МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри, канд. техн. наук,
доцент _____Є. О. Давиденко
«___»____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

РОЗРОБКА CRM-СИСТЕМИ ДЛЯ УПРАВЛІННЯ ПЕРСОНАЛОМ З ПРОДАЖУ ТОВАРІВ

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.21910923

Студент _____Р. І. Шкіль
«___»_____2023 р.

Керівник ст. викладач кафедри ІПЗ _____С. Ю. Боровльова
«___»_____2023 р.

Консультант канд. техн. наук, доцент _____А. О. Алексєєва
«___»_____2023 р.

Миколаїв 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Зав. кафедри _____ Є. О. Давиденко

«_____» _____ 2023 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук

_____Шкіль Роману Ігоровичу_____

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

Розробка CRM-системи для управління персоналом з продажу товарів

Затверджена наказом по ЧНУ від «_17_» березня_2023__ р. № __60_____

2. Строк представлення кваліфікаційної роботи «____» _____ 20__ р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є розроблений та функціонуючий вебзастосунок для керування персоналом, його ефективністю та продуктивністю.

4. Перелік питань, що підлягають розробці:

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до ПЗ;
- визначення архітектури для проектування ПЗ;
- моделювання та проєктування ПЗ;
- реалізація ПЗ;
- тестування роботи ПЗ;
- проведення аналізу результатів розробки ПЗ.

5. Перелік графічних матеріалів

_Презентація до КРБ_____

6. Завдання до спеціальної частини

__Аналіз охорони праці на робочих місцях фахівців з інформаційних технологій__

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алексєєва А.О., канд. техн. наук, доцент (б. в. з.)	Кафедра екології Медичного інституту ЧНУ ім. Петра Могили	Спеціальна частина з охорони праці

Керівник роботи ст. викл. кафедри ІПЗ Боровльова Світлана Юріївна_____

(посада, прізвище, ім’я, по батькові)

_____

(підпис)

Завдання прийнято до виконання

_____Шкіль Роман Ігорович_____

(прізвище, ім’я, по батькові)

_____

(підпис)

Дата видачі завдання «_ _» _____ 2023р.

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: __Розробка CRM-системи для управління персоналом з продажу товарів_____

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КРБ	25.10.2022	30.10.2022	виконано
2.	Пошук інформації та літератури пов'язаної з темою кваліфікаційної роботи	01.11.2022	13.11.2022	виконано
3.	Формування календарного плану роботи	15.11.2022	28.11.2022	виконано
4.	Аналіз предметної області	04.12.2022	24.12.2022	виконано
5.	Розробка проєктних рішень	10.01.2023	07.02.2023	виконано
6.	Моделювання та конструювання ПЗ	18.02.2023	27.02.2023	виконано
7.	Реалізація розробленого ПЗ	03.03.2023	23.03.2023	виконано
9.	Розробка частини КРБ з охорони праці	28.03.2023	07.04.2023	виконано
10.	Відгук керівника КРБ	10.04.2023	12.04.2023	виконано
11.	Оформлення КРБ та презентації КРБ	15.04.2023	14.05.2023	виконано
12.	Попередній захист КРБ	06.06.2023	06.06.2023	виконано
13.	Рецензування КРБ	12.06.2023	14.06.2023	виконано
14.	Завершення оформлення КРБ та презентації	15.06.2023	18.06.2023	виконано
15.	Захист кваліфікаційної роботи	27.06.2023	27.06.2023	виконано

Розробив студент _____Шкіль Роман Ігорович_____ _____

(прізвище, ім'я, по батькові) (підпис)

«__» _____ 2023 р

Керівник роботи _ст. викладач каф. ІПЗ Боровльова Світлана Юріївна_ _____

(посада, прізвище, ім'я, по батькові) (підпис)

«__» _____ 2023 р

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Розробка CRM-системи для управління персоналом з продажу товарів»

Студент 409 групи: Шкіль Роман Ігорович

Керівник: ст. викладач Боровльова Світлана Юріївна

Дана робота присвячена розробці вебзастосунку для керування, моніторингу та контролю успішності персоналу.

Об’єктом є процеси управління персоналом, завданнями та їх ефективністю.

Предметом є – технології, необхідні для реалізації CRM-системи, що виконує всі передбачені функції.

Метою роботи є спрощення моніторингу успішності персоналу, покращення продуктивності та ефективності працівників, шляхом створення вебзастосунку для керування завданнями та управління персоналом.

Кваліфікаційна робота складається зі вступу, n розділів, висновків та переліку джерел посилань.

У вступі визначається актуальність теми, мета, предмет та об’єкт дослідження.

У першому розділі проведено аналіз існуючих вебзастосунків-аналогів. Також проведено формування та опис специфікації вимог вебзастосунку. У другому описано алгоритми роботи, що використовуються в системі. У третьому розділі проведено моделювання та розробку структури застосунку. У четвертому розділі проводиться огляд мов програмування, їх бібліотек та інших технологій, що використовуються для розробки вебзастосунку.

У висновках проводиться аналіз виконаних робіт та отриманих результатів.

Кваліфікаційна робота бакалавра викладена на 69 сторінок, містить 4 розділи, 37 ілюстрацій, 10 таблиць, 17 джерел в переліку посилань.

Ключові слова: *вебзастосунок, розробка вебзастосунку, CRM-система, менеджмент, керування персоналом.*

ABSTRACT

of the Bachelor's Thesis

«Development of a CRM system for the management of sales personnel»

Student: Roman Shkil

Supervisor: Senior Lecturer Svitlana Borovlova

This work is devoted to the development of a web application for managing, monitoring and controlling the success of personnel.

The object is personnel and tasks management processes and their effectiveness.

The subject is the technologies necessary for the implementation of a CRM system that performs all the intended functions.

The goal of the work is to simplify the monitoring and success of personnel, improve the productivity and efficiency of employees, by creating a web application for task management and personnel management.

The qualification work consists of an introduction, n sections, conclusions and a list of reference sources.

The introduction defines the relevance of the topic, the purpose, subject and object of the research.

In the first section, an analysis of existing analogue web applications was carried out. The specification of web application requirements was also formed and described. The second part describes the work algorithms used in the system. In the third section, the modeling and development of the application structure was carried out. The fourth chapter provides an overview of programming languages, their libraries, and other technologies used for web application development.

The conclusions analyze the work performed and the results obtained.

The bachelor's qualification work is laid out on 69 pages, contains 4 chapters, 37 illustrations, 10 tables, and 17 sources in the list of references.

Keywords: *web application, web application development, CRM system, management, personnel management.*

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – База Даних,

ПЗ – Програмне Забезпечення,

СКБД – Система Керування Базами Даних

CLI– Command Line Interface

CRM – Customer Relationship Management

CRUD – Create Read Update Delete

DMS – Document Management System

HR – Human Resources

HTML – HyperText Markup Language

MVC – Model View Controller

PHP – Hypertext Preprocessor

PM – Product Management

UC – User Case

UML – Unified Modeling Language

ВСТУП

CRM-системи для управління персоналом використовуються в різних сферах бізнесу та організацій, де важливим є ефективне управління та розвиток персоналу. Їх функціонал залежить від потреб споживача та може охоплювати великий спектр дій.

Дані системи використовуються в більшості великих корпоративних компаніях, кадрових агентствах, в галузях де важлива взаємодія з клієнтами. Вони дозволяють зберігати і керувати інформацією про співробітників, включаючи дані про зайнятість, навички, навчання та оцінки. CRM-системи допомагають покращити ефективність рекрутингу, зменшити час і зусилля, потрібні для пошуку кандидатів, і забезпечити кращу взаємодію з клієнтами. З їх допомогою можна підтримувати базу клієнтів, відстежувати комунікацію, керувати проектами та завданнями, планувати зустрічі та події, а також аналізувати дані про продажі та взаємодію з клієнтами.

Основні особливості системи, яка буде створена в ході виконання бакалаврської кваліфікаційної роботи:

- керування користувачами та їх ролями;
- створення та призначення завдань працівникам;
- управління доступом до певних функцій системи.

Тема залишається актуальною і важливою в сучасному бізнес-середовищі, оскільки CRM-системи стали популярним інструментом для ефективного управління відносинами з клієнтами, але їх потенціал також можна успішно використовувати для управління персоналом.

Основні тези, що підтверджують актуальність даної теми:

- оптимізація процесів управління персоналом;
- збільшення продуктивності персоналу;
- покращення комунікації та співпраці;
- збереження та управління інформацією про персонал;

- аналітика та звітність.

Об’єктом є процеси управління персоналом, завданнями та їх ефективністю.

Предметом є технології, необхідні для реалізації CRM-системи, що виконує всі передбачені функції.

Метою роботи є збільшення ефективності та спрощення управління персоналом компанії за допомогою розробки та впровадження CRM-системи.

Для досягнення поставленої мети необхідно вирішити наступні **завдання**:

- провести аналіз предметної сфери;
- визначити функціональні та нефункціональні вимоги;
- розробити дизайн і створити логотип для вебзастосунку;
- розробити зовнішній вигляд сайту вебзастосунку;
- виконати верстку сайту вебзастосунку, забезпечивши правильне розміщення контенту та його візуальне оформлення.

CRM-система буде реалізована з використанням таких технологій як: PHP, HTML, MySQL та ін. Система буде розроблена у вигляді вебзастосунку.

Отже, розробка CRM-системи для управління персоналом має значення як для наукової спільноти, де вивчаються технології та методи управління персоналом, так і для практичного застосування, де вона може бути використана для покращення процесів управління персоналом та досягнення кращих результатів в бізнесі.

1 АНАЛІЗ CRM-СИСТЕМ

Аналіз предметної області - це процес дослідження, розуміння та оцінки конкретної галузі знань, діяльності чи проблематики. Він передбачає докладне вивчення і збір інформації про дану область, її основні принципи, поняття, тенденції, проблеми та можливості.

Аналіз предметної області допомагає зрозуміти сутність і специфіку галузі, ідентифікувати ключові чинники, що впливають на неї, і розкрити потенціал для поліпшення. Цей процес дозволяє виявити проблеми, які потребують уваги, і знайти можливі шляхи для розв'язання цих проблем. Аналіз також допомагає визначити переваги, конкурентні переваги та можливості розвитку в даній області.

1.1 Опис предметної області

Для з'ясування специфіки CRM-системи для управління персоналом необхідно провести аналіз наступних аспектів: вимоги бізнесу, технологічні можливості, масштабованість та гнучкість, безпека та конфіденційність.

1.1.1 Вимоги бізнесу

Вимоги бізнесу до CRM-системи для управління персоналом можуть варіюватися в залежності від потреб та особливостей конкретної компанії або організації [1]. Однак, основні вимоги, які часто зустрічаються, включають:

- централізоване зберігання даних, включаючи дані про профіль, контактну інформацію, резюме, досвід роботи, освіту, навички, навчання та оцінки тощо;
- управління процесами персоналу, в тому числі і найм, звільнення, перегляд та оцінку продуктивності, планування навчання та розвитку, адміністрування відпусток та відсутності співробітників, а також управління кар'єрним ростом;

- рекрутинг та підбір персоналу, а саме збір резюме, автоматизований відбір кандидатів за заданими критеріями, планування та організацію співбесід, а також ведення бази даних потенційних кандидатів;
- аналітика та звітність, включаючи показники продуктивності, затримки в роботі, підсумки навчання та розвитку, а також інші показники, які дозволяють оцінити ефективність та внести відповідні зміни у стратегію управління персоналом;
- безпека даних, CRM-система повинна мати вбудовані механізми захисту конфіденційної інформації про персонал, такі як обмеження доступу до даних, шифрування, резервне копіювання та відновлення даних, аудит доступу та інші заходи, щоб запобігти несанкціонованому доступу та втраті даних;
- інтеграція з іншими системами, такими як системи управління відносинами з клієнтами (CRM), системи управління проектами (PM), системи управління документами (DMS) тощо.

В створюваній системі централізоване зберігання даних, управління процесами персоналу та безпека даних. Саме ці пункти були вибрані неспроста, це найпоширеніші вимоги для CMR-системи.

1.1.2 Технологічні можливості

Аналіз технологічних можливостей CRM-системи для управління персоналом включає в себе декілька складових [2].

1. Тип архітектури яку використовує CRM-система. Це може бути клієнт-серверна архітектура, хмарна архітектура або гібридна архітектура.
2. Мови програмування та фреймворки, на яких базується CRM-система. Це можуть бути мови програмування, такі як Java, C#, PHP, Python або JavaScript, та фреймворки, такі як Laravel, Django, ASP.NET або React.
3. Бази даних, які використовуються в CRM-системі. Це можуть бути реляційні бази даних, такі як MySQL, PostgreSQL або Oracle, або нереляційні бази даних, такі як MongoDB або Cassandra.

4. Інтерфейс користувача, які надає CRM-система. Це можуть бути вебінтерфейси, мобільні застосунки, настільні застосунки або комбінація різних типів інтерфейсів.

У системі буде використано: клієнт-серверну архітектуру, мови програмування PHP та JavaScript, базу даних MySQL. Це все буде розроблено у вигляді вебзастосунку. Таким чином буде забезпечено ефективну обробку даних, безпеку, масштабованість та зручний доступ для користувачів.

1.1.3 Масштабованість та гнучкість

Масштабованість та гнучкість є важливими характеристиками CRM-системи для управління персоналом.

Добре розроблена CRM-система повинна мати можливість масштабуватись під зростання обсягу даних та кількості користувачів. Це означає, що система повинна бути готова збільшувати свої обчислювальні ресурси та інфраструктуру для забезпечення швидкої та ефективної обробки даних. Наприклад, система повинна мати можливість додавати нові сервери або розподіляти навантаження між серверами, щоб забезпечити високу продуктивність при зростанні обсягу даних та кількості користувачів [3].

Тому, для даної системи буде передбачено сервер з можливістю розширення, без повного переносу даних.

CRM-система для управління персоналом повинна бути гнучкою і адаптивною до потреб користувачів. Це означає, що система повинна мати можливість налаштовуватись та розширюватись відповідно до вимог та процесів бізнесу. Наприклад, система повинна дозволяти налаштовувати різні типи даних, політики доступу, ролі користувачів та інші функціональні параметри відповідно до потреб організації. Вона також повинна бути здатною інтегруватись з іншими системами та застосунками, що використовуються в компанії [4].

Гнучкість системи буде реалізовано за допомогою відокремлення сутностей в відповідні моделі та їх поведінки в контролерах, що дозволяє в будь-який момент

розширити чи налаштувати поведінку сутності. Також буде впроваджено систему ролей з правами доступу до даних

Обидва ці аспекти дозволяють CRM-системі для управління персоналом адаптуватись до змінних потреб бізнесу і забезпечувати оптимальну продуктивність та ефективність. Вони дозволяють системі розширюватись та пристосовуватись до зростаючих обсягів даних та змін в організаційних потребах.

1.1.4 Безпека та конфіденційність

Безпека та конфіденційність є критичними аспектами для CRM-системи для управління персоналом, оскільки вона містить конфіденційну інформацію про співробітників та клієнтів [5]. Є важливі моменти, що потребують уваги.

