себе інформацію про проєкти, завдання та користувачів. Для кожної з цих сутностей потрібні відповідні таблиці в базі даних, які взаємодіятимуть між собою за допомогою зв'язків. Це дозволить забезпечити належний рівень взаємодії та зручний доступ до необхідної інформації.

Крім того, для кожної сутності потрібно визначити характеристики, які будуть зберігатися в базі даних. Це включає в себе назви проектів, описи завдань, імена користувачів і так далі. Ці дані відображають основні вимоги до системи керування проєктами та дозволяють забезпечити необхідний функціонал для її коректної роботи.

Отже на основі аналізу база даних буде включати в себе наступні таблиці:

- Users;
- Employees;
- Attendance;
- Team_Members;
- Project_Team;
- Position;
- Project_Team;
- Projects;
- Project_Partition;
- Project_Tasks;
- Project_Progress.

Розглянемо ці таблиці детальніше. Таблиця Users потрібна для збереження облікових даних користувачів після реєстрації. У застосунку для керування проєктами, таблиця Users відіграє ключову роль у зберіганні та управлінні інформацією про користувачів. Вона містить дані про всіх осіб, які мають доступ до системи та беруть участь у роботі над проєктами.

Таблиця включає в себе наступні поля:

– User_Id (Id користувача в системі);
– Employee_Id (Id користувача як співробітника);
– User_Type (тип користувача);
– Username (ім’я користувача);
– Password (пароль користувача).

Ця таблиця дозволяє зберігати інформацію про користувачів, їхні облікові дані та ролі в системі, що є важливим елементом управління доступом та контролю прав доступу до функціоналу застосунку для керування проєктами.

Також ця таблиця пов’язана з наступною – Employee. В базі даних застосунку керування проектами це одна з ключових складових, яка дозволяє зберігати та керувати інформацією про співробітників організації.

Основні поля таблиці Employee:

– Employee_Id (Id співробітника);
– First_Name (ім’я);
– Last_Name (прізвище);
– Birthday (дата народження);
– Position_Id (Id посади);
– Status (статус співробітника);
– Date_Added (дата додавання);
– Employee_Picture (зображення співробітника).

Дана таблиця допомагає в ефективному управлінні персоналом, розподілі завдань та спільному виконанні проектів шляхом забезпечення доступу до важливої інформації про працівників.

Таблиця Employee в свою чергу пов’язана з Attendance. Ця таблиця призначена для збереження даних про відвідування застосунку кожним з працівників. Це потрібно для того, щоб контролювати робочий час користувача і використовується в звітах, розрахунку оплати і так далі. Тут є наступні поля:

– Attendance_Id (Id відвідування);
– Employee_Id (Id співробітника);
– Time_In (час входу в систему);
– Time_Out (час виходу з системи);
– Date_Today(дата).

Також таблиця Employee пов’язана з наступною таблицею Team_Member. Вона включає такі поля:

– TeamMember_Id (Id члена команди);
– Team_Id (Id команди);
– Employee_Id (Id) співробітника.

Загалом ця таблиця виконує більш службову роль для прив’язки користувача до проєктної команди і відповідно до проєкту.

Остання таблиця пов’язана з Employee – Position. Таблиця містить дані про посади, які присвоюються кожному з співробітників. Кожна посада має свій тип та рейт, який використовується для обчислення оплати. Структура таблиці:

– Position_Id (Id посади);
– Position_Name (назва посади);
– Position_Type (тип посади);
– Daily_Rate (рейт посади).

Наступна таблиця це Project_Team. Відповідно до своєї назви вона містить в собі інформацію про проекту команду, тобто про користувачів які згруповані в одну команду для конкретного проекту. Поля в цій таблиці:

– Team_Id (Id команди);
– Date_Added (дада додання);
– Employee_Id(Id співробітника).

Дуже важливою є таблиця Projects для організації та відстеження всіх проєктів. Вона дозволяє зберігати та отримувати доступ до ключової інформації про кожен проєкт. Основна мета цієї таблиці – забезпечити централізований доступ до даних про проекти для всіх зацікавлених сторін. Основні поля:

– Project_Id (Id проекту);

– Project_Name (назва проекту);

– Location (місце);

– Overral_Cost (загальна вартість);

– Start_Date (дата початку);

– Deadline (дедлайн);

– Team_Id (Id команди);

– Project_Pic (зображення проекту).

Ця таблиця є однією з основних та поєднана з багатьома іншими таблицями, тобто її можна вважати центральною в базі даних.

Вона має зв’язок з службовою таблицею Project_Partition і міститьв собі наступні поля:

– ProjectPartition_Id (Id учасника);

– Project_Id (Id проекту);

– Project_Tasks_Id (Id задачі).

Ця таблиця в свою чергу поєднана з таблицею Project_Tasks. Ця таблиця також досить невелика і містить в собі інформацію про задачі, які відносяться до кожного окремого проєкту:

– Project Task_Id (Id задачі);

– Task (назва задачі);

– Project_Type (тип проєкту).

Остання в переліку, але не менш важлива за значенням таблиця, що знаходиться в базі даних – Project_Progress. Її основна мета – забезпечити структуроване відстеження прогресу виконання кожного проєкту та його складових завдань. Вона дозволяє зберігати та оновлювати інформацію про різні

аспекти виконання проєктів, що сприяє успішному завершенню та керуванню проєктами. Ця таблиця містить такі поля:

– Progress_Id (Id прогресу);

– ProjectPartititon_Id (Id участі);

– Progress (прогрес);

– Date_Added (дата додавання);

– Partition_img (графічне відображення прогресу).

Отже, на основі вище описаної структури таблиць та їх полів побудовано логічну модель бази даних з необхідними зв'язками та структурою для ефективного зберігання та управління даними (рисунок 2.5).

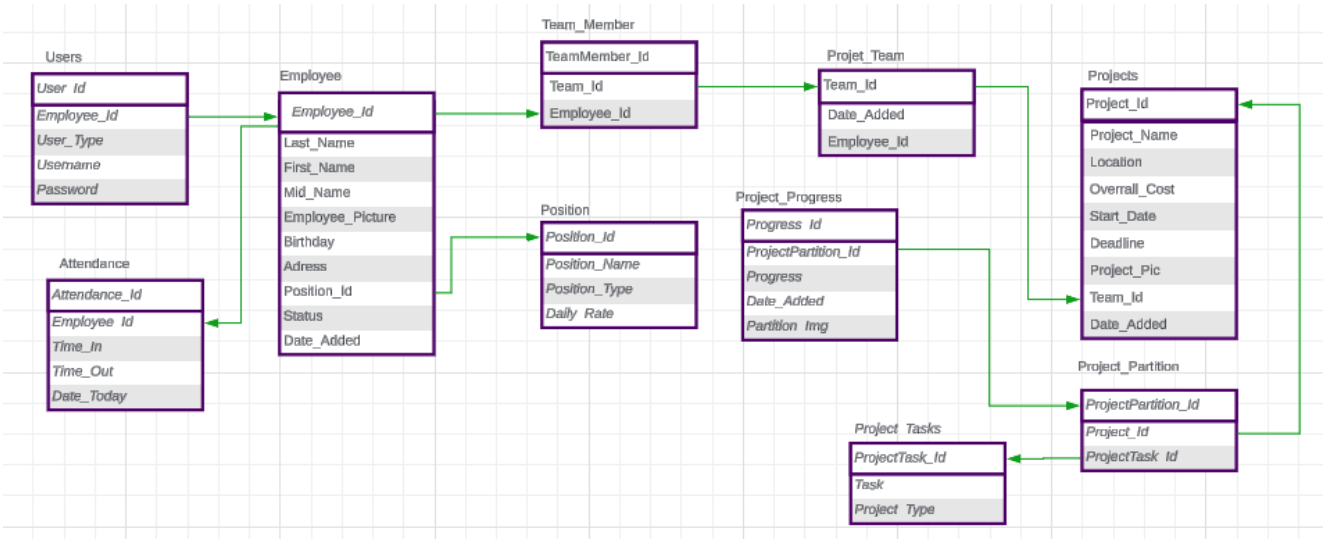


Рисунок 2.5 – Логічна модель БД

Отже підводячи висновок, в поточній базі даних кожна таблиця має власний унікальний ідентифікатор та набір полів, які відображають ключові атрибути кожного об'єкта. Задіяні зв'язки між таблицями дозволяють ефективно відображати взаємозв'язки між проєктами, завданнями та користувачами.

Ця логічна модель бази даних сприяє структурованому зберіганню та ефективному доступу до інформації, що є ключовим аспектом для успішного управління проєктами та досягнення їх цілей.

2.3 Проєктування інтерфейсу користувача

Для оформлення інтерфейсу застосунку для керування проєктами було обрано білий і чорний кольори з елементами синього. Це зроблено для того, щоб застосунок легко сприймався користувачем та зайвий дизайн не відволікав його від основної роботи. Отже застосунок повинен бути максимально легким та без зайвих деталей в дизайні інтерфейсу[24].

Враховуючи вищевказані деталі було спроектовано макет та дизайн інтерфейсу застосунку для керування проєктами в середовищі Figma[25]. Перший екран, який завжди зустрічає користувача, це вікно авторизації (рисунок 2.6).

Тут, для того щоб уникнути звичайного білого фону на весь екран, було прийнято рішення додати тематичне фонове зображення виконане в чорно-білих тонах, щоб не привертати забагато уваги, але при цьому зробити інтерфейс більш привабливим уникнувши стандартного білого екрану.

На екрані відображено стандартну форму авторизації, де користувачеві потрібно ввести свій логін та пароль щоб увійти в систему, та безпосередньо кнопка для входу. Також залежно від правильності введення даних буде залежати додаткове сповіщення внизу форми.

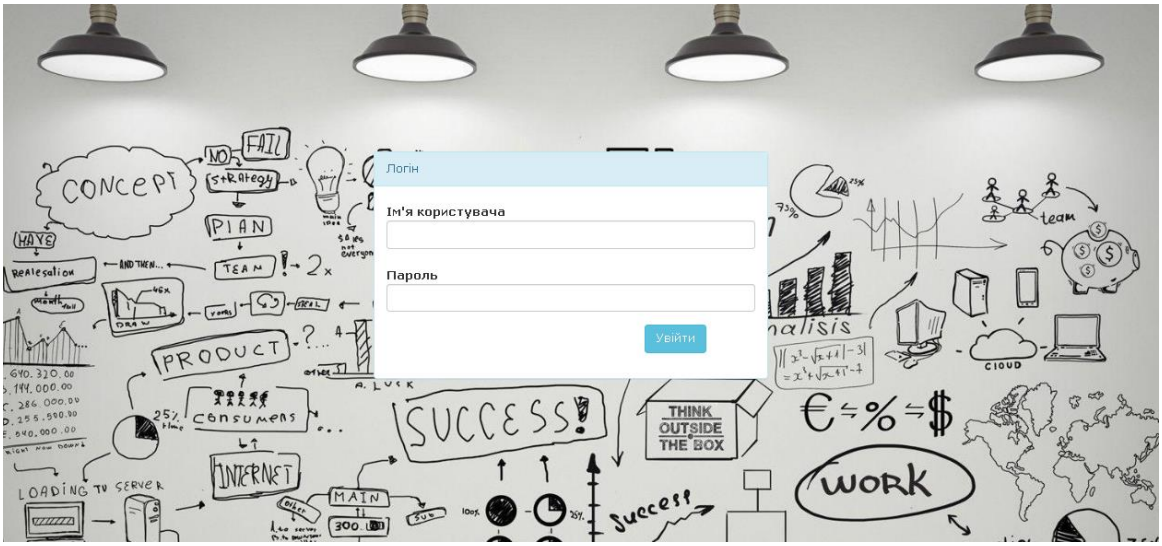


Рисунок 2.6 – Вікно авторизації

Після успішної авторизації користувач потрапляє на головну сторінку застосунку, а саме дешборд (рисунок 2.7). Тут знаходиться бокове меню з усім доступним функціоналом зліва. При виборі будь-якого з цих пунктів відбуватиметься редірект на відповідну сторінку. Тут вгорі вказана назва застосунку і вниз йде список функціоналу з відповідними іконками. Також тут є особливий пункт «Обслуговування» куди винесено менш важливий функціонал, щоб не навантажувати загальне меню великою кількістю пунктів. Доступ до особистого кабінету користувача знаходиться справа зверху. При натисненні на ім’я користувача відкриється меню де можна переглянути свою інформацію, змінити пароль та вийти з акаунту.

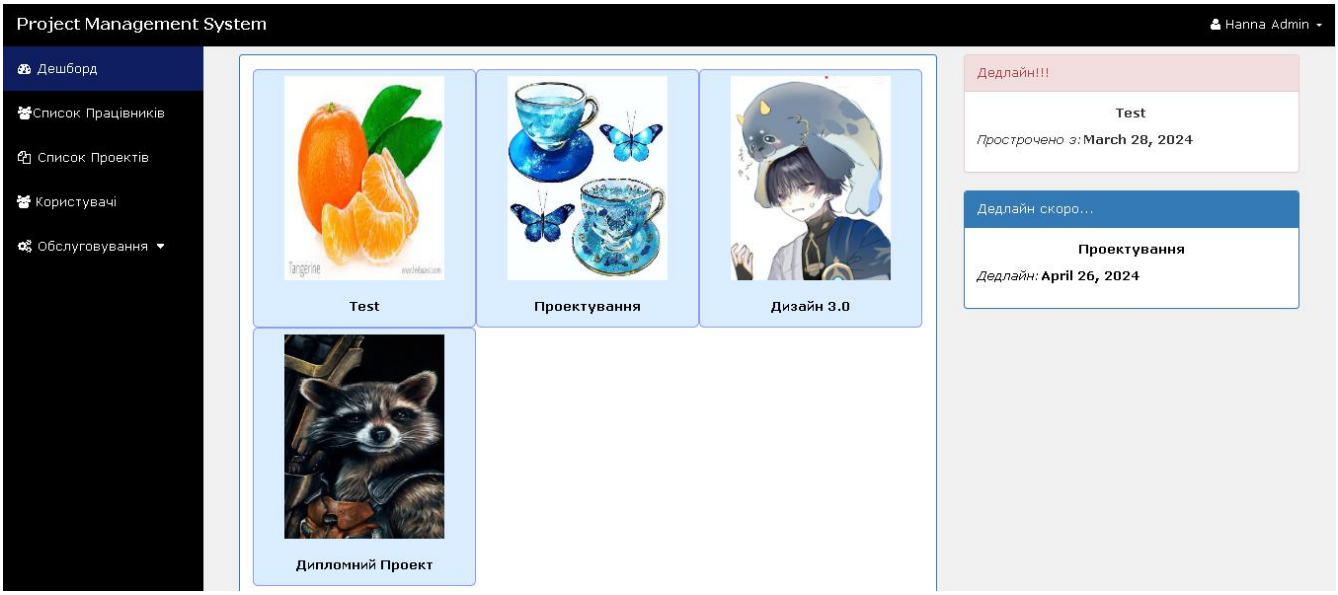


Рисунок 2.7 – Дешборд

Однак головне призначення дешборду – це відображення всіх поточних проєктів та повідомлень про дедлайни. Вони можуть бути двох видів: дедлайн який наближається та прострочений дедлайн. При виборі будь-якого проєкту на дешборді можна переглянути його детальну інформацію (рисунок 2.8).

Тут вказано всю інформацію про проєкт, таку як назва, дедлайн, вартість, керівник, задачі і так далі. Також тут можна налаштувати зображення проєкту, яке буде відображено в його картці та на дешборді.

Одна з найголовніших функцій цього розділу – діаграма прогресу. Тут можна побачити всі задачі які є в конкретному проєкті та відслідкувати прогрес їх виконання в відсотковому співвідношенні. Обравши кнопку редагувати можна робити оновлення по ходу виконання задач та додавати нові за потреби.

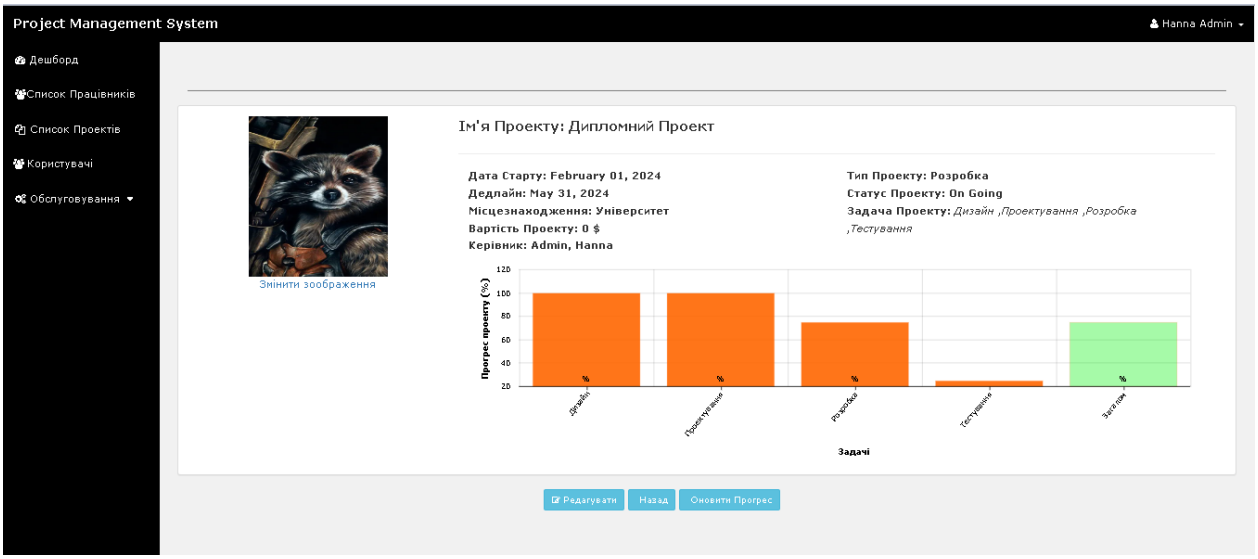


Рисунок 2.8 – Інформація про проєкт

Наступна сторінка призначена для виведення на екран списку працівників в системі (рисунок 2.9). Тут можна переглянути всіх працівників в табличному вигляді. Можна налаштувати відображення та переглядати окремі профілі.

