Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія управління відносинами з клієнтами»

Виконав: студент 2-го курсу, групи 2КН-23м
спеціальності 122 – «Комп'ютерні науки»
Шаргало Д. В.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Крилик Л. В.
(прізвище та ініціали)
«05» 12 2024 р.

Опонент: к.т.н., доцент каф. АІІТ
Богач І. В.
(прізвище та ініціали)
«05» 12 2024 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А. А.
(прізвище та ініціали)
«06» 12 2024 р.

Вінниця - 2024 року

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Системи штучного інтелекту

ЗАТВЕРДЖУЮ
Завідувач кафедри КН

д.т.н., проф. Яровий А. А.

(підпис)
«11» 10 2024 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Шаргало Дмитру Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи: «Інформаційна технологія управління відносинами з клієнтами»
керівник роботи к.т.н., доц., доц. каф. КН, Крилик Л. В.
затверджені наказом вищого навчального закладу від «17» 09 2024 року №310
2. Строк подання студентом роботи 11.11 2024 року.
3. Вихідні дані до роботи: мова програмування: об'єктно-орієнтована; мінімальна кількість рекомендованих модулів CRM: 5 шт.; час відгуку сервера: не має перевищувати 60 секунд для забезпечення ефективної роботи користувачів; кількість клієнтів у системі: розрахована на обслуговування не менше 3 клієнтів; інтерфейс користувача: має бути інтуїтивно зрозумілим для полегшення адаптації співробітників до нової системи.
4. Зміст текстової частини: вступ, обґрунтування доцільності розробки інформаційної технології управління відносинами з клієнтами; моделювання інформаційної технології управління відносинами з клієнтами; програмна реалізація інформаційної технології управління відносинами з клієнтами; економічна частина, висновки, список використаних джерел, додатки.
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): UML-діаграма послідовності взаємодії модулів інформаційної технології управління відносинами з клієнтами; схема системи клієнт-сервер; загальна структурна модель програмного модуля; схема алгоритму роботи інформаційної технології управління відносинами з клієнтами; діаграма діяльності серверу під час процесу автентифікації користувача; діаграма діяльності серверу під час процесу перегляду замовлення користувача; приклад роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконприй
1–3	Крилик Л. В., к.т.н., доц. каф. КН		
4	Адлер О. О., к.т.н., доц. каф. ЕПВМ		

7. Дата видачі завдання 02.09 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування доцільності розробки	02.09.24 – 15.09.24	Розділ 1
2	Моделювання інформаційної технології управління відносинами з клієнтами	16.09.24 – 25.09.24	Розділ 2
3	Програмна реалізація інформаційної технології управління відносинами з клієнтами	26.09.24 – 15.10.24	Розділ 3
4	Підготовка економічної частини	16.10.24 – 28.10.24	Розділ 4
5	Апробація результатів дослідження	29.10.24 – 04.11.24	тези доповіді, авторське право твір, стаття
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	05.11.24 – 11.11.24	Пояснювальна записка, графічн матеріал, презентація

Студент _____ Шаргало
(підпис) (прізвище та ін

Керівник роботи _____ Крилик Л.
(підпис) (прізвище та ін

АНОТАЦІЯ

УДК 004.42

Шаргало Д. В. Інформаційна технологія управління відносинами з клієнтами. Магістерська кваліфікаційна робота зі спеціальності 122 «Комп'ютерні науки», освітня програма «Системи штучного інтелекту». Вінниця: ВНТУ, 2024. 143 с.

Укр. мовою. Бібліогр.: 30 назв; рис.: 28; табл.: 8.

У магістерській кваліфікаційній роботі проведено аналіз предметної галузі управління відносинами з клієнтами (CRM) та розглянуто основні підходи до розробки інформаційних технологій для покращення взаємодії з клієнтами. Виконано аналіз існуючих систем, методів та технологій, що використовуються для вирішення подібних задач, обґрунтовано доцільність створення нової інформаційної технології, яка забезпечує зручність та інтуїтивно зрозумілий інтерфейс для користувачів.

Розроблено структурну модель інформаційної технології управління відносинами з клієнтами та удосконалено математичну модель для прогнозування поведінки клієнтів. Використання цих моделей дозволяє підвищити ефективність прогнозування та персоналізацію взаємодії з клієнтами, що сприяє розширенню функціональних можливостей технології.

Розроблена інформаційна технологія забезпечує користувачам новий інструмент для управління взаємовідносинами з клієнтами, що включає чат та нотатки, а також дозволяє покращити стратегії обслуговування. В роботі наводяться результати досліджень та аналіз ефективності розробленої системи, доведена її економічна доцільність і вигоди від впровадження в бізнес-процеси.

Ілюстративна частина складається з 7 плакатів із результатами моделювання.

Ключові слова: управління відносинами з клієнтами, CRM, інформаційна технологія, автоматизація, персоналізація, аналіз даних.

ABSTRACT

Shargalo D. V. Information technology for managing relations with clients. Master's thesis in the specialty 122 «Computer science», educational program «Artificial intelligence systems». Vinnytsia: VNTU, 2024. 143 p.

In Ukrainian language. Bibliography: 30 titles; fig.: 28; table: 8.

The master's thesis presents an analysis of the subject area of Customer Relationship Management (CRM) and discusses the main approaches to developing information technologies to improve customer interaction. An analysis of existing systems, methods, and technologies used to address similar tasks has been conducted, justifying the need for the creation of a new information technology that provides a user-friendly and intuitive interface.

A structural model of the CRM information technology has been developed, and a mathematical model for predicting customer behavior has been improved. The use of these models enhances the effectiveness of forecasting and personalizing customer interactions, contributing to the expansion of the technology's functional capabilities.

The developed information technology provides users with a new tool for managing customer relationships, which includes chat and notes, and also enables the improvement of service strategies. The thesis presents research results and an analysis of the effectiveness of the developed system, proving its economic viability and the benefits of its implementation in business processes.

The illustrative part consists of 7 posters with simulation results.

Keywords: customer relationship management, CRM, information technology, automation, personalization, data analysis.

ЗМІСТ

ВСТУП

Актуальність теми.

У динамічному та висококонкурентному сучасному бізнес-середовищі, де технології швидко розвиваються, а очікування клієнтів постійно зростають, компанії стикаються з необхідністю не тільки пропонувати високоякісні товари та послуги, але й надавати неперевершений рівень обслуговування клієнтів. У цьому контексті, управління відносинами з клієнтами (CRM) виходить на передній план як критично важливий інструмент, що дозволяє підприємствам підтримувати та розвивати взаємовигідні відносини зі своїми клієнтами. Однак, традиційні CRM системи часто виявляються недостатньо гнучкими або адаптивними до унікальних потреб кожного бізнесу, що призводить до потреби в інноваціях та розробці нових технологічних рішень [1, 2].

Виходячи з цього, дослідницький інтерес охоплює розробка нових функціональних можливостей, таких як розширені аналітичні інструменти для глибокого розуміння потреб клієнтів, автоматизація маркетингових кампаній для ефективного залучення та утримання клієнтів, а також впровадження інноваційних засобів комунікації, включаючи чати та інструменти для створення нотаток, що дозволять компаніям підтримувати постійний діалог зі своїми клієнтами. Тому розробка інформаційної технології управління відносинами з клієнтами є доцільною та має практичне значення. Така розробка потребує врахування найсучасніших тенденцій у технологіях та адаптації до змінних вимог ринку, щоб забезпечити максимально ефективне та гнучке управління відносинами з клієнтами.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська кваліфікаційна робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем».

Мета і завдання дослідження. Метою магістерської кваліфікаційної роботи є розширення функціональних можливостей інформаційної технології управління відносинами з клієнтами за рахунок впровадження нових інструментів, таких як інтерактивний чат та система нотаток. Це дозволить розробити зручний клієнтський додаток для управління взаємовідносинами з клієнтами, який відповідатиме всім запитам користувачів, буде інтуїтивно зрозумілим та зможе конкурувати з іншими системами-аналогами на ринку.

Для досягнення поставленої мети потрібно розв’язати такі завдання:

- Провести аналіз технічного рівня існуючих систем управління відносинами з клієнтами та обґрунтувати доцільність розробки нової інформаційної технології.

- Розробити математичну модель інформаційної технології управління відносинами з клієнтами.

- Розглянути методи та технології, які використовуються для управління взаємовідносинами з клієнтами.

- Спроектувати архітектуру та структуру програмного модуля управління відносинами з клієнтами.

- Проаналізувати та обґрунтувати вибір інструментів для розробки системи.

- Виконати програмну реалізацію програмного модуля інформаційної технології управління відносинами з клієнтами.

- Провести тестування програми та аналіз отриманих результатів.

- Економічно обґрунтувати доцільність розробки інформаційної технології управління відносинами з клієнтами.

Об’єктом дослідження є процес управління відносинами з клієнтами в організаціях.

Предметом дослідження є програмні засоби проектування та реалізації інформаційної технології управління відносинами з клієнтами.

Методи дослідження. У процесі дослідження використовувалися: аналіз сучасних методів управління відносинами з клієнтами, дослідження методів

обробки та аналізу інформації про клієнтів, вивчення методів аналізу отриманих даних, методи та підходи до розробки WEB-орієнтованих програмних додатків, а також методи об’єктно-орієнтованого програмування. Ці методи дозволили розробити ефективну інформаційну технологію, яка забезпечить покращення взаємодії з клієнтами та підвищить якість обслуговування.

Наукова новизна одержаних результатів полягає в наступному:

- удосконалено математичну модель інформаційної технології управління відносинами з клієнтами, яка відрізняється від існуючих ефективним прогнозуванням поведінки клієнтів на основі їхніх дій та вподобань, що дозволяє підприємствам не лише ефективніше взаємодіяти з клієнтами, але й адаптувати свої стратегії в реальному часі;

- удосконалено інформаційну технологію управління відносинами з клієнтами, яка відрізняється від існуючих впровадженням інтерактивного чату та системи нотаток. Використання цієї технології дозволило покращити комунікацію з клієнтами, а також забезпечить зручність у зберіганні та обробці важливої інформації про взаємодію з ними. Інтерактивний чат забезпечує швидкий доступ до консультацій та підтримки, в той час як система нотаток дозволяє фіксувати важливі моменти спілкування та зберігати історію взаємодій. Ці функції суттєво підвищують ефективність та персоналізацію обслуговування клієнтів, дозволяючи швидко реагувати на їхні потреби та уподобання.

Практичне значення одержаних результатів полягає у наступному:

1. Розроблено алгоритм функціонування інформаційної технології управління відносинами з клієнтами.

2. Розроблено діаграму послідовності взаємодії модулів інформаційної технології управління відносинами з клієнтами.

3. Здійснено програмну реалізацію інформаційної технології управління відносинами з клієнтами.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, коректним застосуванням методів під час доведення наукових положень, порівнянням результатів із

відомими та збіжністю результатів моделювання з результатами, що отримані під час впровадження розроблених програмних засобів.

Особистий внесок магістранта. Результати магістерської кваліфікаційної роботи отримані самостійно. В публікації у співавторстві здобувачу належить дослідження перспектив розробки інформаційної технології управління відносинами з клієнтами [1].

Апробація результатів роботи. Результати досліджень було апробовано на LIII науково-технічній конференції підрозділів Вінницького національного технічного університету (НТКП ВНТУ–2024) м. Вінниці у 2024 р [1].

Публікації. За результатами магістерської кваліфікаційної роботи опубліковано тези доповідей на науково-технічній конференції [1], отримано свідоцтво про реєстрацію авторського права на твір у формі комп'ютерної програми – «Інформаційна технологія управління відносинами з клієнтами» [2] та подано до друку статтю в журнал «Вісник Хмельницького національного університету», серія: «Технічні науки», який входить до переліку наукових фахових видань МОНУ (категорія Б) на тему: «Розширення функціональних можливостей програмного модуля інформаційної технології управління відносинами з клієнтами».

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ УПРАВЛІННЯ ВІДНОСИНАМИ З КЛІЄНТАМИ

1.1 Аналіз предметної галузі CRM-систем

CRM-системи, що спрямовані на управління відносинами з клієнтами, є важливим інструментом для сучасних компаній, які прагнуть підвищити ефективність взаємодії з клієнтами та забезпечити довгострокові стосунки. Вони дозволяють централізувати дані про клієнтів та організувати їх таким чином, щоб різні відділи компанії могли легко використовувати цю інформацію для прийняття рішень та оптимізації бізнес-процесів. CRM-системи мають широкий спектр функцій і допомагають автоматизувати процеси у сферах продажів, маркетингу, обслуговування клієнтів та управління проєктами [3].

Ці системи класифікуються за кількома типами, кожен з яких має свої особливості. Наприклад, операційні CRM забезпечують підтримку повсякденних операцій з клієнтами, наприклад, продажі та маркетинг. Вони дозволяють автоматизувати процеси продажів і забезпечують управління лідами та клієнтами на всіх етапах взаємодії. Аналітичні CRM допомагають збирати та аналізувати дані про клієнтів для прийняття обґрунтованих рішень. Завдяки їм компанії отримують можливість краще розуміти потреби клієнтів, прогнозувати попит і вдосконалювати стратегії взаємодії. Колаборативні CRM зосереджені на покращенні взаємодії між відділами компанії та сприяють узгодженій роботі над задоволенням потреб клієнтів.

Крім того, CRM-системи дозволяють компаніям будувати повну історію відносин з клієнтами. Це означає, що співробітники мають доступ до всіх попередніх контактів з клієнтом, можуть бачити його покупки, запити та уподобання. Такий підхід сприяє створенню персоналізованого досвіду обслуговування, що підвищує рівень задоволеності клієнтів. Важливим аспектом є можливість автоматизації маркетингових кампаній, коли система дозволяє

налаштувати персоналізовані розсилки та відстежувати їх результати, що дозволяє точніше орієнтуватися на потреби клієнтів.

CRM-системи значно підвищують продуктивність завдяки автоматизації рутинних завдань та можливості швидкого доступу до інформації про клієнтів. Це дозволяє зосередитися на стратегічно важливих завданнях, таких як покращення якості обслуговування та підвищення ефективності продажів. Система також забезпечує інструменти для аналізу результатів та прогнозування майбутніх продажів, що дозволяє компаніям приймати більш виважені рішення та будувати стратегії розвитку.

Проте CRM-системи мають і певні виклики у впровадженні та використанні. Зокрема, це стосується складнощів у налаштуванні та адаптації системи під потреби компанії. Для повноцінного використання CRM необхідно провести навчання співробітників і, можливо, залучити спеціалістів для інтеграції системи з існуючими процесами. Додатково потрібно врахувати питання конфіденційності та захисту даних клієнтів, оскільки система зберігає велику кількість персональної інформації. Недотримання стандартів безпеки може призвести до витоку даних, що загрожує репутації компанії.

Перспективи розвитку CRM-систем також є багатообіцяючими. Інтеграція штучного інтелекту дозволяє системам краще аналізувати поведінку клієнтів, надавати персоналізовані рекомендації та автоматизувати рутинні завдання. Використання Big Data сприяє глибшому аналізу потреб клієнтів і прогнозуванню майбутніх тенденцій. Багато сучасних CRM пропонують мобільні застосунки, що дозволяє працювати з клієнтами в будь-який час і в будь-якому місці, забезпечуючи більшу гнучкість для менеджерів. Інтеграція з соціальними мережами стає все більш популярною, оскільки дозволяє отримувати зворотний зв'язок від клієнтів та відстежувати актуальні тренди, що відкриває додаткові можливості для залучення нових клієнтів.

Таким чином, CRM-системи є незамінним інструментом для сучасного бізнесу, що дозволяє будувати довготривалі відносини з клієнтами та забезпечувати їм якісний сервіс. Вони сприяють збільшенню рівня задоволеності

клієнтів, підвищують ефективність бізнес-процесів та дозволяють компаніям зберігати конкурентоспроможність.

1.2 Огляд відомих програмних систем-аналогів

CRM-система − це програмне забезпечення, де головний акцент знаходиться на управлінні взаємовідносинами з клієнтами. Компанія використовує CRM для збору, зберігання та аналізу даних про клієнтів, автоматизації процесів продажів та маркетингу, а також для покращення обслуговування клієнтів [4].

Salesforce є однією з найбільш потужних CRM-систем у світі, яку використовують для управління взаєминами з клієнтами та оптимізації бізнес-процесів. Її унікальність полягає в комплексному підході до роботи з клієнтами, що охоплює всі етапи взаємодії — від початкового збору контактної інформації до завершення продажу та надання подальшої підтримки.

Завдяки Salesforce компанії можуть організовувати повний цикл комунікації з клієнтами. Це допомагає систематизувати дані, відслідковувати процеси взаємодії з клієнтами та створювати ефективні стратегії для підвищення продажів. Основна функціональність CRM включає управління контактами, угодами та завданнями, а також ведення бази даних клієнтів, що дозволяє компаніям мати повний доступ до історії взаємодій з кожним клієнтом.

Salesforce також виділяється широкими можливостями для автоматизації бізнес-процесів. Вона пропонує інтегровані модулі для автоматизації маркетингових кампаній, що дозволяє створювати персоналізовані повідомлення та ефективно керувати клієнтськими даними. Такі інструменти допомагають значно полегшити процеси залучення нових клієнтів і підтримки існуючих.

Крім цього, Salesforce забезпечує високий рівень гнучкості завдяки своїм можливостям для розробки додатків. Компанії можуть розробляти кастомізовані рішення безпосередньо на платформі Salesforce, щоб адаптувати її під свої

потреби. Це значно розширює можливості системи і робить її підходящою для бізнесів різних розмірів та індустрій.