1. Аутентифікація та авторизація – система повинна мати механізми аутентифікації, які забезпечують ідентифікацію користувачів і перевірку їх прав доступу. Це допоможе запобігти несанкціонованому доступу до системи та інформації.

2. Захист даних – дані про персонал та клієнтів повинні бути захищені від несанкціонованого доступу, зміни або видалення. Важливо застосовувати методи шифрування для зберігання та передачі даних, а також забезпечити резервне копіювання даних для запобігання втрати інформації.

3. Рівень доступу – система повинна мати гнучкі механізми керування рівнем доступу, які дозволяють встановлювати права доступу до різних функцій та даних відповідно до ролей та обов'язків користувачів. Це забезпечить обмеження доступу до конфіденційної інформації лише для авторизованих користувачів.

4. Захист від зовнішніх загроз – CRM-система повинна мати механізми захисту від зовнішніх загроз, таких як злам, віруси, шкідливі програми тощо. Це може включати використання файрволів, антивірусного програмного забезпечення, регулярні оновлення та моніторинг системи на предмет потенційних загроз.

5. Аудит – важливо мати механізми аудиту, щоб вести записи про дії користувачів, доступ до системи та зміни в інформації. Це допоможе виявити можливі порушення безпеки та встановити відповідальних осіб.

Важливо провести ретельний аналіз цих аспектів безпеки та конфіденційності, щоб забезпечити захищеність інформації та виконання вимог безпеки у CRM-системі для управління персоналом.

В даній системі буде виконано більшу частину з вище перелічених пунктів безпеки та конфіденційності, завдяки чому користувачі зможуть тримати свої дані в безпеці.

1.2 Аналіз аналогічних систем

Для проведення аналізу аналогів обрано наступні вебзастосунки: BambooHR (табл. 1.1), Workday (табл. 1.2), Zoho People (табл. 1.3).

1.2.1 BambooHR

Хмарний сервіс BambooHR – це система для фахівців з персоналу та кадрів, що вирішує основні завдання управління HR-даними у малих та середніх підприємствах [6].

Таблиця 1.1 – Опис BambooHR

Назва	BambooHR
Розробник	Bamboo HR LLC
Архітектура	Web application
Мова реалізації	Salesforce, Zendesk, core-js, HSTS, Google Workspace
Функції	1. Ведення персональних досьє; 2. Відстеження часу роботи; 3. Керування відпустками; 4. Аналіз інформації про персонал.

Продовження таблиці 1.1

Переваги	1. Зручний інтерфейс; 2. Широкий спектр інструментів; 3. Можливість інтеграції з іншими програмними засобами.
Недоліки	1. Обмеження настроювання. 2. Відсутність деяких функцій.
Вебсайт	https://www.bamboohr.com/

Отже, BambooHR представляє собою CRM-систему для управління персоналом, яка надає інструменти для ефективного управління кадрами в організації. Основними функціональними можливостями якої є зберігання та керування даними про співробітників, рекрутинг та працевлаштування, відстеження робочого часу, навчання та розвиток персоналу, а також аналітика та звітність.

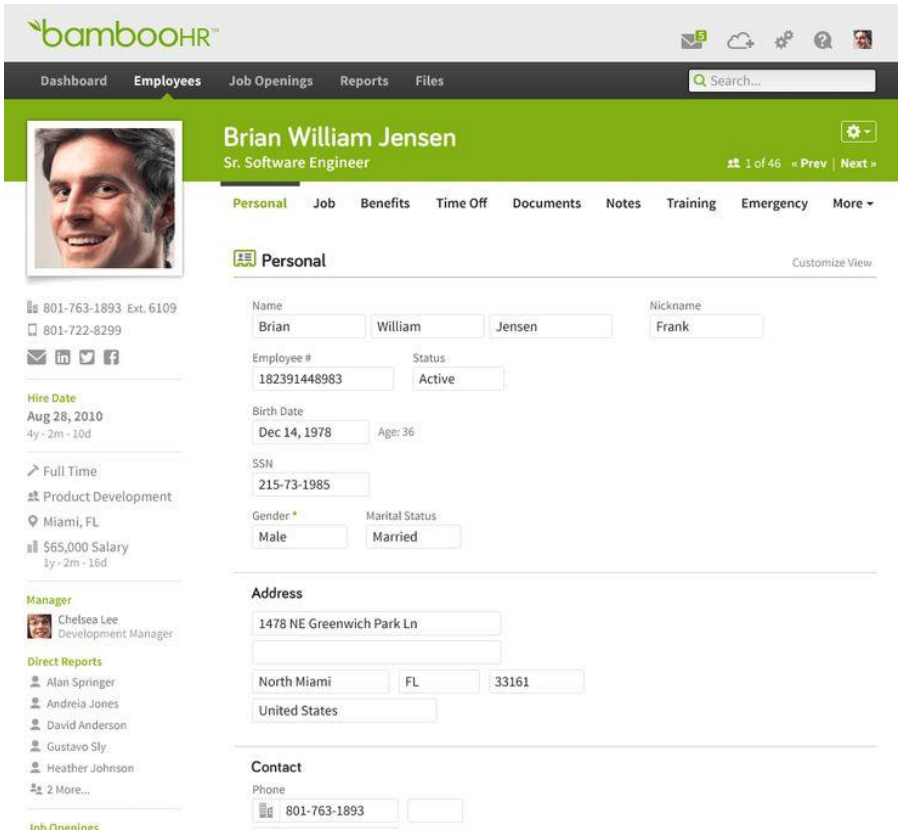

Рисунок 1.1 – Інтерфейс BambooHR

1.2.2 Workday

Workday - це хмарна платформа управління людськими ресурсами (HR) та фінансами для підприємств. Вона пропонує комплексні рішення для автоматизації багатьох аспектів бізнесу, включаючи HR-управління, управління трудовими відносинами, оплату праці, облік кадрів, фінансове планування, аналітику та багато іншого [7].

Workday надає централізований доступ до даних про співробітників, що дозволяє керівникам та HR-фахівцям ефективно керувати всіма аспектами HR-процесів.

Таблиця 1.2 – Опис Workday

Назва	Workday
Розробник	Workday Inc.
Архітектура	Web application
Мова реалізації	Java, Apache HTTP Server, HSTS, jQuery
Функції	1. Керування кадрами; 2. Ведення списків співробітників; 3. Планування робочого часу; 4. Аналітика та звітність.
Переваги	1. Комплексний функціонал; 2. Автоматизація процесів управління персоналом; 3. Моніторинг та аналіз даних.
Недоліки	1. Високі витрати; 2. Складність інтеграції з існуючими системами.
Вебсайт	https://www.workday.com/

Workday - це CRM-система для управління персоналом, яка надає комплексні рішення для організацій у галузі управління персоналом, фінансів, розробки та інших бізнес-процесів. Workday має широкий функціонал, який охоплює багато аспектів управління організацією.

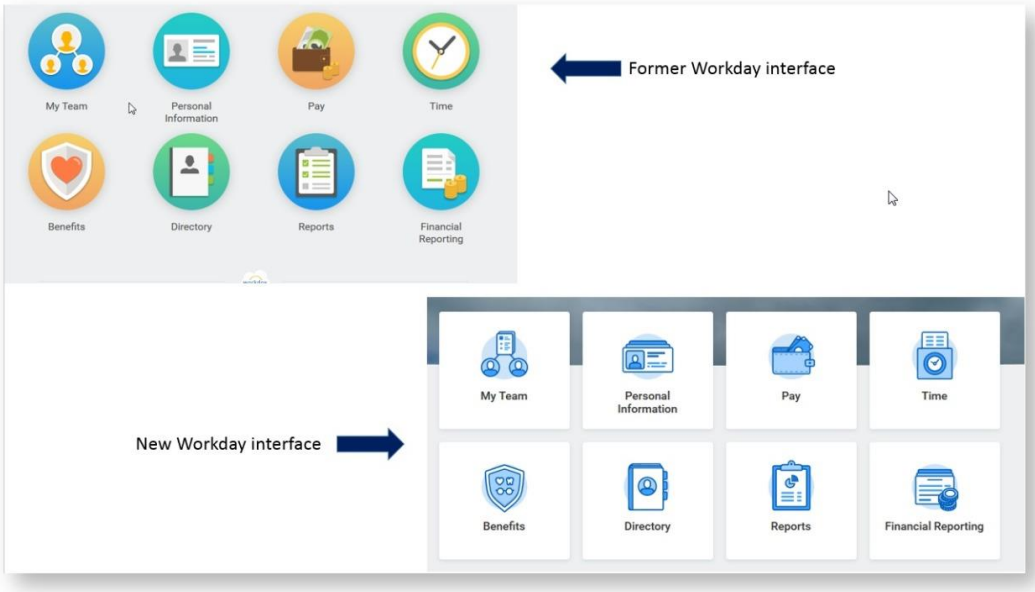

Рисунок 1.2 – Оновлений інтерфейс Workday(справа знизу)

1.2.3 Zoho People

Zoho People - це хмарне програмне забезпечення (SaaS), розроблене компанією Zoho, яке надає рішення для управління людськими ресурсами (HR) в організаціях. Воно дозволяє автоматизувати багато аспектів HR-процесів, включаючи управління персоналом, зберігання даних про співробітників, процеси найму та звільнення, відстеження робочого часу, управління відпустками, оцінку працівників та багато іншого [8].

Zoho People пропонує різноманітні функції, такі як управління профілем співробітників, розклад роботи, зарплатні відомості, витрати на персонал, навчання та розвиток, аналітику та звітність.

Таблиця 1.3 – Опис Zoho People

Назва	Zoho People
Розробник	Zoho Corporation Pvt. Ltd.
Архітектура	Web application
Мова реалізації	CMS Drupal, PHP, Zoho, HSTS, jQuery, Zepto

Продовження таблиці 1.3

Функції	1. Ведення персональних досьє. 2. Керування навчанням та розвитком. 3. Оцінка продуктивності. 4. Керування відпустками.
Переваги	1. Широкий набір інструментів; 2. Можливість інтеграції з іншими системами;
Недоліки	1. Обмеження на кількість користувачів. 2. Непривабливий та неінтуїтивний інтерфейс користувача.
Вебсайт	https://www.zoho.com/people/

Отже, Zoho People – це CRM-система для управління персоналом, яка пропонує широкий спектр функціональних можливостей. Серед її переваг можна виділити комплексність функцій, включаючи рекрутинг, кадровий облік, навчання та оцінку праці. Вона також має зручний інтерфейс та легку інтеграцію з іншими продуктами Zoho та сторонніми застосунками. Додатковою перевагою є мобільний доступ та можливість масштабування системи під зростаючі потреби організації. Однак, серед недоліків можна відзначити обмежену адаптованість, що може потребувати додаткових налаштувань, а також вартість використання, особливо для великих організацій.

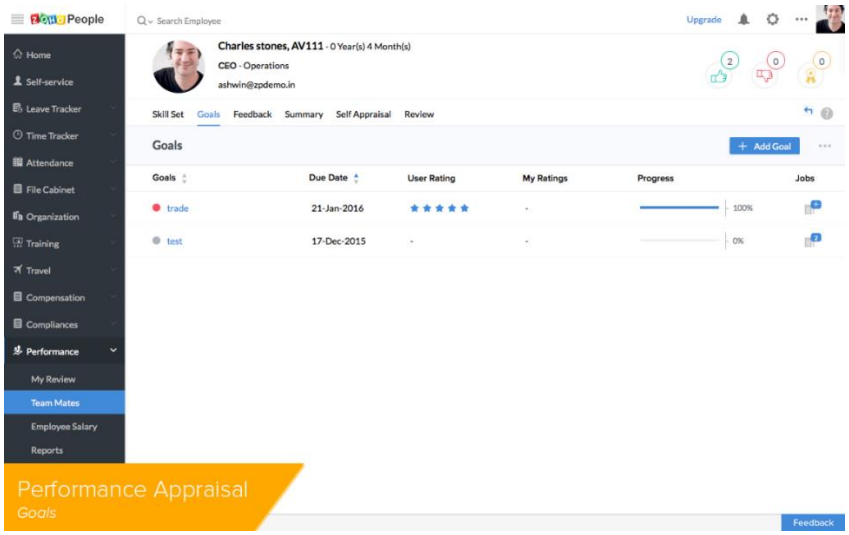

Рисунок 1.3 – Інтерфейс Zoho People

Тож у розглянутих аналогів чітко видно переваги та недоліки, а також можна визначити ті аспекти, які можна вдосконалити в розроблюваній CRM-системі. Порівняння з аналогами допомогло з'ясувати можливі шляхи для подальшого розвитку та покращення системи.

1.3 Специфікація вимог

1.3.1 Опис функціоналу системи

Вебзастосунок CRM-системи для управління персоналу з продажу, який розробляється в ході бакалаврської роботи має такий функціонал:

- реєстрація та авторизація користувачів в системі;
- здійснення CRUD-операцій над даними про *сторінки*;
- здійснення CRUD-операцій над даними про *завдання*;
- зміна статусу виконання активного завдання;
- призначення тегу для завдання та фільтрація за тегом;
- здійснення CRUD-операцій над даними про *ролі* користувачів системи;
- здійснення CRUD-операцій над даними про *користувачів*;
- розмежування прав доступу в залежності від ролі користувача.

1.3.2 Опис ролей користувачів в системі

1. Гість – не має доступу до системи.
2. Працівник – може переглядати, виконувати та змінювати статус прогресу завдання.
3. Керівник – може керувати завданнями.
4. Адміністратор – має найбільше прав в мережі.

Висновки до розділу 1

В ході написання першого розділу була детально розглянута та проаналізована предметна область. Описано вимоги бізнесу, технологічні можливості та інші моменти, що будуть використані в системі. За допомогою аналізу даних аспектів, визначено специфічні функціональні вимоги. Детально розглянуто вимоги до безпеки та конфіденційності в системах такого типу, на основі яких буде побудовано майбутній вебзастосунок.

Також проаналізовано аналоги даного продукту, що користуються популярністю на сучасному ринку. В кожного аналогу виявлено переваги та недоліки, які будуть взяті до уваги при побудові системи.

В останній частині розділу описано функціональні вимоги до системи. Інформація з першого розділу є основою для алгоритмів, що потрібні для коректної роботи системи.

2 АЛГОРИТМИ РОБОТИ CRM-СИСТЕМ

В цьому розділі будуть розглянуті та описані алгоритми, необхідні для функціонування системи. Для зручності подальшої роботи необхідно дослідити наступне: обробка та збереження даних клієнтів та їх алгоритми, алгоритм керування доступом, алгоритм управління замовленнями.

2.1 Обробка та збереження даних клієнтів

2.1.1 Визначення даних клієнта

Дані клієнта – це інформація про клієнта, яку людина зібрала як основна сторона (наприклад, дані з ваших вебсайтів, застосунків, звичайних магазинів та інші відомості, якими клієнти поділилися безпосередньо з нею). Це будь-яка інформація про людину, яку можна ідентифікувати. Ім’я, телефон та електронна пошта, вік, місце роботи, дохід, сімейний стан тощо дають змогу визначити конкретну особу.