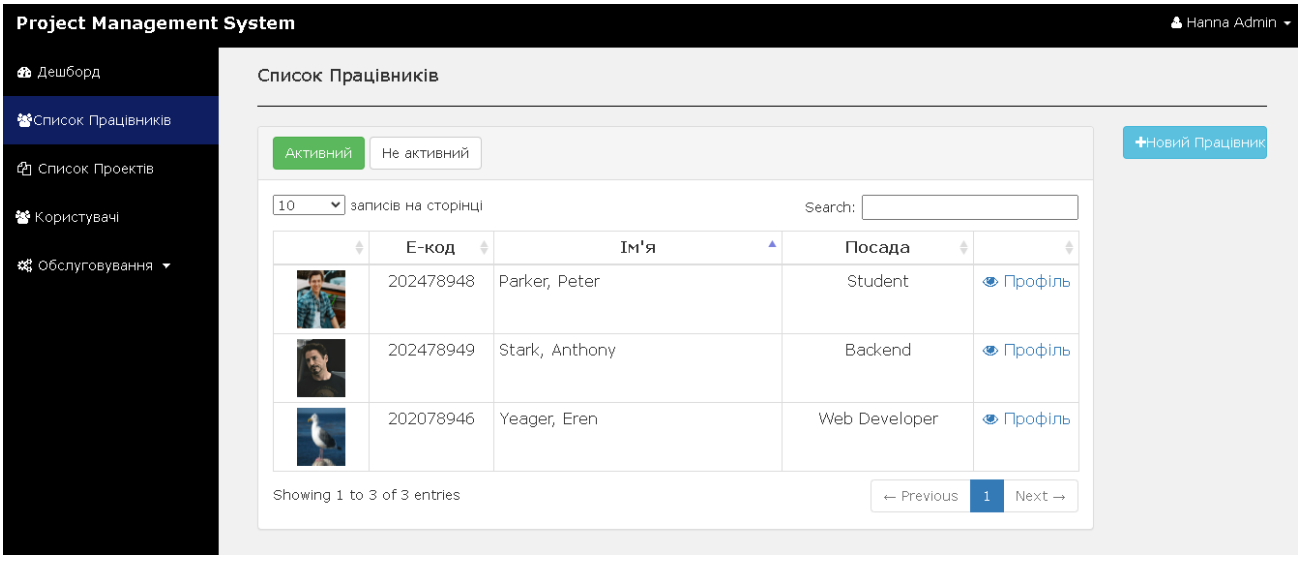


Рисунок 2.9 – Список працівників

При виборі кнопки «Профіль» відкривається сторінка обраного працівника (рисунок 2.10). Тут можна переглянути актуальну інформацію, змінити зображення профілю та редагувати дані.

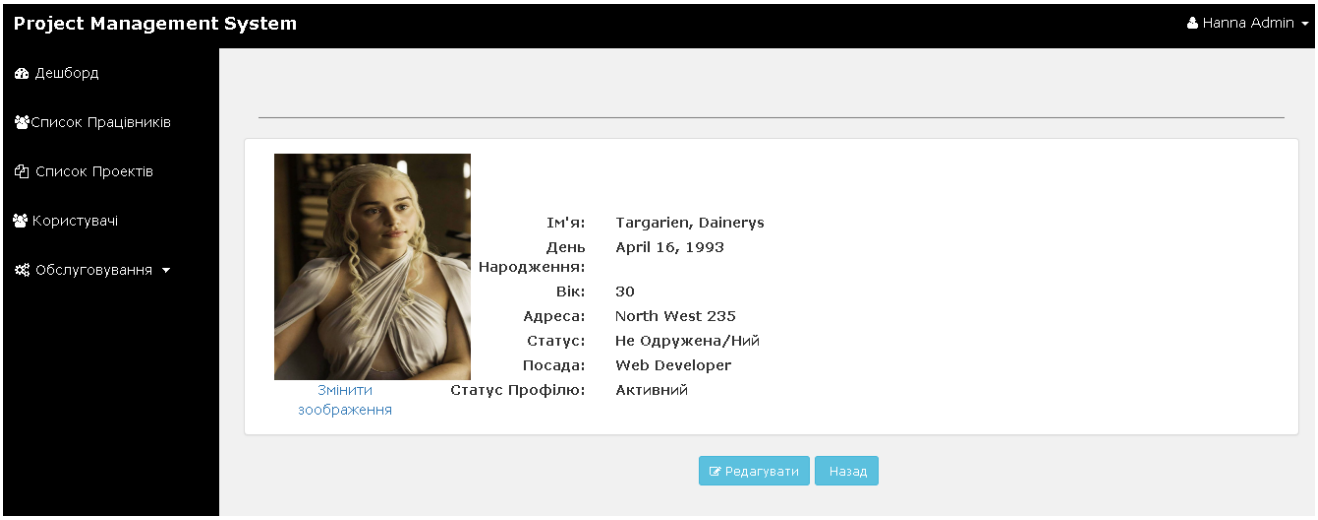


Рисунок 2.10 – Профіль працівника

Система дає можливість додати обраного працівника або учасника проекту, як користувача системи в вкладці «Користувачі» (рисунок 2.11). Для цього потрібно вибрати існуючого співробітника, надати йому ім’я користувача та пароль до системи. Також тут потрібно вказати права, які матиме цей користувач: звичайні чи адміністраторські.

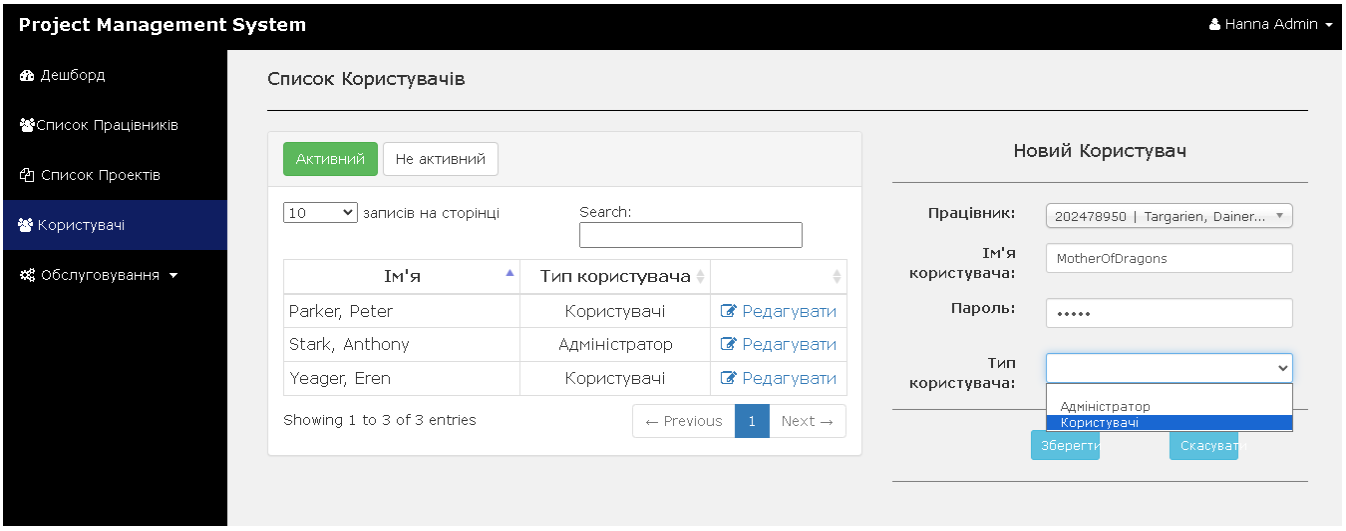


Рисунок 2.11 – Додавання користувача системи

Як можна помітити з попередніх зображень, весь застосунок має спільний стиль і сторінки відрізняються лише даними, які на них відображено. Тому продемонстровано одну з додаткових вкладок які знаходяться в розділі «Обслуговування». Тут показано вкладку «Проєктні команди» (рисунок 2.12).

На цій вкладці можна створити окрему команду для певного проєкту або іншип потреб. Функціонал надає можливість обрати керівника проєкту та учасників даної команди. За потреби можна оновити дані про команду, змінити керівника, додати чи видалити учасника і тому подібне.

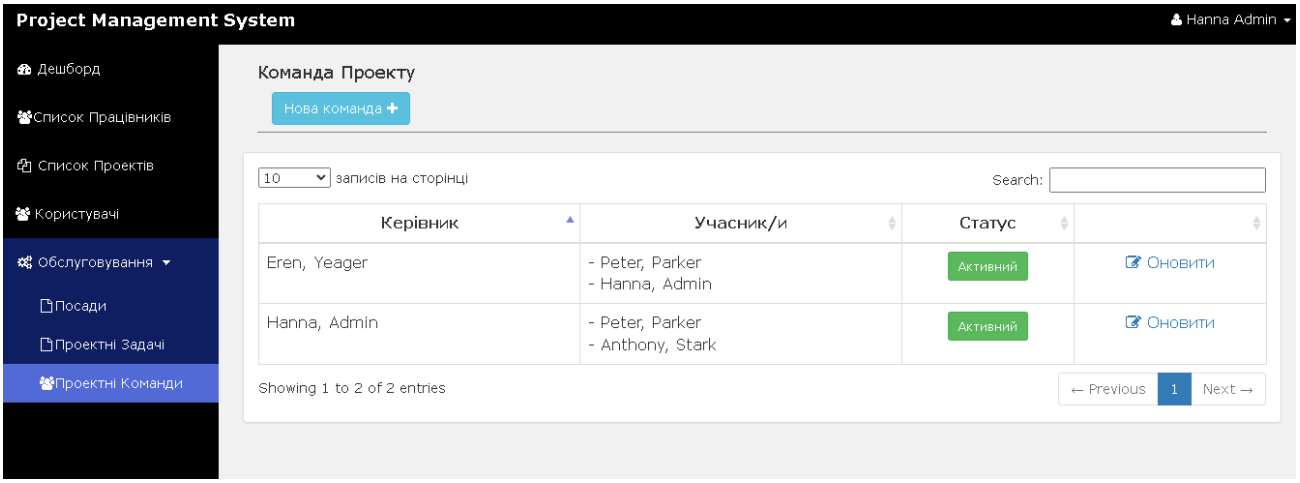


Рисунок 2.12 – Проєктні команди

Загалом, підводячи підсумки проєктування інтерфейсу користувача, застосунок для керування проєктами має свій стиль та дизайн, які адаптовано до кожної сторінки застосунку. Інтерфейс інтуїтивно зрозумілий та не перевантажений, як в деяких готових рішеннях, які було проаналізовано перед розробкою, дозволить користувачам легко зорієнтуватися та використовувати функціонал системи без зайвих зусиль.

Також застосунок забезпечить зручну навігацію по системі, включаючи меню, пошукові функції та шляхи переміщення між різними розділами та екранами. Кольорова палітра та тема застосунку приємна та функціональна, що допоможе покращити сприйняття інформації та забезпечити комфортне користування системою.

2.4 Аналіз та вибір технологій і методів реалізації

На основі попереднього аналізу, визначення вимог та проєктування системи було прийнято рішення, що для розробки такого програмного застосунку найкраще підійде мова програмування PHP.

PHP (Hypertext Preprocessor)[26] – це скриптова мова програмування, яка використовується для створення динамічних веб-сайтів та веб-додатків. Початково PHP означало «Personal Home Page» (особиста домашня сторінка), але згодом було змінено на «Hypertext Preprocessor» – це рекурсивний акронім.

Вона була розроблена Расмусом Лердорфом у 1994 році і швидко стала однією з найпопулярніших мов для розробки веб-додатків. PHP має простий синтаксис, що робить її доступною для новачків у програмуванні, та вона підтримується багатою спільнотою розробників. Вона використовується для створення різноманітних веб-сайтів, таких як блоги, інтернет-магазини, соціальні мережі, та багато інших.

Застосування PHP дозволяє створювати інтерактивні веб-сайти з великою кількістю функцій, таких як форми зворотного зв'язку, авторизація користувачів, робота з базами даних та інші. Вона також ідеально підходить для розробки великих веб-проєктів завдяки своїй гнучкості та розширюваності.

PHP інтегрується з багатьма системами управління базами даних, такими як MySQL, PostgreSQL, та SQLite, що дозволяє легко працювати з даними в веб-додатках. Крім того, PHP має велику кількість розширень та бібліотек, які полегшують розробку різноманітних функцій та функціоналу.

Використання мови програмування PHP в застосунку для керування проєктами має кілька переконливих причин[27]:

– широкі можливості: PHP популярною мовою програмування для веб-розробки з великою кількістю розширень та фреймворків. Це надає можливість легко і швидко розширювати та модифікувати програмний застосунок у майбутньому;

– інтеграція з базами даних: PHP має потужні можливості для роботи з різними системами управління базами даних, такими як MySQL, PostgreSQL, та інші. Це дозволяє зберігати та керувати великою кількістю даних, які потрібні для керування проєктами;

– простота вивчення та використання: PHP має простий та зрозумілий синтаксис. Велика кількість документації, навчальних матеріалів та спільноти розробників також полегшує процес розробки та підтримки проєкту;

– швидкодія: PHP працює на більшості веб-серверів та забезпечує високу швидкодію обробки веб-запитів. Це дозволяє застосунку працювати ефективно та швидко в умовах великого обсягу даних та потоку користувачів;

– гнучкість: PHP є гнучкою мовою програмування, що дозволяє створювати різноманітний функціонал для програмного застосунку. Це дозволить точно відповідати потребам користувачів та ефективно виконувати завдання керування проєктами.

Отже, використання PHP для розроблення застосунку для керування проєктами є раціональним вибором, оскільки вона надає потужність, гнучкість та ефективність, для створення високоякісного програмного застосунку.

Безпосередньо для розробки було обрано середовище PHPStorm. PHPStorm[28] – це інтегроване середовище розробки (IDE) від компанії JetBrains, спеціалізоване на роботі з мовами програмування PHP, HTML, CSS та JavaScript. Воно надає розширені можливості для зручного програмування та налагодження веб-додатків на базі PHP.

Основні функції та переваги PHPStorm включають[29]:

– підтримка PHP: PHPStorm має розширені можливості для роботи з PHP. Автодоповнення коду, рефакторинг, відладка, перегляд документації та інші інструменти, що полегшують роботу з цією мовою;

– підтримка інших мов: Крім PHP, PHPStorm підтримує роботу з HTML, CSS, JavaScript, SQL та іншими мовами програмування, що дозволяє працювати з повністю функціональним стеком технологій в одному інтерфейсі;

– аналіз коду: IDE включає в себе різні інструменти для аналізу коду, як виявлення помилок, відсутність коду та можливості оптимізації;

– підтримка фреймворків: IDE має підтримку популярних фреймворків PHP, таких як Laravel, Symfony, Yii, Zend Framework тощо, що полегшує розробку з їхнім використанням;

– інтеграція з іншими інструментами: PHPStorm інтегрується з різними іншими інструментами розробки, такими як системи контролю версій (наприклад, Git), збірка проєктів, віртуальні середовища, що дозволяє зручно працювати в різних екосистемах розробки.

У загальному, PHPStorm є потужним та зручним інструментом для розробки застосунків на PHP та пов'язаних технологіях та допомагає підвищити продуктивність і якість роботи над проєктом.

Для створення та тестування застосунку на локальному комп'ютері перед розгортанням на живому сервері вирішено використовувати XAMPP.

XAMPP[30] – це набір вільного програмного забезпечення, який містить в собі всі необхідні компоненти для розгортання локального веб-сервера. Він включає в себе такі складові, як веб-сервер Apache, базу даних MySQL, мову програмування PHP та інші корисні інструменти. Використання XAMPP дозволяє зосередитися на розробці програмного забезпечення, забезпечуючи при цьому швидке та зручне середовище для роботи.

Основні компоненти XAMPP[31]:

– Apache Server: Apache є одним з найпоширеніших веб-серверів у світі. Він забезпечує можливість обробки HTTP-запитів, що дозволяє розміщувати веб-сайти та додатки;

– MySQL database: MySQL – це потужна система керування базами даних, яка дозволяє зберігати та керувати даними у веб-застосунках.

– PHP: PHP – скриптова мовою програмування, яка використовується для створення динамічних веб-сайтів. У XAMPP PHP використовується для обробки запитів та генерації веб-сторінок на серверному боці.

Використання XAMPP – це, насамперед, можливість створення власного сервера прямо на комп'ютері. Це дозволяє працювати з програмним забезпеченням безпосередньо на локальній машині, що є дуже зручним для розробки та тестування. Також, використання XAMPP відкриває можливість використовувати весь спектр інструментів, які входять в його склад, для ефективної розробки застосунку.

Не менш важливою в застосунку для керування проєктами є база даних. В вищевказаному контексті обраних технологій та методів реалізації найоптимальнішим рішенням буде використовувати СКБД MySQL[32].

MySQL – це система керування базами даних (СКБД), яка використовується для зберігання та керування великими обсягами даних.

Основні причини вибору MySQL[33]:

– надійність та стабільність: MySQL відомий своєю надійністю та стабільністю, що робить його ідеальним вибором для зберігання важливих даних про проєкти та завдання;

– ефективність: MySQL має високу продуктивність та швидкодію обробки запитів, що дозволяє ефективно працювати з великим обсягом даних;

– безкоштовність та відкритість: MySQL є вільно розповсюджуваним програмним забезпеченням з відкритим вихідним кодом, що робить його доступним для використання безкоштовно та дозволяє модифікувати його під власні потреби.

Враховуючи ці переваги, використання MySQL у розроблюваному ПЗ дозволить ефективно зберігати та управляти даними про проєкти, завдання та користувачів, що є ключовим аспектом застосунку для керування проєктами.

Отже в результаті аналізу було вирішено використовувати PHP як мову програмування, PhpStorm як інтегроване середовище розробки, XAMPP для локального серверу та веб-сервера, а також MySQL як систему управління базами даних. Такий вибір забезпечить зручну та ефективну розробку веб-застосунків з використанням сучасних технологій та інструментів.