Потрібно також зазначити аналітичні можливості Salesforce. Вбудовані інструменти аналітики дозволяють компаніям отримувати глибокі бізнес-інсайти, аналізувати поведінку клієнтів і приймати обґрунтовані рішення на основі зібраних даних. Це допомагає не тільки відслідковувати поточні показники, але й прогнозувати майбутні тенденції, що є важливим елементом успіху будь-якого бізнесу.

Salesforce також підтримує інтеграцію з іншими сервісами та додатками, що робить її дуже зручною для використання в комплексних бізнес-системах [5]. Загальний вигляд інтерфейсного вікна сайту продемонстровано на рисунку 1.1.

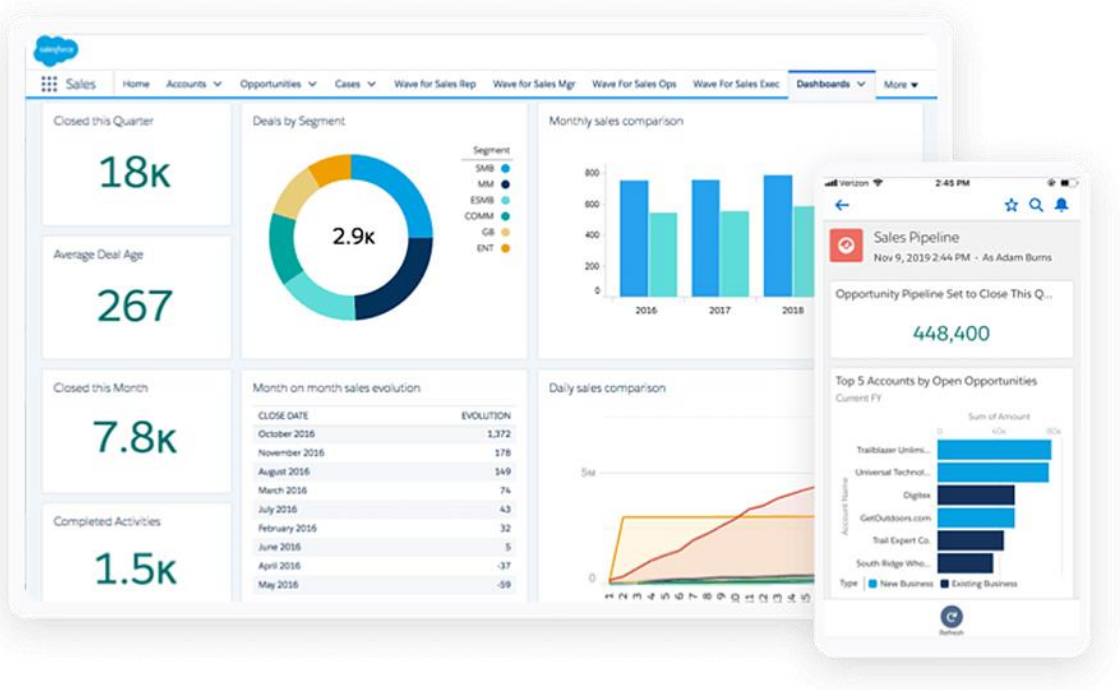

Рисунок 1.1 – Загальний вигляд інтерфейсного вікна сайту CRM-системи Saleforce

HubSpot став справжнім проривом у світі цифрового маркетингу та управління відносинами з клієнтами. Ця платформа пропонує комплексне рішення, яке об'єднує всі ключові аспекти взаємодії з клієнтами під одним дахом.

Основна філософія HubSpot базується на концепції inbound-маркетингу, де головна увага приділяється створенню цінного контенту та побудові довгострокових відносин з клієнтами, замість агресивного просування продуктів. Платформа надає потужні інструменти для створення та управління контентом, включаючи блоги, лендінги та email-розсилки.

У сфері продажів HubSpot пропонує розширені можливості для відстеження та аналізу взаємодій з потенційними клієнтами. Система автоматично збирає та організовує інформацію про контакти, компанії та угоди, допомагаючи sales-менеджерам приймати більш обґрунтовані рішення та ефективніше вести переговори.

Особливу увагу варто приділити можливостям автоматизації в HubSpot. Платформа дозволяє налаштовувати складні послідовності дій, які автоматично запускаються при виконанні певних умов. Це може бути відправка персоналізованих email-повідомлень, оновлення статусу угоди або створення завдань для менеджерів.

HubSpot також відрізняється потужними аналітичними можливостями. Система надає детальні звіти про ефективність маркетингових кампаній, продуктивність sales-команди та загальну результативність бізнес-процесів. Це дозволяє компаніям постійно оптимізувати свої стратегії на основі реальних даних.

Важливою перевагою HubSpot є його гнучкість та масштабованість. Платформа підходить як для невеликих стартапів, так і для великих корпорацій, пропонуючи різні тарифні плани та можливості розширення функціоналу. Крім того, HubSpot має велику кількість інтеграцій з іншими популярними бізнес-інструментами, що дозволяє створити єдину екосистему для управління бізнесом.

Навчання роботі з HubSpot також заслуговує окремої уваги. Компанія надає велику кількість освітніх матеріалів, включаючи онлайн-курси, сертифікації та документацію, що допомагає командам швидко освоїти платформу та почати використовувати її максимально ефективно.

Постійний розвиток та вдосконалення платформи є ще однією сильною стороною HubSpot. Регулярно додаються нові функції та можливості, які відповідають сучасним потребам бізнесу та тенденціям у сфері маркетингу та продажів. Це дозволяє компаніям залишатися конкурентоспроможними та ефективно адаптуватися до змін на ринку [6]. Загальний вигляд інтерфейсного вікна сайту продемонстровано на рисунку 1.2.

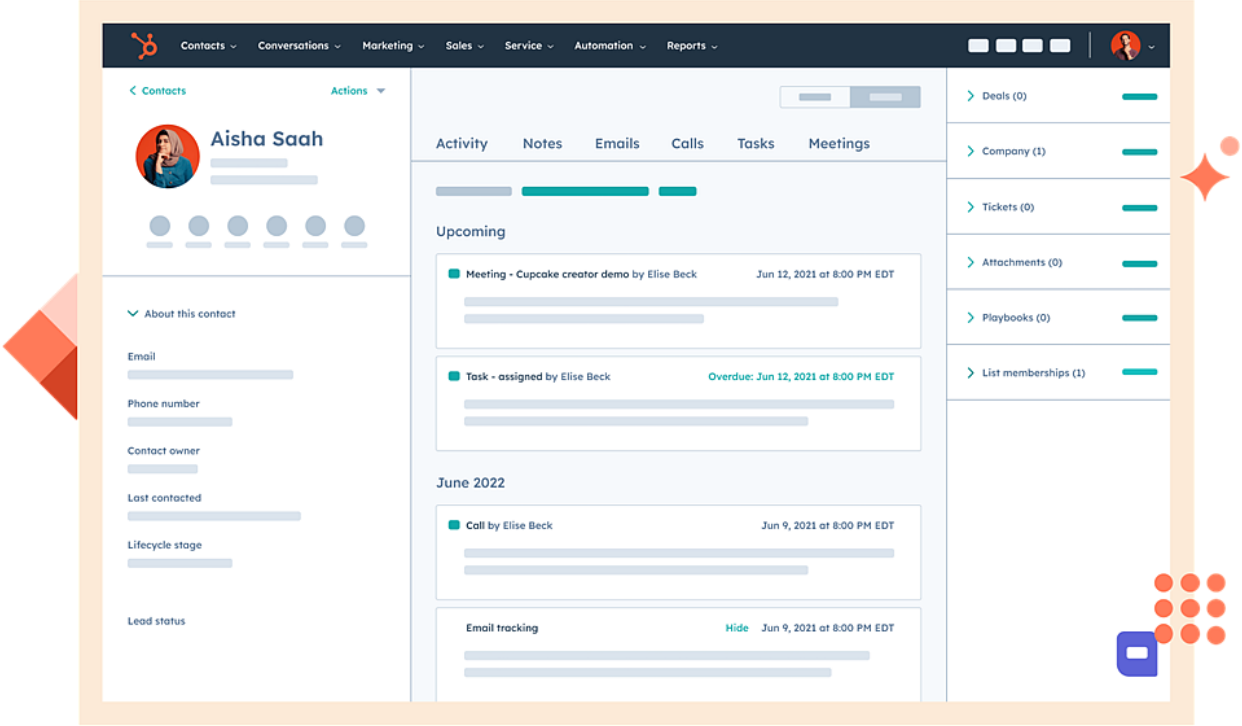

Рисунок 1.2 – Загальний вигляд інтерфейсного вікна сайту CRM-системи HubSpot

Zoho CRM виділяється на ринку систем управління відносинами з клієнтами своєю універсальністю та глибокою інтеграцією з іншими бізнес-інструментами. Ця платформа створена з урахуванням потреб як малого, так і середнього бізнесу, пропонуючи гнучкі рішення для різних галузей та масштабів діяльності.

Основною перевагою Zoho CRM є її здатність автоматизувати рутинні процеси продажів та маркетингу. Система дозволяє створювати складні автоматизовані робочі процеси, які допомагають оптимізувати щоденні операції

та звільнити час співробітників для більш важливих завдань. Це включає автоматичну відправку email-повідомлень, оновлення статусів угод та створення нагадувань.

У сфері управління продажами Zoho CRM надає потужні інструменти для відстеження потенційних клієнтів та управління воронкою продажів. Система допомагає sales-менеджерам ефективно планувати свою роботу, відстежувати прогрес у досягненні цілей та прогнозувати майбутні продажі на основі наявних даних.

Маркетингові можливості Zoho CRM дозволяють створювати та керувати різноманітними кампаніями. Платформа надає інструменти для email-маркетингу, соціальних медіа та WEB-форм, що допомагає залучати нових клієнтів та підтримувати зв'язок з існуючими. Особливо цінною є можливість сегментації клієнтської бази для проведення таргетованих кампаній.

Аналітичні можливості Zoho CRM заслуговують окремої уваги. Система пропонує широкий набір інструментів для створення звітів та dashboards, які допомагають відстежувати ключові показники ефективності та приймати обґрунтовані рішення. Користувачі можуть створювати власні звіти та налаштовувати їх відповідно до специфічних потреб свого бізнесу.

Мобільність є ще однією сильною стороною Zoho CRM. Платформа пропонує зручні мобільні додатки для iOS та Android, що дозволяє співробітникам отримувати доступ до важливої інформації та працювати з системою навіть у дорозі. Це особливо важливо для sales-команд, які багато часу проводять на виїзних зустрічах.

Безпека даних у Zoho CRM реалізована на найвищому рівні. Система забезпечує надійний захист конфіденційної інформації, включаючи шифрування даних, багатофакторну автентифікацію та детальне управління правами доступу для різних користувачів та ролей.

Особливу цінність представляє екосистема Zoho, до якої входить CRM. Платформа легко інтегрується з іншими продуктами Zoho, такими як Zoho Books, Zoho Projects та Zoho Campaigns, створюючи єдиний комплексний

інструмент для управління бізнесом. Крім того, доступні інтеграції з популярними сторонніми сервісами та додатками.

Підтримка користувачів Zoho CRM знаходиться на високому рівні. Компанія надає детальну документацію, навчальні матеріали та відео-уроки, а також забезпечує технічну підтримку через різні канали комунікації. Це допомагає новим користувачам швидко освоїти систему та ефективно використовувати її можливості.

Постійне оновлення та вдосконалення функціоналу є частиною стратегії розвитку Zoho CRM. Регулярно додаються нові функції та можливості, які відповідають сучасним тенденціям у сфері управління клієнтськими відносинами та допомагають бізнесу залишатися конкурентоспроможним [7]. Загальний вигляд інтерфейсного вікна сайту продемонстровано на рисунку 1.3.

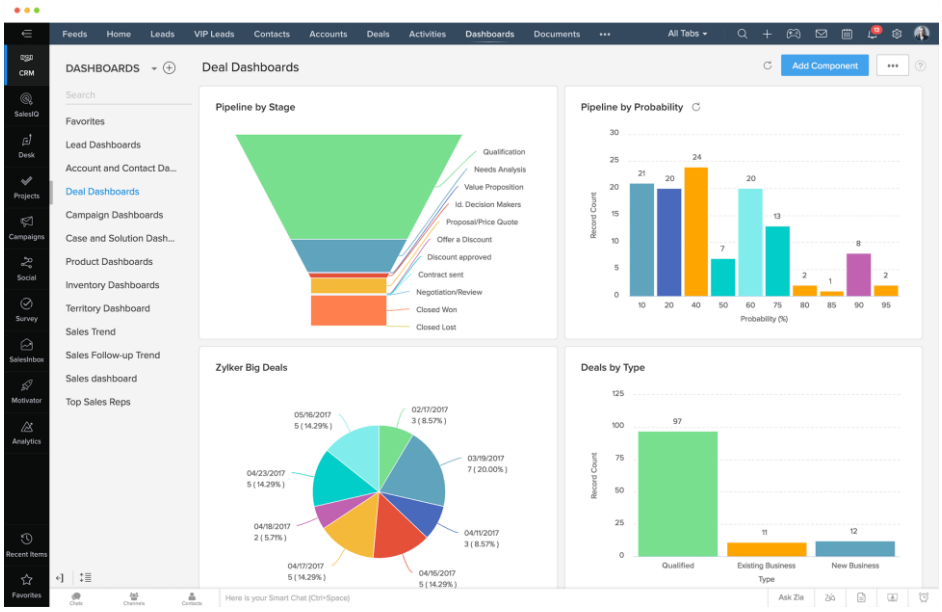

Рисунок 1.3– Загальний вигляд інтерфейсного вікна сайту CRM-системи Zoho CRM

Pipedrive представляє собою унікальне рішення у світі CRM-систем, зосереджуючись передусім на оптимізації процесу продажів. Ця платформа була створена з урахуванням реальних потреб sales-менеджерів та їх щоденних завдань, що робить її особливо ефективною для команд продажів різного масштабу.

Ключовою особливістю Pipedrive є її візуально привабливий та інтуїтивно зрозумілий інтерфейс воронки продажів. Система дозволяє користувачам легко відстежувати прогрес кожної угоди, переміщуючи картки між різними етапами продажу простим перетягуванням. Це значно спрощує процес управління продажами та робить його більш наочним для всіх учасників процесу.

Автоматизація рутинних завдань у Pipedrive реалізована з особливою увагою до деталей. Платформа дозволяє налаштовувати автоматичні нагадування, відправку email-повідомлень та оновлення статусів угод, що допомагає sales-менеджерам зосередитися на найважливіших аспектах своєї роботи – спілкуванні з клієнтами та закритті угод.

Система пропонує потужні інструменти для планування та відстеження активностей. Користувачі можуть легко планувати зустрічі, дзвінки та інші взаємодії з клієнтами, а також отримувати своєчасні нагадування про заплановані події. Це допомагає підтримувати високий рівень організації роботи та не пропускати важливі можливості для продажу.

Аналітичні можливості Pipedrive дозволяють отримувати детальну інформацію про ефективність продажів. Система надає різноманітні звіти та метрики, які допомагають оцінювати продуктивність команди, виявляти проблемні місця в процесі продажів та приймати обґрунтовані рішення щодо оптимізації стратегії.

Мобільний додаток Pipedrive забезпечує повноцінний доступ до всіх функцій системи з мобільних пристроїв. Це особливо важливо для sales-менеджерів, які часто працюють віддалено або проводять зустрічі з клієнтами поза офісом. Додаток дозволяє оперативно оновлювати інформацію про угоди та контакти, планувати активності та відстежувати прогрес продажів.

Інтеграційні можливості Pipedrive вражають своєю різноманітністю. Система легко інтегрується з популярними інструментами для email-маркетингу, бухгалтерського обліку, комунікації та управління проектами. Це дозволяє створити єдину екосистему для ефективного управління бізнес-процесами.

Особлива увага в Pipedrive приділяється безпеці даних. Система забезпечує надійний захист конфіденційної інформації про клієнтів та угоди, використовуючи сучасні методи шифрування та контролю доступу. Це дозволяє компаніям безпечно зберігати та обробляти важливу бізнес-інформацію.

Навчання роботі з Pipedrive відбувається швидко та ефективно завдяки детальній документації та навчальним матеріалам. Компанія пропонує різноманітні ресурси для навчання, включаючи відео-уроки, вебінари та базу знань, що допомагає новим користувачам швидко освоїти систему.

Регулярні оновлення та вдосконалення функціоналу є важливою частиною розвитку Pipedrive. Платформа постійно адаптується до змінних потреб користувачів та нових тенденцій у сфері продажів, додаючи нові функції та покращуючи існуючі можливості.

Підтримка користувачів у Pipedrive організована на високому рівні. Компанія надає професійну технічну підтримку через різні канали комунікації, що дозволяє швидко вирішувати будь-які питання, які можуть виникнути під час використання системи [8]. Загальний вигляд інтерфейсного вікна сайту продемонстровано на рисунку 1.4.