Таку інформацію в більшості випадків зберігають у базах даних.

Сучасні бази даних структуруються у сховищах за певною ознакою, пов'язані одна з одною та використовуються для задоволення тих чи інших потреб їх володільців. Таким чином бази даних мають дві функції – збереження інформації та управління нею.

2.1.2 Алгоритми обробки та збереження даних клієнта

Алгоритми обробки та збереження даних клієнта в CRM-системі використовуються для ефективного управління і збереження інформації про клієнтів. Ці алгоритми дозволяють виконувати операції, такі як додавання нового клієнта до системи, оновлення даних існуючого клієнта, видалення клієнта, пошук і відображення інформації про клієнта. Вони допомагають зберігати і організовувати дані клієнтів для забезпечення кращого обслуговування, аналізу та взаємодії з клієнтами [9].

Для цього в розроблюваній CRM-системі потрібно реалізувати наступні алгоритми обробки та збереження даних.

Додавання клієнта:

- користувач вводить дані клієнта, такі як ім'я, контактні дані тощо;
- перевіряється коректність введених даних;
- створюється новий об'єкт клієнта з введеними даними;
- об'єкт клієнта зберігається в базі даних.

Оновлення даних клієнта:

- користувач обирає клієнта, чиї дані потрібно оновити;
- завантажуються поточні дані клієнта з бази даних;
- користувач вносить необхідні зміни до даних клієнта;
- змінені дані зберігаються в базі даних або оновлюють існуючий запис.

Видалення клієнта:

- користувач обирає клієнта, якого потрібно видалити;
- перевіряється наявність залежностей або обмежень, що вимагають видалення клієнта;
- якщо немає обмежень, клієнт видаляється з бази даних або позначається як видалений;

Пошук клієнта:

- користувач вводить параметри пошуку, такі як ім'я, ідентифікатор або інші характеристики клієнта;
- виконується пошук в базі даних за заданими параметрами;
- знайдені результати відображаються користувачу.

2.2 Контроль доступу

2.2.1 Визначення контролю доступу

Контроль доступу є важливою складовою будь-якої CRM-системи і використовується для забезпечення безпеки та обмеження доступу до певної

інформації. Він визначає, які користувачі мають право отримати доступ до певних функцій, даних або ресурсів в системі, а які - ні.

Контроль доступу базується на принципі надання прав доступу користувачам на основі їх ролей, визначених в системі. Кожна роль має свої привілеї та обмеження, які визначають, до яких даних та функціональностей має доступ користувач. Наприклад, адміністратор системи може мати повний доступ до всіх функцій і даних, тоді як звичайний користувач може мати обмежений доступ лише до певних модулів або інформації.

Контроль доступу включає такі компоненти:

1. Аутентифікація – перевірка ідентифікаційних даних користувача (наприклад, логін та пароль) для підтвердження його ідентичності перед наданням доступу до системи.

2. Авторизація – визначення, до яких ресурсів, функціональностей та даних має доступ користувач на основі його ролі та привілеїв.

3. Аудит – ведення журналу дій користувачів, що включає інформацію про вхід у систему, взаємодію з даними та виконання операцій. Це дозволяє відстежувати та аналізувати активність користувачів, виявляти потенційні загрози безпеці та виявляти випадки порушення правил доступу.

4. Шифрування – застосування методів шифрування для захисту конфіденційної інформації під час її передачі між користувачем та CRM-системою.

Управління правами доступу дозволяє адміністраторам CRM-системи налаштовувати, керувати та змінювати права доступу для окремих користувачів. Вони можуть надавати або забирати права доступу до певних функцій, модулів або даних в залежності від потреб і ролі користувача.

Правильний контроль доступу дозволяє забезпечити безпеку даних, запобігти несанкціонованому доступу та зловживанням, а також зберегти конфіденційність інформації. Це важливий аспект в управлінні CRM-системою, оскільки дозволяє точно налаштувати доступ до різних ресурсів відповідно до

потреб та ролей користувачів, забезпечуючи ефективну та безпечну роботу з системою.

2.2.2 Алгоритм керування доступом

Алгоритм керування доступом в розроблюваній системі включає в себе:

1. Ідентифікація та аутентифікація користувача – користувач, який намагається отримати доступ до системи, повинен пройти процедуру ідентифікації, підтвердивши свою особу.

2. Перевірка прав доступу – після успішної аутентифікації система перевіряє права доступу користувача. Це включає перевірку його ролі, привілеїв та обмежень, які відповідають його профілю користувача.

3. Визначення дозволених дій – на основі перевірки прав доступу система визначає, які дії можуть бути виконані користувачем. Це включає перелік дозволених операцій, таких як перегляд, редагування, створення, видалення завдань тощо.

4. Управління ролями та привілеями – алгоритм включає можливість налаштування ролей, привілеїв та рівнів доступу для кожного користувача або групи користувачів. Це дозволяє адміністраторам системи керувати тим, до яких функціональностей та даних мають доступ користувачі.

5. Керування змінами – при зміні ролей, прав доступу або інших параметрів, алгоритм включає процес керування змінами, щоб забезпечити синхронізацію та оновлення прав доступу та привілеїв.

6. Зміна виконавця та перерозподіл завдань – адміністратори чи користувачі з відповідними правами мають можливість змінити виконавця завдання, алгоритм включає процес перерозподілу завдань між користувачами відповідно до їхньої ролі та доступу.

7. Збереження та реєстрація змін – після здійснення змін виконавця, статусу виконання, важливості тощо, алгоритм забезпечує збереження цих змін та реєстрацію в системі, щоб забезпечити їх доступність та стеження за ними.

2.3 Керування завданнями

2.3.1 Поняття управління завданнями

Керування завданнями - це процес планування, організації та контролю роботи над завданнями з метою досягнення поставлених цілей. В сучасному бізнес-середовищі, де ефективне управління часом та ресурсами є критично важливим, CRM-системи надають потужні інструменти для керування завданнями та їх виконання.

Одним з ключових аспектів керування завданнями є створення завдань і призначення їх виконавцям. Користувачі CRM-системи можуть створювати нові завдання, вказуючи їх деталі. Після створення завдання воно може бути призначене конкретному користувачеві або команді для виконання.

Крім того, керування завданнями включає в себе моніторинг та контроль за ходом виконання завдань. Користувачі можуть встановлювати статуси для завдань, що дозволяє відстежувати прогрес виконання.

Для забезпечення ефективного керування завданнями, CRM-системи надають можливості зміни виконавця, пріоритету, статусу завершеності та інших параметрів. Це дає користувачам гнучкість та контроль над процесом виконання завдань. Крім того, система може надавати повідомлення та нагадування про невиконані завдання.

2.3.2 Алгоритм керування завданням

Алгоритм надає основний опис процесу керування завданнями в CRM-системі та складається з наступних кроків:

Крок 1. Створення завдання:

- користувач створює нове завдання, вказуючи його назву, категорію та дедлайн;
- завдання отримує початковий статус "Нове".

Крок 2. Призначення виконавця:

- адміністратор або відповідальний користувач призначає виконавця для завдання;
- завдання отримує статус "В прогресі" і передається виконавцю.

Крок 3. Виконання завдання:

- виконавець працює над завданням і оновлює його статус, якщо потрібно;
- можливі статуси завдання: "В процесі", "На утриманні", "Відмінене".

Крок 4. Підтвердження завдання:

- після завершення виконання завдання виконавець підтверджує його;
- завдання отримує статус "Завершене" і переходить до наступного кроку.

Крок 5. Завершення завдання:

- завдання може бути остаточно закрито або відновлено для подальшої роботи, в залежності від результатів оцінки;
- якщо завдання успішно виконане, воно закривається;
- якщо завдання частково виконане або невдале, воно може бути відкрите для подальших корекцій або передане на повторне виконання.

Крок 6. Зміна виконавця, статусу та інших параметрів:

- користувач з відповідною роллю доступу або адміністратор має можливість змінювати виконавця, статус завдання, важливість, прогнозовану дату завершення та інші параметри, що дозволяють дати детальнішу інформацію щодо завдання;
- зміни відображаються в системі і впливають на подальшу обробку та відстеження завдання.

Отже, керування завданнями в CRM-системі спрощує процес організації та відстеження роботи, забезпечуючи планування, пріоретизацію та моніторинг виконання завдань. Це допомагає підвищити продуктивність та ефективність роботи команди, зменшити пропуски та помилки, а також забезпечити вчасне виконання завдань для досягнення поставлених цілей бізнесу.

Висновки до розділу 2

Під час написання другого розділу проаналізовано алгоритми, що використовуються у системі для правильного фунціонування.

Застосування відповідних рівнів доступу та автентифікаційних методів дозволяє забезпечити безпеку та конфіденційність даних, обмежуючи доступ лише до відповідних користувачів та команд.

Описано процес керування завданнями в CRM-системі. Він включає створення завдань, призначення виконавців, моніторинг та контроль ходу виконання, підтвердження та завершення завдань. Це дозволяє забезпечити ефективну організацію та виконання завдань, підвищуючи продуктивність та ефективність роботи команди.

Особливу увагу приділено на важливість правильного зберігання та обробки даних клієнтів у CRM-системі. Застосування надійних методів зберігання, шифрування та резервного копіювання даних допомагає запобігти втраті чи пошкодженню важливої інформації. Крім того, використання алгоритмів обробки даних дозволяє здійснювати аналіз та витягування цінних показників зі зібраних даних для покращення стратегій та прийняття рішень.

Описано загальні принципи алгоритмів системи, дано їх детальне визначення. Ці алгоритми допомагають автоматизувати рутинні завдання, оптимізувати процеси та забезпечувати швидку та точну обробку даних. Використання відповідних алгоритмів допомагає зробити CRM-систему більш ефективною та потужною для досягнення бізнес-цілей.

Ці елементи спільно допомагають забезпечити безпеку, ефективність та успішність бізнесу, дозволяючи краще управляти клієнтською інформацією та досягати поставлених цілей.

3 МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ CRM-СИСТЕМИ

Даний розділ присвячений процесу розробки та проектування CRM-системи з урахуванням потреб та вимог бізнесу. У цьому розділі будуть розглянуті ключові етапи проектування системи, включаючи розробку архітектури, моделювання процесів та інші аспекти, необхідні для успішної реалізації CRM-системи.

На основі визначених вимог буде розроблено архітектуру CRM-системи. Це охоплює визначення структури системи, взаємозв'язків між компонентами, вибір технологій та платформи, що використовуються, та інші аспекти, які впливають на функціональність та ефективність системи.

Також у розділі будуть розглянуті аспекти інтерфейсу користувача, дизайну та взаємодії з системою. Застосування принципів вебдизайну та удосконалення користувацького досвіду допомагають створити зручний та привабливий інтерфейс, що сприяє залученню користувачів та покращує їх задоволеність використанням системи.

3.1 Архітектура системи

Архітектура програмного забезпечення є ключовим елементом у розробці ПЗ, оскільки вона надає загальний план для проектування та реалізації програмного продукту з урахуванням вимог замовника, забезпечуючи його якість, масштабованість та довготривалу підтримку.

Архітектура ПЗ - це організаційна структура та концептуальна основа, що визначає принципи, методи та підходи до проектування, розробки, впровадження та підтримки програмного забезпечення. Вона визначає взаємодію між різними компонентами програмного продукту, встановлює правила, засоби та структуру для досягнення вимог замовника та забезпечення його функціональності, надійності, ефективності, розширюваності та інших необхідних характеристик.

Розроблювана система базується на шаблоні MVC. Цей архітектурний шаблон проектування ПЗ використовується для розділення компонентів і логіки

застосунку на три основні ролі: модель (Model), представлення (View) та контролер (Controller). Кожна з цих ролей виконує певні функції і взаємодіє з іншими ролями для досягнення розділення відповідальностей та поліпшення модульності програмного забезпечення.

Застосування шаблону MVC дозволяє досягти розділення логіки застосунку та відображення даних. Це полегшує розробку, підтримку та тестування програмного забезпечення. На рисунку 3.1 проілюстровано принцип роботи шаблону MVC.

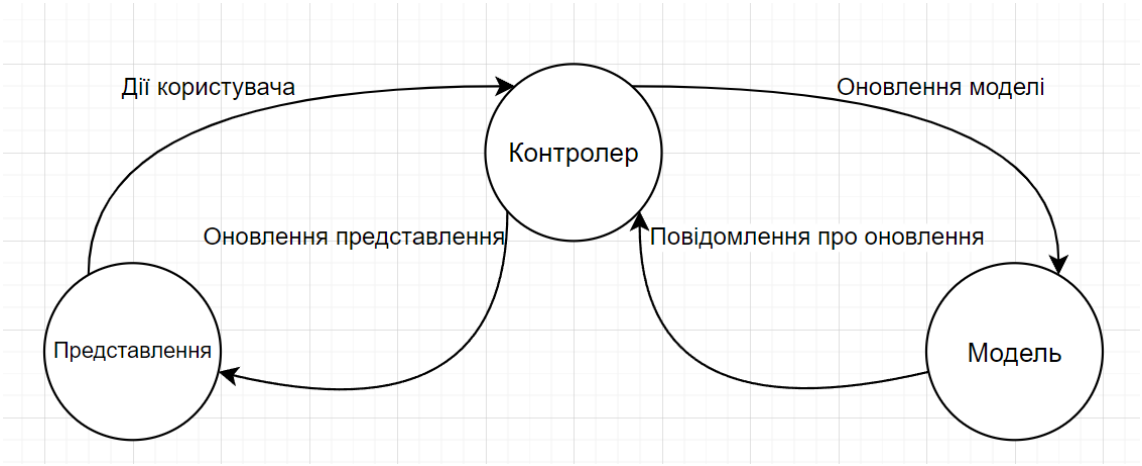

Рисунок 3.1 – Принцип роботи шаблону MVC

Основні компоненти шаблону MVC включають:

1. Модель (Model) – відповідає за управління даними застосунку та бізнес-логікою. Модель представляє собою представлення даних та правил, за якими вони обробляються.

2. Представлення (View) – відповідає за відображення даних моделі та взаємодію з користувачем. Представлення отримує дані з моделі і відображає їх у відповідному форматі для користувача. Воно також може відстежувати події користувача та передавати їх контролеру для подальшої обробки.

3. Контролер (Controller) – відповідає за керування взаємодією між моделлю та представленням. Він отримує вхідні дані від користувача через представлення, обробляє їх та здійснює відповідні дії з моделлю, такі як оновлення

даних. Контролер також може відправляти повідомлення або оновлення до представлення для оновлення відображення.

3.2 Функціональна модель системи

3.2.1 Діаграма та сценарії використання

На основі розробленої раніше специфікації вимог можна створити сценарії використання системи. Сценарії використання системи - це опис послідовності подій або дій, які відбуваються між користувачем (актором) та системою з метою досягнення конкретних цілей або виконання певних функцій. Вони описують, як користувач взаємодіє з системою та як система відповідає на його дії.