2.5 Висновки проєктування програмного забезпечення

В результаті проведення проєктування програмного забезпечення було розроблено детальну архітектуру системи, яка включає в себе визначення ключових компонентів, їх функціональний зв'язок та інтерфейси взаємодії між ними. Це дозволяє забезпечити ефективну організацію роботи програми та зручний доступ до її функцій.

Також була розроблена логічна модель бази даних, яка відображає структуру даних та їх взаємозв'язки. Це дозволяє забезпечити ефективне зберігання та обробку інформації, необхідної для роботи системи.

Під час проєктування інтерфейсу користувача був розроблений інтуїтивно зрозумілий та зручний інтерфейс, який дозволяє легко взаємодіяти з застосунком та виконувати необхідні операції. Це сприяє покращенню користувацького досвіду та збільшенню продуктивності роботи з системою.

Був проведений аналіз різних технологій та методів реалізації, після чого було зроблено вибір оптимальних інструментів та підходів для розробки застосунку для керування проєктами. Це дозволить забезпечити найвищу якість та ефективність роботи системи.

Таким чином, завдяки проведенню проєктування програмного забезпечення було розроблено чіткий план дій та визначено необхідні кроки для подальшої реалізації системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Програмна реалізація модулів

У розробці програмного застосунку для керування проєктами важливим етапом є реалізація функціональності модулів. Цей етап передбачає створення програмного коду, який забезпечить необхідні функції та можливості для ефективного управління проєктами. Організація проєкту по папках допомагає зробити його структурованим та зрозумілим. Кожна папка має своє призначення, що полегшує пошук та роботу з файлами[34].

Отже проєкт має наступну структуру:

– CSS: у цій папці знаходяться файли стилів для веб-сторінок. Це файли CSS, які відповідають за оформлення різних компонентів або сторінок;

– Database: тут містяться файли, що пов'язані з базою даних. Такі як резервні копії баз даних, файли схеми або файли конфігурації;

– Fonts: у цій папці розташовуються файли шрифтів, які використовуються в застосунку;

– Forms: ця папка призначена для зберігання файлів, пов'язаних із формами. Тут знаходяться файли HTML та PHP, які обробляють введені дані;

– Images: це зображення, які використовуються в застосунку;

– JS: у цій папці знаходяться файли JavaScript;

– Pages: ця папка використовується для зберігання файлів сторінок застосунку, а саме PHP-файли, які відображають сторінки користувача. Це головна папка для застосунку в якій міститься вся основна логіка та розроблений програмний код[35].

Розглянемо основні моменти в розробці функціоналу для застосунку керування проєктами на прикладі фрагментів коду. Повний лістинг застосунку наведено в Додатку Б. Перша сторінка, на якій опиняється користувач – це класична index.php. Тут, в першу чергу, буде запущено код для перевірки авторизації та запуску основної сесії, якщо авторизація пройшла успішно:

```
<?php session_start();
      if (isset($_SESSION['UID']))
      { header('location:pages/index.php');
      exit; }
      include_once 'includes/header2.php';
?>
```

Цей PHP-код починає сесію і перевіряє наявність змінної сесії UID, яка вказує на ідентифікатор користувача; якщо вона існує (тобто користувач автентифікований), то він перенаправляється на головну сторінку index.php, в іншому випадку включається файл header2.php, який містить заголовок або частину коду для сторінки з сповіщенням про помилку.

При вході в систему потрібно направити користувача на одну з сторінок застосунку, де він зможе працювати:

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
$page = isset($_GET['page']) && !empty($_GET['page']) ? $_GET['page'] : 'home';
$pages =
array('home','position','employee','employee_profile','division','project_list',
'project_detail','progress','update_progress','user_list','attendance','project_
team');
if (!empty($page)) {
    if(in_array($page,$pages)) {
        $page .= '.php';
        include($page);
    }
    else {
        echo 'Сторінка не знайдена
        <a href="index.php?page=home">На головну</a>';
    }
}
?>
```

Цей код встановлює поточну сторінку на основі параметра запиту «page», перевіряє, чи така сторінка існує в масиві доступних сторінок, і включає відповідний файл сторінки. Якщо сторінка не знайдена, виводиться повідомлення про помилку з посиланням на головну сторінку.

Далі потрібно виконати наступний запит щоб отримати дані про авторизованого користувача:

```
<?php
    include '../includes/db.php';
      $query2=  mysqli_query($conn, "SELECT *,CONCAT(lastname,', ',firstname,'
',midname) as name,users.io as status FROM users natural join employee where uid
= '".$_SESSION['UID']."' ");
       $row2 = mysqli_fetch_assoc($query2);
    ?>
```

Цей код виконує запит до бази даних для отримання інформації про користувача, який увійшов в систему, використовуючи його ідентифікатор користувача (UID). Результат запиту включає дані користувача з профілю та статус активності, який зберігається у стовпці «io».

Далі користувачу потрібно бачити список доступних проєктів, адже це одна з головних задач даної системи

```
<?php
  $pp = mysqli_query($conn,"SELECT * FROM projects order by project ASC ");
    while($pp_row=mysqli_fetch_assoc($pp)){
?>
   <option value="<?php echo $pp_row['project_id'] ?>"><?php echo
ucwords($pp_row['project']) ?></option>
<?php } ?>
```

Код виконує запит до бази даних для отримання списку проєктів. Після цього він створює випадаючий список «select» з ідентифікаторами проєктів у якості значень «option» та назвами проєктів у якості відображених елементів.

На сторінці «Співробітники» потрібно вивести список всіх користувачів та їх зв’язки з загальною структурою проєкту :

```
<?php
   include '../includes/db.php';
   $query=  mysqli_query($conn, "SELECT *,CONCAT(lastname,', ',firstname,'
',midname) as name FROM employee natural join position where io =
'".$_GET['io']."' and eid != 1 order by name ");
    while($row = mysqli_fetch_assoc($query)) {
    $id =$row['eid'];
    $eco= date("Y",strtotime($row['date_added'])).$row['ecode'];
 ?>
```

Цей код виконує запит до бази даних, вибирає працівників з певним статусом та об'єднує їх з їхніми посадами, використовуючи результат унікального ідентифікатора для кожного працівника на основі часу додавання та номера співробітника.

Не менш важливим функціоналом є додавання нових даних в систему. Це може бути додавання нового користувача, проекту, завдання і так далі. Саме на цьому заснована досить велика частина функціоналу застосунку. Розглянемо його на прикладі додавання нового користувача в систему. Для цього написано наступний код:

```
<?php
$action = $_GET['action'];

if($action == 'user'){

	$eid = $_POST['eid'];
	$user = $_POST['user'];
	$pass = $_POST['pass'];
	$u_type = $_POST['u_type'];


	if($query = mysqli_query($conn,"INSERT INTO users
(eid,username,password,user_type,io)VALUES('$eid','$user','$pass','$u_type','1')
")){
		echo '<script>$("#msg").show("SlideDown");</script>';
	}else{
		echo "<script>alert('Помилка збереження даних!')</script>";
	}
}
```

Цей PHP-код перевіряє параметр action у запиті GET і, якщо він дорівнює «user», отримує дані з POST-запиту, такі як ідентифікатор працівника, ім'я користувача, пароль та тип користувача. Після чого він виконує запит INSERT до бази даних для збереження цих даних, а якщо операція вдалася, виводить повідомлення про успішне збереження. Якщо виникає помилка, виводиться відповідне повідомлення.

В такому ж контексті додаються і складніші сутності, наприклад такі як прогрес проєкту:

```
if($action =='progress'){
	foreach ($_POST as $var => $value)
		$$var = $value;
	$rd2 = mt_rand(1000, 9999);
			$filename = basename($_FILES['image']['name']);
			$ext = substr($filename, strrpos($filename, '.') + 1);
			$file = $rd2. "_" .$filename;

	(move_uploaded_file($_FILES['image']['tmp_name'],'../images/'.$file));
}
	$query= mysqli_query($conn,"INSERT INTO project_progress
(pp_id,progress,date_added,partition_img)values('$div','$prog',now(),'$file')
");
	if($query){
		echo "<script>location.replace(document.referrer);</script>";
	}
```

Тут код перевіряє значення параметра $action. Якщо воно дорівнює «progress», то відбувається наступне: для кожного елементу POST-запиту створюється змінна з іменем, що відповідає ключу, і значенням, переданим у

POST-запиті. Потім генерується випадковий номер для ідентифікатора файлу, отримується ім'я файлу, його розширення та нове ім'я файлу, що складається з випадкового номера та оригінального імені файлу. Файл завантажується вказану директорію. Після цього виконується запит INSERT до бази даних для збереження інформації про прогрес проекту, включаючи шлях до зображення. Якщо запит успішний, відбувається перенаправлення на попередню сторінку.

Додавання нового прогресу має досить багато нюансів і потребує необхідних блоків коду в декількох представленнях. Отже тут потрібен код для відстеження всіх підзадач проєкту[36]:

```
<?php
          $div= mysqli_query($conn,"SELECT * FROM project_partition natural join
project_division where project_id = '$id' and pp_id != '".$row2['pp_id']."'
order by division ");
          while($div_row = mysqli_fetch_assoc($div)){
          $test = mysqli_query($conn,"SELECT sum(progress) as prog FROM
project_progress where pp_id = '".$div_row['pp_id']."' ");
          $test_row= mysqli_fetch_assoc($test);
          if(mysqli_num_rows($test) > 0){
            $prog = $test_row['prog'];
          }else{
            $prog = 0;
          }
          if($prog < 100 ){ ?>
            <option value="<?php echo $div_row['pp_id'] ?>"><?php echo
ucwords($div_row['division']) ?></option>
<?php }} ?>
```

Цей код виконує запит до бази даних, щоб отримати всі підрозділи проекту, які не мають повного прогресу. Для кожного підрозділу обчислюється загальний прогрес, використовуючи запит до таблиці project_progress. Якщо загальний прогрес менше 100%, то підрозділ додається, як варіант в випадаючому списку.

Також окремо потрібно додати код, який перевіряє загальний прогрес певного підрозділу проєкту, обчисленого як сума прогресу всіх завдань в цьому підрозділі. Якщо загальний прогрес менше або дорівнює 100%, виводиться повідомлення про те, скільки ще потрібно прогресувати для досягнення 100%. Якщо прогрес перевищує 100%, виводиться повідомлення про перевищення та приховується кнопка збереження:

```
<?php
include '../includes/db.php';
foreach ($_GET as $var => $value)
      $$var = $value;
 $query = mysqli_query($conn,"SELECT sum(progress) as prog FROM project_progress
where pp_id = '$id'");
 $row = mysqli_fetch_assoc($query);
 if(mysqli_num_rows($query) > 0 ){
      $total = $row['prog'];
 }else{
      $total = 0;
 }
 if($prog <= 100){
 $left = 100 - $total;
      if($left < $prog ){
      echo "<div class='alert alert-danger'>".$left."% прогресу залишилося, щоб
вважати цю сферу 100%.</div>";
      echo "<script>$('#btn_save').hide();</script>";
 }}else{
      echo "<div class='alert alert-danger'>значення перевищує 100%.</div>";
      echo "<script>$('#btn_save').hide();</script>";
 }?>
```

Також варто окремо відзначити функцію додавання зображення :

```
if($action == 'change_pic'){
      $id = $_GET['id'];
            $rd2 = mt_rand(1000, 9999);
            $filename = basename($_FILES['file']['name']);
            $ext = substr($filename, strrpos($filename, '.') + 1);
            $file = $rd2. "_" .$filename;
      (move_uploaded_file($_FILES['file']['tmp_name'],'../images/'.$file));
      $query = mysqli_query($conn,"UPDATE employee set e_pic = '$file' where eid
= '$id'");
      if($query){
      echo '<script> location.replace(document.referrer);</script>';
      }
```

Тут перевіряється значення параметра $action. Якщо воно дорівнює «change_pic», то відбувається наступне: отримується ідентифікатор користувача ($id) з параметру GET, генерується випадковий номер для ідентифікатора файлу, отримується ім'я файлу, його розширення та нове ім'я файлу, що складається з випадкового номера та оригінального імені файлу. Файл завантажується у вказану директорію. Після цього виконується запит UPDATE до бази даних для оновлення імені зображення користувача. Якщо запит виконується успішно, відбувається перенаправлення на попередню сторінку.

Отже у підсумку, тут наведено основний функціонал, який міститься в застосунку. Решта програмного коду має схожу структуру та принципи роботи, тому тут буде не доцільно описувати абсолютно всі методи та функції.

3.2 Розробка бази даних

Створення бази даних для застосунку для керування проєктами відбувається на сервері XAMMP з використанням MySQL запитів. Беручи за основу спроектовану модель бази даних з розділу 2.2 необхідно створити всі таблиці для збереження даних застосунку. Демонстрація створення таблиць бази даних на прикладі таблиці «Проєкти». Таким чином створено всі таблиці, потрібні для коректної роботи застосунку:

```
CREATE TABLE `projects` (
  `project_id` int(10) NOT NULL,
  `project` varchar(100) NOT NULL,
  `location` varchar(100) NOT NULL,
  `overall_cost` varchar(10) NOT NULL,
  `start_date` date NOT NULL,
  `deadline` date NOT NULL,
  `site_pic` varchar(100) NOT NULL,
  `tid` int(10) NOT NULL,
  `proposed_project` int(5) NOT NULL,
  `date_added` date NOT NULL,
  `io` int(5) NOT NULL
) ENGINE=InnoDB;
```

Ця команда SQL створює таблицю з назвою projects, що містить інформацію про різні аспекти проектів, такі як: назва, місце розташування, вартість, дати початку та завершення, зображення сайту, теги проекту та дату додавання, з використанням InnoDB для двигуна таблиці.

Наступним кроком буде створення первинних ключів для кожної з таблиць в базі даних. Для цього використовується код:

```
ALTER TABLE `employee`
  ADD PRIMARY KEY (`eid`);

ALTER TABLE `position`
  ADD PRIMARY KEY (`pid`);

ALTER TABLE `projects`
  ADD PRIMARY KEY (`project_id`);
```

Ці SQL-команди змінюють структуру бази даних, додаючи первинний ключ для таблиць employee, position та projects, який буде ідентифікувати унікальний запис у кожній таблиці.

Також при створенні бази даних необхідно передбачити авто інкремент, який дозволяє автоматично призначати унікальні ідентифікатори при додаванні нових записів в таблиці:

```
ALTER TABLE `project_team`
  MODIFY `tid` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

ALTER TABLE `team_member`
  MODIFY `tm_id` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

ALTER TABLE `users`
  MODIFY `uid` int(10) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=5;
COMMIT;
```

Ці SQL-команди змінюють структуру бази даних, змінюючи налаштування для автоматичного збільшення значення поля tid в таблиці project_team, tm_id в таблиці team_member та uid в таблиці users, що дозволяє автоматично призначати унікальні ідентифікатори при додаванні нових записів. Це необхідно прописати для всіх таблиць в базі даних так само як і первинний ключ, для коректної роботи застосунку.

Отже, виконавши вищеописані дії для всіх таблиць в базі даних, а саме створивши таблиці, надати їм первинні ключі та призначати унікальні ідентифікатори при додаванні нових записів, можна побачити загальну структуру готової бази даних (рисунок 3.1).

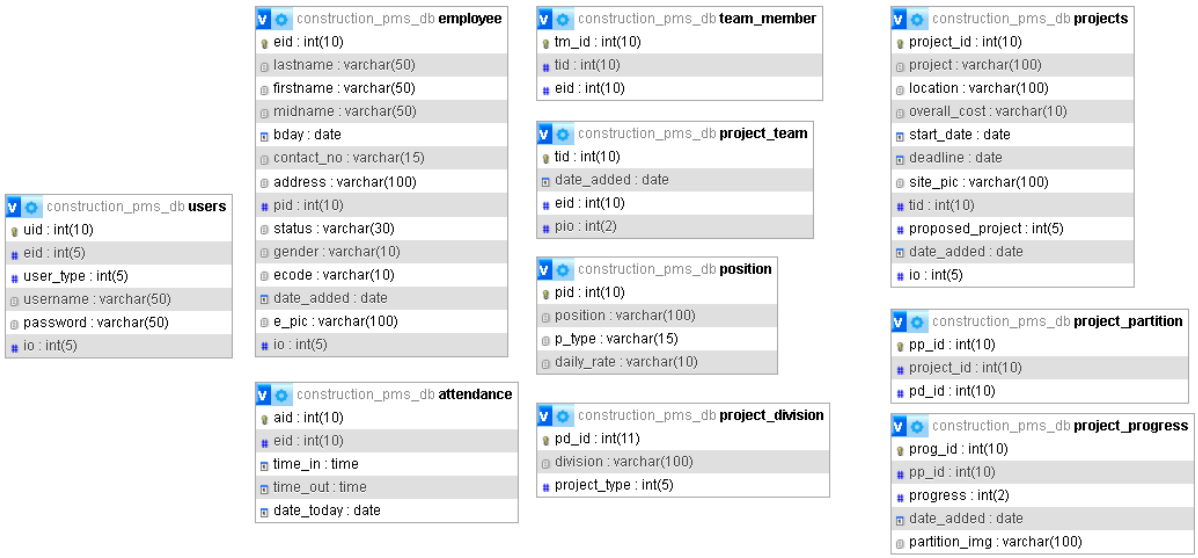


Рисунок 3.1 – Структура Бази Даних

Тепер, коли структура бази даних готова до використання, необхідно додати першого користувача з адміністраторськими правами для доступу в систему. На даному етапі це буде зроблено в ручну. Додамо декілька користувачів з різними правами для початку роботи:

```
INSERT INTO `users` (`uid`, `eid`, `user_type`, `username`, `password`, `io`)
VALUES
(1, 1, 1, 'Admin', 'admin', 1),
(3, 2, 2, 'Eren', '123', 1),
(4, 3, 2, 'Spider', '12345', 1);
```

Ця SQL-команда додає нові записи до таблиці users, вказуючи значення для кожного поля. Кожен рядок представляє одного користувача з його унікальним ідентифікатором (uid), ідентифікатором працівника (eid), типом користувача (user_type), ім'ям користувача (username), його паролем (password) і ознакою (io).

Останнім, але не менш важливим кроком, буде підключення бази даних безпосередньо до застосунку, для цього використано наступний код:

```
<?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 код встановлює з'єднання з базою даних MySQL. Він використовує змінні для збереження даних про сервер, ім'я користувача, пароль та назву бази даних. Після цього він викликає функцію mysqli_connect, щоб підключитися до сервера бази даних. Якщо підключення не вдається, скрипт завершиться, виведенням повідомлення про помилку.