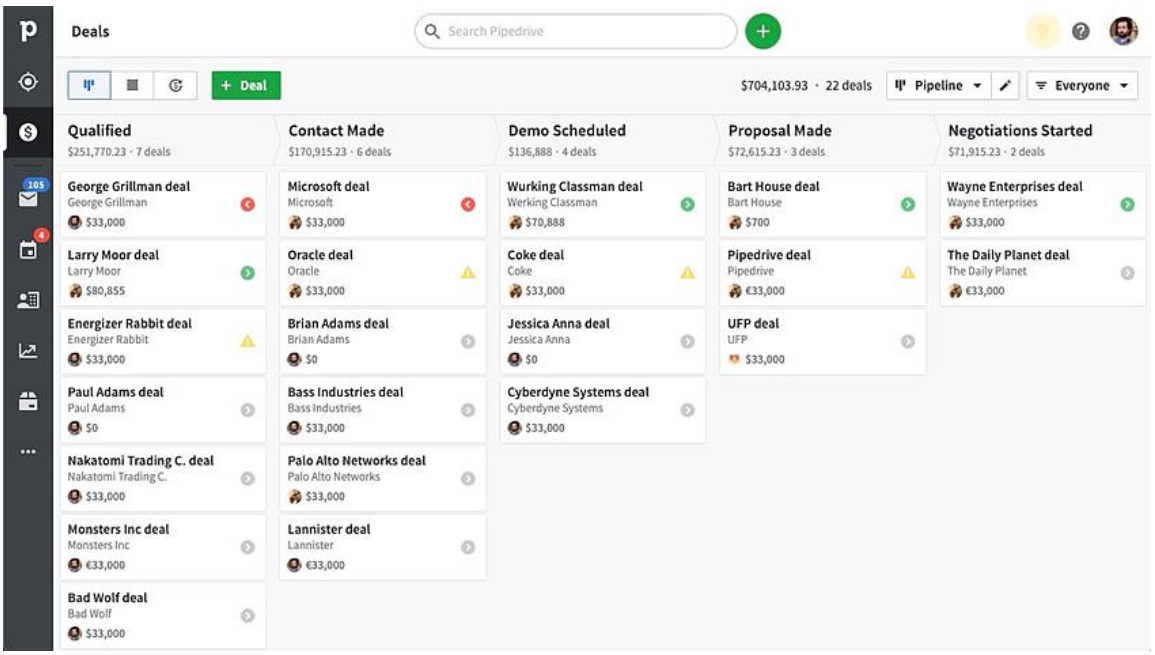

Рисунок 1.4 – Загальний вигляд інтерфейсного вікна сайту CRM-системи Pipedrive

1.3 Огляд методів та програмних засобів для управління відносинами з клієнтами

Протягом останнього десятиліття управління клієнтськими відносинами зазнало кардинальних змін завдяки стрімкому розвитку технологій. Незважаючи на численні інновації, головна мета залишається незмінною − виявлення ключових факторів, що впливають на якість взаємодії з клієнтами. Сучасні компанії все більше усвідомлюють, що успіх бізнесу безпосередньо залежить від здатності будувати міцні та довготривалі відносини зі споживачами.

В основі сучасного CRM лежить глибокий аналіз факторів впливу на клієнтський досвід. Компанії ретельно вивчають як внутрішні показники, зокрема якість обслуговування та сприйняття бренду, так і зовнішні чинники − споживчу поведінку та вподобання клієнтів. Важливим елементом цього процесу є порівняння очікуваних результатів із реальними показниками, що дозволяє приймати виважені управлінські рішення. Особлива увага приділяється аналізу зворотного зв'язку від клієнтів, який допомагає виявляти проблемні моменти та вдосконалювати сервіс [9].

Значна увага приділяється оцінці споживчого попиту через аналіз клієнтської активності. Це включає вивчення звернень до служби підтримки, моніторинг відгуків у соціальних мережах та математичну обробку зібраних даних. Такий підхід дозволяє краще розуміти потреби клієнтів та прогнозувати їхню поведінку. Компанії активно впроваджують автоматизовані системи збору та аналізу даних, які дозволяють оперативно реагувати на зміни у споживчих настроях та адаптувати свої стратегії відповідно до нових викликів.

Сучасний погляд на CRM передбачає його розгляд як частини ширшої бізнес-системи. При цьому враховуються ринкові умови, конкурентне середовище та актуальні соціальні тренди. Це дає можливість формувати більш повну картину факторів, що впливають на задоволеність та лояльність клієнтів. Важливо розуміти, що успішна CRM-стратегія має бути гнучкою та адаптивною, здатною швидко реагувати на зміни у зовнішньому середовищі.

Технологічний прогрес приніс у сферу CRM використання штучних нейронних мереж. Ці передові інструменти здатні аналізувати величезні масиви даних, виявляти приховані закономірності та прогнозувати майбутні потреби клієнтів. Особливо цінною є здатність нейромереж адаптуватися до швидких змін у поведінці споживачів. Використання машинного навчання дозволяє створювати персоналізовані пропозиції та покращувати якість обслуговування на індивідуальному рівні.

Поряд з інноваційними підходами, традиційні математичні методи зберігають свою актуальність у CRM-аналітиці. Вони забезпечують об'єктивну оцінку ефективності різних стратегій роботи з клієнтами та допомагають у прогнозуванні результатів. Хоча ці методи можуть бути менш гнучкими порівняно з новітніми технологіями, вони залишаються надійним фундаментом для прийняття управлінських рішень. Важливо відзначити, що найкращих результатів досягають компанії, які вміло поєднують класичні та інноваційні підходи.

Сучасні CRM-системи все частіше інтегруються з іншими бізнес-процесами компанії, створюючи єдиний інформаційний простір. Це дозволяє забезпечити безперервність взаємодії з клієнтами та підвищити ефективність роботи всіх підрозділів. Інтеграція даних з різних джерел допомагає створити повний профіль клієнта та краще розуміти його потреби та очікування.

Особлива роль у розвитку CRM належить цифровим каналам комунікації. Соціальні мережі, месенджери та інші онлайн-платформи стають важливими джерелами інформації про клієнтів та їхню поведінку. Компанії активно використовують ці канали для побудови діалогу з клієнтами, отримання зворотного зв'язку та поширення інформації про свої продукти та послуги.

Успіх у сфері управління клієнтськими відносинами сьогодні залежить від вміння компаній ефективно поєднувати різні підходи та методи. Організації, які знаходять оптимальний баланс між традиційними та інноваційними інструментами, здатні краще задовольняти потреби клієнтів та досягати кращих бізнес-результатів у конкурентному середовищі. Важливо також постійно

навчати персонал новим технологіям та методам роботи з клієнтами, адже саме людський фактор часто відіграє вирішальну роль у побудові успішних клієнтських відносин.

Розвиток технологій штучного інтелекту та аналітики великих даних відкриває нові можливості для вдосконалення CRM-систем. Компанії можуть використовувати предиктивну аналітику для передбачення поведінки клієнтів, автоматизовані системи для оптимізації взаємодії та персоналізовані рекомендації для підвищення лояльності [10]. Це дозволяє створювати більш ефективні стратегії утримання клієнтів та збільшення їхньої цінності для бізнесу.

Переваги та недоліки розглянутих методів можна представити за допомогою таблиці. 1.1.

Таблиця 1.1 – Переваги та недоліки методологій дослідження управління відносинами з клієнтами

Метод	Переваги	Недоліки
Фундаментальний аналіз	Дозволяє отримати цілісну картину взаємодії з клієнтами. Допомагає виявити ключові фактори, що впливають на лояльність клієнтів.	Вимагає значних ресурсів для збору та аналізу даних. Може не враховувати швидко змінювані тенденції на ринку.
Кількісна оцінка попиту	Простота збору та обробки даних. Легка інтеграція в математичні моделі.	Необхідність у досвіді для правильної інтерпретації результатів. Можливість «шуму» інформації, що може спотворити результати.
Системний підхід	Дозволяє розглядати CRM як частину більшої системи. Враховує взаємозв'язки між різними елементами ринку.	Може бути складним для реалізації через різноманіття факторів, які потрібно врахувати. Вимагає детального аналізу, що може займати багато часу.

Продовження таблиці 1.1

Метод	Переваги	Недоліки
Використання нейронних мереж	Може обробляти великі обсяги даних і виявляти складні закономірності. Адаптується до швидких змін у поведінці споживачів.	Потребує висококваліфікованих фахівців для налаштування та навчання моделей. Складність у поясненні результатів та процесу прийняття рішень.
Традиційні методи	Об'єктивність та точність в аналізі даних. Використання перевірених статистичних моделей.	Можуть бути недостатніми для швидко змінюваних, нелінійних процесів. Обмежена здатність до прогнозування в умовах динамічних ринкових змін.

1.4 Постановка задачі дослідження

Поєднання різноманітних видів діяльності з IT-технологіями все частіше набуває широкого застосування та суттєво удосконалює різні сфери нашого життя. У сучасному бізнес-середовищі управління відносинами з клієнтами (CRM) відіграє ключову роль у забезпеченні успішності компаній. Завдяки стрімкому розвитку інформаційних технологій, CRM-системи стають все більш ефективними та функціональними.

Сьогодні бізнеси стикаються з викликом: необхідність мати простий, безкоштовний та одночасно багатофункціональний інструмент для управління взаємовідносинами з клієнтами. Це підкреслює важливість розробки сучасних технологій, які забезпечують точний аналіз споживчих даних, підтримують належну комунікацію з клієнтами та сприяють вдосконаленню обслуговування.

Загальний опис функціоналу інформаційних технологій для управління відносинами з клієнтами включає:

- Реалізацію входу до інформаційної системи за допомогою графічного інтерфейсу, що забезпечує зручність користування.

- Можливість вибору періоду для аналізу даних, що дає змогу адаптувати систему до потреб конкретного бізнесу.

- Виведення результуючих графіків та звітів, що дозволяє швидко оцінити ефективність взаємодії з клієнтами.

- Збереження та виведення історії взаємодій з клієнтами, що допомагає виявляти проблемні моменти та вдосконалювати стратегії обслуговування.

У рамках цієї роботи необхідно розробити інформаційну технологію для управління відносинами з клієнтами, яка надасть користувачам інструменти для аналізу даних та оптимізації своїх стратегій обслуговування. Основна мета розробки полягає в підвищенні точності прогнозування потреб клієнтів та зручності використання системи. Це досягається завдяки впровадженню гнучкого механізму вибору періодів для аналізу, що зазвичай є платною функцією в інших аналогічних програмах, а також можливістю перегляду власної історії взаємодії з клієнтами.

Розроблений програмний модуль буде інтегрований із зовнішніми базами даних, що дозволить автоматично оновлювати інформацію про клієнтів та їхні уподобання. Для цього планується використовувати API популярних платформ, таких як Firebase, для забезпечення швидкого доступу до даних та їх обробки.

Таким чином, для вирішення проблеми управління відносинами з клієнтами необхідно вирішити такі основні завдання:

1. Проектування структури інформаційної технології для управління відносинами з клієнтами.
2. Розробка математичної моделі аналізу та прогнозування потреб клієнтів.
3. Реалізація програмного модуля управління відносинами з клієнтами.
4. Проведення тестування інформаційної технології для виявлення її ефективності та зручності [12].

Отже, метою дослідження є розробка інформаційної технології, яка дозволить компаніям ефективно управляти відносинами з клієнтами, забезпечуючи точний аналіз та своєчасне реагування на їхні потреби.

1.5 Висновок до розділу 1

У першому розділі було розглянуто основні поняття управління відносинами з клієнтами, проведено аналіз існуючих технологічних рішень для цього напрямку та доведено актуальність розробки нової інформаційної технології для управління взаємовідносинами з клієнтами. Було проаналізовано системи-аналоги, такі як CRM-платформи, які надають різноманітні інструменти для аналізу клієнтських даних та підтримки комунікації з ними. Однак виявлено, що існуючі рішення часто є комерційними сервісами, які не розкривають усіх деталей реалізації або мають обмежений функціонал, що не дозволяє користувачам отримувати повну інформацію для прийняття рішень.

Подана інформація стала основою для розробки інформаційної технології управління відносинами з клієнтами, яка надасть користувачам більше можливостей для аналізу даних та ефективного управління взаємодією. Завдання, пов'язані з управлінням відносинами з клієнтами, мають схожість із завданнями прогнозування інших бізнес-процесів, проте вони включають специфічні аспекти, які потребують окремого розгляду. Важливість соціальних факторів у поведінці клієнтів є суттєвою, але часто недооцінюється в математичному аналізі, тому створення нової інформаційної технології є доцільним та актуальним.

У розділі також були представлені основні методи та програмні засоби, які можуть бути використані для розробки цієї технології. З метою створення ефективної системи було обрано метод прогнозування на основі математичних моделей, що використовують часові ряди. Цей метод дозволяє отримувати прогнози з необхідною точністю при наявності актуальних та цілісних даних, що є важливим для галузі управління відносинами з клієнтами.

Також були визначені задачі дослідження та вимоги клієнтів, це допоможе у подальшому формуванні структури системи, виборі доцільних технологій та мов програмування для реалізації інформаційної технології управління відносинами з клієнтами.

2 МОДЕЛЮВАННЯ ІНФОРМАЦІЇНОЇ ТЕХНОЛОГІЇ УПРАВЛІННЯ ВІДНОСИНАМИ З КЛІЄНТАМИ

2.1 Обгрунтування вибору методів реалізації інформаційної технології управління відносинами з клієнтами

Протягом останніх років інтерес до систем управління відносинами з клієнтами (CRM) значно зріс серед дослідників та бізнес-спільноти. Хоча цій темі присвячено чимало досліджень, ключовим викликом залишається визначення комплексу факторів та метрик, що впливають на результативність CRM-рішень. Також важливим є аналіз еволюції галузі, виявлення ключових показників ефективності та розуміння взаємозв'язків між різними аспектами клієнтського досвіду.

На сьогодні відсутня універсальна методика, яка б всебічно охоплювала всі нюанси CRM з точки зору аналітики та прогнозування поведінки клієнтів. Фахівці, що досліджують цю сферу, зосереджуються на виявленні ключових факторів успіху CRM-стратегій, але цього недостатньо для повноцінного розуміння динаміки клієнтських відносин [13].

Один з підходів до оцінки CRM-стратегій − це комплексний аналіз. Він передбачає глибоке вивчення реальної цінності CRM-ініціатив через призму різноманітних економічних, фінансових та інших кількісних і якісних параметрів. Аналітики розглядають широкий спектр факторів, включаючи загальноекономічні тенденції та специфіку галузі, які можуть вплинути на успішність впровадження CRM. Мета такого аналізу − отримати конкретні показники, що демонструють ефективність або неефективність обраних CRM-підходів.

Цінність CRM-систем для бізнесу визначається ринковими механізмами. Попит на конкретні функції та можливості CRM прямо впливає на їх значущість для компаній. Цей попит формується на основі практичних переваг, які надає CRM-система. Поява інноваційного CRM-рішення, здатного суттєво підвищити

задоволеність клієнтів, може викликати стрімке зростання інтересу до нього. На формування попиту впливають успішні приклади впровадження, інноваційні розробки та анонси від лідерів ринку. Позитивна репутація CRM-рішень, схвальні відгуки користувачів та історії успішних впроваджень сприяють зростанню попиту на певні CRM-платформи. Зі збільшенням обізнаності про переваги якісних CRM-рішень, все більше компаній прагнуть інвестувати в ці технології для вдосконалення своєї взаємодії з клієнтами [14].

Крім того, на ефективність CRM-систем впливає їх поширення та адаптація в різних галузях бізнесу. Останнім часом CRM-рішення отримали широке висвітлення у світових ділових ЗМІ, що безумовно сприяло зростанню їх популярності та впровадження.

Оскільки успішність CRM-стратегій значною мірою залежить від їх відповідності потребам бізнесу та клієнтів, одним зі способів прогнозування ефективності CRM є кількісна оцінка попиту на певні функції та можливості, а також подальший аналіз впливу цього попиту на розвиток CRM-технологій. Висновки про попит та популярність тих чи інших CRM-рішень можна зробити, використовуючи дані про частоту обговорення цієї теми в інтернеті. Наприклад, одним з таких показників може бути кількість згадувань певної CRM-системи у професійних соціальних мережах, як LinkedIn.

Переваги цього показника:

- Кількісний характер;
- Технічна доступність для отримання чи обчислення;
- Простота використання у математичних моделях.

Недоліки:

- Можливі обмеження доступу до даних;
- Обмеження щодо обсягу та швидкості отримання інформації.

3. Ще один метод дослідження розвитку CRM-технологій – системний підхід. Цей метод розглядає ринок CRM-рішень як систему, яка впливає на розвиток інших аспектів бізнесу, будучи частиною загальної системи управління підприємством. Ринок CRM є відкритою, складною, динамічною та керованою

системою, яка постійно розвивається. У цій системі є суб'єкти (розробники CRM-систем, консультанти) та об'єкти керування (компанії-користувачі CRM), а розвиток системи здійснюється сукупністю елементів, що утворюють механізм постійного вдосконалення та адаптації CRM-рішень до потреб бізнесу [15].

Такий підхід дозволяє комплексно оцінювати перспективи розвитку CRM-технологій, враховуючи взаємозв'язки між різними аспектами управління відносинами з клієнтами та загальними бізнес-процесами підприємств.

Одним з перспективних напрямків розвитку CRM-технологій є використання нейронних мереж для прогнозування поведінки клієнтів та оптимізації взаємодії з ними. Штучні нейронні мережі (ANN) як математичні моделі, засновані на принципах функціонування біологічних нейронних мереж, можуть бути ефективно застосовані в CRM-системах.

У контексті CRM, нейронні мережі можуть використовуватися для:

- Прогнозування ймовірності покупки певним клієнтом;
- Сегментації клієнтської бази;
- Персоналізації маркетингових пропозицій;
- Прогнозування відтоку клієнтів;
- Оптимізації ціноутворення.

ANN в CRM-системах представляють собою взаємопов'язану мережу простих процесорів (штучних нейронів), які обробляють вхідні дані про клієнтів, їхню поведінку та історію взаємодії з компанією. Хоча кожен окремий нейрон виконує прості операції, їх колективна робота дозволяє вирішувати складні завдання аналізу та прогнозування в сфері управління відносинами з клієнтами.