Таблиця 3.1 – Сценарій використання UC-01

UC-01: Реєстрація	
Діючі особи	Користувач та система
Мета	Зареєструватись у системі
Передумова	Користувач неавторизований
Успішний сценарій: 1. Користувач переходить на сторінку реєстрації. 2. Користувач заповнює відповідні поля. 3. Система здійснює валідацію даних. 4. Після валідації система зберігає обліковий запис до бази даних. 5. Користувач переадресовується на сторінку входу. 6. Користувач заповнює дані, вказані при реєстрації. **7.** Авторизований користувач перенаправляється на головну сторінку застосунку.	
Результат	Створено новий обліковий запис користувача
Розширення	
	Немає доступу до бази даних. Система повідомляє про помилку. **Результат**: обліковий запис не створено.

Продовження таблиці 3.1

	Користувач вводить недопустиме значення у поля вводу. **Результат**: система повідомляє про відповідну помилку та переадресовує користувача на сторінку реєстрації.

Таблиця 3.2 – Сценарій використання UC-02

UC-02: Авторизація	
Діючі особи	Користувач та система.
Мета	Авторизуватись у системі.
Передумова	Користувач неавторизований, проте зареєстрований у системі.
Успішний сценарій: 1. Користувач переходить на сторінку авторизації. 2. Користувач заповнює відповідні поля. 3. Система здійснює валідацію даних. **4.** Авторизований користувач перенаправляється на головну сторінку застосунку.	
Результат	Користувач авторизувався в системі.
Розширення	
	Немає доступу до бази даних. Система повідомляє про помилку. **Результат**: користувач не може авторизуватись в системі.
	Користувач вводить недопустиме значення у поля вводу. **Результат**: система повідомляє про відповідну помилку та переадресовує користувача на сторінку авторизації.

Таблиця 3.3 – Сценарій використання UC-03

UC-03: Управління даними про ролі користувачів	
Діючі особи	Адміністратор та система
Мета	Здійснення CRUD-операцій над даними про ролі користувачів
Передумова	Адміністратор авторизований в системі
Успішний сценарій: 1. Адміністратор переходить на сторінку з ролями користувачів. 2. Адміністратор обирає одну з ролей в системі. 3. Адміністратор обирає CRUD-операцію. 4. Адміністратор змінює (створює, редагує або видаляє) дані про роль. 5. Система здійснює валідацію внесених даних. 6. Система зберігає зміни. 7. Система відображає оновлену сторінку зі списком доступних ролей користувачів.	
Результат	Здійснено управління даними про ролі користувачів.
Розширення	
	Немає доступу до бази даних. Система повідомляє про помилку. **Результат**: дані про роль користувачів не змінено.
	Адміністратор вводить недопустиме значення у поля вводу. **Результат**: система повідомляє про відповідну помилку та переадресовує адміністратора на сторінку з ролями.

Таблиця 3.4 – Сценарій використання UC-04

UC-04: Управління даними про користувачів	
Діючі особи	Адміністратор та система
Мета	Здійснення CRUD-операцій над даними про користувачів
Передумова	Адміністратор авторизований в системі
Успішний сценарій: 1. Адміністратор переходить на сторінку з користувачами. 2. Адміністратор обирає одного з користувачів в системі. 3. Адміністратор обирає CRUD-операцію. 4. Адміністратор змінює (створює, редагує або видаляє) дані про обраного користувача. 5. Система здійснює валідацію внесених даних. 6. Система зберігає зміни.	
Результат	Здійснено управління даними про ролі користувачів.
Розширення	
	Немає доступу до бази даних. Система повідомляє про помилку. **Результат**: дані про користувача не змінено.
	Адміністратор вводить недопустиме значення у поля вводу. **Результат**: система повідомляє про відповідну помилку та переадресовує адміністратора на сторінку з користувачами.

Таблиця 3.5 – Сценарій використання UC-05

UC-05: Управління даними про завдання	
Діючі особи	Адміністратор та система
Мета	Здійснення CRUD-операцій над даними про завдання
Передумова	Адміністратор авторизований в системі
Успішний сценарій: 1. Адміністратор переходить на сторінку з завданнями. 2. Адміністратор обирає одне з доступних завдань. 3. Адміністратор обирає CRUD-операцію. 4. Адміністратор змінює дані про завдання. 5. Система здійснює валідацію внесених даних. 6. Система зберігає зміни. 7. Система відображає оновлену сторінку зі списком завдань.	
Результат	Здійснено управління даними про завдання.
Розширення	
	Немає доступу до бази даних. Система повідомляє про помилку. **Результат**: дані про завдання не змінено.
	Адміністратор вводить недопустиме значення у поля вводу. **Результат**: система повідомляє про відповідну помилку та переадресовує адміністратора на сторінку з завданнями.

Таблиця 3.6 – Повний сценарій використання

Primary actor	Користувач з роллю Editor
Scope	CRM-система для управління персоналом з продажу товарів.
Level	Створення та редагування завдання
Preconditions	Користувач вказав коректні дані при вході.
Stakeholders and interests	Працівник: Зацікавлений у перегляді наявних та запланованих завдань. Редактор: Зацікавлений у коректному додаванні завдання та подальшій перевірці коректності його додавання. Керівник проєкту: Зацікавлений у коректності відображення даних та коректності доступу користувачів.
Main success scenario	1. Після коректного вводу даних для входу, користувач опинився на сторінці перегляду завдань. 2. Натиснув на кнопку «додати завдання» та перейшов в редактор завдання. 3. Ввів необхідну інформацію завдання. 4. Зберіг та опублікував завдання. 5. Перейшов на сайт, переглянув коректність додавання завдання. 6. Повернувся до редагування завдання. 7. Додав необхідні відомості про завдання(нагадування, важливість, статус). 8. Зберіг оновлену інформацію завдання. 9.Перевірив коректність відображення завдання. 10. Вийшов з панелі керування завданнями.
Result	Користувач з роллю Editor створив нове завдання

Продовження таблиці 3.6

Extensions	1. Користувач ввів неправильні дані користувача при вході. 2. Перевищений розмір одного з полів при заповненні інформації. 3. Збій системи на сервері, внаслідок чого вебзастосунок працює некоректно. 4. Збій читання бази даних при редагуванні, повідомлення про помилку. 5. Перевищений розмір БД, завдання не додано, отримання помилки.
Frequency of occurrence	Вебзастосунок працює безперервно, за виключенням моментів його оновлення, перезавантаження, або перебоїв з різних видів на стороні серверу та інтернетом.

На рисунку 3.2 продемонстрована діаграма варіантів використання CRM-системи для управління персоналом з продажу.

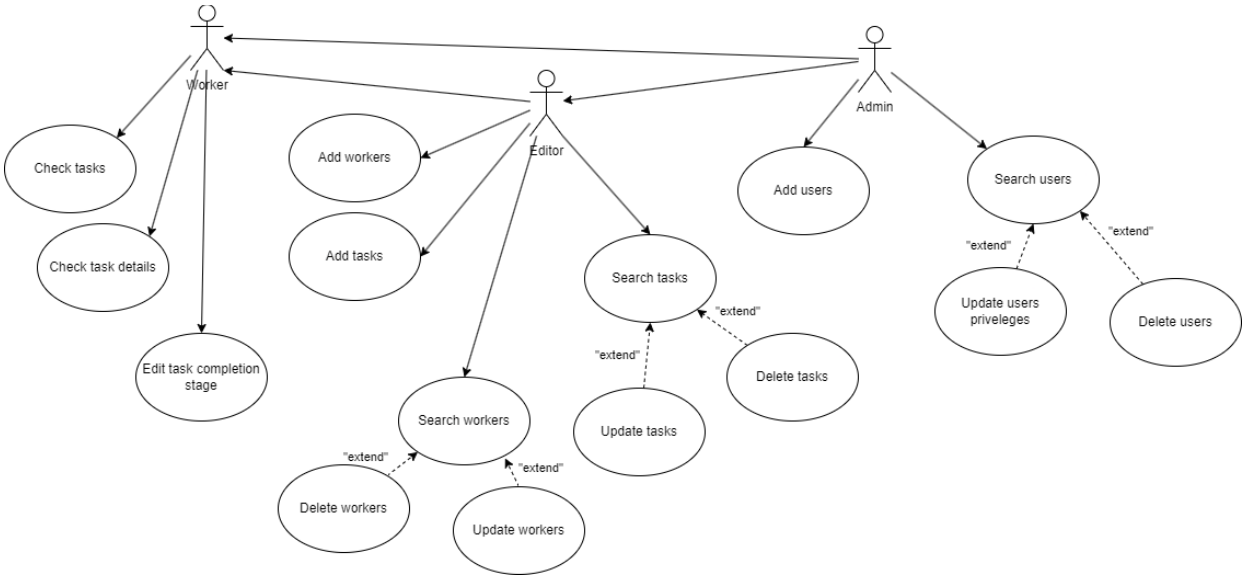

Рисунок 3.2 – Діаграма варіантів використання системи

Діаграма варіантів використання дозволяє зрозуміти функціональність системи, ідентифікувати взаємодію з акторами та виділити основні сценарії використання.

3.2.2 Діаграма класів

Діаграма класів є одним з основних видів UML-діаграм і використовується для візуального представлення структури класів в системі, їх атрибутів, методів та взаємозв'язків між ними. Вона дозволяє моделювати статичний аспект системи та деталізувати класи, які використовуються у розроблюваній CRM-системі (рис 3.3).

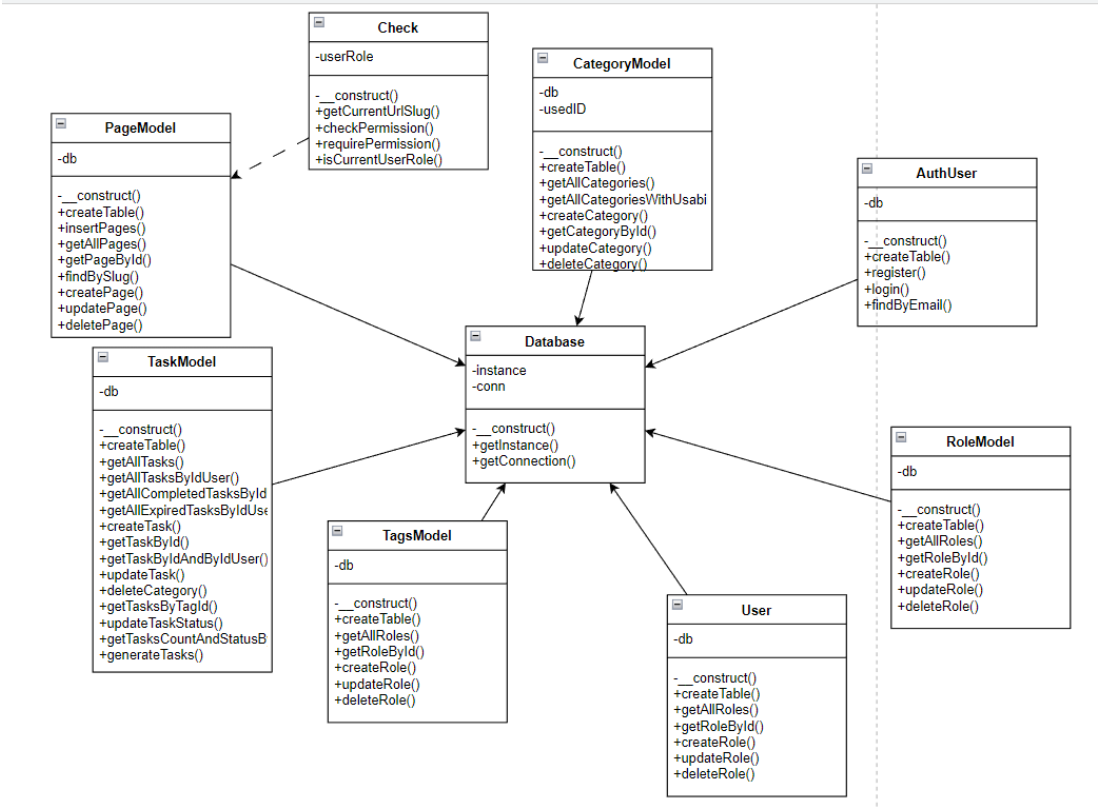

Рисунок 3.3 – Діаграма класів системи

3.2.3 Діаграми станів та переходів

Діаграми станів та переходів (statechart diagrams) використовуються для моделювання поведінки об'єктів або системи з точки зору їх станів, подій та переходів між станами. Вони дозволяють вам відображати різні стани об'єкта та умови, за яких вони можуть змінюватися. Вони особливо корисні для моделювання систем зі складними станами або систем, які мають складність у відповіді на події.

Для майбутньої системи було розроблено 3 діаграми станів та переходів:

1) діаграма станів завдання (рис. 3.4);

2) діаграма станів користувача (рис. 3.5);
3) діаграма стану логіну в систему (рис. 3.6).