Отже підсумовуюючи, в такому вигляді базу даних можна спокійно використовувати в застосунку та робити різноманітні запити щоб отримати весь необхідний функціонал для керування проєктами.

3.3 Керівництво користувача

Застосунок для керування проєктами – закрита система, що має потенціал задовольнити потреби як окремого користувача, так і команди, або навіть цілої компанії. Його основна мета – надати повну структуру та ієрархію для ефективного управління проєктами.

Користування застосунком починається з первинного адміністратора, який отримує доступ до системи та власноруч створює і додає решту користувачів. Таким чином будь-яка особо не зможе отримати доступ до даних тої чи іншої компанії.

Коли користувач отримає свій логін та пароль до застосунку, він вводить його в форму авторизації, та, якщо дані введені вірно, отримує доступ до всіх можливостей системи.

В правому верхньому куті завжди знаходиться особистий кабінет користувача. При натисненні на свій логін користувач може побачити свою персональну інформацію та при потребі змінити пароль. Також тут так само можна вийти зі свого акаунту.

Також, на кожній зі сторінок завжди буде прикріплене меню користувача. Тут вказані всі функції до яких конкретний користувач має доступ. Меню включає в себе вкладки: «Дешборд», «Список працівників», «Список Проєктів», «Користувачі», та вкладка «Обслуговування» – яка містить в собі ще три службові вкладки: «Посади», «Проектні задачі» та «Проектні команди».

При вході в застосунок першим вікном буде «Дешборд». Тут користувач може бачити вже існуючі проєкти та сповіщення що до дедлайнів. Звідси можна перейти до інформації про проєкт натиснувши на потрібне зображення.

В картці проєкту буде прописано всю інформацію про проєкт, його титульне зображення (яке можна замінити) та візуальне відображення прогресу проєкту в вигляді діаграми. Тут також присутній функціонал редагування проєкту та додавання нового прогресу, який відразу відобразиться на попередній діаграмі в вигляді відсотку.

На вкладці «Список працівників» можна побачити список всіх людей, які працюють в цій групі чи компанії. Тут можна налаштувати кількість записів для відображення та знайти людину в пошуку, якщо записів багато. Також є можливість додати нового працівника та відфільтрувати активних та неактивних працівників.

При виборі будь-якого з профілів можна переглянути загальну інформацію про працівника. Користувач з адміністраторським акаунтом має змогу редагувати дані та змінити фотографію працівника.

Вкладка «Список проєктів» працює аналогічним чином як попередня, тобто виводить список всіх проєктів. Тут так само можна створити новий проєкт, передивитись інформацію про існуючий та відфільтрувати проєкти за статусом: В процесі, Завершений або Скасований.

Вкладка «Користувачі» відображає всіх користувачів системи, тобто людей, які безпосередньо мають доступ до застосунку. Ця вкладка доступна тільки для адміністраторів, оскільки призначена для додавання нових користувачів та зміни даних про їх акаунти. Важливий нюанс: щоб додати нового користувача необхідно спершу додати його як співробітника.

Вкладка «Посади» містить інформацію про можливі посади в компанії та дає можливість редагувати та додавати нові посади.

Вкладка «Проєктні задачі» працює за такою ж логікою. Вона надає функціонал створення нових задач відповідно до певного типу проєкту та можливість редагувати цю інформацію.

Остання вкладка «Проєктні команди» дає можливість групувати певних співробітників в команду. Тут є можливість вибору керівника проєкту та учасників команди. Також склад таких команд можна змінювати за потреби, тобто змінювати керівника, додавати чи прибирати працівників. Також команда може мати активний або неактивний статус на певний момент часу.

Таким чином, система має досить зручний та зрозумілий користувацький інтерфейс і за потреби можна звертатись до даного керівництва користувача щоб покращити всій досвід використання застосунку для керування проєктами.

3.4 Вимоги до технічних та програмних засобів

Для успішного запуску та стабільної роботи застосунку для керування проєктами пристрій повинен мати стабільне підключення до мережі Інтернет а також задовольняти мінімальні системні вимоги:

Мінімальні вимоги:

– операційна система: Windows 7+, macOS 10.12+ або сучасний дистрибутив Linux з оновленим ядром;

– процесор: двоядерний процесор з тактовою частотою 2 ГГц ;

– оперативна пам'ять: мінімум 2 ГБ оперативної пам'яті;

– монітор: роздільна здатність екрану не менше 1280x800 пікселів;

– браузер: сучасний веб-браузер, такий як Google Chrome, Mozilla Firefox, Safari або Microsoft Edge.

Рекомендовані вимоги:

– операційна система: Windows 10, macOS 11+, або остання версія Linux з підтримкою оновлень;

– процесор: чотирьохядерний процесор з тактовою частотою 3 ГГц;

– оперативна пам'ять: мінімум 4 ГБ оперативної пам'яті;

– монітор: роздільна здатність екрану не менше 1920x1080 пікселів для кращої роботи з інтерфейсом застосунку;

– браузер: остання версія будь-якого сучасного веб-браузера з підтримкою HTML5 та CSS3.

Отже по цих даних можна зробити висновок що для успішного запуску та стабільної роботи застосунку для керування проєктами важливо мати на увазі мінімальні та рекомендовані системні вимоги. Мінімальні вимоги забезпечують базовий рівень функціональності та продуктивності. Врахування цих вимог дозволяє забезпечити ефективну роботу застосунку на різних пристроях та операційних системах, що сприяє успішному управлінню проєктами та досягненню їх цілей.

3.5 Вибір та обґрунтування методів тестування додатку

Тестування важливий етап в розробці будь-якого програмного забезпечення, включаючи системи керування проєктами. Воно дозволяє перевірити функціональність, надійність і ефективність програми перед її впровадженням в реальне середовище.

Тестування допомагає забезпечити високу якість програмного забезпечення. Воно дозволяє виявити та усунути помилки та дефекти ще до того, як програма потрапить до кінцевих користувачів.

Коректно проведене тестування забезпечує високий рівень якості та надійності програмного забезпечення, що сприяє підвищенню довіри користувачів до системи[37].

На даний час, існує багато різноманітних видів тестування програмного забезпечення, які використовуються для перевірки його якості та відповідності вимогам. Наприклад, функціональне тестування спрямоване на перевірку функціональних можливостей програми, нефункціональне тестування оцінює характеристики, такі як продуктивність та безпека, модульне тестування проводиться для перевірки окремих модулів програми, а інтеграційне тестування — для перевірки взаємодії між ними. Це лише кілька прикладів, і використання конкретного типу тестування залежить від конкретних потреб та характеристик програмного продукту.

Після аналізу різних можливостей та потреб проєкту було обрано функціональне тестування і тестування рівня даних, як ключові етапи в процесі тестування даного програмного забезпечення, оскільки дуже важливо щоб застосунок відповідав всім заявленим функціональним вимогам. І в зв’язку з тим, що практично всі ці вимоги тим чи іншим чином пов’язані з базою даних буде доцільно провести тестування рівня даних, так як вони являють собою всю логіку застосунку та включають в себе майже весь функціонал застосунку. Тому їх правильність надзвичайно важлива в цьому контексті.

Функціональне тестування[38] важливе для перевірки коректності роботи програми з точки зору її функціональних можливостей. Під час цього типу тестування перевіряються всі функції програмного забезпечення, включаючи вхідні та вихідні дані, взаємодію з користувачем та інші аспекти роботи програми. Результати функціонального тестування допомагають виявити та виправити будь-які помилки в програмі перед випуском її в експлуатацію.

Тестування рівня даних[39], з іншого боку, спрямоване на перевірку правильності обробки та зберігання даних у базі даних. Це важливо для того, щоб упевнитися, що дані програми зберігаються та обробляються належним чином, що є критичним аспектом для її коректної роботи. Під час тестування рівня даних перевіряються операції додавання, зміни та видалення даних, а також їхня взаємодія з іншими компонентами програми.

Обидва типи тестування є важливими для забезпечення якості програмного забезпечення. Вони допомагають виявити та виправити помилки, забезпечуючи надійність та стабільність програми в реальних умовах експлуатації. Тестування рівня даних і функціональне тестування є невід'ємною частиною процесу розробки програмного забезпечення, які сприяють покращенню якості та задоволенню потреб користувачів.

Для проведення тестів буде використовуватись PHP Unit. Це популярний фреймворк для тестування PHP-додатків. Він надає зручні інструменти для написання та виконання тестів на рівні модулів, які дозволяють перевірити різні частини коду на очікувані результати.

Одиниці тестування PHP Unit[40] можуть бути написані для окремих функцій, класів або навіть цілих модулів програми. Це дозволяє перевірити кожну частину коду і переконатися, що вона працює, як очікувалося.

Крім того, PHP Unit надає зручний інтерфейс для перегляду результатів тестування, включаючи інформацію про тестовані методи, кількість помилок та невдач, а також покриття коду тестами. Загалом, використання PHP Unit дозволяє забезпечити якість PHP-коду шляхом ефективного тестування та виявлення помилок на ранніх етапах розробки.

3.6 Тестування додатка

Для перевірки функціоналу застосунку на відповідність всім заявленим вимогам виконано функціональне тестування з використанням PHPUnit Test. Щоб впевнитись що функціонал працює коректно, потрібно перевірити правильність додавання та зміни даних в базі даних, так як це є основою роботи всього застосунку. Для забезпечення якості роботи застосунку і відповідності його функціоналу вимогам виконано комплексне функціональне тестування. Цей процес включає ретельне аналізування різних аспектів програмного застосунку, таких як обробка даних, взаємодія з базою даних, коректність відображення інтерфейсу користувача та інші.

Далі наведено приклад коду для тестування додавання нового користувача в систему:

```
class Test extends TestCase
{

    public function testUserAction()
    {
        // Створюємо об'єкт класу для тестування дії 'user'
        $action = new Action('user');

        // Викликаємо метод, що тестує дію 'user'
        $result = $action->perform([
            'eid' => 1,
            'user' => 'test_user',
            'pass' => 'test_password',
            'u_type' => 'test_user_type'
        ]);

        // Перевіряємо, чи був успішно доданий користувач
        $this->assertTrue($result);
    }
}
```

Під час тестування слід переконатися, що функції, такі як додавання користувачів, зміна даних та інші операції, працюють належним чином і не порушують цілісність даних.

Таким чином, потрібно протестувати весь основний функціонал застосунку. Можливі сценарії функціонального тестування застосунку для керування проєктами наведено у таблиці 3.1.

Таблиця 3.1 – Тестові сценарії функціонального тестування

№	Функціональна вимога	Вихідні дані	Очікуваний результат
1	Авторизація в системі	Користувач вводить свій логін і пароль	При коректних даних користувач отримує доступ до системи, при неправильному вводі – повідомлення про помилку
2	Зміна паролю	Користувач відкриває свій кабінет та вводить новий пароль	При правильному підтвердженні попереднього пароль успішно змінено. При неправльному – повідомлення про помилку
3	Додання співробітника	Користувач заповнює форму з даним и про нового співробітника	Якщо всі дані введено коректно, в систему додається новий співробітник, додається зображення профілю
4	Додання користувача	Користувач з правами адміністратора заповнює форму користувача	Користувач без прав адміністратора не має доступу до цього функціоналу. Додається новий користувач в систему за умови якщо його попередньо створено як співробітника
5	Додання проєкту	Користувач заповнює форму з інформацією про проєкт	За умови коректного заповнення всіх полів додається новий проєкт в списку проєктів та на Дешборді
6	Редагування проєкту	Користувач відкриває інформацію про проєкт та змінює частину інформації	За умови коректного введення змінюється інформація про проєкт
7	Оновлення прогресу	Користувач обирає задачу проєкту і вписує новий прогрес	На графіку прогресу проєкту відображається доданий прогрес задачі в відсотковому форм аті
8	Додання нової посади	Користувач заповнює форму з інформацією про нову посаду	В списку доступних посад для співробітника з’являється новий варіант
9	Створення задачі	Користувач додає нову задачу для певного типу проєкту	За умови коректного введення при створенні нового чи редагуванні вже існуючого проєкту відображається, як варіант, нова задача
10	Створення команди	Користувач обирає керівника проєкту, його учасників та створює команду	Додається нова проєктна команда, яка буде пропонуватись як варіант при створенні новго проєкту
11	Нагадування про дедлайн	Користувач натискає на нагадування на Дешборді	Відкривається сторінка проєкту де скоро дедлайн виконання
12	Фільтрація записів	Користувач обирає потрібні фільтри в списку користувачів, проєктів чи співробінників	Виводиться інформація яка відповідна обраному фільтрові

Також для застосунку важливо провести тестування рівня даних, через те, що практично весь функціонал використовує базу даних та виконує різні операції по її змінах. Для прикладу протестовано функціонал додавання нового проєкту в систему. Завданням цього тесту є перевірка, чи правильно дані про новий проєкт зберігаються в базі даних після додавання.

```
use PHPUnit\Framework\TestCase;

class ProjectDataTest extends TestCase
{
    protected $conn;

    protected function setUp(): void
    {
        // Підключення до бази даних перед кожним тестом
        $this->conn = new mysqli("localhost", "username", "password",
"database");
    }
    public function testProjectInsertion()
    {
        // Введення нового проекту в базу даних
        $result = $this->conn->query("INSERT INTO projects (project, location,
overall_cost, start_date, deadline) VALUES ('Test Project', 'Location A',
'1000', '2024-04-15', '2024-05-15')");
       // Перевірка, чи вдалося вставити дані
        $this->assertTrue($result);
    }
    protected function tearDown(): void
    {
        // Відключення від бази даних після кожного тесту
        $this->conn->close();
    }
}
```

У цьому прикладі створюється новий проєкт у базі даних та перевіряється, чи вдалося коректно вставити дані. Якщо тест пройде успішно, це означатиме, що дані про новий проєкт правильно зберігаються в базі даних.

Отже підсумовуючи, такі тести також необхідно виконати для всієї бази даних, оскільки вони дуже важливі для перевірки коректності функціонування бази даних в цілому. Вони допомагають переконатися, що дані додавалися, змінювалися та видалялися відповідно до очікуваних результатів, а також що запити до бази даних повертають очікувані результати, так як на зв’язку з базою даних та коректністю її роботи та правильністю внесених змін тримається вся логіка застосунку для керування проєктами.

3.7 Аналіз результатів тестування

У результаті тестування було успішно протестовано описані сценарії. Для кожного з цих сценаріїв було написано окремий PHPUnit Test для симуляції прописаних умов та перевірки на відповідність очікуваним результатам. Результати функціонального тестування проведеного в попередньому підрозділі продемонстровано в таблиці 3.2, де відображено вимогу яка тестувалась, вхідні дані для тесту, отриманий результат та загальний результат на коректність.

Таблиця 3.2 – Результати функціонального тестування

№	Функціональна вимога	Вхідні дані	Отриманий результат	Результат
1	Авторизація в системі	Логін і пароль	Користувач отримує доступ до системи, або повідомлення про помилкові дані	Правильно
2	Зміна паролю	Старий і новий пароль	Пароль успішно змінено	Правильно
3	Додання співробітника	Дані про нового співробітника	Додано нового співробітника	Правильно
4	Додання користувача	Дані про нового користувача	Додано нового користувача, його створено як співробітника	Правильно
5	Додання проєкту	Дані про проєкт	Додано новий проєкт в список проєктів та на Дешборд	Правильно
6	Редагування проєкту	Оновлена інформація	Інформацію про проєкт змінено	Правильно
7	Оновлення прогресу	Дані оновлення прогресу	На графіку прогресу проєкту відображається прогрес задачі в відсотковому форматі	Правильно
8	Додання нової посади	Дані про нову посаду	В списку посад співробітника з’являється новий варіант	Правильно
9	Створення задачі	Дані про нову задачу	При створенні нового чи редагуванні вже існуючого проєкту показано нову задачу	Правильно
10	Створення команди	Керівник проєкту, список учасників команди	Нова проєктна команда, яка буде пропонуватись як варіант при створенні нового проєкту	Правильно
11	Нагадування про дедлайн	Нагадування на дешборді	Відкривається сторінка проєкту від якої прийшло сповіщення про дедлайн виконання	Правильно
12	Фільтрація записів	Фільтри списку користувачів, проєктів чи співробіників	Виводиться інформація яка відповідає обраному фільтрові	Правильно

Також розглянуто результати тестування рівня даних для застосунку керування проєктами з попереднього підрозділу. Запуск Unit Test демонструє результат проходження тесту в консолі (рисунок 3.2).

```
PHPUnit 9.5.10 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 00:00.021, Memory: 8.00 MB

OK (1 test, 1 assertion)
```

Рисунок 3.2 – Результати тесту

Таким чином було проведено тести всіх таблиць бази даних і було отримано позитивний результат. Це означає що база даних працює коректно, та всі дані зі сторони застосунку записуються правильно.

3.8 Висновки програмної реалізації та тестування

Підсумовуючи даний розділ, було програмно реалізовано застосунок для керування проєктами, розроблено та підключено базу даних да застосунку.

В результаті отримано готовий застосунок, для якого підготовано інструкцію користувача. Для впевненості в якості продукту його було протестовано двома методами.

Підсумки проведеного тестування застосунку для керування проєктами можна вважати цілком успішними. Отримані результати свідчать про те, що система працює коректно. Це означає, що основний функціонал системи працює належним чином, і користувачі можуть використовувати систему без помилок. Такий позитивний результат підтверджує правильність функціонування системи і підвищує рівень довіри до неї з боку користувачів.