5. Крім нейронних мереж, у розвитку CRM-технологій також застосовуються традиційні методи аналізу та прогнозування з використанням математичних моделей. Ці методи, які успішно використовуються в інших сферах (прогнозування погоди, продажів, захворюваності), можуть бути адаптовані для потреб CRM.

Переваги використання математичних моделей у CRM:

- Об'єктивність: дозволяють уникнути впливу суб'єктивних факторів при прийнятті рішень;

- Точність: використання точних історичних даних для навчання моделей і виявлення закономірностей;

- Масштабованість: можливість обробки великих обсягів даних про клієнтів;

- Автоматизація: зменшення потреби в ручному аналізі даних.

Математичні моделі в CRM можуть застосовуватися для:

- Прогнозування життєвого циклу клієнта;

- Оцінки ймовірності успіху крос-продажів;

- Оптимізації маркетингових кампаній;

- Прогнозування попиту на продукти чи послуги.

Використання цих передових методів аналізу та прогнозування в CRM-системах дозволяє компаніям більш ефективно управляти відносинами з клієнтами, підвищувати лояльність та збільшувати прибутковість. Розвиток цих технологій є ключовим напрямком у вдосконаленні інформаційних технологій управління відносинами з клієнтами [16].

Традиційні методи аналізу даних у CRM-системах, які базуються на припущеннях стаціонарності математичних моделей, часто не відображають швидкозмінні, нелінійні процеси взаємодії з клієнтами. Тому сучасні CRM-технології все частіше звертаються до нових моделей та методів дослідження, зокрема до прогнозування на основі часових рядів.

У контексті CRM часовий ряд може представляти собою:

- Послідовність транзакцій клієнта;
- Історію взаємодій клієнта з компанією;
- Динаміку зміни лояльності клієнта;
- Послідовність маркетингових кампаній та їх результатів.

Особливості часових рядів у CRM:

1. Взаємозалежність показників: наприклад, частота покупок клієнта в різні періоди часу може бути взаємопов'язана.

2. Різна інформативність показників: недавні взаємодії з клієнтом можуть бути більш значущими для прогнозування його майбутньої поведінки.

3. Обмежена корисність історичних даних: зі зміною ринкових умов або поведінки клієнта старі дані можуть втрачати актуальність.

Аналіз часових рядів у CRM може включати:

1. Побудову та вивчення графіків клієнтської активності.

2. Перевірку рядів на стаціонарність: чи змінюються середні показники взаємодії з клієнтом з часом?

3. Розрахунок числових характеристик: середня частота покупок, варіація суми покупок, кореляція між різними типами взаємодій.

4. Виділення тренду: наприклад, зростання або спадання активності клієнта.

5. Аналіз сезонності: чи є періодичні коливання в активності клієнта?

Переваги використання аналізу часових рядів у CRM:

- Можливість виявлення складних патернів у поведінці клієнтів;
- Врахування динаміки зміни клієнтських переваг;
- Покращення точності прогнозів майбутньої поведінки клієнтів;
- Оптимізація часу та змісту маркетингових комунікацій.

Перспективи розвитку CRM-технологій з використанням аналізу часових рядів:

1. Створення динамічних моделей клієнтської поведінки.

2. Розробка алгоритмів прогнозування відтоку клієнтів на основі часових патернів.

3. Впровадження систем раннього попередження про зміни в поведінці клієнтів.

4. Удосконалення персоналізації пропозицій з урахуванням історичних даних та прогнозів.

Використання методів аналізу часових рядів у CRM-системах дозволить компаніям більш точно розуміти та прогнозувати поведінку клієнтів, що

призведе до підвищення ефективності маркетингових стратегій та покращення якості обслуговування клієнтів [17].

У контексті CRM-систем, аналіз трендів та сезонності в поведінці клієнтів є критично важливим для розуміння та прогнозування їхніх дій. Розглянемо, як ці концепції можуть бути застосовані в CRM:

1. Виділення трендів у CRM:

a) Тренд середнього:

- Зміна середньої суми покупок клієнта з часом;
- Тенденція зростання або спадання частоти взаємодій з компанією.

b) Тренд дисперсії:

- Зміна варіативності в поведінці клієнта (наприклад, збільшення розкиду сум покупок);
- Зростання нестабільності у частоті звернень до служби підтримки.

c) Тренд автокореляції:

- Зміна зв'язку між послідовними покупками клієнта;
- Еволюція патернів взаємодії клієнта з різними каналами комунікації.

Застосування в CRM:

- Використання параметричних тестів (Стьюдента, Фішера) для виявлення значущих змін у поведінці клієнтів;
- Моделювання трендів клієнтської поведінки за допомогою поліноміальної апроксимації;
- Застосування методу найменших квадратів для визначення параметрів моделей клієнтської поведінки.

2. Аналіз сезонності в CRM:

- Виявлення сезонних коливань у активності клієнтів (наприклад, збільшення покупок перед святами);
- Аналіз циклічних патернів у зверненнях до служби підтримки;
- Дослідження сезонних змін в ефективності маркетингових кампаній.

Методи аналізу сезонності в CRM:

- Застосування методу ковзного середнього для виділення сезонного компонента;

- Побудова адитивних та мультиплікативних моделей сезонності для різних аспектів клієнтської поведінки.

Перспективи розвитку CRM-технологій з урахуванням аналізу трендів та сезонності:

1. Розробка динамічних моделей клієнтської поведінки:

- Створення алгоритмів, що автоматично виявляють та адаптуються до змін у трендах поведінки клієнтів;

- Впровадження систем раннього попередження про зміни в трендах лояльності клієнтів.

2. Вдосконалення персоналізації:

- Розробка механізмів адаптації пропозицій з урахуванням індивідуальних трендів та сезонних патернів кожного клієнта;

- Створення предиктивних моделей для оптимізації часу та змісту комунікацій з клієнтами.

3. Оптимізація ресурсів компанії:

- Прогнозування навантаження на службу підтримки з урахуванням сезонності;

- Планування маркетингових кампаній з урахуванням виявлених трендів та сезонних коливань.

4. Покращення аналітичних можливостей:

- Інтеграція методів машинного навчання для більш точного виявлення складних трендів та сезонних патернів;

- Розробка візуальних інструментів для представлення трендів та сезонності в інтуїтивно зрозумілому форматі.

5. Прогнозування життєвого циклу клієнта:

- Створення моделей, що враховують довгострокові тренди в поведінці клієнтів для передбачення їхньої цінності для компанії;

- Розробка стратегій утримання клієнтів на основі прогнозованих змін у їхній поведінці.

Впровадження цих технологій дозволить CRM-системам стати більш адаптивними та ефективними у відповідь на динамічні зміни у поведінці клієнтів, що в кінцевому підсумку призведе до підвищення лояльності клієнтів та зростання бізнес-показників компанії [18].

Отже, можна зазначити, що ця сфера є відносно новою, проте має багато спільного з іншими задачами автоматизації бізнес-процесів, таких як управління продажами чи підтримка клієнтів. Водночас, є й суттєві відмінності.

Нині існує кілька способів вирішення завдань управління відносинами з клієнтами, зокрема, комерційні рішення, які здебільшого пропонують закриті моделі та не завжди розкривають повні деталі їхньої реалізації. Крім того, було виявлено, що поведінка клієнтів та ринкові тенденції суттєво впливають на ефективність CRM-систем, проте цей аспект ще не достатньо проаналізований з точки зору математичного моделювання та прогнозування.

Тому, проблема створення більш досконалих та адаптованих CRM-рішень залишається відкритою та актуальною для подальших досліджень, зокрема у напрямку автоматизації персоналізованих підходів до клієнтів, використання великих даних для аналізу поведінки споживачів.

Для реалізації інформаційної технології управління відносинами з клієнтами (CRM) обрано метод аналізу часових рядів. Цей підхід дозволяє ефективно прогнозувати поведінку клієнтів на основі їхніх попередніх дій і взаємодій з компанією. Використовуючи історичні дані, можна передбачити, які продукти чи послуги будуть актуальними для клієнта в майбутньому, що допомагає зберігати лояльність і стимулювати покупки.

Аналіз часових рядів також дозволяє розподіляти клієнтів на різні групи в залежності від схожих патернів їхньої поведінки, що дає можливість адаптувати маркетингові стратегії для кожної групи. Крім того, він допомагає прогнозувати ймовірність відтоку клієнтів, це дозволяє вчасно вжити заходів для їхнього утримання. Зібрані дані про активність клієнтів у часі також дають змогу

оптимізувати маркетингові кампанії, роблячи їх більш персоналізованими та орієнтованими на потреби конкретного клієнта.

Завдяки цьому методу CRM-система може ефективно враховувати динамічні зміни в поведінці клієнтів, що дозволяє створювати більш персоналізовані стратегії комунікації і підвищувати ефективність роботи з клієнтами. Такий підхід відкриває можливості для прогнозування змін у лояльності клієнтів і оптимізації ресурсів компанії, що в свою чергу сприяє покращенню загальних бізнес-показників.

2.2 Математична модель інформаційної технології управління відносинами з клієнтами

Розробка оптимальної математичної моделі щодо управління відносинами з клієнтами є ключовим етапом у розробці ефективної CRM-системи. Цей процес вимагає ретельного аналізу та врахування багатьох факторів, які впливають на успішну взаємодію з клієнтами.

1. Аналіз часових рядів

Для прогнозування поведінки клієнтів та частоти їх взаємодій можна використовувати модель ARIMA (Autoregressive Integrated Moving Average):

$$Y_t = c + \varphi_1 * Y_{t-1} + \ldots + \varphi_p + Y_{t-p} + \Theta_1 * \varepsilon_{t-1} + \ldots + \Theta_q * \varepsilon_{t-q} + \varepsilon_t, \qquad (2.1)$$

де Y_t – значення ряду в момент t,

c – константа,

φ_i – параметри авторегресії,

θ_j – параметри ковзного середнього,

ε_t – білий шум [19].

2. Кластерний аналіз

Для сегментації клієнтів можна застосувати метод *k*-середніх:

$$J = \Sigma_i = 1\text{^}k\ \Sigma_x \in S_i||\text{x} - \mu_i||\text{^}2\ , \quad (2.2)$$

де k − кількість кластерів;

S_i − i-й кластер,

μ_i − центроїд і-го кластера.

3. Логістична регресія

Для прогнозування ймовірності певної дії клієнта (наприклад, здійснення покупки) можна застосувати логістичну регресію:

$$P(Y = 1|\text{X}) = 1\ /\ (1\ +\ \text{e}\text{^}(-\beta_0\ - \beta_1 X_1 - \ldots - \beta_n X_n)), \quad (2.3)$$

де Y − бінарний результат,

X_1 − вхідні змінні,

β_1 − коефіцієнти регресії [20].

4. Аналіз текстових даних

Для обробки чатів і нотаток можна використовувати модель TF-IDF (Term Frequency-Inverse Document Frequency):

$$\text{TF} - \text{IDF}(t, d, D) =\ \text{TF}(t, d)\ *\ \text{IDF}(t, D), \quad (2.4)$$

де t – термін,

d – документ,

D – корпус документів,

$\text{TF}(t, d)$ − (кількість входжень t в d) / (загальна кількість слів в d),

$\text{IDF}(t, D)$ – (загальна кількість документів / кількість документів, що містять t).

5. Марковські ланцюги

Для моделювання послідовності дій клієнта застосовують марковські ланцюги:

$$P(X_{(n+1)} = x \mid X_n = X_n, \dots, X_1 = X_1) = P(X_{(n+1)} = x \mid X_n = X_n), \quad (2.5)$$

де X_n – стан системи в момент n.

6. Теорія черг

Для оптимізації обслуговування клієнтів у чаті застосовують теорію черг:

$$W_q = \lambda / (\mu(\mu - \lambda)), \quad (2.6)$$

де λ – середня швидкість надходження запитів,

μ – середня швидкість обслуговування;

W_q – середній час очікування в черзі [21].

7. Регресійний аналіз

Для прогнозування числових показників (наприклад, суми покупки) можна застосувати регресійний аналіз:

$$Y = \beta_0 + \beta_1 X_1 \beta_2 X_2 + \dots + \beta_n X_n + \varepsilon, \quad (2.7)$$

де Y – залежна змінна,

X_i – незалежні змінні,

β_i – коефіцієнти регресії,

ε – випадкова помилка.

Ця комплексна модель дозволяє:

- Прогнозувати поведінку клієнтів;
- Сегментувати клієнтську базу;
- Аналізувати текстові дані з чатів і нотаток;
- Оптимізувати процеси обслуговування;
- Передбачати ключові показники.

Використання цих методів дає можливість створити ефективну систему управління відносинами з клієнтами, яка враховує особливості чату і нотаток,

забезпечуючи персоналізований підхід та покращення якості обслуговування [22]. Виходячи з наведених моделей, комплексна модель для CRM-системи, що прогнозує поведінку клієнтів, сегментує їх, аналізує текстові дані і оптимізує процеси обслуговування. Кожна частина рівняння (5.8) відповідає певному аспекту CRM-аналізу і виглядає так:

$$Yt = ARIMA(t) + K - Means(Xi) + Logistic(X1, X2, \dots, Xn) + TF - \\ -IDF(t, d, D) + Queueing(Wq), \quad (2.8)$$

де $ARIMA(t)$ − прогноз поведінки клієнтів,

$K - Means(Xi)$ − сегментація клієнтів,

$Logistic(X1, X2, \dots, Xn)$ − прогноз ймовірності дії клієнта,

$TF - IDF(t, d, D)$ − аналіз тексту

$Queueing(Wq)$ − оптимізація часу очікування клієнта.

Рівняння (2.8) охоплює всі основні аспекти управління взаємодією з клієнтами і може бути використане для комплексної аналітики в CRM-системі. Значення математичної моделі для практики полягає в її здатності забезпечувати даними для прийняття рішень, що базуються на аналітичних висновках, а також у підвищенні ефективності маркетингових зусиль через точне націлювання на сегменти клієнтів. Завдяки такій моделі підприємства отримують можливість оптимізувати свої ресурси та покращити якість обслуговування, що врешті-решт веде до збільшення прибутку та конкурентоспроможності на ринку.

2.3 Розробка UML-діаграми послідовності інформаційної технології управління відносинами з клієнтами

Для глибшого розуміння роботи програмного модуля інформаційної технології управління відносинами з клієнтами, доцільно створити його UML-діаграму послідовності. UML (Unified Modeling Language) є уніфікованою мовою

моделювання, яка є невід'ємною частиною процесу розробки програмного забезпечення. Ця мова надає стандартизовані графічні позначення для створення абстрактних моделей систем.

UML-моделі поділяються на два основні типи: структурні діаграми та діаграми поведінки. Оскільки наша увага зосереджена на описі робочого циклу програми, найбільш доречними для цього будуть діаграми поведінки, а саме: діаграма прецедентів (Use Case Diagram), діаграма діяльності (Activity Diagram) та діаграма послідовності (Sequence Diagram).

Діаграма послідовності є особливо корисною, оскільки вона наочно демонструє порядок обміну повідомленнями між об'єктами в системі в часовій послідовності. Така діаграма надає повне уявлення про роботу програмного застосунку, показуючи, як різні компоненти взаємодіють один з одним.

Для інформаційної технології управління відносинами з клієнтами, діаграма послідовності може включати такі ключові компоненти, як клієнт, менеджер, CRM-система та база даних.

Такий підхід наочно демонструє послідовність дій та обмін повідомленнями між різними елементами системи управління відносинами з клієнтами. Це дає змогу краще зрозуміти, як взаємодіють ключові компоненти програмного модуля, що відповідає за цю технологію [23].

Таким чином, розробка UML-діаграми послідовності є ефективним інструментом для глибшого розуміння роботи програмного модуля інформаційної технології управління відносинами з клієнтами. Вона допомагає візуалізувати та чітко представити порядок взаємодії між різними елементами системи. Розроблена діаграма послідовностей зображена на рисунку 2.1.