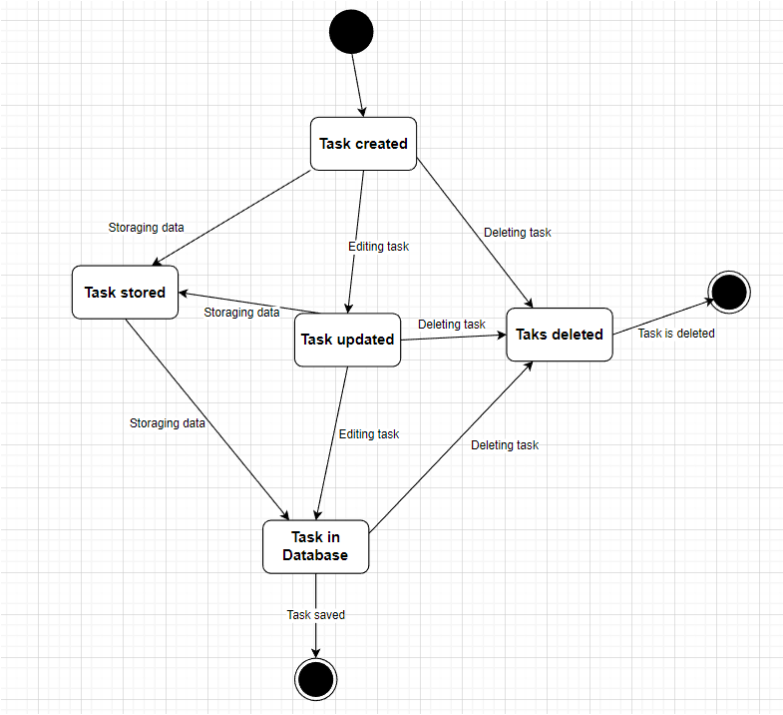

Рисунок 3.4 – Діаграма станів завдань

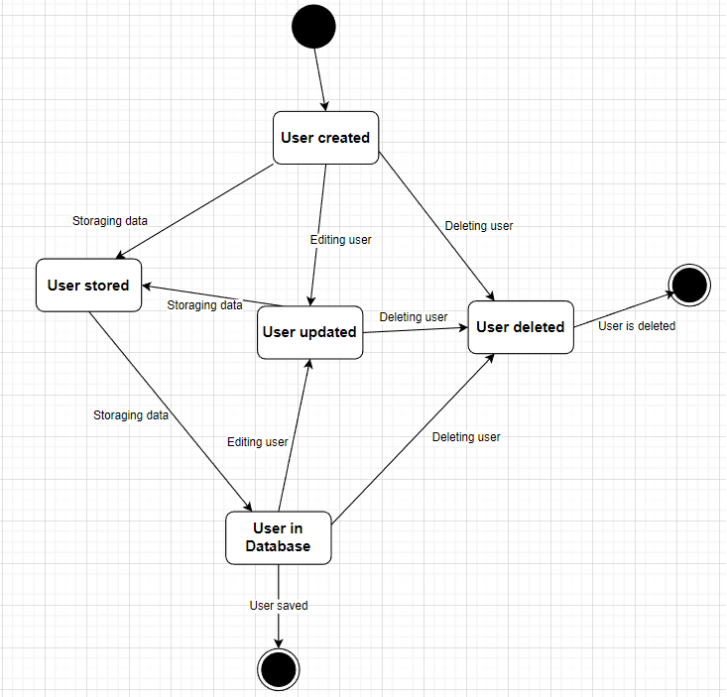

Рисунок 3.5 – Діаграма станів користувачів застосунку

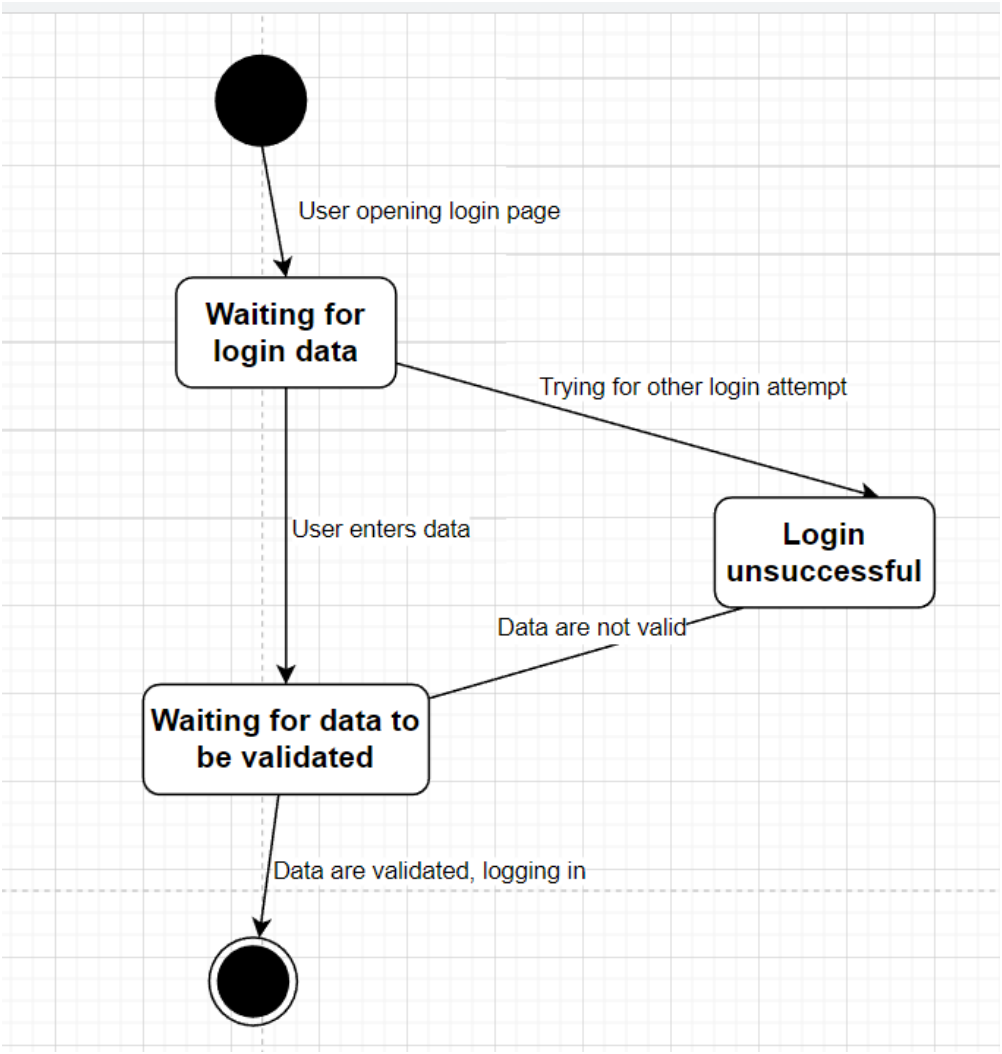

Рисунок 3.6 – Діаграма стану логіну в систему

3.2.4 Діаграми взаємодії

Діаграми взаємодії є графічними зображеннями, що використовуються для моделювання та відображення взаємодії між об'єктами або компонентами в системі. Ці діаграми зазвичай використовуються в області розробки програмного забезпечення, особливо при проектуванні об'єктно-орієнтованих систем.

Діаграми взаємодії допомагають візуалізувати взаємодію між різними об'єктами або компонентами в системі, а також послідовність обміну повідомленнями між ними. Вони дозволяють розуміти, як об'єкти взаємодіють між собою для досягнення певного результату або виконання певного функціоналу.

Діаграми взаємодії допомагають команді розробників та зацікавленим сторонам краще зрозуміти архітектуру та поведінку системи. Вони є потужним інструментом для аналізу, проектування та документування систем, а також для виявлення помилок та уточнення вимог до системи. Під час моделювання застосунку було розроблено 3 діаграми взаємодії (рис. 3.7-3.9).

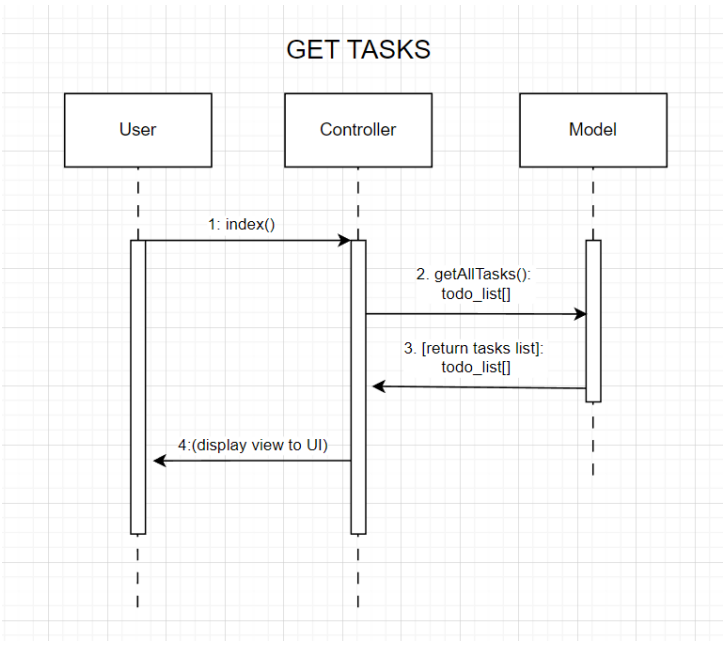

Рисунок 3.7 – Діаграма взаємодії перегляду поточних завдань

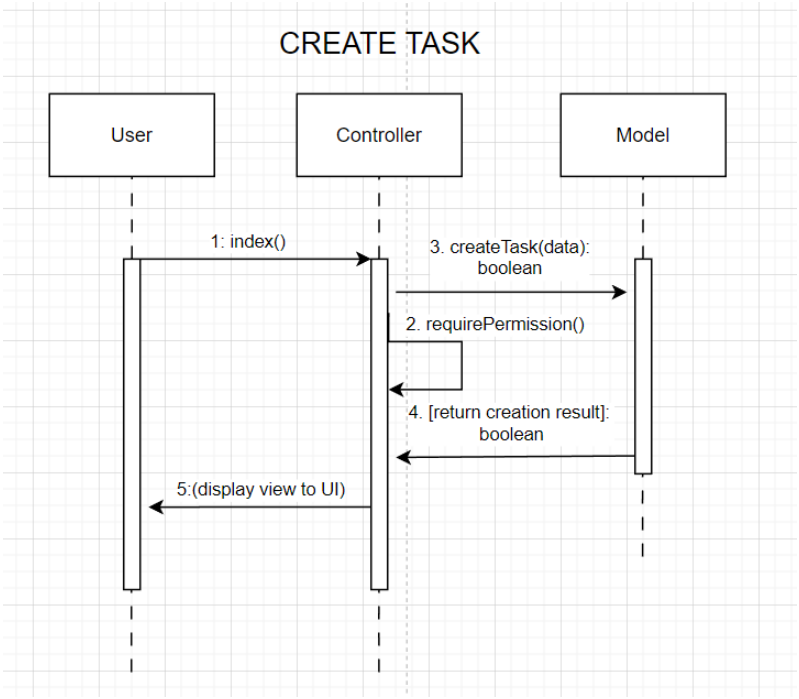

Рисунок 3.8 – Діаграма взаємодії створення нового завдання

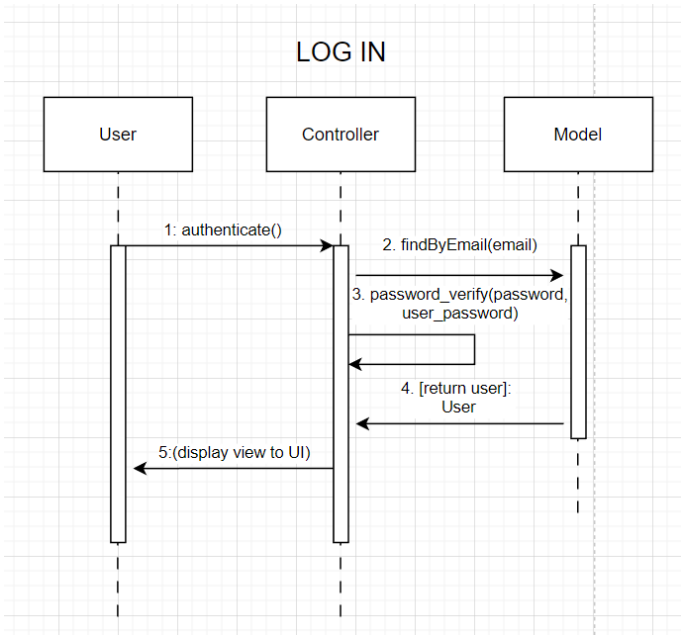

Рисунок 3.9 – Діаграма взаємодії входу в систему

На діаграмі взаємодії користувача, який виконує вхід у систему зображено як оперуються дані користувача, а саме логін та пароль

3.2.5 Діаграма розгортання

Діаграма розгортання (рис. 3.8) використовується для візуалізації фізичного розгортання компонентів програмного забезпечення, апаратного забезпечення та зв'язків між ними. Вона допомагає описати, як програмне забезпечення розташоване на фізичних пристроях і як вони взаємодіють між собою. На відміну від діаграм логічного представлення, діаграма розгортання є єдиною для системи в цілому, оскільки повинна повністю відображати особливості її реалізації.

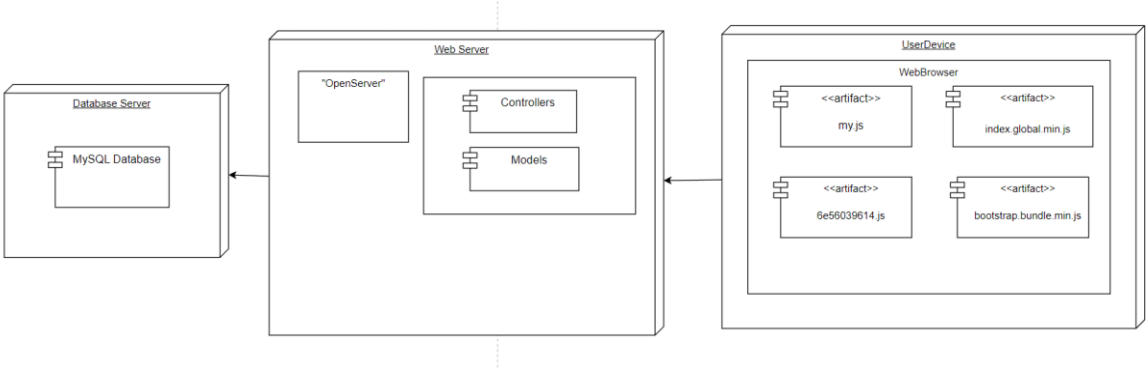

Рисунок 3.8 – Діаграма розгортання

У розроблюваній системі використовується нормалізована база даних. Нормалізована база даних означає, що дані в ній організовані згідно з принципами нормалізації даних. Нормалізація - це процес структурування бази даних з метою зменшення дублювання даних, забезпечення цілісності даних та полегшення маніпулювання даними.

У нормалізованій базі даних використовуються нормальні форми, які описують правила для організації даних (табл. 3.5).

Таблиця 3.5 – Нормальні форми БД

Перша нормальна форма (1NF)	Вимагає, щоб кожне поле у таблиці мав атомарне значення, тобто не може містити масиви або повторювані значення.
Друга нормальна форма (2NF)	Вимагає, щоб кожне поле у таблиці залежало від усього первинного ключа, а не від окремих частин ключа.
Третя нормальна форма (3NF)	Вимагає, щоб кожне поле у таблиці залежало від первинного ключа і нічого крім первинного ключа. Вона також вимагає видалення транзитивних залежностей між полями.

Нормалізована база даних має кілька переваг, таких як ефективнішу організацію даних, зменшення дублювання, полегшення змін та оновлень, а також забезпечення цілісності даних. Однак, нормалізація також може призвести до більшої кількості таблиць і складніших запитів, що може вплинути на продуктивність системи. Тому при проектуванні бази даних потрібно знайти баланс між нормалізацією та ефективністю виконання запитів.

Висновки до розділу 3

Моделювання та розробка архітектури ПЗ є важливим кроком у створенні ефективної та функціональної CRM-системи, яка задовольняє потреби бізнесу та користувачів. Він забезпечує фундаментальні засади для подальшого розроблення та реалізації системи, а також сприяє досягненню успішних результатів у впровадженні CRM-системи у підприємство.

Завдяки поставленій меті на початку розділу створено та описано модель даних, архітектуру, сценарії використана та низку діаграм що так чи інакше показують архітектуру та логіку системи.

Також під час написання третього розділу було проаналізовано архітектурний шаблон MVC для системи.

Окрім того, нормалізація бази даних відіграла важливу роль у забезпеченні цілісності даних та ефективного доступу до них. Цей процес дозволяє уникнути дублювання даних та забезпечує оптимальну організацію таблиць та зв'язків між ними.