ВИСНОВКИ

Метою дослідження була проектування та розробка високоефективного застосунку для керування проєктами, що дасть можливість вирішити завдання управління проєктами, і забезпечити зручний та інтуїтивно зрозумілий інтерфейс для користувачів.

Завданням був аналіз потреб користувачів, проєктування, розробка застосунку для керування проєктами та його тестування, забезпечення безпеки та оптимізації його функціональності.

Робота мала на меті розкрити всі аспекти проєктування та розробки застосунку для керування проєктами.

Для реалізації поставленої мети було проведено дослідження предметної області та постановку задач, спроєктовано програмне забезпечення, розроблено і протестовано застосунок та підведено підсумки.

Для вирішення задачі були виконані наступний об’єм робіт:

– дослідження предметної області: протягом даного етапу безпосередньо було проведено змістовний аналіз предметної області, проаналізовано декілька наявного програмно-технічного забезпечення даної предметної області та визначено вимоги до програмного забезпечення і на основі цих вимог розроблено технічне завдання. Кінцевим результатом етапу є детальний перелік вимог, які необхідно виконати в процесі реалізації проєкту;

– проєктування програмного забезпечення: на даному етапі було проаналізовано декілька типів архітектур та обрано ту, яка найбільше підходить для реалізації всіх вимог до застосунку, розроблено логічну структуру бази даних, спроєктовано та описано інтерфейс користувача та проаналізовано та вибрано технології та методи реалізації застосунку для керування проєктами. Як результат було обрано архітектуру для застосунку, спроектовану модель бази даних, та готовий дизайн інтерфейсу для подальшої розробки. Також було обрано інструменти та середовище розробки;

– програмна реалізація та тестування: під час цього етапу було програмно реалізовано модулі застосунку, розроблено базу даних, додано керівництво користувача та вимоги до технічних та програмних засобів. Фінальною частиною даного етапу було вибрано методи тестування додатку, його безпосереднє тестування та проведено аналіз результатів. Результатом є готовий протестований застосунок для керування проєктами.

Загальний висновок полягає в тому, що розроблений застосунок для керування проєктами має дружній інтерфейс, підтримує різноманітні функціональні можливості, такі як: створення проєктів, призначення завдань, відстеження прогресу, створення команд і так далі. Завдяки відповідним системним вимогам і рекомендаціям щодо технічних характеристик, користувачі можуть максимально ефективно використовувати застосунок для досягнення своїх цілей у керуванні проєктами. раціональною та ефективною для вирішення поставленої задачі.

Отримані результати показали, що розроблений програмний застосунок може успішно виконувати поставлені завдання та функціональні вимоги. Досягнуті результати є основою для подальшого розвитку та покращення застосунку для керування проєктами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Сороко С. В. Керування проектами. Сучасні підходи : підручник. Київ : Лібра ЛТД, 2021. 440 с.

2. Загородній О. В. Цифрова трансформація проектного менеджменту. Київ : НДЦ Інститут економіки та управління НАН України, 2020. 272 с.

3. Swaber K. Essential Scrum: a practical guide for agile teams. Express polygraph, 2020. 352 с.

4. Project Management Institute (PMI). URL: https://www.pmi.org/ (дата звернення: 20.01.2024).

5. Association for Project Management (APM). URL: https://www.apm.org.uk/ (дата звернення: 20.01.2024).

6. Якимчук Г.Б., Праворська Н.І., Програмний застосунок для керування проектами. Збірник наукових праць за матеріалами XV Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2023». Хмельницький. 2023. – 329-332 с.

7. Jira. URL: https://www.atlassian.com/software/jira (дата звернення: 10.02.2024).

8. Trello. URL: https://trello.com/tour (дата звернення: 10.02.2024).

9. Microsoft Project. URL: https://www.microsoft.com/en-us/microsoft-365/project/project-management-software (дата звернення: 10.02.2024).

10. Distch D. Data flow diagrams: a visual guide to system modeling and analysis. 2023. 464 с.

11. Lucidchart. URL: https://www.lucidchart.com (дата звернення: 15.02.2024).

12. Crispin L. User Stories and Agile Software Development. 2018. 286 с.

13. VisualParadigm. URL: https://online.visual-paradigm.com/ (дата звернення: 15.02.2024).

14. Perri M. User Stories and Acceptance Criteria: Bridging the Gap Between Development and Design. 2020. 354 с.

15. The Ultimate Guide to Writing User Stories. URL: https://www.productplan.com/glossary/user-story/ (дата звернення: 18.02.2024).

16. Shalloway Alan, James R. Trott. Software Requirements: A Critical Perspective. 2020. 315 с.

17. Mark Richards Fundamentals of Software Architecture. O'Reilly Media, 2020. – 419 с.

18. Gregor Hohpe The Software Architect Elevator: Redefining the Architect's Role in the Digital Enterprise. O'Reilly Media, 2020. – 365 с.

19. Amazon Web Services: Архітектура клієнт-сервер. URL: https://aws.amazon.com/ru/compare/the-difference-between-web-server-and-application-server/ (дата звернення: 25.02.2024).

20. Understanding the Model-View-Controller (MVC) Architecture. URL: https://www.sitepoint.com/model-view-controller-mvc-architecture-rails/ (дата звернення: 25.02.2024).

21. API Evangelist: REST API Tutorial. URL: https://apievangelist.com/ (дата звернення: 25.02.2024).

22. Microservices.io. URL: https://microservices.io/ (дата звернення: 25.02.2024).

23. Kleppmann M. Designing Data-Intensive Applications. O'Reilly Media. 2019. 728 с.

24. Tidwell Jenifer. Designing Interfaces: Patterns for Effective Interaction Design (3rd edition). O'Reilly Media, 2020. 599 с.

25. Figma. URL: https://www.figma.com (дата звернення: 28.02.2024).

26. Офіційний веб-сайт PHP. URL: https://www.php.net/manual/en/index.php (дата звернення: 20.03.2024).

27. Sklar David, Trachtenberg Adam. PHP Cookbook, 5th Edition. Birmingham, UK: Packt Publishing, 2022. 1120 с.

28. Документація PHPStorm. URL: https://www.jetbrains.com/phpstorm/ (дата звернення: 20.03.2024).

29. Liebler Eric. PHPStorm Unleashed: A Comprehensive Guide to PHP Development with PHPStorm. Birmingham, UK: Packt Publishing, 2020. 584 с.

30. Документація XAMPP. URL: https://www.apachefriends.org/docs/ (дата звернення: 20.03.2024).

31. Lenox Jack. Working with XAMPP: A Comprehensive Guide to Installing, Configuring, and Using XAMPP for Web Development. Birmingham, UK: Packt Publishing, 2021. 448 с.

32. Документація MySQL. URL: https://dev.mysql.com/doc/ (дата звернення: 20.03.2024).

33. Foretic Marko. MySQL in Depth: Unleashing the Power of Relational Data Management. Birmingham, UK: Packt Publishing, 2022. 1176 с.

34. Tejaswini Jog. A Roadmap with Instructions for the Effective Implementation of Features, Codes, and Programs (English Edition). BPB Publications, 2022. 326 с.

35. Документація PHP. URL: https://www.php.net/manual/en/index.php (дата звернення: 03.04.2024).

36. Pollock Josh. Modern PHP: Next-Generation Web Development. Packt Publishing, 2021. 656 с.

37. Crispin, Lisa. Practical Software Testing: A Comprehensive Guide. Addison-Wesley Professional, 2018. 672 с.

38. Functional Testing: A Complete Guide. overview of the methods and best practices. URL: https://www.geeksforgeeks.org/software-testing-functional-testing/ (дата звернення: 23.04.2024).

39. What is Data-Driven Testing? A Complete Guide. URL: https://www.tutorialspoint.com/data_driven_testing.ht (дата звернення: 23.04.2024).

40. Документація PHPUnit. URL: https://phpunit.de/documentation.html (дата звернення: 28.04.2024).

ДОДАТОК А
(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту автоматизації керування проєктом.

1 Підстава для розробки

Підставою для розробки є «Завдання на дипломний проект», затверджене завідувачем кафедри інженерії програмного забезпечення.

Найменування розробки: Програмний застосунок для керування проєктами

2 Призначення розробки

Застосунок для керування проєктами призначений для полегшення та оптимізації процесів управління проєктами будь-якої складності. Основними завданнями розробленого застосунку є забезпечення ефективного планування, контролю та виконання проєктів, а також підвищення продуктивності команди та зниження ризиків затримок та неуспішного завершення проєктів.

Користувачами застосунку є проектні менеджери, команди проєктів та інші учасники, які відповідають за виконання конкретних завдань у межах проєктів. Вони можуть спільно працювати над завданнями, встановлювати пріоритети та відстежувати прогрес кожного етапу роботи.

Розроблений застосунок надає можливості для планування та розподілу завдань, відстеження прогресу, а також аналізу ризиків та прийняття рішень на основі зібраної статистики. Він дозволяє адаптуватися до різних методологій управління проєктами, та забезпечує високу гнучкість та простоту використання для різних типів проєктів та команд.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Програмний застосунок для керування проєктами повинен забезпечувати виконання наступних функцій:

Авторизація та аутентифікація:

– система повинна забезпечити можливість авторизації користувачів з різними ролями (менеджери проєктів, команди, адміністратори тощо) та аутентифікації їх ідентичності.

Управління проєктами:

– користувач повинен мати можливість створювати нові проєкти, вказуючи їх назву, опис та інші додаткові атрибути;

– після створення проєкту користувач повинен мати змогу редагувати інформацію про нього, включаючи назву, опис, терміни, призначені ресурси, тд;

– користувач повинен мати можливість видаляти проєкти, які більше не потрібні або завершилися, з можливістю попереднього підтвердження дії.

Управління завданнями:

– користувач має можливість створювати нові завдання для кожного проєкту, вказуючи назву, опис, термін виконання, пріоритет та інші деталі;

– редагування завдань включає можливість зміни будь-яких параметрів, включаючи зміну дедлайну, пріоритету або опису завдання;

– під час створення або редагування завдання, користувач може призначити одну чи кілька осіб, які відповідають за виконання цього завдання;

– користувач може відстежувати прогрес виконання кожного завдання через визначення статусу завдання (наприклад: «В роботі», «Завершено», «Відкладено»);

– система автоматично відображає відсоток завершення завдання та дедлайн, щоб користувачі могли легко оцінити прогрес роботи;

– система надає можливість спільного доступу до завдань для всіх учасників команди, що сприяє колективному розв'язанню проблем та досягненню спільних цілей.

Аналітика та звітність:

– система надає можливість генерувати детальні звіти про прогрес кожного проєкту, включаючи інформацію про завдання, терміни виконання, відповідальних осіб та статус завдань;

– графіки можуть відображати прогрес проектів у часі, розподіл ресурсів, витрати, пріоритети завдань тощо, що дозволяє швидко зрозуміти стан проектів та приймати обґрунтовані управлінські рішення.

Безпека та захист даних:

– доступ до даних обмежується за допомогою ролей та прав доступу, що забезпечує обмеження доступу лише до необхідної інформації.

Ці функціональні вимоги мають бути ретельно розроблені та реалізовані з метою забезпечення повного функціонального покриття для всіх користувачів та сценаріїв використання застосунку.

3.2 Вимоги до надійності

Розроблюване програмне забезпечення повинно мати наступні характеристики надійності:

– здатність до автоматичного відновлення після виникнення збоїв, таких як втрата електроживлення або непередбачені помилки системи;

– захищеність за допомогою пароля при запуску застосунку для забезпечення обмеженого доступу до нього;

– обмеження несанкціонованого доступу до даних, забезпечуючи безпеку та конфіденційність інформації;

– можливість регулярного резервного копіювання інформаційної бази для запобігання втрати даних у випадку виникнення непередбачених ситуацій;

– розмежування прав користувачів для контролю доступу та забезпечення безпеки даних.

3.3 Умови експлуатації та вимоги до технічних засобів

Системні вимоги для роботи програмного застосунку повинні мати:

– тактова частота процесора – 1200 Гц;

– обсяг оперативної пам’яті – 64 Мб;

– обсяг вільного дискового простору – 50 Мб;

– роздільна здатність монітора 1024 ×768;

3.4 Вимоги до інформаційної та програмної сумісності

Сумісність з різними веб-браузерами: Веб-сайт повинен коректно відображатися на найпопулярніших веб-браузерах, таких як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge та Opera.

Сумісність з різними операційними системами: Сайт має бути доступним для користувачів з різними операційними системами, такими як Windows, macOS, Linux тощо.

Сумісність з версіями PHP: Веб-сайт повинен працювати на різних версіях PHP, забезпечуючи коректне виконання коду та запитів.

3.5 Спеціальні вимоги

Програма має мати зручний та інтуітивно зрозумілий користувацький інтерфейс, що легко сприймається користувачами з середнім рівнем комп'ютерної грамотності. Мова програмування обирається виконавцем.

4 Вимоги до програмної документації

Під час розроблення застосунку має бути підготовлено: текст програми, опис програми, програма і методика випробувань, керівництво користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки застосунку для керування проєктами прописані у таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.24 – 31.01.24	Обґрунтування необхідності розробки програми	Коротка характеристика програмного забезпечення; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 01.02.24 – 26.02.24	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; уточнення середовища програмування; розробка і опис загальної алгоритмічної структури
Технічний проект 29.02.24 – 19.03.24	Розробка технічного проекту	Уточнення структури вхідних і вихідних даних; розробка докладного алгоритму; розробка структури програми
Робочий проект 20.03.24 – 15.04.24	Розробка програмного забезпечення	Реалізація програмного забезпечення; відлагодження; проведення попереднього тестування
Розробка програмної документації 16.04.24 – 22.04.24	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням

Кінець таблиці А.1

1	2	3
Тестування системи 23.04.24 – 30.04.24	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Розробка програмної документації 16.04.24 – 22.04.24	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням

6 Порядок контролю та приймання

Контроль і приймання розробленого застосунку здійснюються на основі розробленої методики випробувань. Виконання всіх функцій застосунку перевіряється шляхом програмного та мануального тестування групою потенційних користувачів. Прийом програмного забезпечення здійснюється після успішного тестування.

ДОДАТОК Б
(обов'язковий)

КОД (ЛІСТИНГ) ЗАСТОСУНКУ

Index.php

```
<?php
session_start();
if (isset($_SESSION['UID'])) {
    header('location:pages/index.php');
    exit;
}
include_once 'includes/header2.php';
?>
<style>
    title1 {
       display: block;
       width:50%;
       height:90px;
       background-color: white;
       padding:1px;
       border-radius:5px;
       position:fixed;
       top:30%;
    }
    #main-bod{
           background: url(images/1930875.jpg);
           background-repeat: no-repeat;
          background-size: cover;
          display:flex;
          height:calc(100%);
          width:calc(100%);
          align-items:center;
          justify-content:center;
          top: 0;
          margin:unset
    }
</style>
<body id="main-bod">
       <div class="col-lg-4">
          <div class="panel panel-info" style="">
             <div class="panel-heading">
                Логін
             </div>
             <div class="panel-body">
                 <div class="container-fluid">
                 <form class="form-horizontal" method="POST" id="login_form">
                           <div class="form-group" id="form-login">
                              <label for="" class="control-label">Ім'я
користувача</label>
                              <input class="form-control" id="user" name="user"
type="text">
                           </div>
                           <div class="form-group">
                              <label for="" class="control-
label">Пароль</label>
                              <input type="password" name="pass" id="pass"
class="form-control">
                           </div>
                           <div class="form-group" id="msg">
                           <div class="col-sm-8 col-sm-offset-8">
```

```
                              <button type="submit" class="btn btn-
info">Увійти</button> <br>
                              </div>

                              <div class="col-sm-12">
                                 <div class="alert alert-success" id="correct">
Вхід виконано успішно!</div>
                                 <div class="alert alert-danger" id="error">
Помилка входу </div>
                              </div>
                              </div>
                           </form>
                  </div>
               </div>
            </div>
         </div>
</body>
<script>
       jQuery(document).ready(function(){
                jQuery(document).ready(function(){
                   $("#correct").hide();
                   $("#error").hide();
                   jQuery("#login_form").submit(function(e){
                         e.preventDefault();
                         var formData = jQuery(this).serialize();
                         $.ajax({
                            type: "POST",
                            url: "includes/login.php",
                            data: formData,
                            success: function(html){
                            if(html=='true' )
                            {
                               $('#error').hide();
                               $("#correct").slideDown();
                               var delay = 2000;
                               setTimeout(function(){ window.location =
'pages/index.php?page=home';   }, delay);
                            }else{
                            $('#error').slideDown();
                               var delay = 2000;
                               setTimeout(function(){ $('#error').slideUp();  },
delay);
                            }
                            }
                         });
                               return false;
                   });
                   });
                   });

</script>
```