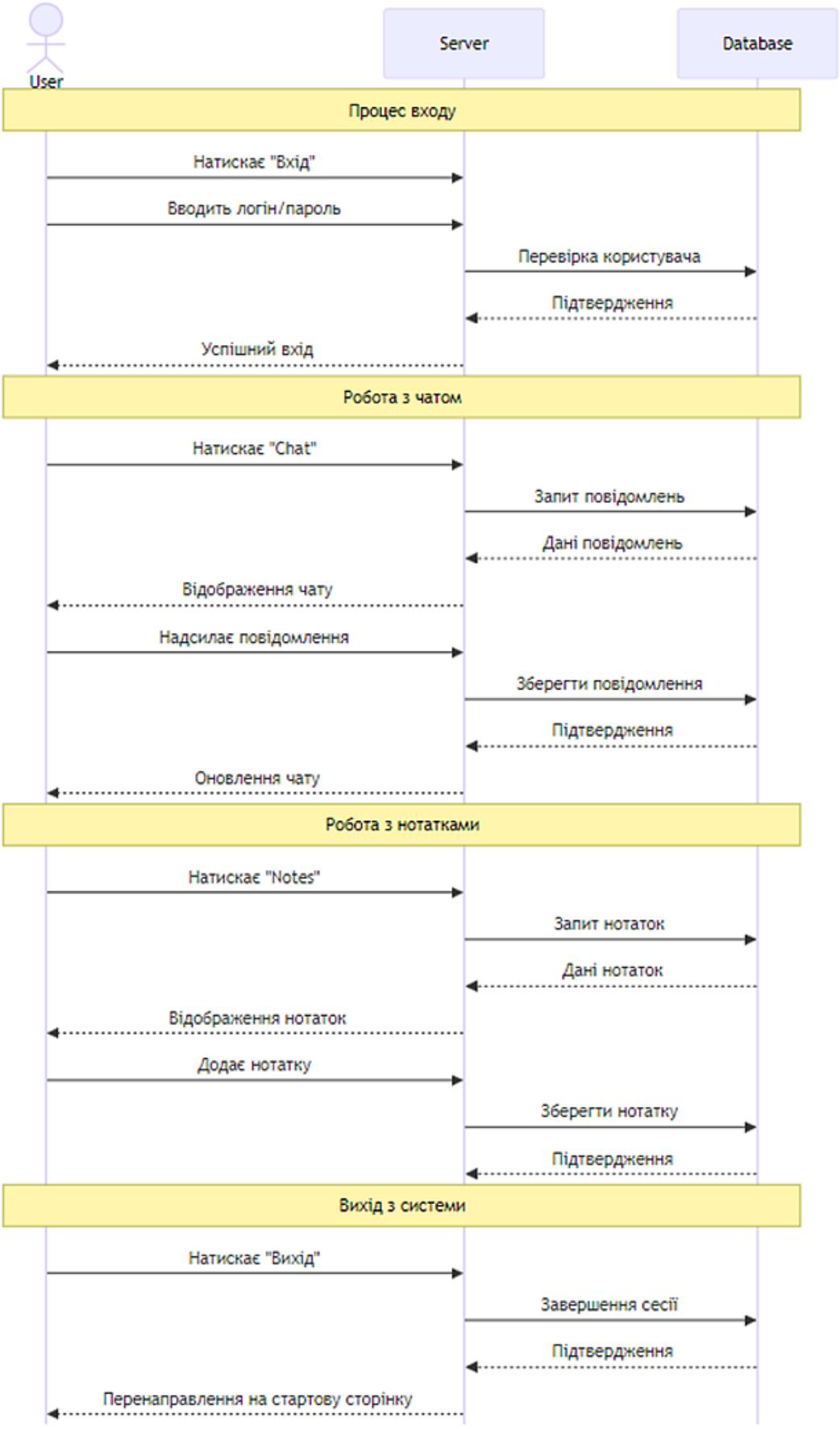

Рисунок 2.1 – UML-діаграма послідовності взаємодії модулів інформаційної технології управління відносинами з клієнтами

2.4 Висновок до розділу 2

У другому розділі було розглянуто методи моделювання інформаційних технологій управління відносинами з клієнтами (CRM) та обґрунтовано використання методу аналізу даних. Удосконалено математичну модель програмного модуля для управління взаємовідносинами з клієнтами, яка дозволяє аналізувати та прогнозувати поведінку споживачів на основі історичних даних. Модель відповідає за генерування рекомендацій та стратегій взаємодії з клієнтами на основі переданої користувачем інформації. Було створено UML-діаграму послідовності для опису роботи програмного модуля інформаційної технології управління відносинами з клієнтами.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ УПРАВЛІННЯ ВІДНОСИНАМИ З КЛІЄНТАМИ

3.1 Обґрунтування вибору архітектури інформаційної технології управління відносинами з клієнтами

Архітектура програмного забезпечення – це структура програми, системи або модуля, яка відображає взаємодію між елементами системи, їх властивості та логічні частини, за які відповідає той чи інший елемент. Побудова архітектури є першим етапом розробки програмного забезпечення. При проектуванні модуля, орієнтуючись на поставлене технічне завдання, необхідно розробити найбільш оптимальну архітектуру, враховуючи швидкість роботи майбутнього проекту, зв'язки основних елементів системи, а також закласти можливість масштабування архітектури. Архітектура програмного забезпечення створює обмеження для написання функціоналу, декларує єдиний стиль коду та закріплює ключові кроки, створені на ранніх етапах розробки.

Інформаційна технологія управління відносинами з клієнтами також має клієнт-серверну архітектуру. Сервер виступає потужною обчислювальною машиною, призначеною для збереження та обробки інформації, а також для її передачі між користувачами. Клієнт, у свою чергу, відповідає за прийом і обробку введеної користувачем інформації, а також за відображення існуючих даних.

Принцип роботи такої архітектури полягає в тому, що клієнт відповідає за пряму взаємодію з користувачем через створений інтерфейс. Клієнт збирає необхідну інформацію від користувача додатку, обробляє та модифікує її за потреби, а потім відправляє на сервер за допомогою HTTP або TCP протоколів передачі даних. TCP протокол дозволяє створити постійне підключення між клієнтом і сервером, що забезпечує динамічну передачу змін у випадку оновлення даних в будь-якому з елементів системи.

Сервер відповідає за обробку та збереження інформації, надісланої від клієнта. Він може обробляти запити від кількох клієнтів або містити декілька серверів, які виконують різні функції в залежності від побудованої архітектури. Ці сервери зв'язуються між собою також за допомогою HTTP або TCP протоколів. Сервер має доступ до бази даних системи, відповідає за безпеку доступу до неї та здійснює операції з інформацією.

При побудові архітектури можна вказати пріоритетність запитів до сервера, що дозволяє йому в першу чергу обробляти найбільш важливі функції для роботи системи. Це забезпечує ефективне управління взаємовідносинами з клієнтами та підвищує загальну продуктивність системи [24].

Структурна схема взаємодії клієнт-серверної архітектури зображено на рисунку 3.1. Схема відображає взаємодію між сервером, базою даних, та багатьма клієнтами інформаційної технології безконтактного обміну персональними даними.

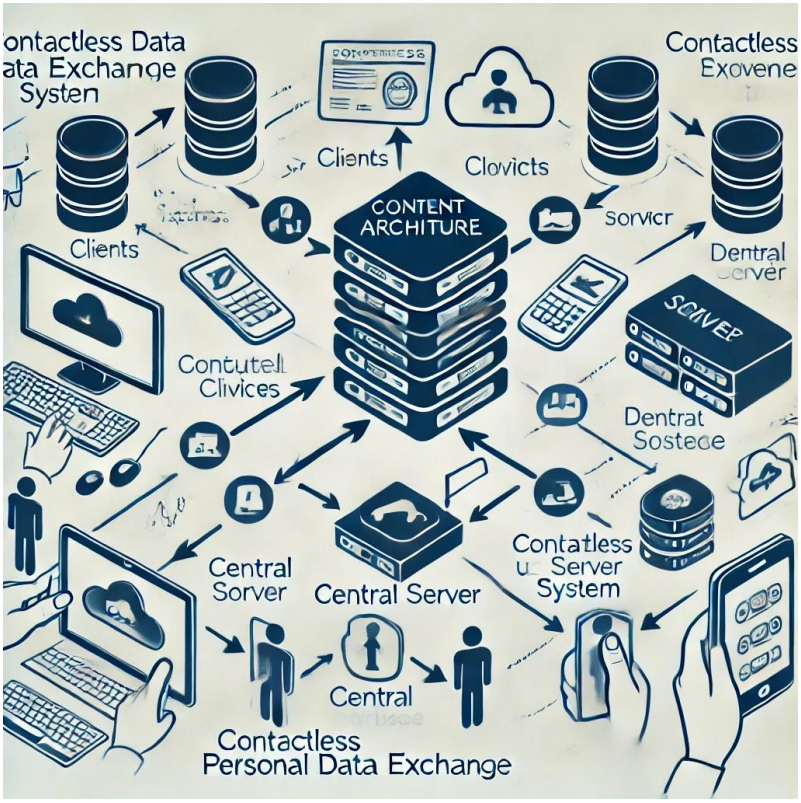

Рисунок 3.1 – Схема системи клієнт-сервер

Функції сервера:

- Зберігання, доступ, захист і резервне копіювання даних:с відповідає за безпечне зберігання персональних даних, їх доступність для клієнтів, а також за регулярне резервне копіювання для запобігання втратам.
- Обробка клієнтського запиту: сервер отримує запити від клієнтів, аналізує їх і виконує відповідні дії.
- Відправлення результату (відповіді) клієнту: після обробки запиту сервер надсилає результати назад до клієнта, забезпечуючи інтерактивність системи.

Функції клієнта:

- Надання користувацького інтерфейсу: клієнтська частина забезпечує зручний інтерфейс для взаємодії користувача з системою, що дозволяє легко формулювати запити.
- Формулювання запиту до сервера і його відправка: клієнт готує запити на основі дій користувача та надсилає їх на сервер для обробки.
- Отримання результатів запиту і відправка додаткових команд: клієнт отримує відповіді від сервера та може надсилати додаткові команди, такі як запити на додавання, оновлення або видалення даних.

3.2 Проектування структури та розробка алгоритму функціонування інформаційної технології управління відносинами з клієнтами

Оскільки CRM-система має клієнт-серверну архітектуру, для реалізації проекту необхідно розробити кілька частин, таких як сервер, клієнт і базу даних. Для забезпечення швидкої роботи системи та можливості подальшої її підтримки доцільно використати модульний підхід.

Модульне програмування − це підхід до створення програмного забезпечення, при якому система розділяється на менші незалежні частини або модулі. Кожен модуль являє собою своєрідну міні-програму, що виконує певну частину загальної функціональності. Основні принципи модульного програмування включають:

1. Висока згуртованість − усе всередині модуля тісно пов’язане та спрямоване на досягнення однієї мети.

2. Слабкий зв’язок − модулі взаємодіють один з одним лише через зовнішні інтерфейси, що зменшує залежність між ними та дозволяє їм працювати незалежно.

3. Абстракція − кожен модуль приховує свою внутрішню реалізацію, відкриваючи для інших лише ті аспекти, які необхідні для його використання.

4. Автономність − модулі мають усе необхідне для самостійного функціонування, що зменшує залежність від інших компонентів системи.

Цей підхід дозволяє створити прозору логіку роботи системи, спрощує написання коду та його тестування, знижує витрати на технічне обслуговування і модернізацію, а також підвищує стійкість до можливих помилок.

У програмі будуть представлені такі модулі: модуль інтерфейсу, модуль сервера, модуль для управління відносинами з клієнтами, модуль для роботи з постачальниками даних. Структура програмного модуля для управління відносинами з клієнтами наведена на рисунку 3.2.

Рисунок 3.2 – Загальна структурна модель програмного модуля

Для реалізації інформаційної технології управління відносинами з клієнтами (CRM) було розроблено схему алгоритму її роботи. Ця схема

відображає основні етапи взаємодії користувача з системою та логіку обробки даних для надання персоналізованих пропозиційі рекомендацій. Схема алгоритму зображена на рисунку 3.3.

І випадок

Алгоритм у випадку натиснення на кнопку «Вхід» складається з таких кроків:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує.

ІІ випадок

Алгоритм для кнопки «Calendar» => «Завдання»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 8: Користувач натискає кнопку «Calendar»;

Крок 9: Виведення інформації з категорії Calendar;

Крок 11: Користувач обирає кнопку: «Завдання»;

Крок 12: Виведення інформації про завдання.

ІІІ випадок

Алгоритм для кнопки «Calendar» => «День»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 8: Користувач натискає кнопку «Calendar»;

Крок 9: Виведення інформації з категорії Calendar;

Крок 13: Користувач обирає кнопку «День»;

Крок 14: Введення даних;

Крок 15: Збереження даних у БД.

IV випадок

Алгоритм для кнопки «Calls»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 16: Користувач натискає кнопку «Calls»;

Крок 17: Виведення інформації з категорії Calls;

Крок 18: Користувач натискає кнопку «Додавання дзвінка»;

Крок 19: Введення даних про дзвінок;

Крок 20: Збереження даних у БД.

V випадок

Алгоритм для кнопки «Tasks»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 21: Користувач натискає кнопку «Tasks»;

Крок 22: Виведення інформації з категорії Tasks;

Крок 23: Користувач натискає кнопку «Додати завдання»;

Крок 24: Введення даних про завдання;

Крок 25: Збереження даних у БД.

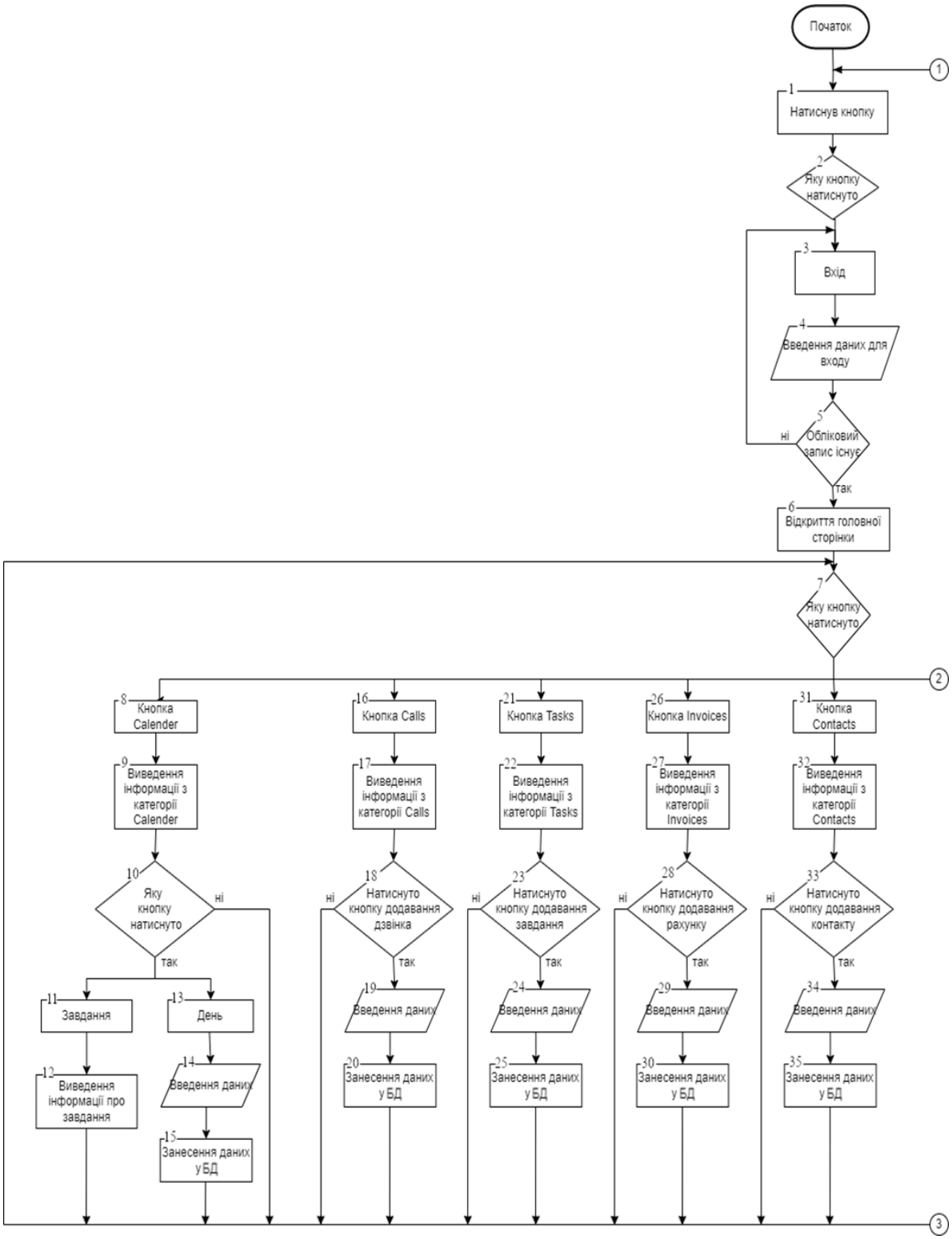

Рисунок 3.3 – Схема алгоритму роботи інформаційної технології управління відносинами з клієнтами

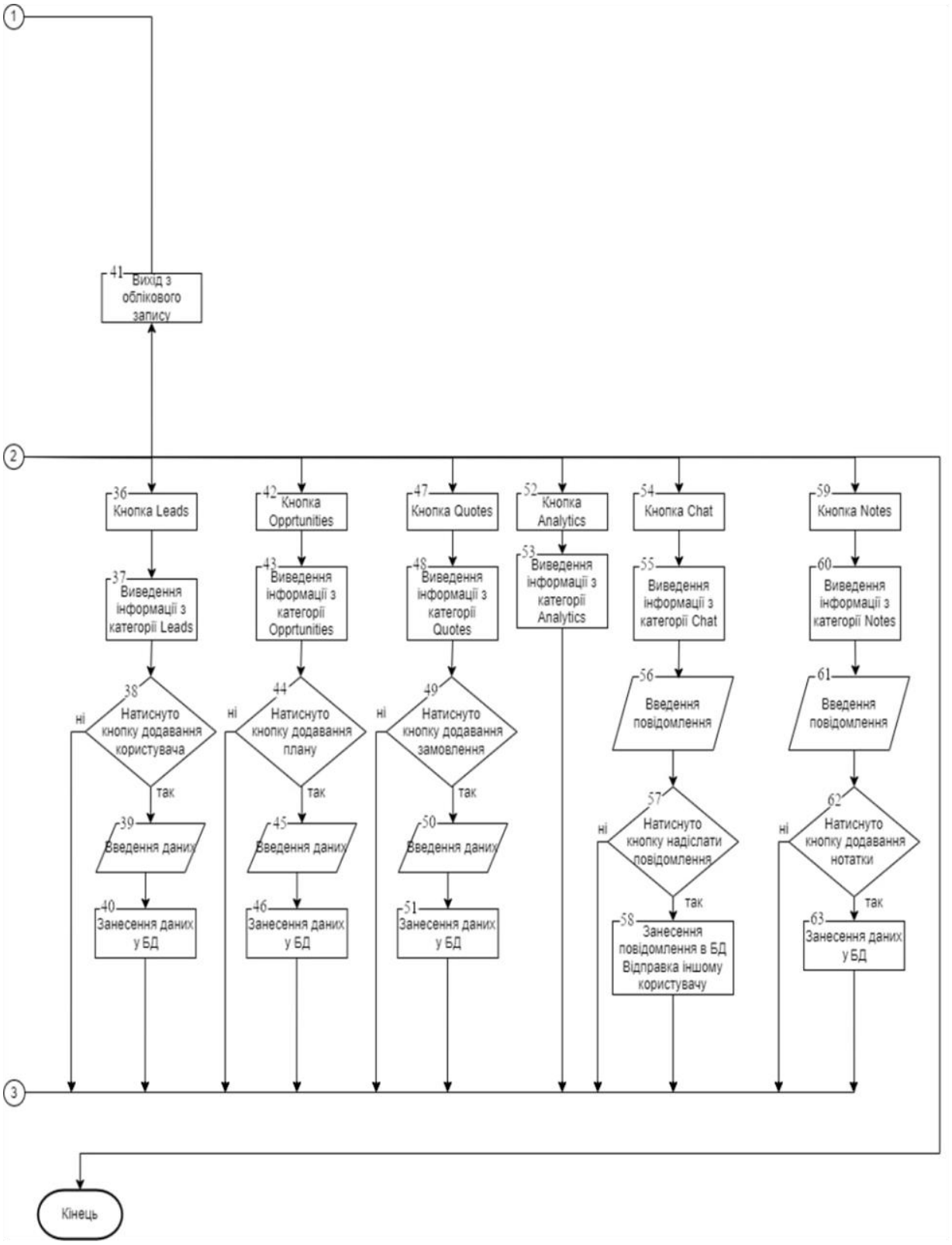

Рисунок 3.3, аркуш 2

VI випадок

Алгоритм для кнопки «Invoices»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 26: Користувач натискає кнопку «Invoices»;

Крок 27: Виведення інформації з категорії Invoices;

Крок 28: Користувач натискає кнопку «Додати рахунок»;

Крок 29: Введення даних про рахунок;

Крок 30: Збереження даних у БД.