У результаті моделювання та розробки архітектури ПЗ було створено міцний фундамент для подальшої реалізації CRM-системи. Цей процес дозволив нам чітко визначити структуру системи, забезпечити ефективну обробку даних та зручний інтерфейс для користувачів.

4 РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ CRM-СИСТЕМИ

Реалізація програмного забезпечення - це процес перетворення проекту або концепції програмного забезпечення в робочу, функціональну програму. Цей етап включає в себе написання, тестування, налагодження та оптимізацію коду, а також розгортання та підтримку створеного програмного продукту. Основна мета реалізації полягає в тому, щоб забезпечити виконання функцій, описаних у вимогах до програмного забезпечення.

Метою в четвертому розділі є вибір технологій, опис кроків реалізації програмного забезпечення CRM-системи та написання керівництва користувача.

4.1 Пояснення вибору технологій для реалізації системи

Як зазначено у розділі 1: у системі буде використано клієнт-серверну архітектуру, мови програмування PHP та JavaScript, базу даних MySQL. Розглянемо їх переваги та недоліки, та чому саме вони були обрані для створення вебзастосунку CRM-системи.

Для розробки серверної частини вебзастосунку обрано мову програмування PHP. Це скриптова мова програмування, розроблена спеціально для веброзробки. Вона використовується для створення динамічних вебсайтів та вебзастосунків. Основна ідея PHP полягає в тому, щоб забезпечити можливість генерації HTML-коду на основі даних, що знаходяться на сервері, тим самим дозволяючи створювати вебсторінки, які можуть взаємодіяти з користувачем та базами даних.

PHP виконується на стороні сервера, що означає, що код PHP обробляється на сервері, а результат відправляється до клієнта у вигляді згенерованої HTML-сторінки. Це дає можливість створювати динамічний контент, як-от сторінки з унікальним вмістом, форми введення даних, обробку запитів, роботу з базами даних, керування сеансами та багато іншого.

PHP застосовується в широкому спектрі веброзробки. Він може бути використаний для створення простих вебсайтів, таких як особисті блоги або невеликі компаній, а також для складних вебзастосунків, таких як соціальні мережі, електронна комерція, системи управління контентом (CMS), форуми, бронювання та інші.

PHP є основним компонентом популярних вебфреймворків, таких як Laravel, Symfony, CodeIgniter, які спрощують процес розробки вебзастосунків, надаючи структуру, готові рішення та підтримку для роботи з базами даних, маршрутизацією, шаблонами та іншими функціями.

Крім веброзробки, PHP також може бути використаний для створення командних інтерфейсів (CLI) та розробки застосунків, які виконуються на сервері без прив'язки до веббраузера.

Переваги PHP:

1. PHP має простий синтаксис, який легко вивчити для початківців у програмуванні. Це зменшує поріг входу і дозволяє швидше розпочати розробку.

2. Має одну з найбільших спільнот розробників, що означає наявність багато допоміжних матеріалів, документації, форумів і бібліотек. Ви можете легко знайти відповіді на свої питання або підтримку від інших розробників.

3. Вбудована підтримка для багатьох типів баз даних, що полегшує роботу з ними. Ви можете легко виконувати операції збереження, оновлення та вибору даних.

4. Має велику кількість фреймворків, таких як Laravel, Symfony, CodeIgniter, які спрощують розробку вебзастосунків і прискорюють процес.

5. PHP має хорошу швидкодію, особливо в контексті веброзробки. Використання прискорювачів, оптимізація коду та кешування дозволяють досягти високої продуктивності.

Недоліки PHP:

1. PHP використовує нестрогу типізацію, що означає, що він не накладає жорстких обмежень на типи даних. Це може призводити до проблем з помилками, пов'язаними з типами даних.

2. Історично PHP мала деякі проблеми з безпекою, особливо коли використовуються застарілі або неправильно налаштовані скрипти. Проте, з випуском нових версій PHP та дотриманням найкращих практик, ці проблеми зменшуються.

3. Деякі аспекти PHP можуть бути складними для масштабування при роботі з великими проектами або високою навантаженістю. Потрібно ретельно планувати архітектуру застосунків, щоб забезпечити масштабованість.

4. Оскільки PHP є відкритою мовою, немає жорсткого контролю якості для всіх сторонніх бібліотек та фреймворків. Це означає, що можуть виникати проблеми з якістю коду або відсутністю оновлень.

Загалом, PHP є потужним і популярним інструментом для розробки вебзастосунків, завдяки своїй простоті, широкій функціональності та великій спільноті розробників. Він застосовується в різних сферах веброзробки і забезпечує можливість створення динамічних та інтерактивних вебсайтів. Вибір PHP для розробки дозволяє швидко розробляти і розгортати функціональні динамічні вебсайти. Логотип PHP зображений на рисунку 4.1.

Рисунок 4.1 – Логотип PHP

Клієнтська частина вебзастосунку реалізована за допомогою HTML, CSS, Bootstrap та JavaScript (рис 4.2-4.5).

HTML є основною мовою розмітки для створення вебсторінок. Вона використовується для структурування та відображення вмісту вебсторінки. Основні елементи HTML включають заголовки, абзаци, списки, таблиці, посилання та інші. Переваги використання HTML:

– має простий синтаксис, що дозволяє швидко освоїти основи мови;

– базується на встановлених стандартах, що забезпечує сумісність між різними браузерами та пристроями;

– надає можливість використовувати семантичні теги для кращого розуміння вмісту сторінки як людьми, так і пошуковими системами.

Рисунок 4.2 – Логотип HTML

CSS використовується для оформлення і стилізації вебсторінок, що були структуровані за допомогою HTML. CSS дозволяє змінювати вигляд елементів, таких як кольори, шрифти, розташування, розміри і багато іншого. Переваги використання CSS включають:

1. CSS дозволяє відокремити структуру вебсторінки від її візуального представлення, що робить код чистішим та підтримкою стилів простішою.

2. Він надає можливість створювати стилі, які можна використовувати на кількох сторінках, що забезпечує єдність вигляду та полегшує зміни.

3. CSS пропонує можливості для гнучкого дизайну, дозволяючи створювати вебсторінки, які пристосовуються до різних розмірів екранів і пристроїв.

Рисунок 4.3 – Логотип CSS

Bootstrap є популярним фреймворком CSS, який надає набір готових стилів, компонентів та скриптів для швидкої розробки вебсторінок. Деякі переваги використання Bootstrap включають:

1. Bootstrap надає готові компоненти та шаблони, що прискорює процес розробки вебінтерфейсу.

2. Bootstrap має вбудовану підтримку респонсивного дизайну, що дозволяє створювати вебсторінки, які пристосовуються до різних пристроїв та розмірів екранів.

3. Bootstrap забезпечує сумісність з різними браузерами, що зменшує необхідність у вирішенні багатьох проблем відображення.

Рисунок 4.4 – Логотип Bootstrap

JavaScript є мовою програмування, яка використовується для надання динамічності та інтерактивності вебсторінкам. Вона дозволяє взаємодіяти з користувачем, маніпулювати вмістом сторінки, виконувати запити до сервера та багато іншого. Переваги використання JavaScript включають:

1. JavaScript дозволяє перевіряти дані, введені користувачем у форми, без необхідності звертатися до сервера.

2. JavaScript дає можливість створювати динамічні ефекти, анімацію та змінювати вміст сторінки без перезавантаження.

3. JavaScript дозволяє виконувати асинхронні запити до сервера без перезавантаження сторінки, що робить вебзастосунки швидшими.

Рисунок 4.5 – Логотип JavaScript

Ці мови та технології працюють разом, надаючи зручність, гнучкість та можливості для розробки зручного та естетично привабливого вебзастосунку. Комбінація цих інструментів дозволяє створювати потужні та інтерактивні інтерфейси, які привертають користувачів та забезпечують їх зручність у використанні.

В якості СКБД вибір зупинився на MySQL (рис. 4.6). MySQL є однією з найпопулярніших систем керування базами даних (СКБД) і має декілька переваг, які роблять його привабливим вибором для великої кількості застосунків і проектів. Ось детальний опис та переваги MySQL:

MySQL - це реляційна СКБД, розроблена для зберігання, управління та доступу до великих обсягів даних. Вона є безкоштовною та відкритою системою, що означає, що її можна використовувати безкоштовно та вносити зміни в її вихідний код відповідно до своїх потреб.

Вона має простий синтаксис SQL, який легко вивчити і використовувати. Це дозволяє швидко створювати таблиці, виконувати запити та модифікувати дані. Ця система є швидкою та ефективною СКБД, яка дозволяє оптимізувати запити та забезпечує швидкий доступ до даних. Вона підтримує індексування, кешування та інші методи покращення продуктивності.

MySQL здатна обробляти великі обсяги даних і масштабуватися залежно від потреб вашого проекту. Він підтримує розподілені системи, кластеризацію та реплікацію, що дозволяє збільшити продуктивність і надійність бази даних. Має високу надійність і вбудовані механізми захисту даних. Він підтримує резервне копіювання, відновлення даних та шифрування, що дозволяє забезпечити безпеку та цілісність ваших даних.

Обрана СКБД є сумісною з багатьма платформами і мовами програмування, такими як PHP, Python, Java та інші. Це робить її універсальною і легкою для інтеграції з іншими технологіями.

MySQL є надійним та потужним вибором для багатьох проектів, від невеликих вебзастосунків до великих корпоративних систем. Вона надає широкі можливості для роботи з даними, забезпечує продуктивність та безпеку, а також має велику спільноту підтримки.

Рисунок 4.6 – Логотип MySQL

4.2 Опис програмної реалізації

Програмна реалізація вебзастосунку включає розробку наступних компонентів: серверної частини, клієнтської частини та бази даних.

Комбінація серверної частини, клієнтської частини і бази даних дозволяє створити повноцінний вебзастосунок. Серверна частина обробляє запити користувачів, взаємодіє з базою даних і надсилає відповіді клієнтській частині. Клієнтська частина відповідає за відображення інтерфейсу користувача та обробку дій користувача, таких як заповнення форм, взаємодія з елементами сторінки та відправка запитів на сервер. База даних забезпечує зберігання і доступ до даних, що використовуються в застосунку.

4.2.1 Загальна структура проєкту

Реалізацією системи, що була змодельована в попередніх розділах є проєкт, загальну структуру якого представлено на рисунку 4.7.

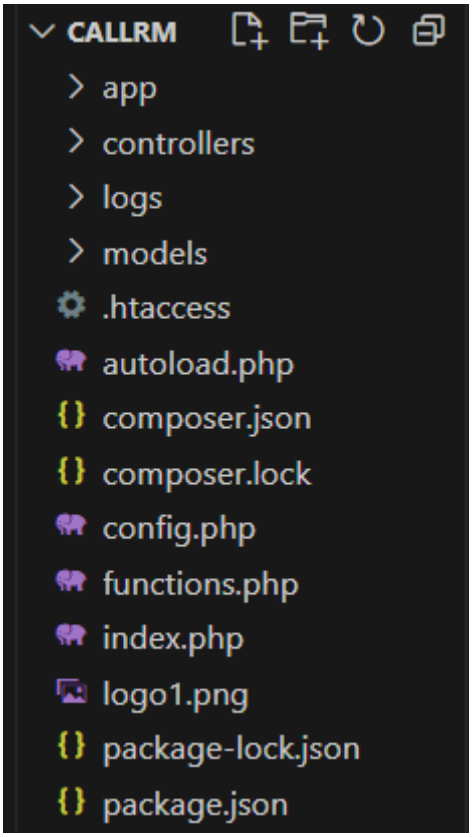

Рисунок 4.7 – Загальна структура проєкту

Вміст складових:

- застосунок «*app*» містить маршрутизатор та представлення проєкту;
- застосунок «*controllers*» відповідає за керування взаємодією між моделями та представленнями;
- застосунок «*models*» відповідає за управління даними застосунку;
- у директорії «*logs*» знаходяться файли журналу – логи;
- файл «*autoload.php*» містить функцію що відповідає за автозавантаження файлів у системі;
- у файлі «*config.php*» наведено налаштування та деякі константні змінні проєкту;
- у файлі «*functions.php*» знаходяться додаткові функції проєкту, що не стосуються існуючих моделей;
- у файлі «*index.php*» реалізовано початок сесії, підключення конфігураційних файлів та маршрутизатора;
- файл «*logo.png*» є логотипом проєкту;
- файли «*composer.json*», «*composer.lock*», «*package-lock.json*», «*package.json*» є системними файлами.

4.2.2 Застосунок «app»

Даний застосунок містить файл маршрутизатора «*router.php*», представлення видів «views» та файли стилів та скриптів – «*style.css*», «*my.js*». Загальним шаблоном є файл «*layout.php*», який наслідують всі сторінки проєкту. У файлі «*index.php*» міститься вигляд головної сторінки системи. Загальна структура даного застосунку зображена на рисунку 4.8.

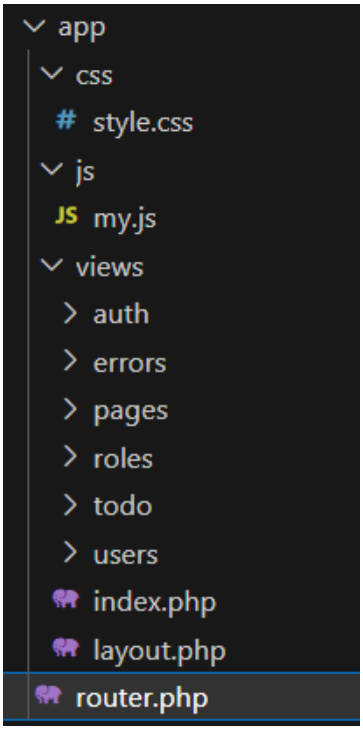

Рисунок 4.8 – Структура застосунку «*app*»

У файлі маршрутизатора знаходиться функція **run()**, що відповідає за перегляд шаблону URL та перенаправляє на відповідну сторінку якщо рядок адреси введено правильно (рис 4.9).

```php
public function run() {
    $uri = $_SERVER['REQUEST_URI'];
    $controller = null;
    $action = null;
    $params = null;
    foreach ($this->routes as $pattern => $route) {
        if (preg_match($pattern, $uri, $matches)) {
            $controller = "controllers\\" . $route['controller'];
            $action = $route['action'] ?? $matches['action'] ?? 'index';
            $params = array_filter($matches, 'is_string', ARRAY_FILTER_USE_KEY);
            break;
        }
    }
```

Рисунок 4.9 – Функція run() маршрутизатора

У файлі де знаходяться скрипти, знаходяться обробники подій при взаємодії користувача з сайтом. Основним обробником подій є сортування завдань за пріоритетом, що буде часто використовуватись при роботі системи – sortTasksByPriority() (рис 4.10).

```
function sortTasksByPriority(priority) {
    const tasksAccordion = document.querySelector('#tasks-accordion');
    const tasks = Array.from(tasksAccordion.querySelectorAll('.accordion-item'));

    tasks.sort((a, b) => {
        const aPriority = a.querySelector('.accordion-button').getAttribute('data-priority');
        const bPriority = b.querySelector('.accordion-button').getAttribute('data-priority');

        if (aPriority === priority && bPriority !== priority) {
            return -1;
        } else if (aPriority !== priority && bPriority === priority) {
            return 1;
        } else {
            return 0;
        }
    });

    tasks.forEach((task) => {
        tasksAccordion.appendChild(task);
    });
}
```

Рисунок 4.10 – Функція сортування завдань за пріоритетом

У директоріях що знаходяться у «views» знаходяться представлення для CRUD операцій.