add_forms.php

```
<?php include '../includes/db.php' ?>
<?php
$action = $_GET['action'];

if($action == 'user'){
      $eid = $_POST['eid'];
      $user = $_POST['user'];
      $pass = $_POST['pass'];
```

```
	$u_type = $_POST['u_type'];
	if($query = mysqli_query($conn,"INSERT INTO users
(eid,username,password,user_type,io)VALUES('$eid','$user','$pass','$u_type','1')
")){
		echo '<script>$("#msg").show("SlideDown");</script>';
	}else{
		echo "<script>alert('Помилка збереження даних!')</script>";
	}
}
if($action == 'position'){

	$pos = $_POST['position'];
	$dr = $_POST['dr'];
	if($query = mysqli_query($conn,"INSERT INTO position
(position,daily_rate)VALUES('$pos','$dr')")){
		include '../includes/msg_box.php';
	}else{
		echo "<script>alert('Помилка збереження даних!')</script>";
	}
}
if($action == 'attendance'){

	foreach($_GET as $var=>$value)
		$$var =$value;
	if($task == 'in'){
		$query = mysqli_query($conn,"INSERT INTO attendance
(eid,time_in,date_today) VALUES('$id',now(),'$d') ");
	}
	if($task == 'del'){
		$query = mysqli_query($conn,"DELETE from attendance where eid ='$id'
and date_today = '$d' ");
	}
	if($task == 'out'){
		$query2 = mysqli_query($conn,"UPDATE attendance set time_out = now()
where eid ='$id' and date_today = '$d' ");
	}
	if($task == 'out'){
		$query2 = mysqli_query($conn,"UPDATE attendance set time_out = ''
where eid ='$id' and date_today = '$d' ");
	}
	if($query){

		echo '<script>
		window.location.reload();
		</script>';
	}else{
		echo "<script>alert('Помилка збереження даних!')</script>";
		echo '<script>
		window.location.reload();
		</script>';
	}
	if($query2){
		echo '<script>
		window.location.reload();
		</script>';
	}else{
		echo "<script>alert('Працівник ще не встиг прийти.')</script>";
		echo '<script>
		window.location.reload();
		</script>';
	}
}
if($action == 'division'){

	$division = $_POST['division'];
	$p_type =  $_POST['p_type'];
```

```
	if($query = mysqli_query($conn,"INSERT INTO project_division
(division,project_type)VALUES('$division','$p_type')")){
		include '../includes/msg_box.php';
	}else{
		echo "<script>alert('Помилка збереження даних!')</script>";
	}
}
if($action == 'employee'){
	$fname =$_POST['fname'];
	$lname =$_POST['lname'];
	$mname =$_POST['mname'];
	$address =$_POST['address'];
	$gender =$_POST['gender'];
	$bday =$_POST['bday'];
	$cn =$_POST['cn'];
	$position =$_POST['position'];
	$status =$_POST['status'];
		$file = "no_image.jpg";
		$e_query = mysqli_query($conn,"SELECT * FROM employee order by eid
DESC limit 1");
	$e_row = mysqli_fetch_assoc($e_query);
	if($e_row > 0 && $e_row['ecode'] != ''){
		$ecode = $e_row['ecode']+'1';
	}else { $ecode = '1001';}

	$query = mysqli_query($conn,"INSERT INTO employee
(lastname,firstname,midname,bday,contact_no,address,pid,status,gender,ecode,e_pi
c,io,date_added)

	VALUES('$lname','$fname','$mname','$bday','$cn','$address','$position','$s
tatus','$gender','$ecode','$file','1',NOW())");
	$last_id = mysqli_insert_id($conn);
	if($query){
		echo '<script>$("#suc_msg").show("slidedown");
		 var delay = 1500;
			setTimeout(function(){  window.location =
"index.php?page=employee_profile&id='. $last_id.'&dattyp=new";   }, delay);
		</script>';
	}else{
		echo '<script>$("#err_msg").show("slidedown");</script>';

	}
}
if($action == 'project'){
	$pname =$_POST['pname'];
	$location =$_POST['location'];
	$cost =$_POST['cost'];
	$deadline =$_POST['deadline'];
	$sdate =$_POST['sdate'];
	$tid =$_POST['tid'];
	$p_type =$_POST['p_type'];
	$file = "no_image.jpg";

	$query = mysqli_query($conn,"INSERT INTO projects
(project,location,overall_cost,start_date,deadline,site_pic,tid,date_added,io,pr
oposed_project)

	VALUES('$pname','$location','$cost','$sdate','$deadline','$file','$tid',no
w(),'1','$p_type')");
	$last_id = mysqli_insert_id($conn);
	if(isset($_POST['divs'])){

	$divs= $_POST['divs'];
	$cd = count($divs);
	for($i=0; $i < $cd; $i++){
```

```
            $query2 = mysqli_query($conn,"INSERT INTO project_partition
(project_id,pd_id)
            VALUES('$last_id','$divs[$i]')");
      }}
      if($query && $query2){
            echo '<script>$("#suc_msg2").show("slidedown");
            var delay = 1500;
                  setTimeout(function(){  window.location =
"index.php?page=project_detail&id='. $last_id.'&dattyp=new";   }, delay);
            </script>';
      }else{
            echo $query;
      }
}
if($action == 'team'){
      $fid = $_POST['fid'];

      $q1 = mysqli_query($conn,"INSERT INTO project_team (eid,date_added,pio)
VALUES('$fid',now(),'1')");
      $id = mysqli_insert_id($conn);
      if(isset($_POST['mid'])){
            $mid = $_POST['mid'];
            $mc=count($mid);
            for($i = 0 ; $i < $mc;$i++){
                  $q2 = mysqli_query($conn,"INSERT INTO team_member (tid,eid)
VALUES('$id','$mid[$i]')");
            }
      }
      if($q1){
            echo "true";
      }
}
if($action =='progress'){
      foreach ($_POST as $var => $value)
            $$var = $value;
      $rd2 = mt_rand(1000, 9999);
                  $filename = basename($_FILES['image']['name']);
                  $ext = substr($filename, strrpos($filename, '.') + 1);
                  $file = $rd2. "_" .$filename;

      (move_uploaded_file($_FILES['image']['tmp_name'],'../images/'.$file));
}
      $query= mysqli_query($conn,"INSERT INTO project_progress
(pp_id,progress,date_added,partition_img)values('$div','$prog',now(),'$file')
");
      if($query){
            echo "<script>location.replace(document.referrer);</script>";
      }
 ?>
```

update_forms.php

```
<?php
session_start();
include '../includes/db.php';

$action = $_GET['action'];
if($action == 'user'){

      $uid = $_POST['uid'];
      $user = $_POST['user'];
      $pass = $_POST['pass'];
      $u_type = $_POST['u_type'];
```

```
      $status = $_POST['status'];
      if($query = mysqli_query($conn,"UPDATE users SET username = '$user',
                                                                  password =
'$pass',

      user_type = '$u_type',
                  io = '$status' where uid = '$uid' ")){
            echo '<div class="alert alert-success" id="msg2"><i class="fa fa-
check"></i> Дані успішно оновлені! </div>
      <script>$("#msg2").show("SlideDown");
      </script>';
      }else{
            echo "<script>alert('Помилка оновлення даних!.')</script>";
      }
}
if($action == 'user2'){
      $uid = $_POST['uid'];
      $query20= mysqli_query($conn,"SELECT * FROM users where uid ='$uid'");
      $row20 = mysqli_fetch_assoc($query20);
      $cpass = $_POST['current'];
      $user = $_POST['user'];
      $npass = $_POST['npass'];
      if($cpass == $row20['password']){

      if($query = mysqli_query($conn,"UPDATE users SET username = '$user',
                                                                  password =
'$npass' where uid = '$uid' ")){
            echo '<script>$("#msg20").show("SlideDown");
            $("#msg21").hide();
       var delay = 2000;
                  setTimeout(function(){  window.location.reload();   },
delay);
      </script>';
      }else{
            echo "<script>alert('Помилка оновлення даних!')</script>";
      }}else{
      echo '<script>$("#msg21").show("SlideDown");
      </script>';}
}
if($action == 'position'){
      $id= $_POST['id'];
      $pos= $_POST['pos'];
      $dr= $_POST['dr'];
      if(mysqli_query($conn,"UPDATE position SET daily_rate = '$dr' , position =
'$pos' where pid = '$id'")){
            echo '<h4 class="alert alert-success"><i class="fa fa-fw fa-
check"></i>  Дані успішно оновлені!</h4>';
      }else{
            echo "<script>alert('Помилка оновлення даних!')</script>";
      }
}
if($action == 'division'){
      $id = $_POST['id'];
      $division = $_POST['division'];
      $p_type = $_POST['p_type'];
      if($query = mysqli_query($conn,"UPDATE project_division set division =
'$division', project_type = '$p_type' where pd_id = '$id'")){
            include '../includes/msg_box.php';
            echo '<script>$("#msg").html("Дані успішно оновлені!")</script>';
      }else{
            echo "<script>alert('Помилка оновлення даних!')</script>";
      }
}
if($action == 'change_pic'){
      $id = $_GET['id'];
                  $rd2 = mt_rand(1000, 9999);
```

```
                  $filename = basename($_FILES['file']['name']);
                  $ext = substr($filename, strrpos($filename, '.') + 1);
                  $file = $rd2. "_" .$filename;

      (move_uploaded_file($_FILES['file']['tmp_name'],'../images/'.$file));
      $query = mysqli_query($conn,"UPDATE employee set e_pic = '$file' where eid
= '$id'");
      if($query){
            echo '<script> location.replace(document.referrer);</script>';
      }
}if($action == 'change_pic2'){
      $id = $_GET['id'];
                  $rd2 = mt_rand(1000, 9999);
                  $filename = basename($_FILES['file']['name']);
                  $ext = substr($filename, strrpos($filename, '.') + 1);
                  $file = $rd2. "_" .$filename;

      (move_uploaded_file($_FILES['file']['tmp_name'],'../images/'.$file));
      $query = mysqli_query($conn,"UPDATE projects set site_pic = '$file' where
project_id = '$id'");
      if($query){
            echo '<script> location.replace(document.referrer);</script>';
      }
}
if($action == 'employee'){
      $id =$_POST['id'];
      $fname =$_POST['fname'];
      $lname =$_POST['lname'];
      $mname =$_POST['mname'];
      $address =$_POST['address'];
      $gender =$_POST['gender'];
      $bday =$_POST['bday'];
      $cn =$_POST['cn'];
      $position =$_POST['position'];
      $status =$_POST['status'];
      $ps =$_POST['ps'];

      $query = mysqli_query($conn,"UPDATE employee SET lastname = '$lname',
                                                                  firstname =
'$fname',

      midname = '$mname',

      bday = '$bday',

      contact_no = '$cn',

      address = '$address',

      pid = '$position',

      status = '$status',

      gender = '$gender',
                                                                        io
= '$ps' where eid = '$id' ");
      if($query){
            echo '<script>$("#suc_msg1").show("slidedown");
             var delay = 1500;
                   setTimeout(function(){  window.location =
"index.php?page=employee_profile&id='. $id.'";   }, delay);
            </script>';
      }else{
            echo '<script>$("#err_msg1").show("slidedown");</script>';

}}
```

```
if($action == 'project'){
	$id =$_POST['id'];
	$pname =$_POST['pname'];
	$location =$_POST['location'];
	$cost =$_POST['cost'];
	$deadline =$_POST['deadline'];
	$sdate =$_POST['sdate'];
	$tid =$_POST['tid'];
	$p_type =$_POST['p_type'];
	$stats =$_POST['stats'];

	$query = mysqli_query($conn,"UPDATE projects SET project = '$pname',

	location = '$location',

	overall_cost = '$cost',

	deadline = '$deadline',

	start_date = '$sdate',

	tid = '$tid',
                                                                                io
= '$stats',

	proposed_project ='$p_type' where project_id = '$id' ");
	if($query ){
		echo '<script>$("#suc_msg2").show("slidedown");
		var delay = 1500;
			setTimeout(function(){  window.location =
"index.php?page=project_detail&id='. $id.'&stats='.$stats.' ";   }, delay);
		</script>';
	}else{
		echo $query;
	}
}
if($action == 'progress'){
	foreach($_POST as $var=>$value)
		$$var = $value
	$query = mysqli_query($conn,"UPDATE project_progress set pp_id =
'$div',progress = '$prog' where prog_id = '$id' ");
	if($query){
		echo '<script>location.replace(document.referrer);</script>';
	}
}
if($action == 'team'){
	$id = $_POST['id'];
	$fid = $_POST['fid'];
	$q1 = mysqli_query($conn,"UPDATE project_team SET eid = $fid where tid =
'$id' ");
	if(isset($_POST['mid'])){
		$mid = $_POST['mid'];
		$mc=count($mid);
		for($i = 0 ; $i < $mc;$i++){
			$q2 = mysqli_query($conn,"INSERT INTO team_member (tid,eid)
VALUES('$id','$mid[$i]')");
		}
	}
	if($q1){
		echo "true";}}
if($action == 'team_stats'){
	$io = $_GET['io'];
	$id = $_GET['id'];
	$update=mysqli_query($conn,"UPDATE project_team set pio = '$io' where
tid='$id' ");
	if($update){
```

```
            echo '<script>location.replace(document.referrer);</script>';}}
if($action == 'attendance'){
      foreach($_GET as $var=>$value)
            $$var =$value;
      if($task == 'out'){
            $query2 = mysqli_query($conn,"UPDATE attendance set time_out = now()
where eid ='$id' and date_today = '$d' ");
      }
      if($task == 'odel'){
            $query2 = mysqli_query($conn,"UPDATE attendance set time_out = ''
where eid ='$id' and date_today = '$d' ");
      }
      if($query2){
            echo '<script>
            window.location.reload();
            </script>';
      }else{
            echo "<script>alert('Employee haven't time in yet.')</script>";
            echo '<script>
            window.location.reload();
            </script>';
      }
}
?>
```

auth.php

```
< ?php
      //Start session
      session_start();
      //Check whether the session variable SESS_MEMBER_ID is present or not
      if(!isset($_SESSION['ID']) || (trim($_SESSION['ID']) == '')) {
            header("location: ../");
            exit();
      }
?>
```

db.php

```
<?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());
}
?>
```

login.php

```
<?php
global $conn;
session_start();
      $errmsg_arr = array();
      $errflag = false;
      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) {
			session_regenerate_id();
			$name = $row['firstname'] . ' ' . $row['lastname'];
			$_SESSION['ID'] = $row['eid'];
			$_SESSION['UID'] = $row['uid'];
			$_SESSION['TYPE'] = $row['user_type'];
			$_SESSION['NAME'] = $name;
			echo "true";}}
		?>php
```