VII випадок

Алгоритм для кнопки «Contacts»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 31: Користувач натискає кнопку «Contacts»;

Крок 32: Виведення інформації з категорії Contacts;

Крок 33: Користувач натискає кнопку «Додати контакт»;

Крок 34: Введення даних про контакт;

Крок 35: Збереження даних у БД.

VIII випадок

Алгоритм для кнопки «Leads»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 36: Користувач натискає кнопку «Leads»;

Крок 37: Виведення інформації з категорії Leads;

Крок 38: Користувач натискає кнопку «Додати потенційного клієнта»;

Крок 39: Введення даних про потенційного клієнта4

Крок 40: Збереження даних у БД.

IX випадок

Алгоритм для кнопки «Opportunities»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 42: Користувач натискає кнопку «Opportunities»;

Крок 43: Виведення інформації з категорії Opportunities;

Крок 44: Користувач натискає кнопку «Додати можливість»;

Крок 45: Введення даних про можливість;

Крок 46: Збереження даних у БД.

X випадок

Алгоритм для кнопки «Quotes»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 47: Користувач натискає кнопку «Quotes»;

Крок 48: Виведення інформації з категорії Quotes;

Крок 49: Користувач натискає кнопку «Додати комерційну пропозицію»;

Крок 50: Введення даних про пропозицію;

Крок 51: Збереження даних у БД.

XI випадок

Алгоритм для кнопки «Analytics»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 52: Користувач натискає кнопку «Analytics»;

Крок 53: Виведення інформації з категорії Analytics.

XII випадок

Алгоритм для кнопки «Chat»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 54: Користувач натискає кнопку «Chat»;

Крок 55: Виведення інформації з категорії Chat;

Крок 56: Введення повідомлення;

Крок 57: Користувач натискає кнопку «Надіслати повідомлення»;

Крок 58: Збереження повідомлення у БД та виведення його іншим користувачам.

XIII випадок

Алгоритм для кнопки «Notes»:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 59: Користувач натискає кнопку «Notes»;

Крок 60: Виведення інформації з категорії Notes;

Крок 61: Користувач натискає кнопку «Додати нотатку»;

Крок 62: Введення даних про нотатку;

Крок 63: Збереження даних у БД.

XIV випадок

Алгоритм у випадку натиснення на кнопку «Вихід з облікового запису» складається з таких кроків:

Крок 1. Користувач натискає кнопку «Вхід»;

Крок 4. Введення вхідних даних: Ім’я користувача, пароль;

Крок 5. Перевірка чи є такий користувач у БД;

Крок 6. Вхід у систему, якщо запис існує;

Крок 41: Користувач натискає кнопку «Вихід з облікового запису»;

Крок 1: Система завершує сесію користувача і повертає на стартову сторінку.

3.3 Обґрунтування вибору мови програмування та фреймворків

Еволюція мов програмування демонструє захоплюючу історію технологічного розвитку, де кожна нова мова з'являлася для вирішення специфічних викликів свого часу. Python, створений Гвідо ван Россумом у 1991 році, став однією з найвпливовіших мов програмування завдяки своїй простоті та читабельності. Сьогодні Python домінує у різних сферах: від WEB-розробки з використанням фреймворків Django та Flask до машинного навчання та аналізу даних з бібліотеками TensorFlow, PyTorch та scikit-learn. Його інтуїтивний синтаксис та величезна кількість бібліотек роблять його ідеальним вибором як для новачків, так і для досвідчених розробників.

C++ залишається незамінною мовою для розробки високопродуктивних систем, ігрових движків та системного програмного забезпечення. Сучасні стандарти C++17 та C++20 значно розширили можливості мови, додавши кращу підтримку паралельного програмування та метапрограмування. Фреймворк Qt дозволяє створювати кросплатформні додатки з графічним інтерфейсом, а бібліотека Boost суттєво розширює функціональність стандартної бібліотеки.

JavaScript, створена Бренданом Айхом у 1995 році, з часом стала однією з найпопулярніших мов програмування у світі. Особливо сильні позиції JavaScript займає у сфері WEB-розробки, де вона використовується для створення динамічних, інтерактивних та потужних WEB-додатків.

З моменту свого створення JavaScript постійно еволюціонувала, адаптуючись до мінливих потреб розробників та індустрії в цілому. У контексті

WEB-розробки ключовим етапом стала поява AJAX (Asynchronous JavaScript and XML) у 2005 році. Ця технологія дозволила розробникам створювати асинхронні WEB-додатки, значно покращивши користувацький досвід.

Наступним важливим кроком стала поява потужних фреймворків та бібліотек, таких як jQuery (2006), AngularJS (2010), React (2013) та Vue.js (2014). Ці інструменти значно розширили можливості JavaScript у WEB-розробці, спростивши створення складних односторінкових додатків (SPA).

JavaScript має ряд характеристик, які роблять її відмінним вибором для розробки WEB-додатків:

Універсальність: JavaScript може використовуватися як на фронтенді, так і на бекенді (Node.js), що дозволяє розробникам використовувати одну мову для всього стеку.

Динамічність та гнучкість: JavaScript − це динамічно типізована мова, що дозволяє швидко прототипувати та легко адаптувати код.

Багата екосистема: JavaScript має величезну кількість бібліотек та фреймворків, доступних через npm (Node Package Manager), що спрощує розробку WEB-додатків.

Асинхронність: JavaScript має вбудовану підтримку асинхронного програмування, що робить її ідеальною для обробки мережевих запитів та інших операцій введення-виведення [25].

Постійний розвиток: регулярні оновлення стандарту ECMAScript забезпечують появу нових можливостей та покращень мови.

У світі JavaScript WEB-розробки існує кілька ключових технологій:

DOM маніпуляції: JavaScript дозволяє динамічно змінювати структуру та вміст WEB-сторінок через Document Object Model (DOM).

Розглянемо існуючі фреймворки, які сприятимуть розробці інформаційної технології управління відносинами з клієнтами.

AJAX та Fetch API: ці технології дозволяють асинхронно обмінюватися даними з сервером без перезавантаження сторінки.

SPA фреймворки (React, Vue.js, Angular): ці фреймворки дозволяють створювати складні односторінкові додатки з багатим користувацьким інтерфейсом.

Node.js: це середовище виконання JavaScript на сервері, яке дозволяє використовувати JavaScript для бекенд-розробки.

Express.js: популярний WEB-фреймворк для Node.js, який спрощує створення WEB-серверів та API.

Незважаючи на численні переваги, використання JavaScript для WEB-розробки має певні виклики:

Різноманітність інструментів: велика кількість фреймворків та бібліотек може ускладнити вибір правильного інструменту.

Динамічна типізація: хоча це може бути перевагою, відсутність статичної типізації може призвести до помилок, які важко виявити.

Безпека: JavaScript, що виконується на клієнтській стороні, може бути вразливим до атак, якщо не вжити належних заходів безпеки.

Продуктивність: для складних обчислень JavaScript може бути менш ефективним порівняно з компільованими мовами.

Незважаючи на ці виклики, JavaScript продовжує відігравати ключову роль у WEB-розробці. Постійні оновлення мови та платформи, такі як введення класів та модулів в ES6, асинхронних функцій в ES2017 та нових можливостей в наступних версіях, демонструють прагнення до подальшого розвитку.

Крім того, поява нових технологій, таких як WebAssembly, які дозволяють запускати код, написаний на інших мовах, у браузері, відкриває нові можливості для взаємодії JavaScript з високопродуктивним кодом [26].

JavaScript залишається потужним інструментом для створення динамічних, інтерактивних WEB-додатків. Її гнучкість, універсальність та постійний розвиток забезпечують їй міцні позиції у світі WEB-розробки на найближче майбутнє.

Rust, розроблений Mozilla Research, стрімко набирає популярність завдяки своїй безпеці роботи з пам'яттю та високій продуктивності. Його інноваційна

система власності (ownership system) запобігає багатьом поширеним помилкам програмування без втрати швидкодії. Rust активно використовується у системному програмуванні, розробці браузерних компонентів та створенні високопродуктивних WEB-серверів.

Go, створена в Google, виділяється своєю простотою та ефективністю в розробці масштабованих серверних додатків. Її вбудована підтримка конкурентності через горутини (goroutines) та канали робить її відмінним вибором для створення мікросервісів та розподілених систем. Фреймворки Gin та Echo спрощують створення REST API та WEB-сервісів.

Swift, розроблений Apple для iOS та macOS розробки, поступово розширює свої можливості і зараз активно використовується також для серверної розробки через Swift on Server фреймворки як Vapor та Kitura. Його сучасний синтаксис та потужна система типів забезпечують надійну та продуктивну розробку.

TypeScript, надбудова над JavaScript від Microsoft, став стандартом для великих фронтенд-проектів завдяки своїй статичній типізації та покращеній підтримці об'єктно-орієнтованого програмування. У поєднанні з фреймворками Angular та Next.js, TypeScript забезпечує більш надійну та масштабовану розробку WEB-додатків .

У сфері функціонального програмування Elixir, побудований на віртуальній машині Erlang, знаходить все більше застосування у розробці високонавантажених розподілених систем. Його фреймворк Phoenix дозволяє створювати швидкі та надійні WEB-додатки, а вбудована підтримка паралелізму через акторну модель робить його ідеальним для обробки даних у реальному часі.

Для розробки інформаційної технології управління відносинами з клієнтами (CRM) обрано JavaScript як основну мову програмування, оскільки вона має високий рівень універсальності та гнучкості, що дозволяє створювати інтерактивні та динамічні WEB-додатки. Завдяки своєму інтуїтивно зрозумілому синтаксису та великій екосистемі бібліотек і фреймворків, JavaScript ідеально

підходить для створення складних CRM-систем, які потребують інтеграції з різними сервісами та швидкої обробки даних.

Для побудови клієнтської частини застосовувався React, один з найбільш популярних фреймворків для створення односторінкових додатків (SPA). Він дозволяє ефективно керувати станом додатка та взаємодіяти з користувачем, забезпечуючи динамічну зміну контенту без перезавантаження сторінки. Використання React дозволяє створити зручний та інтуїтивно зрозумілий інтерфейс для взаємодії з клієнтами.

Для серверної частини використовується Node.js, оскільки він дозволяє запускати JavaScript на сервері, забезпечуючи високу продуктивність при обробці численних запитів одночасно. Це ідеально підходить для CRM-систем, де важливо ефективно обробляти запити від багатьох користувачів. Для спрощення розробки серверної частини застосовується Express.js − популярний фреймворк для Node.js, який полегшує створення API та серверів для обробки запитів.

Усі ці технології взаємодіють між собою, це дозволяє створити потужну та масштабовану CRM-систему, здатну ефективно обробляти дані про клієнтів та адаптувати взаємодію з ними в реальному часі.

3.4 Обґрунтування вибору бази даних для розробки

У світі баз даних відбувається значна трансформація − поряд з традиційними реляційними СУБД як PostgreSQL та MySQL, все більшої популярності набувають NoSQL рішення. MongoDB став стандартом для документо-орієнтованих баз даних, особливо у проектах з динамічною схемою даних. Redis відмінно виконує роль швидкого сховища даних в пам'яті та системи кешування. Cassandra від Apache забезпечує високу масштабованість для розподілених систем.

Сучасні інструменти розробки також еволюціонують − Visual Studio Code став де-факто стандартним редактором коду для багатьох розробників завдяки своїй гнучкості та багатій екосистемі розширень. Docker зробив революцію в розгортанні додатків, забезпечуючи стабільне середовище виконання через контейнеризацію. Kubernetes став стандартом для оркестрації контейнерів у великих розподілених системах.

Системи контролю версій як Git стали невід'ємною частиною процесу розробки, а платформи як GitHub та GitLab забезпечують не тільки хостинг коду, але й інструменти для continuous integration та continuous deployment (CI/CD). GitLab CI та GitHub Actions автоматизують процеси тестування та розгортання.

У сфері тестування популярними стали фреймворки як Jest для JavaScript, PyTest для Python. Інструменти для автоматизації тестування користувацького інтерфейсу як Selenium та Cypress дозволяють забезпечити якість фронтенд-частини додатків [27].

Спостерігається також тренд на використання serverless архітектури, де такі платформи як AWS Lambda, Google Cloud Functions та Azure Functions дозволяють розробникам зосередитись на бізнес-логіці, не турбуючись про інфраструктуру. Важливо розуміти, що вибір мови програмування та технологічного стеку залежить від багатьох факторів: вимог проекту, наявних ресурсів, масштабованості, продуктивності та екосистеми. Кожна мова має свої сильні сторони та області застосування, і часто найкращим рішенням є використання декількох мов у рамках одного проекту, де кожна мова виконує ті задачі, для яких вона найкраще підходить.

Сучасна розробка програмного забезпечення стає все більш комплексною, і важливо не тільки знати конкретні мови програмування, але й розуміти принципи архітектури програмного забезпечення, патерни проектування та кращі практики розробки. Це дозволяє створювати більш якісні, надійні та масштабовані рішення незалежно від обраного технологічного стеку.

Окрему увагу варто приділити розвитку крос-платформної розробки, де такі фреймворки як Flutter та React Native дозволяють створювати мобільні

додатки для різних платформ використовуючи єдину кодову базу. Це значно прискорює процес розробки та зменшує витрати на підтримку додатків.

Вибір мови програмування для розробки є критично важливим етапом у процесі створення програмного забезпечення, і я обрав JavaScript. Ця мова демонструє величезну універсальність і адаптивність, що робить її ідеальним вибором для WEB-розробки. Завдяки своїй здатності працювати як на фронтенді, так і на бекенді (за допомогою Node.js), JavaScript дозволяє створювати повноцінні веб-додатки, що підвищує ефективність та спрощує процес розробки.

JavaScript має потужну екосистему, що включає численні фреймворки та бібліотеки, такі як React, Angular та Vue.js, які дозволяють розробникам створювати динамічні, інтерактивні та масштабовані додатки. Крім того, наявність таких технологій, як AJAX та Fetch API, спрощує асинхронний обмін даними між клієнтом і сервером, що значно покращує користувацький досвід.

Постійний розвиток стандарту ECMAScript забезпечує появу нових функцій, що дозволяють розробникам писати більш чистий і зрозумілий код. Важливою перевагою є також активна спільнота розробників, яка постійно вдосконалює мову та її екосистему [28].

Незважаючи на виклики, такі як динамічна типізація та питання безпеки, переваги JavaScript в контексті веб-розробки роблять її моїм пріоритетним вибором. Зокрема, можливість створення односторінкових додатків (SPA) і підтримка сучасних архітектурних рішень, таких як серверна архітектура без серверів, надають мені можливість реалізувати ідеї у зручний і продуктивний спосіб.

Для розробки інформаційної технології управління відносинами з клієнтами обрано MongoDB як базу даних. MongoDB є популярним документо-орієнтованим рішенням, яке добре підходить для проектів, що мають динамічну схему даних. Ця база даних дозволяє зберігати великі обсяги інформації в гнучкому форматі, що є ідеальним для зберігання різноманітних даних про клієнтів, їхні взаємодії з компанією та історію покупок.

Основною перевагою MongoDB є її здатність до горизонтального масштабування, що дозволяє ефективно обробляти великі обсяги даних навіть при зростанні навантаження. Крім того, MongoDB підтримує швидкий доступ до даних за допомогою індексів, що дозволяє забезпечити високу продуктивність при виконанні запитів.

MongoDB також надає можливість працювати з документами в форматі JSON, що робить її інтеграцію з WEB-додатками, побудованими на JavaScript, дуже зручною. Ця характеристика значно спрощує роботу з даними, оскільки дозволяє безпосередньо взаємодіяти з ними без необхідності в додаткових перетвореннях [29].