Призначення директорій представлень:

– «auth» відповідає за представлення сторінок авторизації та реєстрації;

– «errors» відповідає за представлення сторінок помилок;

– «pages» відповідає за представлення сторінок з існуючими маршрутами;

– «roles» відповідає за представлення сторінок з ролями користувачів;

– «todo/categories» відповідає за представлення сторінок з категоріями завдань;

– «todo/tasks» відповідає за представлення сторінок де завдання відфільтровані за станом виконання;

– «users» відповідає за представлення сторінок користувачів.

4.2.3 Застосунок «controllers»

Структура застосунку «controllers» зображена на рисунку 4.11.

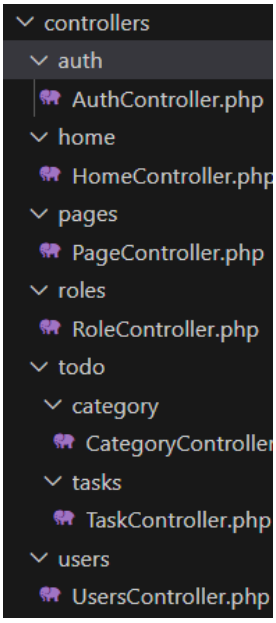

Рисунок 4.11 – Структура застосунку «controllers»

Вебзастосунок відповідає за збереження класів контролерів, які виконують обробку запитів користувача, взаємодію з моделями та керування логікою застосунку. Згідно до концепції патерну MVC даний застосунок містить контролери моделей. Деякі з цих контролерів містять методи що відповідають за CRUD операції з відповідною моделлю.

Призначення контролерів:

– «AuthController.php» містить логіку авторизації та реєстрації користувачів в системі;

– «HomeController.php» відповідає за логіку головної сторінки;

– «PageController.php» містить CRUD операції з представленнями;

– «RoleController.php» містить CRUD операції з ролями;

– «CategoryController.php» містить CRUD операції з категоріями завдань;

– «TaskController.php» містить CRUD операції з завданнями а також інші логічні дії з ними;

– «UsersController.php» містить CRUD операції з користувачами та логіка профілю користувача.

При авторизації до системи, створюється нова сесія у браузері, дані записуються у cookies, також створюється об'єкт User, його логіка прописана у контролері. На рисунку 4.12 показано код конструктора контролера, що відповідає за створення cookies.

```php
public function __construct()
{
    $userRole = isset($_SESSION['user_role']) ? $_SESSION['user_role'] : null;
    $this->userId = isset($_SESSION['user_id']) ? $_SESSION['user_id'] : null;
    $this->check = new Check($userRole);
}
```

Рисунок 4.12 – Код конструктора файлу «*UserController.php*»

Також з рисунка видно, що відбувається перевірка ролі користувача та видача відповідних прав доступу.

4.2.4 Застосунок «models»

Структура даного застосунку зображена на рисунку 4.13, як можна помітити, вона схожа на структуру застосунку «controllers», через те що у контролерах реалізовано логіку усіх моделей.

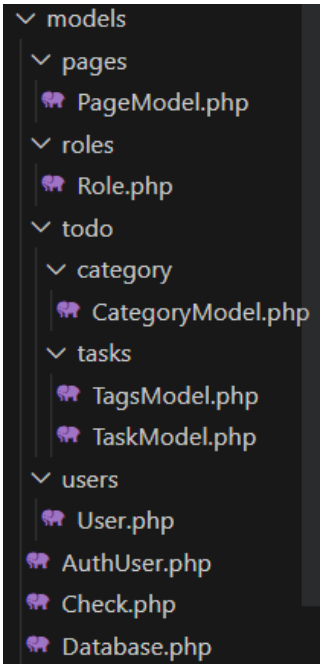

Рисунок 4.13 – Структура застосунку «models»

В даному застосунку представлені сутності, що існують в системі, саме вони були зображені на діаграмі класів при моделюванні системи. В структурі вебзастосунку відповідає за збереження класів моделей об'єктів, які представляють дані та логіку, пов'язану з цими даними.

Моделі включають методи для збереження, оновлення, видалення та отримання даних з бази даних, а також методи для валідації та перевірки даних.

Важливо підтримувати чистоту та розсудливість у структурі моделей. Тому кожна модель повинна відповідати одному конкретному типу об'єкту і мати чітку відповідальність.

Призначення моделей:

- «PageModel.php» містить клас моделі сторінок представлень;
- «Role.php» містить клас моделі ролі користувача;
- «CategoryModel.php» містить клас моделі категорії завдання;
- «TagsModel.php» містить клас моделі тегів завдання;
- «TaskModel.php» містить клас моделі завдання;
- «User.php» містить клас моделі користувача;
- «AuthUser.php» містить клас моделі авторизації та методи валідації;
- «Check.php» містить клас моделі перевірки ролі, за допомогою якого видається доступ до функцій системи;
- «Database.php» - містить клас моделі бази даних та методи підключення до неї.

Основним є клас моделі бази даних, оскільки через нього відбувається вся взаємодія з даними системи. Підключення до бази даних зображено на рисунку 4.14.

```php
private function __construct(){
    $db_host = DB_HOST;
    $db_user = DB_USER;
    $db_pass = DB_PASS;
    $db_name = DB_NAME;

    try{
        $dsn = "mysql:host=$db_host;dbname=$db_name";
        $this->conn = new \PDO($dsn, $db_user, $db_pass);
        $this->conn->setAttribute(\PDO::ATTR_ERRMODE, \PDO::ERRMODE_EXCEPTION);
    } catch(\PDOException $e){
        echo "Connect failed: " . $e->getMessage();
    }
```

Рисунок 4.14 – Код підключення до бази даних

Користувачі моделей (такі як контролери) взаємодіють з моделями, викликаючи їх методи для отримання та зміни даних, виконання різних операцій та реалізації бізнес-логіки застосунку.

4.3 Інструкція користувача

Інструкція користувача є важливою складовою будь-якого програмного забезпечення. Воно надає користувачам необхідну інформацію та інструкції щодо використання програмного продукту.

4.3.1 Авторизація та реєстрація

Авторизація необхідна для отримання доступу до системи. На рисунку 4.15 зображено сторінку авторизації.

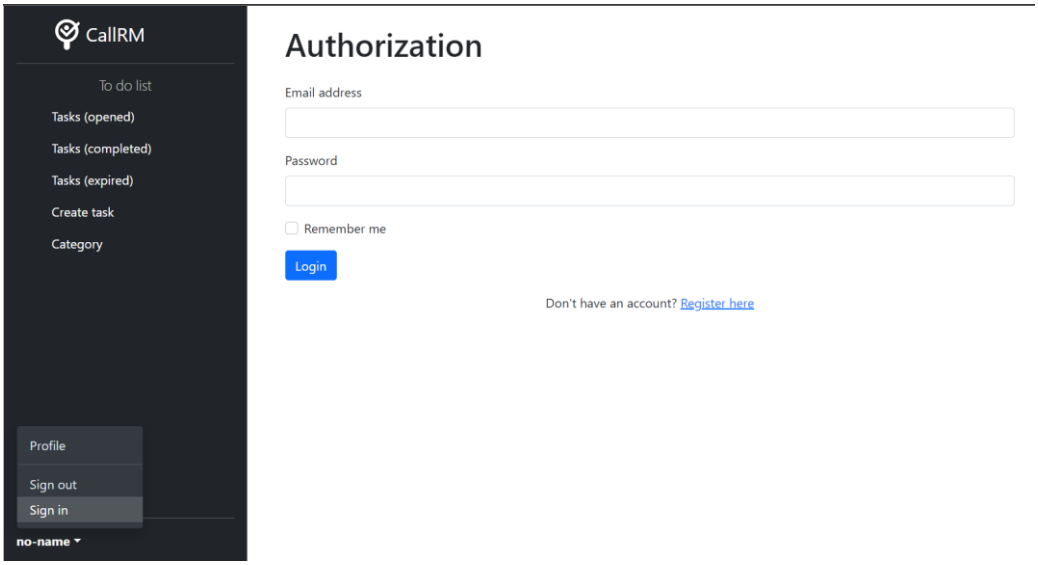

Рисунок 4.15 – Сторінка авторизації

В даній формі є два поля – «Email address» і «Password» які потрібно заповнити та чекбокс «Remember me», який відповідає за збереження інформації у куках. Після введення даних потрібно натиснути «Login». Якщо користувач досі не зареєстрований у системі, то він може перейти до форми реєстрації натиснувши на посилання «Register here».

При вводі некоректних даних або даних що не співпадають з базою даних, користувач отримає відповідну помилку (рис. 4.16).

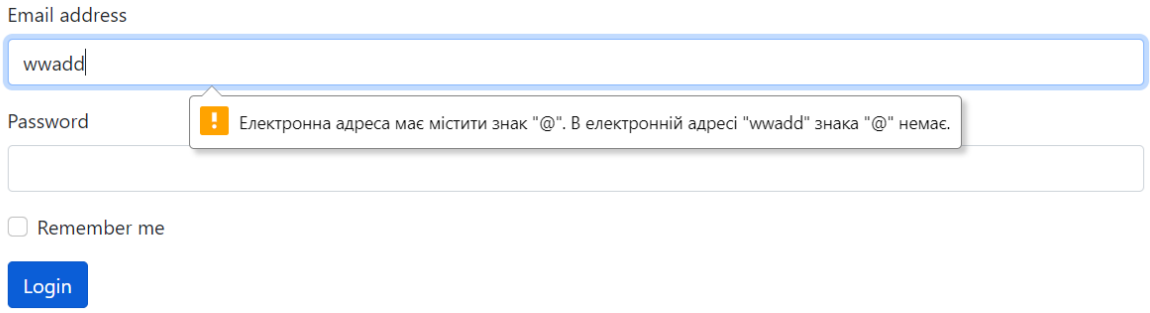

Рисунок 4.16 – Повідомлення про помилку вводу

На рисунку 4.17 зображено форму реєстрації нового користувача.

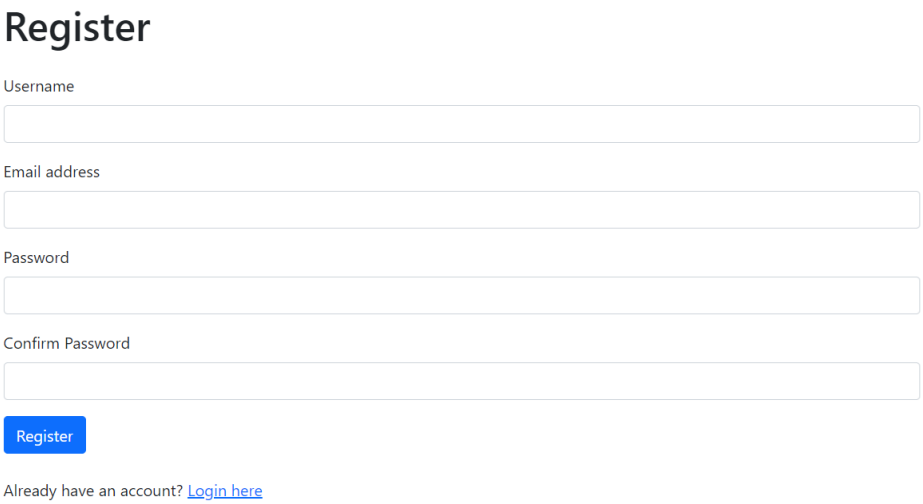

Рисунок 4.17 – Форма реєстрації користувача

Для успішної реєстрації потрібно коректно заповнити поля:

- «Username» (логін);
- «Email» (електронна адреса користувача);
- «Password» (пароль);

– «Confirm Password» (підтвердження пароля).

Після заповнення полів потрібно натиснути кнопку «Register». Також є можливість повернутись на сторінку авторизації натиснувши на посилання «Login here».

4.3.2 Робота з категоріями

Після авторизації, користувач повинен створити категорії для завдань, це свого роду групи. Для цього потрібно перейти на сторінку «Category» і натиснути кнопку «Create category». Форма створення категорії зображена на рисунку 4.18.

Рисунок 4.18 – Форма створення категорії

Тут є два поля – «Title», що відповідає за назву категорії, і «Description», яке є описом категорії. Після вводу даних потрібно натиснути «Create category», після чого користувача перенаправить на сторінку з категоріями.

4.3.3 Робота з завданнями

Після створення хоча б однієї категорії користувач має можливість створити завдання. Це можна зробити перейшовши на сторінку «Create task» і заповнити форму. Зовнішній вигляд форми зображено на рисунку 4.19.

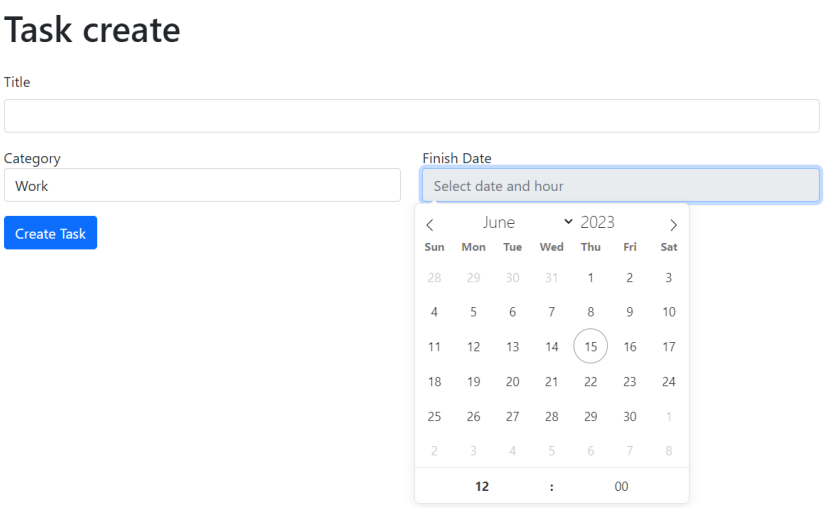

Рисунок 4.19 – Форма створення завдання

Для створення завдання достатньо ввести «Title» (назва) обрати категорію з випадного списку і обрати дату та час за допомогою інтерактивного календаря. Після створення завдання користувача буде перенаправлено до списку активних завдань (рис 4.20).