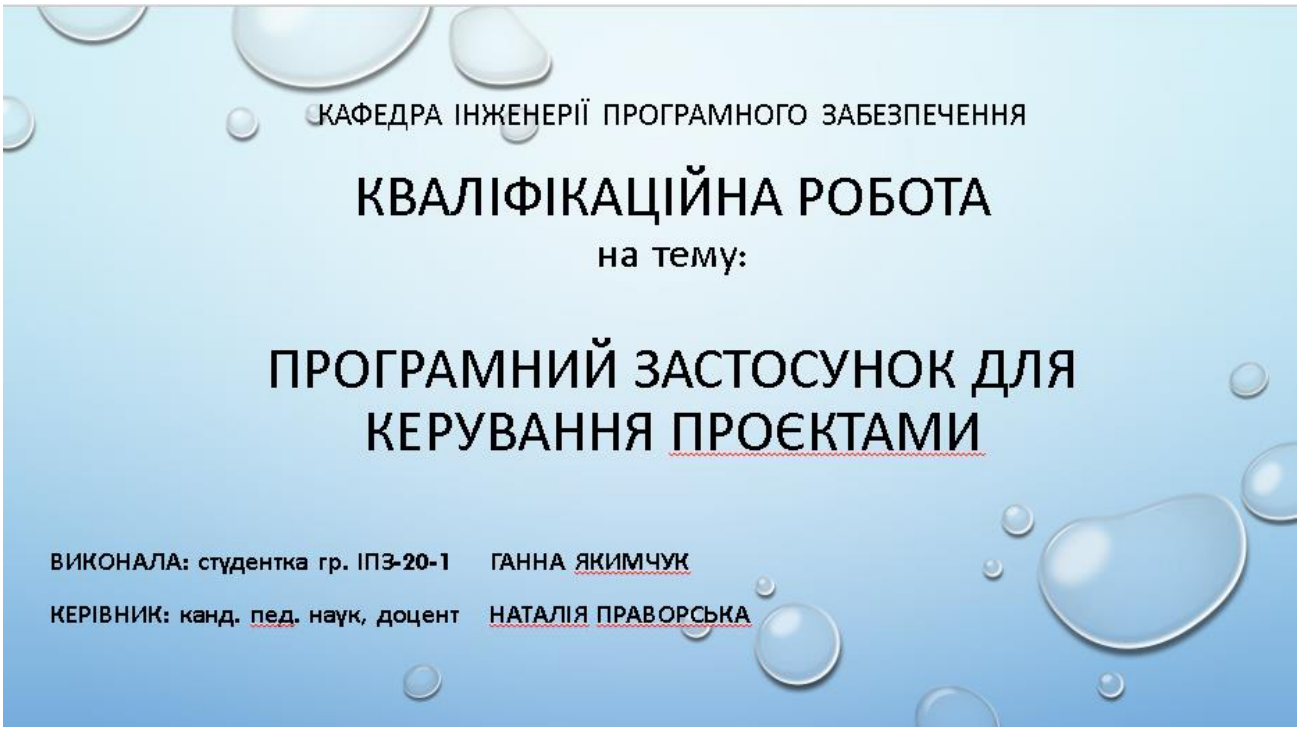
home.php

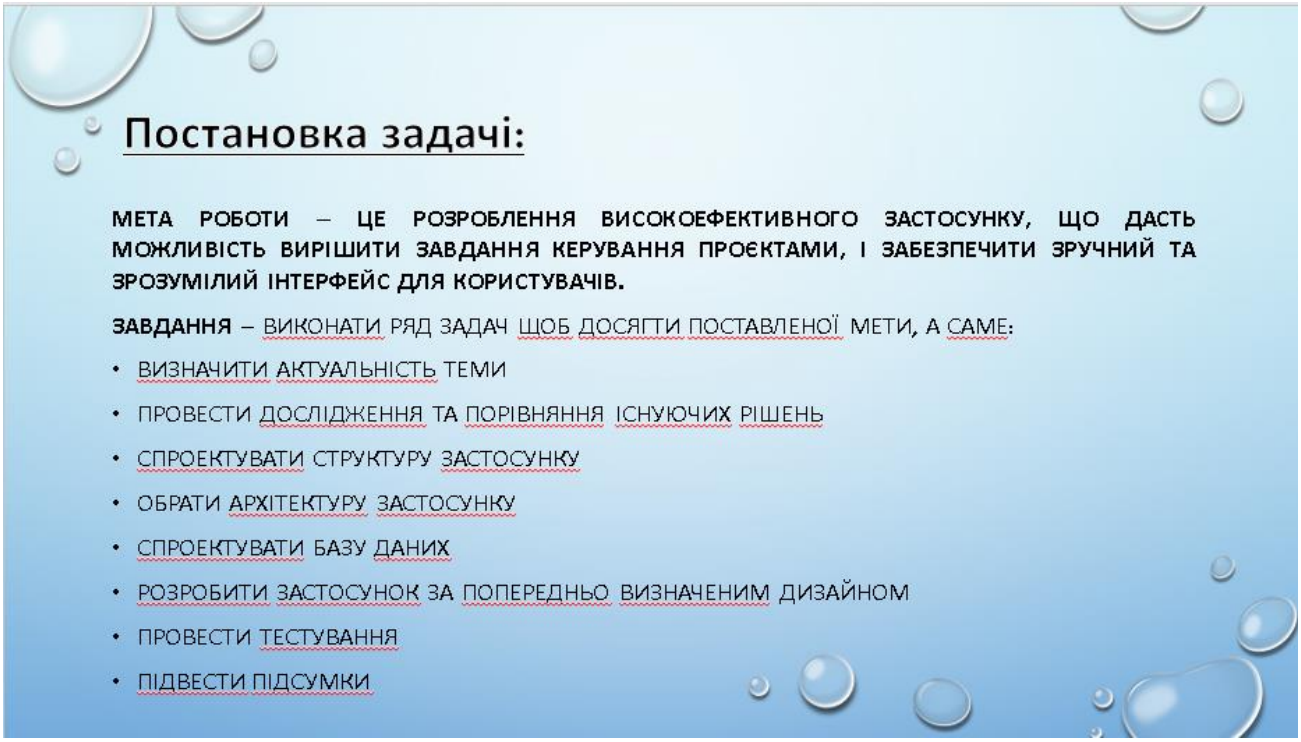
```
<?php global $conn; ?>
<style>
	#notif{
		display:none
	}
	.col-sm-4.project-item {
		background: #008aff24;
		padding: .5em;
		border-radius: 6px;
		border: 1px solid #0000ff59;
	}
	.col-sm-4.project-item:hover {
		background:#008aff8c;
	}
</style>
<div class="col-lg-12" id="project-field">
	<div class="panel panel-primary">
		<div class="panel-body">
			<?php include '../includes/db.php';
			$query = mysqli_query($conn,"SELECT * FROM projects where io =
'1' ");
			while($row = mysqli_fetch_assoc($query)){
				?>
				<div class="col-sm-4 project-item">
				<a href="index.php?page=project_detail&id=<?php echo
$row['project_id'] ?>&action=view details" style="color:black">
				<center><img src="../images/<?php echo $row["site_pic"]
?>" width="180px" height="230px" alt=""></center>
				<br>
				<center><label style="text-transform:capitalize"><?php
echo $row['project'] ?></label></center></a>
				</div>
			<?php } ?>
		</div>
	</div>
</div>
<div class="col-lg-4" id="notif">
	<div id="" style="">

		<?php include '../includes/db.php';
		$query1 = mysqli_query($conn,"SELECT * FROM projects where io = '1'
");
		while($row1 = mysqli_fetch_assoc($query1)){
			$d1= date("Ymd",strtotime($row1['deadline'].' -15 days'));
```

```
                  $d2= date("Ymd");
                  if($d2 >= $d1 && date("Ymd",strtotime($row1['deadline'])) >
$d2 ){
                  ?>
                  <a href="index.php?page=project_detail&id=<?php echo
$row1['project_id'] ?>&action=upcoming deadline" style="color:black">
                        <div class="panel panel-primary">
                              <div class="panel-heading">
                                    Дедлайн скоро...
                              </div>
                              <div class="panel-body">
                                    <center><p><b><?php echo
ucfirst($row1['project']) ?></b></p></center>
                                    <p><i>Дедлайн:</i><b><?php echo date("F d,
Y",strtotime($row1['deadline'])) ?></b></p>
                              </div>
                        </div>
                  </a>
                  <?php }elseif(date("Ymd",strtotime($row1['deadline'])) < $d2){
?>
                        <div class="panel panel-danger">
                              <div class="panel-heading">
                                    Дедлайн!!!
                              </div>
                              <div class="panel-body">
                                    <center><p><b><?php echo
ucfirst($row1['project']) ?></b></p></center>
                                    <p><i>Прострочено з:</i><b><?php echo
date("F d, Y",strtotime($row1['deadline'])) ?></b></p>
                              </div>
                        </div>
                  <?php }} ?>
      </div>
</div>
<script>
      if($('#notif .panel').length > 0){
            $('#project-field').removeClass('col-lg-12').addClass("col-md-8")
            $('#notif').show()
      }
</script>
```

ДОДАТОК В
(обов'язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ





Актуальність теми:

Потреба в ефективному управлінні: Практично кожна організація, що займається розробленням, потребує ведення нових та існуючих проєктів. Програми для керування проєктами стають невід'ємною частиною процесу.

Зростання обсягів та складності: З розвитком технологій і збільшенням конкуренції проєкти стають складнішими. Інструменти управління проєктами необхідні для забезпечення структури та контролю з метою полегшення процесу розроблення.

Постійна еволюція технологій: З появою нових технологій і методологій керування проєктами з'являється потреба в актуальних інструментах та рішеннях. Нові програмні застосунки забезпечують можливість швидко адаптуватися до змінних вимог та використовувати передові практики для досягнення успіху у проєктах.

Аналіз існуючих рішень:

ДЛЯ АНАЛІЗУ БУЛО ОБРАНО ТРИ ПОПУЛЯРНИХ ЗАСТОСУНКИ ДЛЯ КЕРУВАННЯ ПРОЄКТАМИ, А САМЕ: **JIRA, TRELLO** ТА **MS PROJECT.**

JIRA – ЦЕ ПОПУЛЯРНИЙ ІНСТРУМЕНТ ДЛЯ КЕРУВАННЯ ПРОЄКТАМИ, ЯКИЙ ВИКОРИСТОВУЮТЬ КОМАНДИ ЩОБ ВІДСТЕЖУВАТИ ЗАВДАННЯ, ПЛАНУВАТИ SPRINTИ, ВЕСТИ BACKLOG-СПИСКИ ТА СПІЛКУВАТИСЯ.

TRELLO – ЦЕ ВІДОМИЙ ІНСТРУМЕНТ ДЛЯ ВІЗУАЛЬНОГО КЕРУВАННЯ ПРОЄКТАМИ, ЯКИЙ ВИКОРИСТОВУЄТЬСЯ КОМАНДАМИ ДЛЯ СТРУКТУРУВАННЯ ЗАВДАНЬ, СПІЛЬНОГО ДОСТУПУ ДО ІНФОРМАЦІЇ ТА КОМУНІКАЦІЇ.

MS PROJECT – ЦЕ ВИСОКОПОТУЖНИЙ ІНСТРУМЕНТ ДЛЯ КЕРУВАННЯ ПРОЄКТАМИ, ЯКИЙ ВИКОРИСТОВУЄТЬСЯ КОМАНДАМИ ДЛЯ ПЛАНУВАННЯ, СТВОРЕННЯ GANTT-ДІАГРАМ, ВІДСТЕЖЕННЯ ЗАДАЧ, СКЛАДАННЯ ЗВІТІВ ТА БАГАТЬОХ ІНШИХ ЗАВДАНЬ.

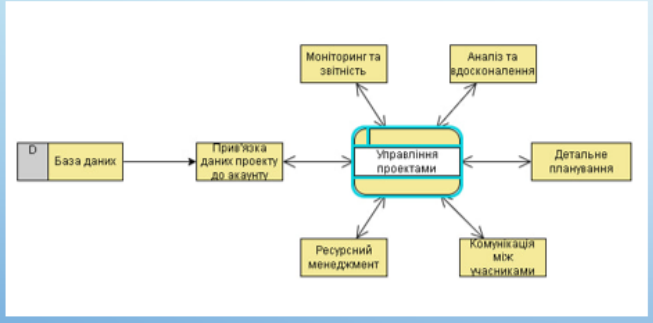
Аналіз існуючих рішень:

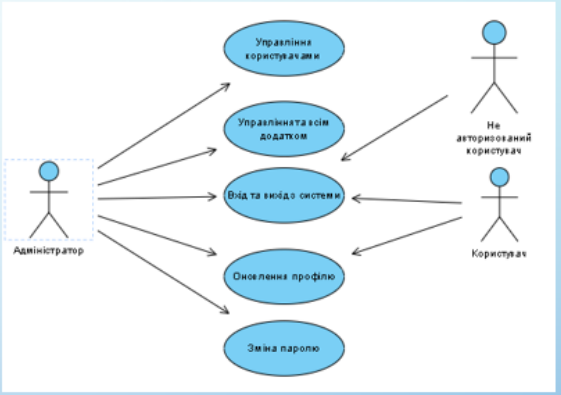
ПОРІВНЯЛЬНА ХАРАКТЕРИСТИКА ЗАСТОСУНКІВ JIRA, TRELLO ТА MS PROJECT:

Критерій	Jira	Trello	MS Project
Призначення	Управління проектами, відстеження завдань, agile-розробка	Візуальне управління проектами, командна робота	Професійне управління проектами, планування, Gantt-діаграми
Фірма-розробник	Atlassian	Trello Enterprises	Microsoft
Інтерфейс	Дошки, списки, картки, календарі	Дошки, картки, списки	Gantt-діаграми, таблиці, календарі
Переваги	Гнучкість, налаштування, agile-методології, інтеграції	Простота, візуалізація, командна робота	Потужні інструменти, планування, Gantt-діаграми.
Недоліки	Складність, висока ціна	Обмежена функціональність, не для складних проектів	Складність, висока ціна, не підходить для agile-команд

Структура застосунку:

НА РИСУНКАХ ЗООБРАЖЕНО ОСНОВНІ ХАРАКТЕРИСТИКИ ЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ ПРОЄКТАМИ, ЩО ПЕРЕХОДЯТЬ У ФУНКЦІОНАЛ ЗАСТОСУНКУ ТА ВЗАЄМОДІЮТЬ МІЖ СОБОЮ. ТАКОЖ НАВЕДЕНО ДІАГРАМУ ВАРІАНТІВ ВИКОРИСТАННЯ ДЛЯ РІЗНИХ ТИПІВ КОРИСТУВАЧІВ ЗАСТОСУНКУ

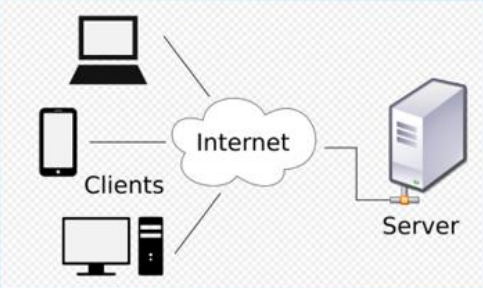




Архітектура застосунку:

ДЛЯ РОЗРОБКИ БУЛО ОБРАНО КЛІЄНТ-СЕРВЕРНУ АРХІТЕКТУРУ

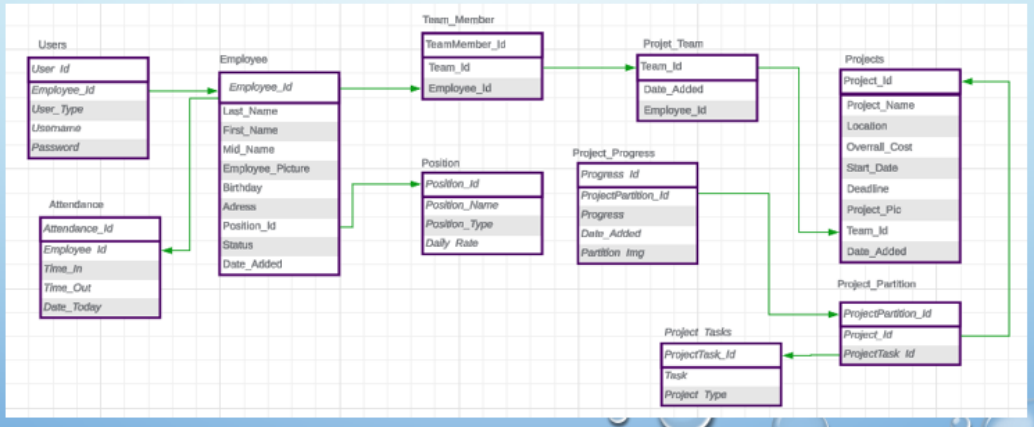
РОЗРОБКА ЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ ПРОЄКТАМИ НА ОСНОВІ КЛІЄНТ-СЕРВЕРНОЇ АРХІТЕКТУРИ ПЕРЕДБАЧАЄ ЧІТКЕ РОЗДІЛЕННЯ ВІДПОВІДАЛЬНОСТІ МІЖ КЛІЄНТОМ І СЕРВЕРОМ.



КЛІЄНТСЬКА ЧАСТИНА ВІДПОВІДАЄ ЗА ІНТЕРФЕЙС КОРИСТУВАЧА ТА ВІДОБРАЖЕННЯ ДАНИХ, ТОДІ ЯК СЕРВЕРНА ЧАСТИНА ВІДПОВІДАЄ ЗА БІЗНЕС-ЛОГІКУ, ЗБЕРІГАННЯ ТА ОБРОБКУ ДАНИХ. ЦЕЙ ПІДХІД ЗАБЕЗПЕЧУЄ ОПТИМАЛЬНИЙ ОБМІН ДАНИМИ, ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ТА МАСШТАБОВАНІСТЬ СИСТЕМИ, А ТАКОЖ ЗРУЧНІСТЬ У ВИКОРИСТАННІ ДЛЯ РОЗРОБНИКІВ.

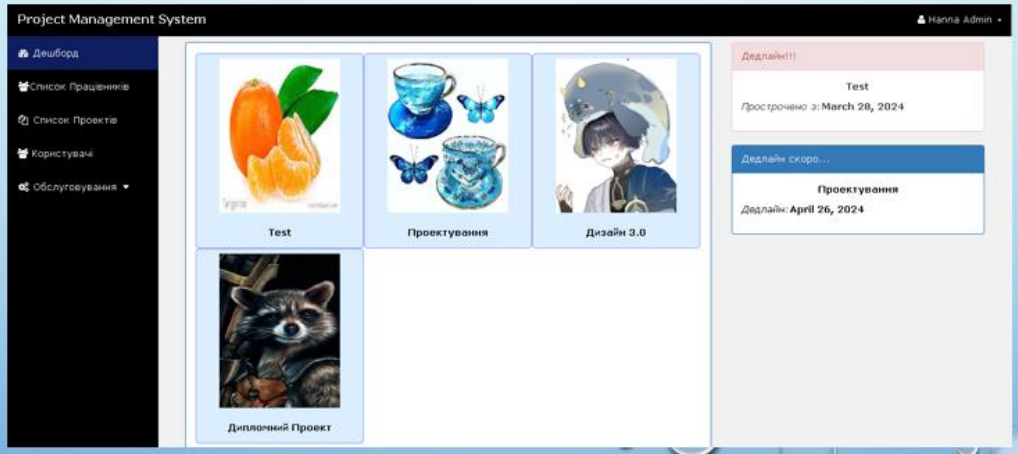
Проектування Бази Даних:

ЛОГІЧНА МОДЕЛЬ БАЗИ ДАНИХ З НЕОБХІДНИМИ ЗВ'ЯЗКАМИ ТА СТРУКТУРОЮ ПОБУДОВАНА НА ОСНОВІ СТРУКТУРИ НЕОБХІДНИХ ТАБЛИЦЬ ТА ЇХ ПОЛІВ:



Інтерфейс Застосунку:

ДИЗАЙН ЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ ПРОЄКТАМИ ВИКОНАНО В МІНІМАЛІСТИЧНОМУ СТИЛІ, ЩОБ НЕ ВІДВОЛІКАТИ КОРИСТУВАЧА ВІД РОБОТИ ТА В НЕЙТРАЛЬНИХ КОЛЬОРАХ:



Тестування:

ДЛЯ ПЕРЕВІРКИ ФУНКЦІОНАЛУ ЗАСТОСУНКУ НА ВІДПОВІДНІСТЬ ВСІМ ЗАЯВЛЕНИМ ВИМОГАМ ВИКОНАНО ФУНКЦІОНАЛЬНЕ ТЕСТУВАННЯ З ВИКОРИСТАННЯМ UNIT TEST

У РЕЗУЛЬТАТІ ФУНКЦІОНАЛЬНОГО ТЕСТУВАННЯ БУЛО ОПИСАНО МОЖЛИВІ СЦЕНАРІЇ РОБОТИ З ЗАСТОСУНКОМ, ЯКІ УСПІШНО ПРОТЕСТОВАНО

№	Функціональна вимога	Вхідні дані	Отриманий результат	Результат
1	Авторизація в системі	Логін і пароль	Користувач отримує доступ до системи, або повідомлення про помилкові дані	Правильно
2	Зміна паролю	Старий і новий пароль	Пароль успішно змінено	Правильно
3	Додання співробітника	Дані про нового співробітника	Додано нового співробітника	Правильно
4	Додання користувача	Дані про нового користувача	Додано нового користувача, його створено як співробітника	Правильно
5	Додання проекту	Дані про проект	Додано новий проект в список проектів та на Дешборд	Правильно

```
PHPUnit 9.5.10 by Sebastian Bergmann and contributors.

.                                                   1 / 1 (100%)

Time: 00:00.021, Memory: 8.00 MB

OK (1 test, 1 assertion)
```

РЕЗУЛЬТАТИ ТЕСТУВАННЯ РІВНЯ ДАНИХ ПІСЛЯ ЗАПУСКУ UNIT TEST

Участь у наукових конференціях:

РОЗРОБКА БУЛА ВИСВІТЛЕНА В ТЕЗАХ ВСЕУКРАЇНСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ «АКТУАЛЬНІ ПРОБЛЕМИ КОМПЮТЕРНИХ НАУК АПКН-2023

ПОСИЛАННЯ:

ЯКИМЧУК Г.Б., ПРАВОРСЬКА Н.І., ПРОГРАМНИЙ ЗАСТОСУНОК ДЛЯ КЕРУВАННЯ ПРОЕКТАМИ. ЗБІРНИК НАУКОВИХ ПРАЦЬ ЗА МАТЕРІАЛАМИ XV ВСЕУКРАЇНСЬКОЇ НАУКОВО-ПРАКТИЧНОЇ КОНФЕРЕНЦІЇ «АКТУАЛЬНІ ПРОБЛЕМИ КОМП'ЮТЕРНИХ НАУК АПКН-2023». ХМЕЛЬНИЦЬКИЙ. 2023. – 329-332 С.