У контексті CRM-систем MongoDB дає можливість гнучко управляти даними про клієнтів і швидко адаптувати базу даних до змінюваних потреб бізнесу, що робить її оптимальним вибором для цього проекту.

3.5 Реалізація програмного модуля інформаційної технології управління відносинами з клієнтами

Для початку роботи програми потрібно провести авторизацію користувача.

Спочатку функція приймає два параметри: req (запит) та res (відповідь). У середині функції розпочинається блок try, що дозволяє ловити можливі помилки під час виконання.

У тілі функції з об'єкта req.body вилучаються username та password, які вводить користувач. Потім відбувається запит до бази даних для пошуку користувача за наданим username. У запиті також враховується, що поле deleted має бути false, що дозволяє уникати видалених користувачів. Для отримання додаткової інформації про користувача, функція використовує метод populate для заповнення полів, пов'язаних з ролями користувача.

Якщо користувач не знайдений, функція повертає статус 401 і повідомлення про невдалу аутентифікацію. У випадку успішного знаходження користувача, код переходить до порівняння пароля. Метод bcrypt.compare

використовується для перевірки, чи введений пароль відповідає хешованому паролю, який зберігається в базі даних. Якщо паролі не співпадають, функція також повертає статус 401 з відповідним повідомленням про помилку.

Якщо порівняння паролів успішне, функція генерує JWT-токен за допомогою бібліотеки jsonwebtoken. Токен створюється на основі ідентифікатора користувача (user._id) і має термін дії один день.

Після цього функція встановлює заголовок Authorization у відповіді, додаючи до нього токен, та відправляє статус 200 разом із токеном і даними користувача в форматі JSON. У разі виникнення помилки під час виконання будь-якого з попередніх кроків, функція ловить цю помилку в блоці catch і повертає статус 500 з повідомленням про помилку.

Діаграма діяльності сервера під час процесу автентифікації користувача зображена на рисунку 3.4.

Рисунок 3.4 – Діаграма діяльності серверу під час процесу автентифікації користувача

Зображений вище алгоритм (рис. 3.4) можна записати у вигляді коду:

```
const login = async (req, res) => {
  try {
    const { username, password } = req.body;
    // Find the user by username
    const user = await User.findOne({ username, deleted: false }).populate({
      path: 'roles'
    });
    if (!user) {
      res.status(401).json({ error: 'Authentication failed, invalid username' });
      return;
    }
    // Compare the provided password with the hashed password stored in the database
    const passwordMatch = await bcrypt.compare(password, user.password);
    if (!passwordMatch) {
      res.status(401).json({ error: 'Authentication failed,password does not match' });
      return;
    }
    // Create a JWT token
    const token = jwt.sign({ userId: user._id }, 'secret_key', { expiresIn: '1d' });

    res.status(200).setHeader('Authorization', `Bearer${token}`).json({ token: token, user });
  } catch (error) {
    res.status(500).json({ error: 'An error occurred' });
  }
}
```

Спочатку, за допомогою хука useState, визначаються кілька станів: chats, який ініціалізується як порожній масив для зберігання списку чатів; message, що зберігає текст повідомлення; chatId та зберігає ідентифікатор вибраного чату; chatMessages, ініціалізований як null для зберігання повідомлень чату.

З використанням useSelector з Redux витягуються дані про користувача (userData) з глобального стану.

Функція getChats є асинхронною і виконує запит до API за допомогою axios для отримання списку чатів з ендпоінту /chat. Після отримання даних, функція оновлює стан chats через метод setChats.

Наступна функція subscribeGetMessages також є асинхронною. Вона приймає параметр id, що є ідентифікатором чату і намагається виконати запит до API для отримання повідомлень чату з ендпоінту /message/${id}. Успішно отримавши повідомлення, вона оновлює стан chatMessages. Якщо отримання повідомлень пройшло успішно, функція викликає сама себе для підписки на нові повідомлення, це дозволяє постійно оновлювати список повідомлень.

Якщо виникає помилка під час запиту, блок catch обробляє цю помилку, і функція повторно викликається через 500 мілісекунд, що забезпечує спробу повторного підключення.

Хук useEffect виконує функцію getChats при монтуванні компонента. Це означає, що при першому завантаженні компонента список чатів буде отримано з сервера.

Функція sendMessage є асинхронною і використовується для надсилання нового повідомлення. Вона виконує POST-запит до API на ендпоінт /message, передаючи текст повідомлення та ідентифікатор чату. Після надсилання повідомлення функція очищує поле вводу, оновлюючи стан message, і додає нове повідомлення до стану chatMessages, при цьому використовуються дані про користувача для зазначення відправника.

Остання функція getMessages є асинхронною і приймає id як параметр. Вона виконує GET-запит до API на ендпоінт /messages/${id} для отримання повідомлень, пов’язаних з певним чатом. Отримані повідомлення зберігаються у стані chatMessages. Діаграма діяльності сервера під час процесу перегляду замовлення користувача зображена на рисунку 3.5.

Рисунок 3.5 – Діаграма діяльності серверу під час процесу перегляду замовлення користувача

Зображений вище алгоритм (рис. 3.5) можна записати у вигляді коду:

```
const [chats, setChats] = useState([]);
  const [message, setMessage] = useState('');
  const [chatId, setChatId] = useState('');
  const [chatMessages, setChatMessages] = useState(null)

   const { userData } = useSelector(state => state.auth.data)

  const getChats = async () => {
    const chatsObj = await axios.get('/chat')
     setChats(chatsObj.data)

  }
  const subscribeGetMessages = async (id) => {

    try {
```

```
    console.log(id)
    setChatId(id)
   const messages = await axios.get(/message/${id});
    setChatMessages(messages)
    subscribeGetMessages(id)
  }
  catch (err) {
    setTimeout(() => {
      subscribeGetMessages(id)
    }, 500)
  }
}

React.useEffect(() => {
  getChats();

}, [])


const sendMessage = async () => {
   await axios.post('/message', { message, chatId })
  setMessage(' ')
  setChatMessages([...chatMessages, {
    sender: userData.customer,
    message: message
  }])
}

const getMessages = async (id) => {
   const messages = await axios.get(/messages/${id});
   setChatMessages(messages.data)

}
```

Спочатку, за допомогою хука useState, визначаються два стани: notes, ініціалізований як порожній масив для зберігання нотаток і note, ініціалізований як порожній рядок для зберігання тексту нової нотатки, яку вводить користувач.

Потім визначається функція handleAddNote, яка виконується при натисканні кнопки "Add Note". У тілі цієї функції спочатку перевіряється, чи не є введений текст порожнім (після видалення пробілів). Якщо текст не порожній, функція оновлює стан notes, додаючи до нього нову нотатку, шляхом створення нового масиву, що містить всі попередні нотатки плюс нову. Після цього поле для введення очищується, скидаючи значення note до порожнього рядка. Також виконується POST-запит до API на ендпоінт /notes, де передається текст нотатки.

У методі return компонента відбувається рендеринг JSX-елементів. Створюється контейнер div з певними стилями, в якому міститься заголовок "Notes". Після заголовка йде блок для введення, що складається з текстового поля textarea і кнопки "Add Note". Текстове поле прив’язане до стану note, і зміна його значення обробляється за допомогою функції onChange, яка оновлює стан note при введенні користувача.

Кнопка "Add Note" має обробник onClick, що викликає функцію handleAddNote при натисканні.

Внизу компонента знаходиться контейнер div, в якому відображаються всі нотатки з масиву notes. За допомогою методу map кожна нотатка рендериться як новий div з унікальним ключем key, що дорівнює індексу нотатки в масиві. Кожен з цих div містить текст нотатки, що забезпечує простий інтерфейс для перегляду нотаток, які були додані користувачем.

```
const Notes = () => {
  const [notes, setNotes] = useState([]);
  const [note, setNote] = useState("");

  const handleAddNote = () => {
    if (note.trim() !== "") {
      setNotes([...notes, note]);
```

```
    setNote("");
    axios.post('/notes', note)
  }
 };

 return (
  <div style={styles.container}>
   <h1 style={styles.header}>Notes</h1>
   <div style={styles.inputContainer}>
    <textarea
     style={styles.textarea}
     value={note}
     onChange={(e) => setNote(e.target.value)}
     placeholder="Enter your note here..."
    />
    <button style={styles.button} onClick={handleAddNote}>
     Add Note
    </button>
   </div>
   <div style={styles.notesContainer}>
    {notes.map((note, index) => (
     <div key={index} style={styles.note}>
      {note}
     </div>
    ))}
   </div>
  </div>
 );
};
```

Спочатку визначається стан isLogoutScheduled за допомогою хука useState, ініціалізуючи його значенням false. Цей стан може використовуватися для відстеження, чи запланований вихід.

Наступною є функція logOut, яка приймає параметр message. У тілі цієї функції спочатку виконується очищення локального та сесійного сховища за допомогою методів localStorage.clear() і sessionStorage.clear(). Це видаляє всі збережені дані користувача, зокрема токени або інформацію про сесію, що гарантує, що користувач вийде з системи без залишення слідів.

Після очищення сховищ функція викликає метод navigate з бібліотеки react-router, щоб перенаправити користувача на сторінку аутентифікації, яка зазвичай є сторінкою входу в систему.

Потім відбувається перевірка, чи був переданий параметр message. Якщо так, то відображається повідомлення про помилку з використанням функції toast.error, що показує текст, переданий у параметрі message. Якщо ж параметр не був переданий, функція відображає повідомлення про успішний вихід з системи, використовуючи toast.success з текстом "Log out Successfully".

На завершення функція оновлює стан isLogoutScheduled, встановлюючи його в true, це може сигналізувати про те, що вихід було заплановано або виконано. Це може бути корисно для умовної логіки в компоненті, пов'язаної з відображенням певних елементів або поведінкою після виходу з системи.

```
const [isLogoutScheduled, setIsLogoutScheduled] = useState(false);

const logOut = (message) => {
    localStorage.clear();
    sessionStorage.clear();
    navigate("/auth");
    if (message) {
        toast.error(message);
    } else {
        toast.success("Log out Successfully");
    }
    setIsLogoutScheduled(true);
};
```

На початку функції за допомогою блоку try забезпечується обробка можливих помилок під час виконання коду. Функція приймає два параметри: req (об'єкт запиту) та res (об'єкт відповіді).

Спочатку з об'єкта req.body вилучаються дані, введені користувачем: username, password, firstName, lastName та phoneNumber. Далі виконується запит до бази даних для перевірки наявності користувача з таким же ім'ям користувача, використовуючи метод findOne моделі User. Якщо користувач з таким іменем вже існує, функція повертає відповідь зі статусом 401 і повідомленням про помилку, що вказує на те, що такий користувач вже зареєстрований.

Якщо користувача не знайдено, код продовжує виконання. Наступним кроком є хешування пароля за допомогою бібліотеки bcrypt, де метод hash приймає пароль та значення "10", що є кількістю раундів хешування. Хешований пароль зберігається у змінній hashedPassword.

Потім створюється новий об'єкт користувача на основі моделі User, де передаються всі необхідні поля, включаючи хешований пароль і дату створення облікового запису (createdDate), що встановлюється на поточну дату.

Після цього відбувається збереження нового користувача в базі даних за допомогою методу save. Якщо збереження проходить успішно, функція повертає відповідь зі статусом 200 і повідомленням про успішну реєстрацію користувача.

Якщо під час виконання будь-якого з попередніх кроків виникає помилка, вона ловиться у блоці catch, і функція повертає відповідь зі статусом 500, разом з інформацією про помилку. Це дозволяє правильно обробляти будь-які проблеми, що можуть виникнути під час процесу реєстрації.

```
const register = async (req, res) => {
  try {
    const { username, password, firstName, lastName, phoneNumber } = req.body;
    const user = await User.findOne({ username: username })

    if (user) {
      return res.status(401).json({ message: "user already exist please try another email" })
```

```
    } else {
      // Hash the password
      const hashedPassword = await bcrypt.hash(password, 10);
      // Create a new user
      const user = new User({ username, password: hashedPassword, firstName, lastName,
phoneNumber, createdDate: new Date() });
      // Save the user to the database
      await user.save();
      res.status(200).json({ message: 'User created successfully' });
    }
  } catch (error) {
    res.status(500).json({ error });
  }
}
```

3.6 Тестування інформаційної технології управління відносинами з клієнтами та аналіз отриманих результатів

Тестування є критично важливим етапом при розробці нового програмного продукту, оскільки воно дозволяє переконатися, що програма виконує всі очікувані функції та працює стабільно, без збоїв і помилок. Основною метою тестування є виявлення усіх можливих проблем та недоліків у роботі програми ще до того, як вона стане доступною для користувачів. Це допомагає підвищити якість продукту та уникнути можливих труднощів у майбутньому, що також позитивно позначається на рівні задоволеності користувачів.

Для перевірки розробленого CRM-продукту було проведено понад 200 тестових запусків, протягом яких ретельно перевірено його функціональні можливості та оцінено стабільність роботи кожного модуля. Тестування підтвердило здатність програми виконувати всі заявлені функції, що свідчить про її високу надійність і відповідність поставленим вимогам.

Для того, щоб почати роботу з інформаційною технологією, потрібно мати персональний комп’ютер. Головною умовою є стабільне підключення до мобільної мережі інтернету або Wifi.

На рисунку 3.6 зображено вікно авторизації користувача. Після успішної авторизації користувач переадресовується на вікно головного меню програми. Це вікно містить всю необхідну користувачу інформацію. Наприклад, у цьому вікні відображається статистика клієнтів та інформація про них, деяка інформація про активність менеджерів, а саме: відображає працює даний менеджер чи ні. Також у цьому вікні знаходиться коротка таблиця історії останніх замовлень користувача. Ця таблиця містить колонки: Completed, Pending, In Progress, Todo та On Hold. Також можна побачити на графіку звіт про дані модуля в системі.

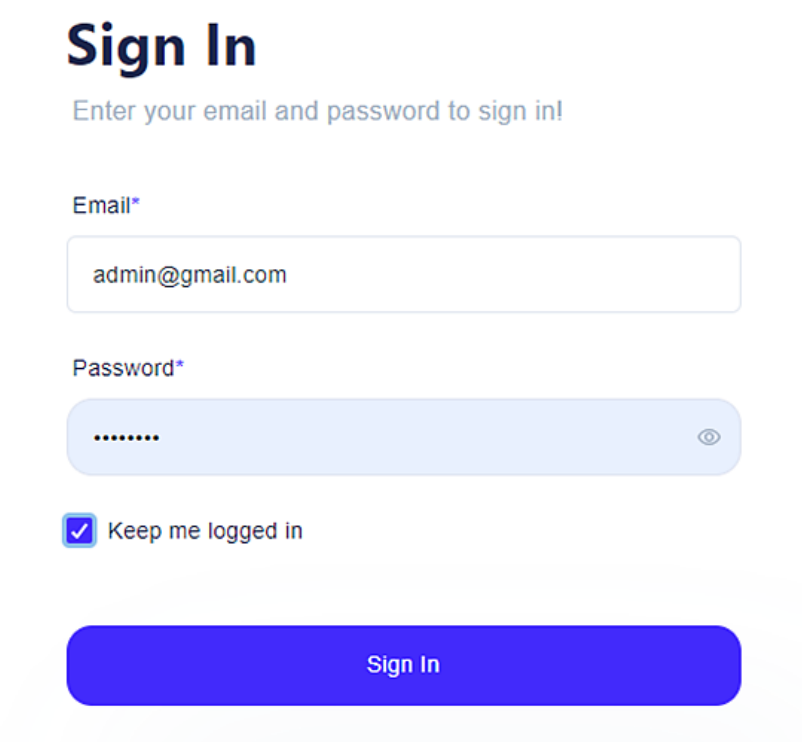

Рисунок 3.6 – Загальний вигляд інтерфейсного вікна авторизації користувача

На рисунку 3.7 подано «Головне вікно програми», де відображається графік, статистика та навігаційне меню.

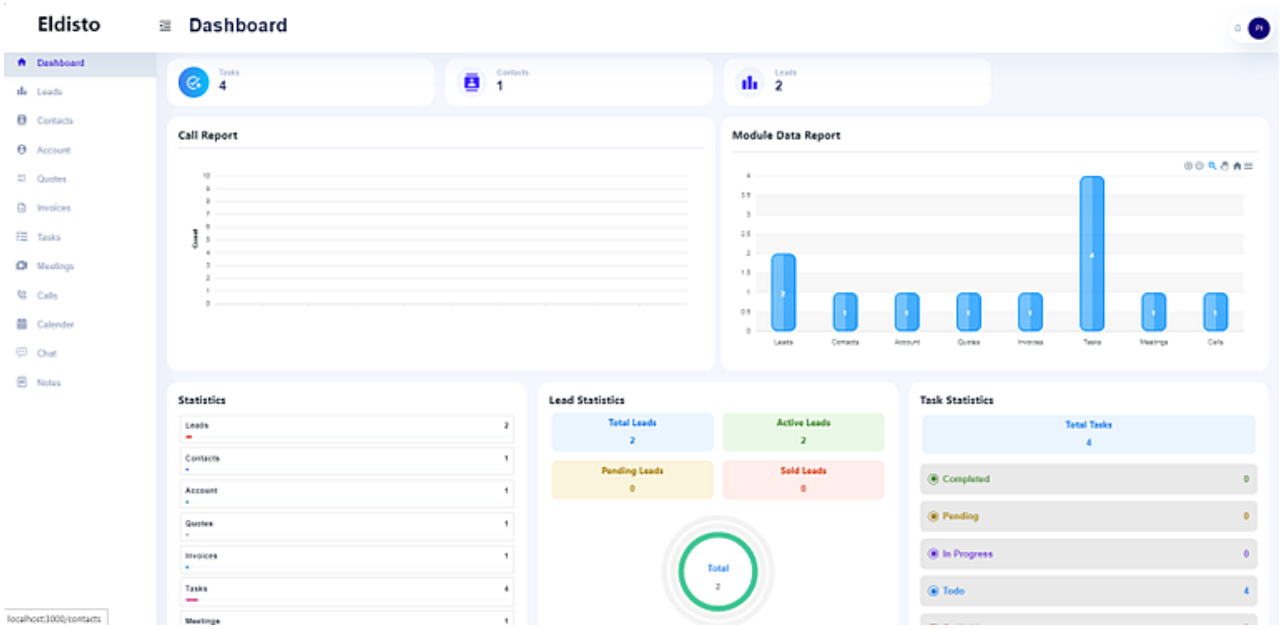

Рисунок 3.7 – Загальний вигляд інтерфейсного вікна головного меню з графіком

На рисунку 3.8 подано «Leads», де відображається інформація про менеджерів та керівництво.