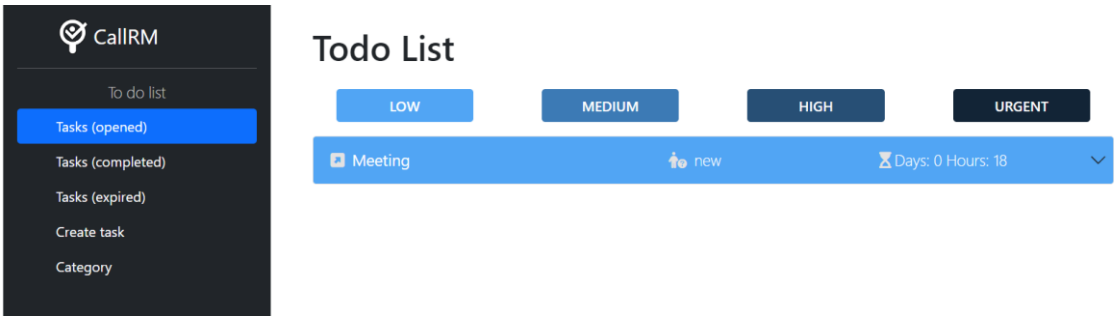

Рисунок 4.20 – Сторінка активних завдань

На скріншоті видно градацію пріоритетів, за замовчуванням завдання створюється з низьким пріоритетом. Кнопки з рівнем пріоритету є клікабельними, при натисканні завдання з відповідним пріоритетом сортуються вгору списку. При натисканні на завдання буде можливість його редагувати – клавіша «Edit». На рисунку 4.21 зображено форму редагування завдання.

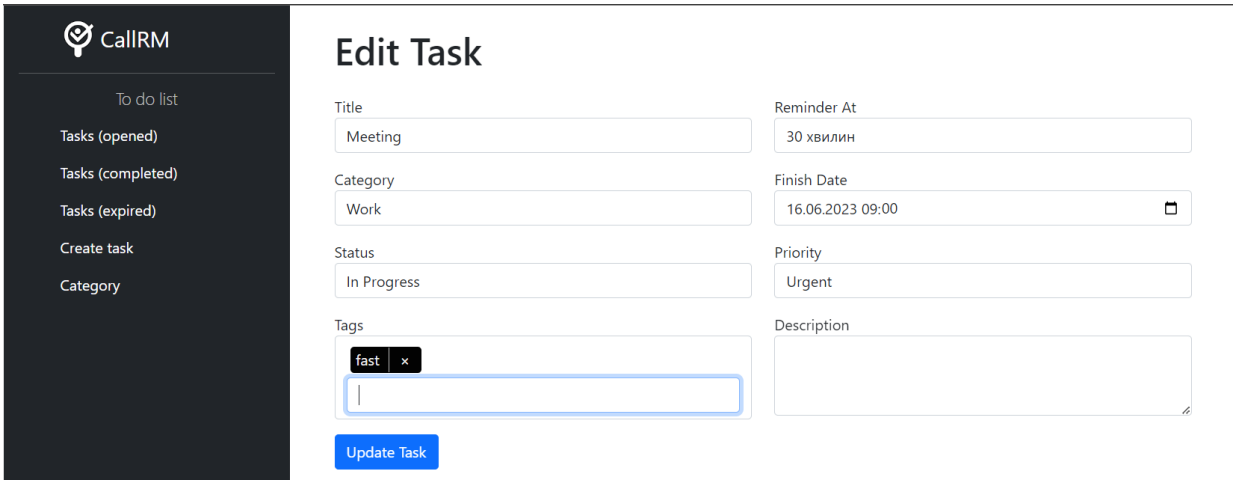

Рисунок 4.21– Форма редагування завдання

В редагуванні завдання додались поля:

- «Reminder At» (нагадати через);
- «Status» (статус);
- «Priority» (пріоритет)
- «Tags» (тег);
- «Description» (опис).

Поля «Reminder At», «Status» та «Priority» є випадними списками. Теги це додатковий тип фільтру, можна додавати декілька тегів. Для цього потрібно написати назву тегу і ввести знак коми «,», для видалення потрібно натиснути хрестик біля назви тегу. Щоб зберегти зміни потрібно натиснути «Update Task». Також користувач може видалити завдання натиснувши «Delete». Після створення завдання, воно буде відображатись на головній сторінці в календарі до дати завершення включно (рис 4. 22).

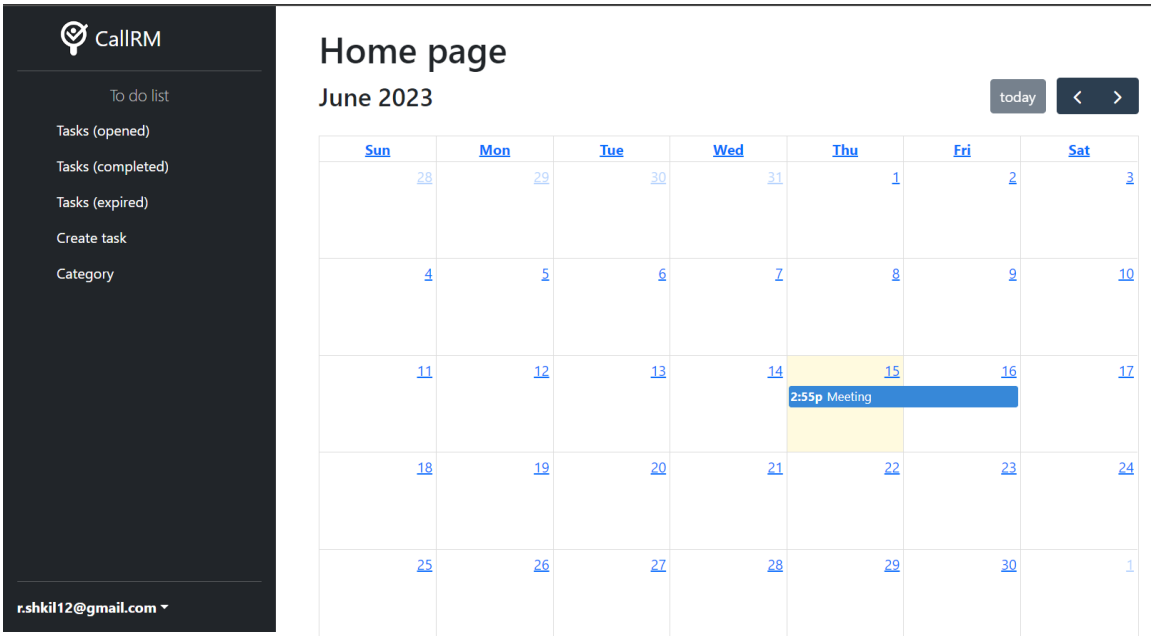

Рисунок 4.22 – Головна сторінка проєкту

На цьому можливості звичайного користувача завершуються.

Для адміністратора доступно більше функцій, а саме керування користувачами «Users», керування ролями «Roles» та керування доступом до сторінок «Pages», як зображено на рисунку 4.23.

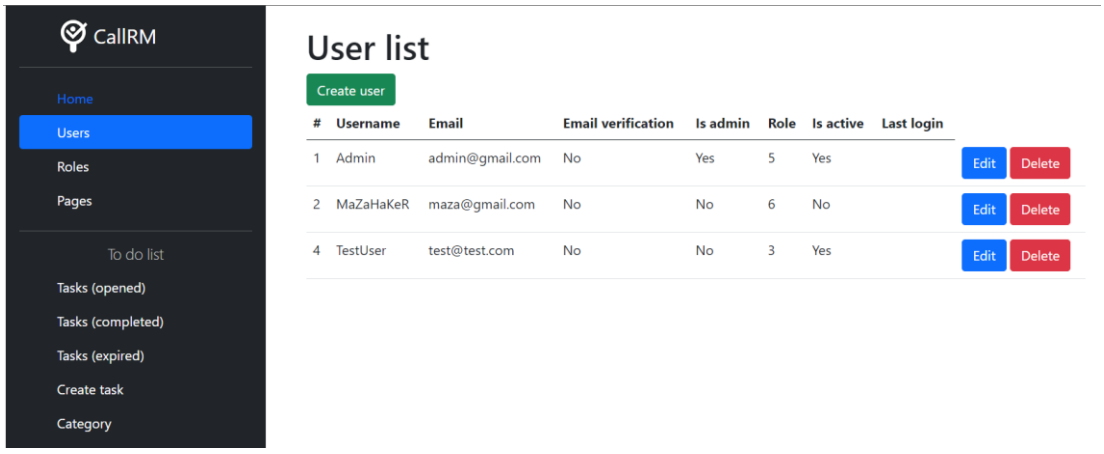

Рисунок 4.23 – Сторінка керування користувачами

Для зміни ролі користувача адміністратору потрібно натиснути «» навпроти потрібного користувача та обрати одну з наявних ролей в системі.

На рисунку 4.24 зображено сторінку налаштування доступу до представлень системи.

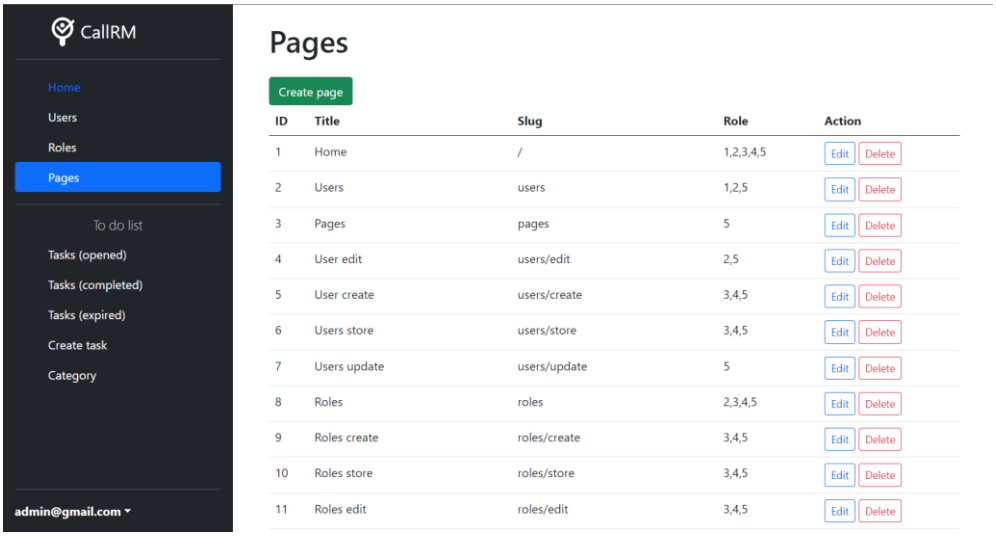

Рисунок 4.24 – Сторінка керування доступом до функцій проєкту

Висновки до розділу 4

В четвертому розділі було обґрунтовано вибір технологій для реалізації системи, детально описано код та структуру проєкту. Крім того, було написано інструкцію користувача. Використання обраних програмних засобів дозволило розробити вебзастосунок ефективно та швидко. Результатом роботи є створений вебзастосунок, який повністю задовольняє вимоги. Таким чином було успішно досягнуто поставленої мети.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи бакалавра було успішно реалізовано вебзастосунок, який збільшує ефективність та спрощує керування персоналом. Таким чином, було досягнуто поставленої мети. Для досягнення мети було виконано наступні завдання:

- проведено аналіз предметної сфери;
- визначено функціональні та нефункціональні вимоги вебзастосунку;
- розроблено дизайн і створено логотип для вебзастосунку;
- розроблено зовнішній вигляд сайту вебзастосунку;
- виконано верстку сайту вебзастосунку.

Під час реалізації вебзастосунку CRM-системи було отримано значний досвід, який дозволяє зробити наступні висновки:

1. Важливо правильно та чітко зрозуміти потреби та вимоги користувачів CRM-системи.

2. При розробці системи, важливо правильно спланувати архітектуру проєкту. Це включає вибір правильних технологій, розбиття функціональності на модулі та забезпечення масштабованості та легкості супроводу.

3. CRM-система потребує потужної та ефективної бази даних для зберігання та обробки великої кількості інформації. Розуміння реляційних баз даних та оптимізації запитів є важливими навичками, щоб забезпечити швидкий доступ до даних.

4. Оскільки CRM-система містить конфіденційну та важливу інформацію, безпека є першочерговим завданням. Для забезпечення безпеки користувачів є необхідними налаштування доступу, шифрування даних та захист від зламу.

Отже, розроблена система відповідає всім вимогам та своєму призначенню. В майбутньому вона може бути вдосконалена та розширена.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Customer-centric business management system (CRM – system) / A. O. Ongdash et al. European Journal of Humanities and Social Sciences. 2016. P. 58–61. URL: https://doi.org/10.20534/ejhss-16-1-58-61 (date of access: 15.06.2023).

2. How to build a custom CRM software for your business | LANARS. Lanars LLC. URL: https://lanars.com/blog/how-to-build-a-custom-crm-software (date of access: 28.04.2023).

3. Leadership: Style or circumstance? / ed. by C. Productions, C. E. Films. Released by CRM Educational Films. 30 p.

4. How to Design a CRM System: An Extensive Step-by-Step Guide. SaaS UI/UX Design Agency Eleken. URL: https://www.eleken.co/blog-posts/how-to-design-a-crm-system-all-you-need-to-know-about-custom-crm (date of access: 03.05.2023).

5. Kakovkina V. How to make sure your CRM data is secure. NetHunt Blog | Sales, Marketing, and CRM. URL: https://nethunt.com/blog/crm-data-protection/ (date of access: 05.05.2023).

6. BambooHR. URL: https://www.bamboohr.com/ (дата звернення: 28.05.2023).

7. Workday. URL: https://www.workday.com/ (дата звернення: 28.05.2023).

8. Zoho People. URL: https://www.zoho.com/people/ (дата звернення: 28.05.2023).

9. Худий А. М. Ефективні методи та алгоритми обробки сигналів і даних в системних та нейронних середовищах : автореферат, канд. техн. наук. Львів, 2002. 16 с.

10. Nixon R. Learning PHP, MySQL and JavaScript. O'Reilly Media, Incorporated, 2014.

11. Curioso A., Galbraith P., Bradford R. Expert PHP and MySQL. Wiley & Sons, Incorporated, John, 2010. 587 p.

12. Stevens G. Learn HTML and CSS with the Ultimate Crash Course on HTML and CSS. Independently Published, 2022.

13. Prettyman S. Learn PHP 8: Using MySQL, JavaScript, CSS3, and HTML5. 2nd ed. Apress, 2020. 583 p.

14. Helgeson L. CRM for Dummies. Wiley & Sons, Incorporated, John, 2017. 368 p.

15. Duckett J. PHP & MySQL: Server-side Web Development. Wiley, 2022. 672 p.

16. Про затвердження Типового порядку обробки персональних даних у базах персональних даних : Наказ М-ва юстиції України від 30.12.2011 р. № 3659/5 : станом на 1 січ. 2014 р. URL: https://zakon.rada.gov.ua/laws/show/z0001-12#Text (дата звернення: 30.05.2023).

17. Supaartagorn C. PHP Framework for Database Management Based on MVC Pattern. International Journal of Computer Science and Information Technology. 2011. Т. 3, № 2. С. 251–258. URL: https://doi.org/10.5121/ijcsit.2011.3219 (date of access: 02.06.2023)