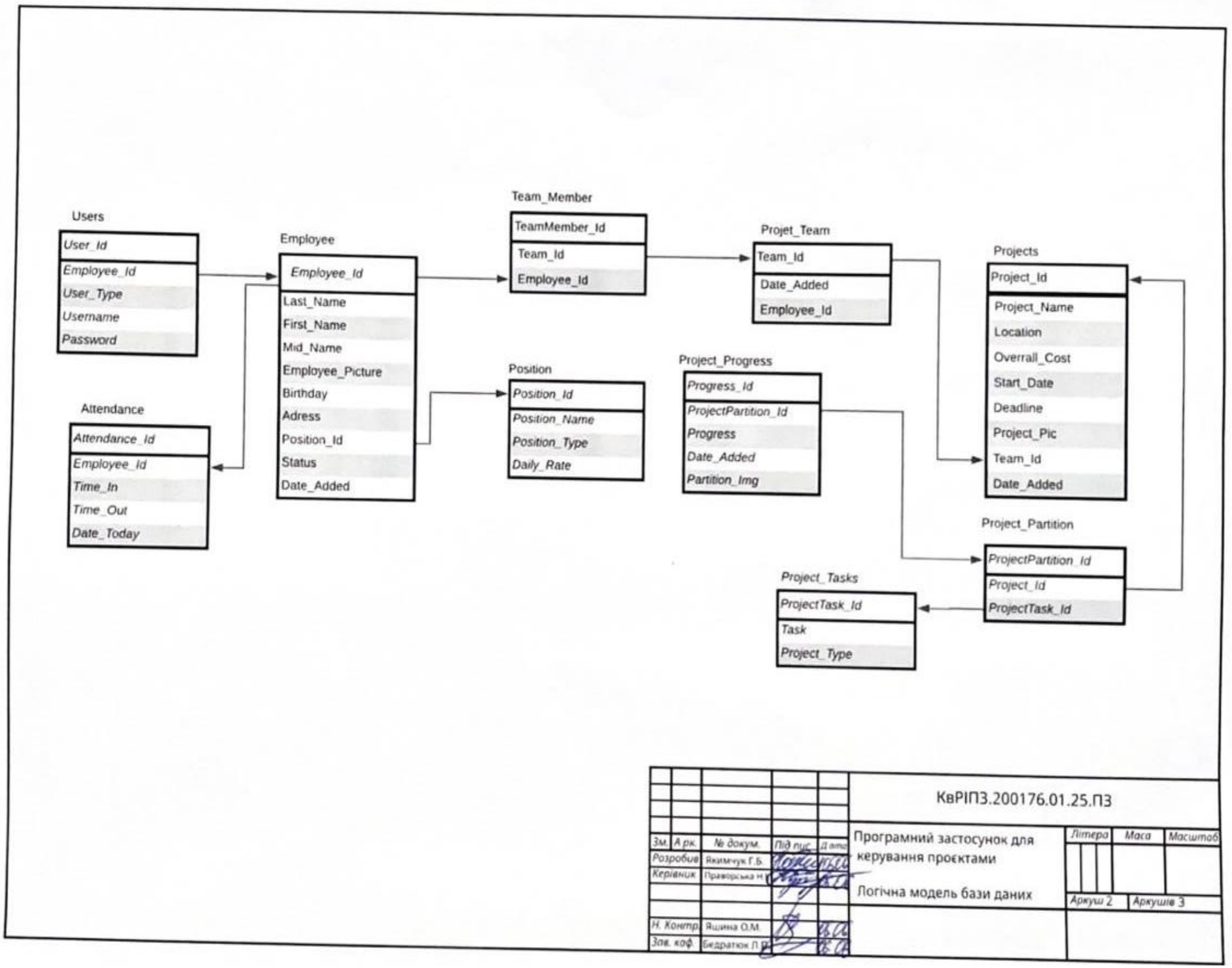
Висновки:

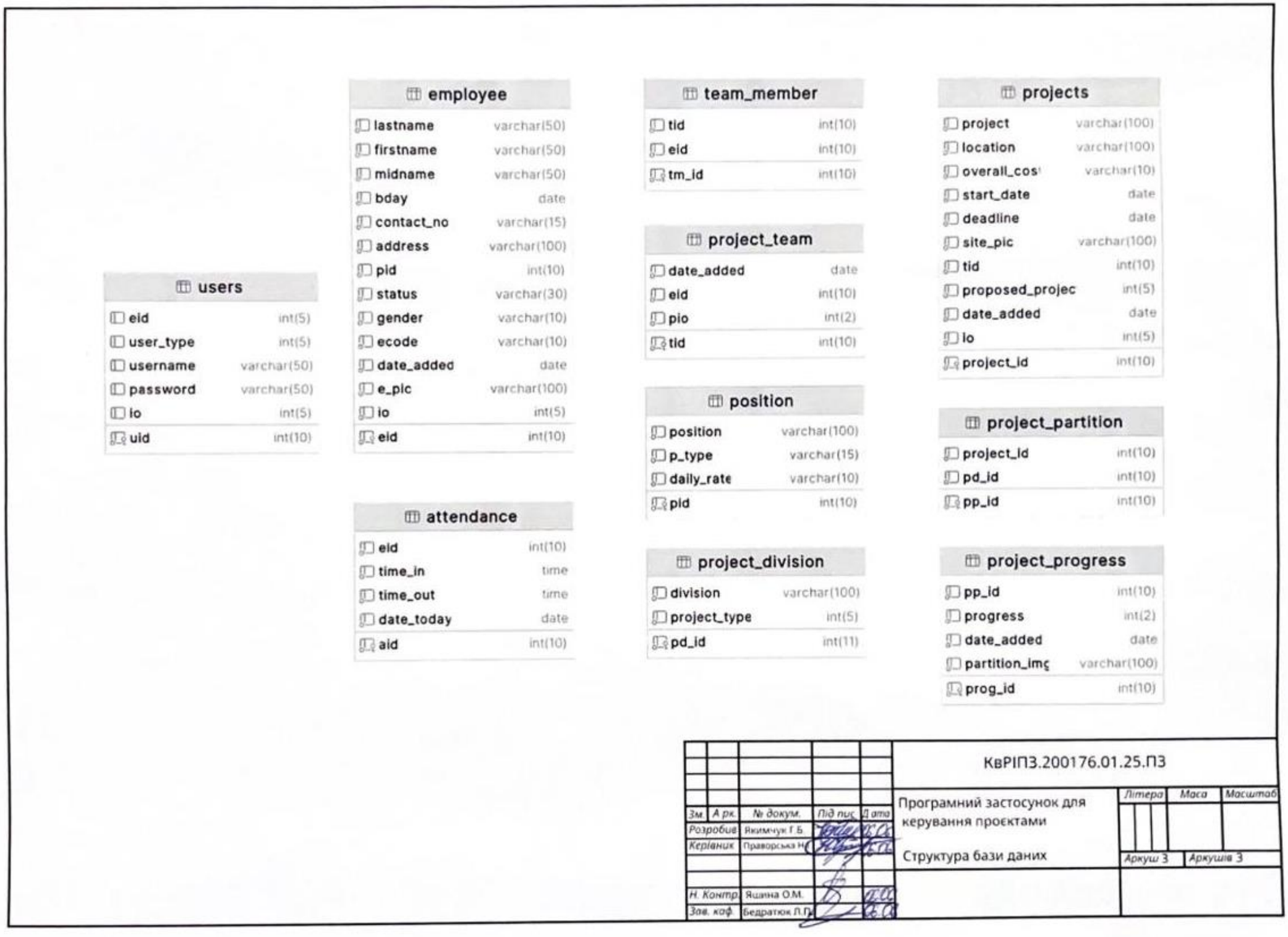
- ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ МАЛО НА МЕТІ РОЗКРИТИ ВСІ АСПЕКТИ ПРОЄКТУВАННЯ ТА РОЗРОБЛЕННЯ ЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ ПРОЄКТАМИ.
- ДЛЯ РЕАЛІЗАЦІЇ ПОСТАВЛЕНОЇ МЕТИ БУЛО ПРОВЕДЕНО ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКУ ЗАДАЧ, СПРОЄКТОВАНО ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, РОЗРОБЛЕНО І ПРОТЕСТОВАНО ЗАСТОСУНОК ТА ПІДВЕДЕНО ПІДСУМКИ.
- ПІДСУМОВУЮЧИ, РОЗРОБЛЕНИЙ ЗАСТОСУНОК ДЛЯ КЕРУВАННЯ ПРОЄКТАМИ МАЄ ДРУЖНІЙ ІНТЕРФЕЙС, ПІДТРИМУЄ РІЗНОМАНІТНІ ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ, ТАКІ ЯК: СТВОРЕННЯ ПРОЄКТІВ, ДОДАВАННЯ ЗАДАЧ, ВІДСТЕЖЕННЯ ПРОГРЕСУ, СТВОРЕННЯ КОМАНД І ТАК ДАЛІ.
- ЗАВДЯКИ ВІДПОВІДНИМ СИСТЕМНИМ ВИМОГАМ І РЕКОМЕНДАЦІЯМ ЩОДО ТЕХНІЧНИХ ХАРАКТЕРИСТИК, КОРИСТУВАЧІ МОЖУТЬ МАКСИМАЛЬНО ЕФЕКТИВНО ВИКОРИСТОВУВАТИ ЗАСТОСУНОК ДЛЯ ДОСЯГНЕННЯ СВОЇХ ЦІЛЕЙ У КЕРУВАННІ ПРОЄКТАМИ. РАЦІОНАЛЬНОЮ ТА ЕФЕКТИВНОЮ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ.
- ОТРИМАНІ РЕЗУЛЬТАТИ ПОКАЗАЛИ, ЩО РОЗРОБЛЕНИЙ ПРОГРАМНИЙ ЗАСТОСУНОК МОЖЕ УСПІШНО ВИКОНУВАТИ ПОСТАВЛЕНІ ЗАВДАННЯ ТА ФУНКЦІОНАЛЬНІ ВИМОГИ. ДОСЯГНУТІ РЕЗУЛЬТАТИ Є ОСНОВОЮ ДЛЯ ПОДАЛЬШОГО РОЗВИТКУ ТА ПОКРАЩЕННЯ ЗАСТОСУНКУ ДЛЯ КЕРУВАННЯ ПРОЄКТАМИ


Дякую за увагу!

ГРАФІЧНА ЧАСТИНА

					КвРІПЗ.200176.01.25.ПЗ			
					Програмний застосунок для керування проєктами	Літера	Маса	Масштаб
Зм.	А рк.	№ докум.	Під пис	Дата				
Розробив		Якимчук Г.Б.						
Керівник		Праворська Н.І.			Діаграма варіантів використання	Аркуш 1	Аркушів 3	
Н. Контр		Яцина О.М.						
Зав. каф.		Бедратюк Л.П.						


Users
User_Id
Employee_Id
User_Type
Username
Password
Employee
Employee_Id
Last_Name
First_Name
Mid_Name
Employee_Picture
Birthday
Adress
Position_Id
Status
Date_Added
Team_Member
TeamMember_Id
Team_Id
Employee_Id
Projet_Team
Team_Id
Date_Added
Employee_Id
Projects
Project_Id
Project_Name
Location
Overrall_Cost
Start_Date
Deadline
Project_Pic
Team_Id
Date_Added
Position
Position_Id
Position_Name
Position_Type
Daily_Rate
Project_Progress
Progress_Id
ProjectPartition_Id
Progress
Date_Added
Partition_Img
Attendance
Attendance_Id
Employee_Id
Time_In
Time_Out
Date_Today
Project_Partition
ProjectPartition_Id
Project_Id
ProjectTask_Id
Project_Tasks
ProjectTask_Id
Task
Project_Type
КвРІПЗ.200176.01.25.ПЗ
Програмний застосунок для керування проєктами
Логічна модель бази даних
Літера
Маса
Масштаб
Аркуш 2
Аркушів 3
Зм.
А рк.
№ докум.
Під пис.
Дата
Розробив
Якимчук Г.Б.
Керівник
Н. Контр.
Ящина О.М.
Зав. каф.
Бедратюк Л.Р.



КвРІПЗ.200176.01.25.ПЗ

Зм.	Арк.	№ докум.	Під пис.	Дата
Розробив		Якимчук Г.Б.		
Керівник		Праворська Н.		
Н. Контр.		Яшина О.М.		
Зав. каф.		Бедратюк Л.Г.		

Програмний застосунок для керування проєктами

Структура бази даних

Літера	Маса	Масштаб
Аркуш 3	Аркушів 3	

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти

Якимчук Г.Б.

Прізвище, ініціали

факультет ІТ, 4 курс, група ІПЗ-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті», згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлена. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщена та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та/або Anti-Plagiarism) і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

31.05.2024

дата

підпис

Wed Jun 05 10:55:31 EEST 2024, Форкун Юрій Вікторович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 15%**

ID: 128477 Назва: БКР Програмний застосунок для керування проектами Додано в БД: 2024-06-05 Автора: ЯКИМЧУК Ганна Керівники: ПРАВОРСЬКА Н.І., канд. пед. наук, доцент Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	84819	757	2255 (3%)	33 (4%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK
by Turnitin

Ім'я користувача:
ІПЗ

ID перевірки:
1016316942

Дата перевірки:
03.06.2024 23:59:39 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
04.06.2024 09:07:14 EEST

ID користувача:
100012953

Назва документа: БКР_Програмний Застосунок для керування проектами_Якимчук Г.Б_Приворська Н.І

Кількість сторінок: 68 Кількість слів: 12806 Кількість символів: 101890 Розмір файлу: 3.62 MB ID файлу: 1016114514

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

8.26%
Схожість

Найбільша схожість: 2.32% з джерелом з Бібліотеки (ID файлу: 1016114512)

6.85% Джерела з Інтернету	662	Сторінка 70
3.4% Джерела з Бібліотеки	220	Сторінка 74

0.16% Цитат

Цитати	2	Сторінка 75

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	8
Підозріле форматування	13 сторінок

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Якимчук Ганна Богданівна

Тема Програмний застосунок для керування проєктами

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 ; кількість сторінок записки 68

1.Короткий зміст пояснювальної записки та прийнятих рішень В рамках даної кваліфікаційної роботи було проведено глибоке дослідження та аналіз предметної області. На основі цього аналізу було вивчено існуючі на ринку рішення, їхні переваги та недоліки. Отримані знання дозволили сформулювати чіткі вимоги до програмного забезпечення, яке планувалося розробити. Для вибору технології та розроблення структури програмного застосунку оперувались визначеними вимогами. Для зручного користування програмним забезпеченням було розроблено прототип інтерфейсу користувача. Після завершення розробки було проведено ряд тестувань, спрямованих на перевірку відповідності програмного забезпечення вимогам та коректної роботи. За результатами тестувань можна зробити висновок, що розроблене програмне забезпечення готове до використання користувачами.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до усіх поставлених вимог та завдань.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі даної кваліфікаційної роботи показано актуальність теми роботи, поставлено мету та основні завдання. Перший розділ демонструє дослідження предметної області, розглядає та аналізує декілька найпопулярніших існуючих рішень з наведенням їхніх переваг та недоліків на основі чого виконано постановку задач та вимог. У другому розділі було проведено аналіз та порівняння декількох архітектурних шаблонів для майбутнього застосунку. Також було розроблено прототип інтерфейсу користувача та здійснено проєктування бази даних для правильного функціонування застосунку. В третьому розділі описано програмну реалізацію застосунку та бази даних з наведенням коду основних модулів. Після розроблення застосунку було проведено тестування для перевірки на відповідність попередньо визначеним вимогам.

4. Позитивні сторони роботи Проведено досить детальний аналіз предметної області з описом всіх необхідних вимого до програмного застосунку. Для аналізу готових рішень використано наочні зображення та порівняльна табличка. Детально описано декілька варіантів архітектурних шаблонів. При виборі засобів розробки було виконано детальний опис технологій з наведенням переваг та недоліків та принципами їхньої роботи. Розроблено привабливий та зручний дизайн кожного контекстного вікна застосунку.

5. Негативні сторони роботи Готовий застосунок можна покращити шляхом додавання нового та розширенням вже існуючого функціоналу, можна покращити механізм реєстрації, так як він є обмеженим на даний момент.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано у вигляді рисунків та діаграм та відповідають темі кваліфікаційної роботи. Пояснювальна записка виконана відповідно до вимог і стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Пояснювальна частина відзначається чіткою структурою та деталізацією, що ретельно описує всі аспекти відповідної тематики. Графічний матеріал наочно демонструє усі ключові моменти виконаної роботи.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі та відповідає всім поставленим вимогам та завданням і заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ Мартинюк Валерій Володимирович, доктор технічних наук, професор, зав. кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки ХНУ.

«6» 06 2024 р. _____
(підпис)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів:

Назва кваліфікаційної роботи: «Програмний застосунок для керування проєктами»
Автор: Якимчук Ганна Богданівна
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Праворська Наталія Іванівна, кандидат педагогічних наук, доцент
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені у роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою перевірки на плагіат Unicheck виявлено схожість з деякими документами у частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів, у рамках основних написів, у назвах публікацій переліку джерел посилання;

2) в якості запозичень системою Unicheck було зафіксовано деякі послідовності вихідного коду і посилання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) запозичення, виявлені в тексті роботи, є фрагментарними.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, складає 1,0%. Обсяг запозичень, визначений системою Unicheck виявлення збігів ідентичності/схожості, складає 8,26% і адресується до 662 джерел з Інтернету і 220 джерел з бібліотеки, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Дата 06.06.2024 р.

Завідувач кафедри Леонід БЕДРАТЮК

Гарант освітньої програми Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи Наталія ПРАВОРСЬКА