Рисунок 3.8 – Загальний вигляд інтерфейсного вікна «Leads» з інформацією та кнопкою активності

На рисунку 3.9 подано «Contacts», де відображається таблиця з інформацією про клієнта.

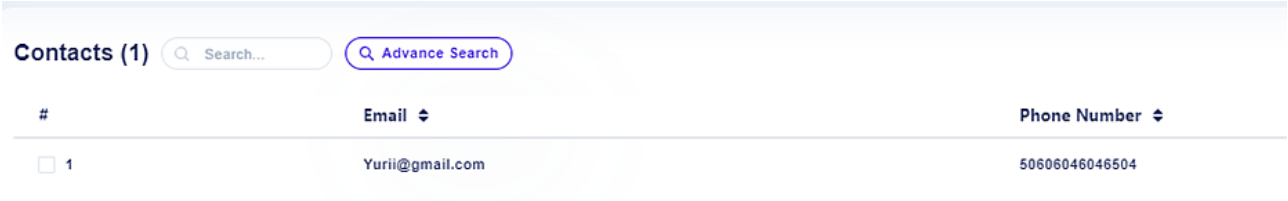

Рисунок 3.9 – Загальний вигляд інтерфейсного вікна «Contacts» з інформацією про користувача

На рисунку 3.10 подано «Account», де відображається інформація про створені акаунти і можна додати нові акаунти.

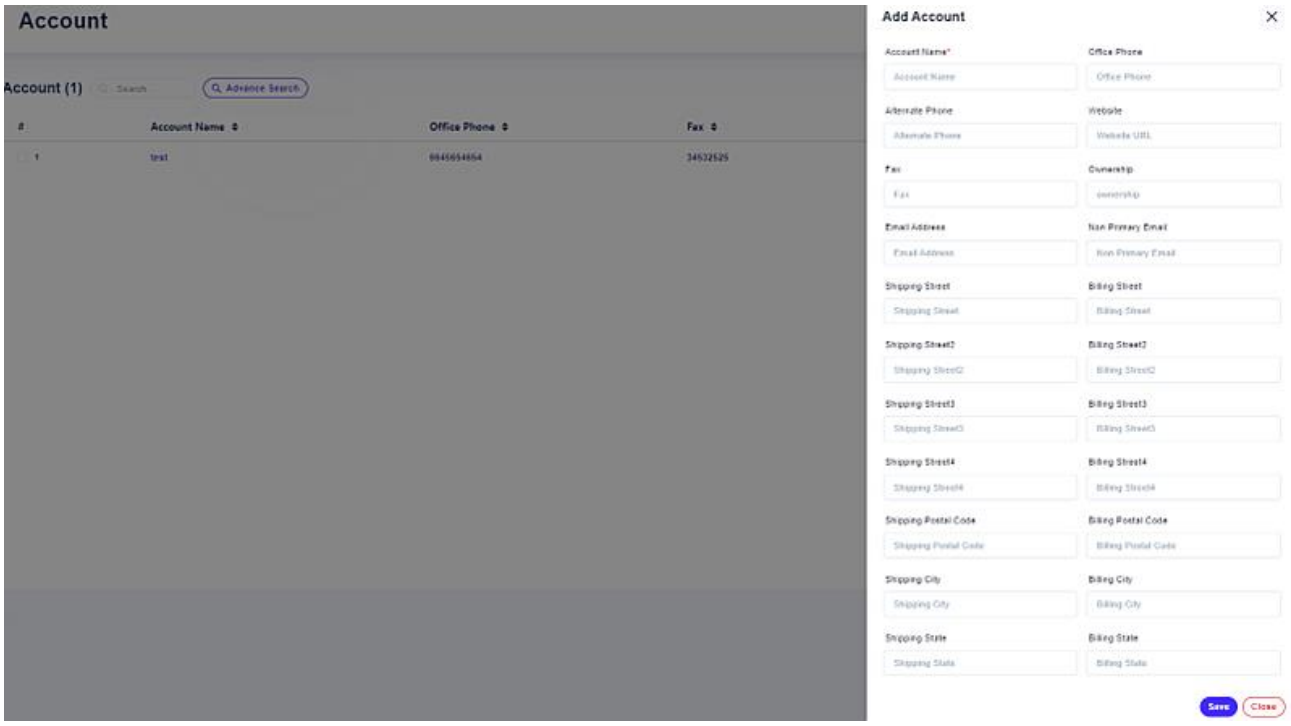

Рисунок 3.10 – Загальний вигляд інтерфейсного вікна «Account»

На рисунку 3.11 та рисунку 3.12 відображено вікно інформацію про замовлення і можна додати нове замовлення.

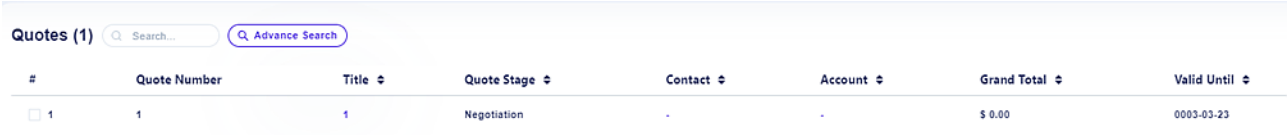

Рисунок 3.11 – Загальний вигляд інтерфейсного вікна «Quotes»

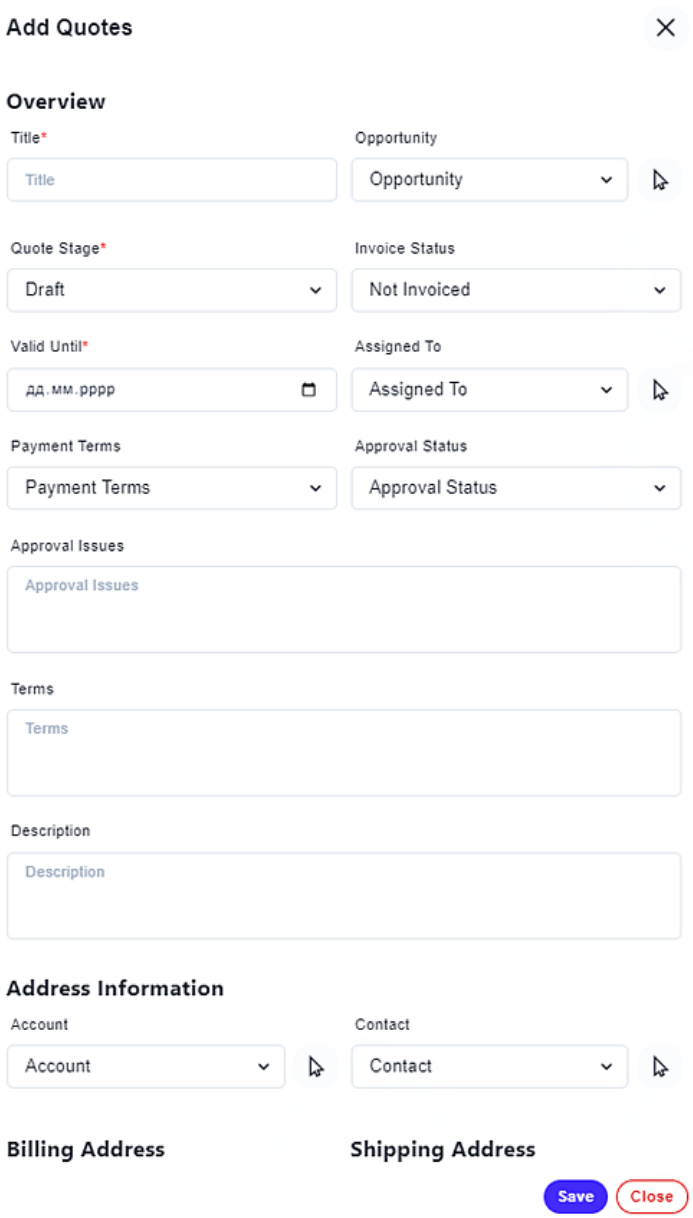

Рисунок 3.12 – Загальний вигляд інтерфейсного вікна додавання інформації про замовлення

На рисунку 3.13 подано «Invoices», де відображається інформація про статус замовлення.

Рисунок 3.13 – Загальний вигляд інтерфейсного вікна інформації про замовлення

На рисунку 3.14 та рисунку 3.15 подано «Tasks», де відображається інформація завдань для менеджерів, а також кнопка додавання нового завдання.

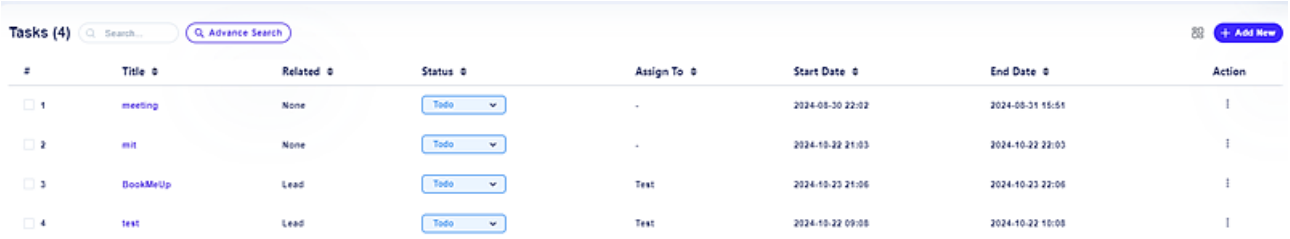

Рисунок 3.14 – Загальний вигляд інтерфейсного вікна інформації про завдання

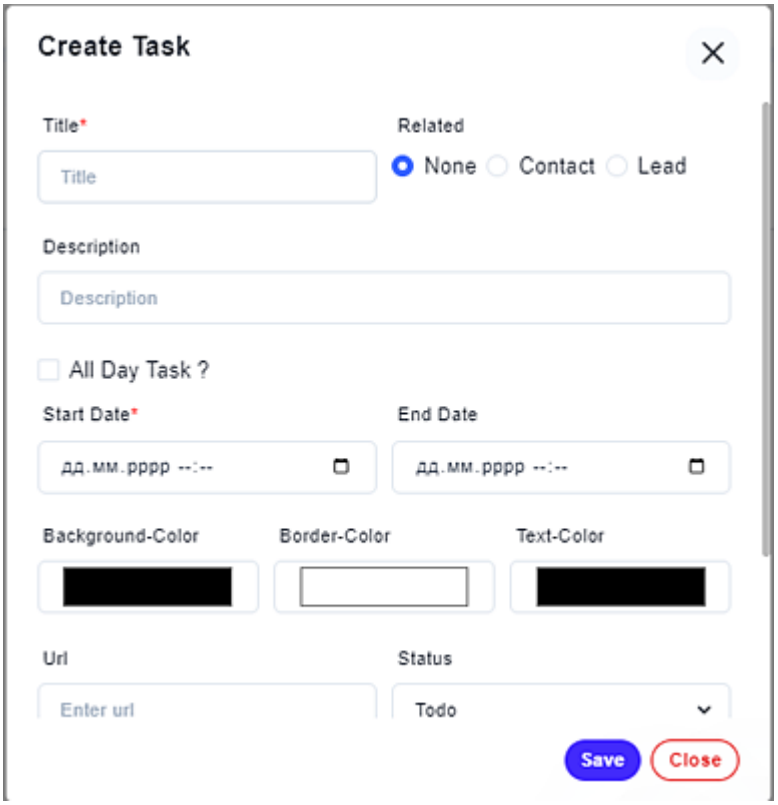

Рисунок 3.15 – Загальний вигляд інтерфейсного вікна додавання нового завдання

На рисунку 3.16 подано «Meetings», де відображається інформація про заплановані мітинги, на яких буде оговорюватись загальна інформація про продажі та підвищення продуктивності менеджерів, також на цій сторінці можна додати новий мітинг, що і зображено на рисунку 3.17.

Рисунок 3.16 – Загальний вигляд інтерфейсного вікна «Meeting»

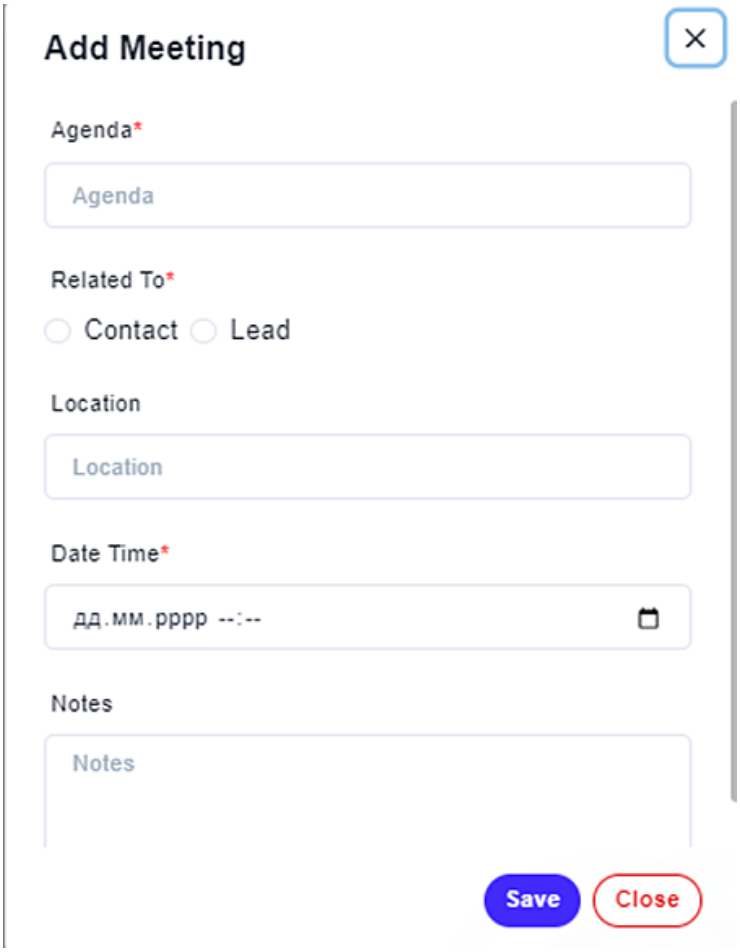

Рисунок 3.17 – Загальний вигляд інтерфейсного вікна додавання мітингу

На рисунку 3.18 подано «Calls», де відображається інформація про дзвінки до клієнтів або від клієнтів для узгодження замовлення.

Рисунок 3.18 – Загальний вигляд інтерфейсного вікна «Calls»

На рисунку 3.19 подано «Calender», де відображається інформація про завдання для менеджерів.

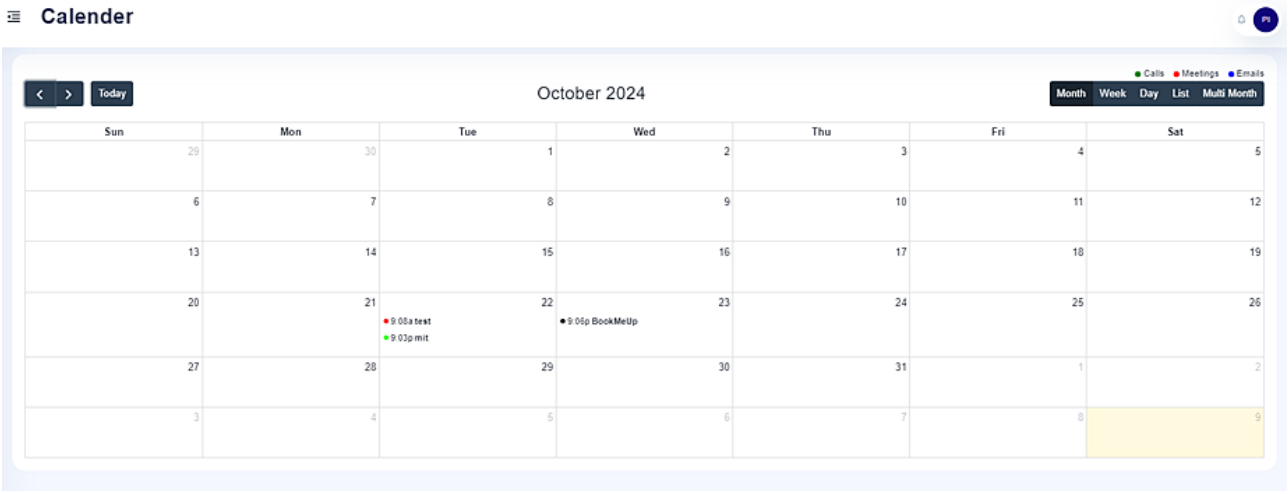

Рисунок 3.19 – Загальний вигляд інтерфейсного вікна «Calender»

На рисунку 3.20 подано «Chat», де відображається чат з клієнтом.

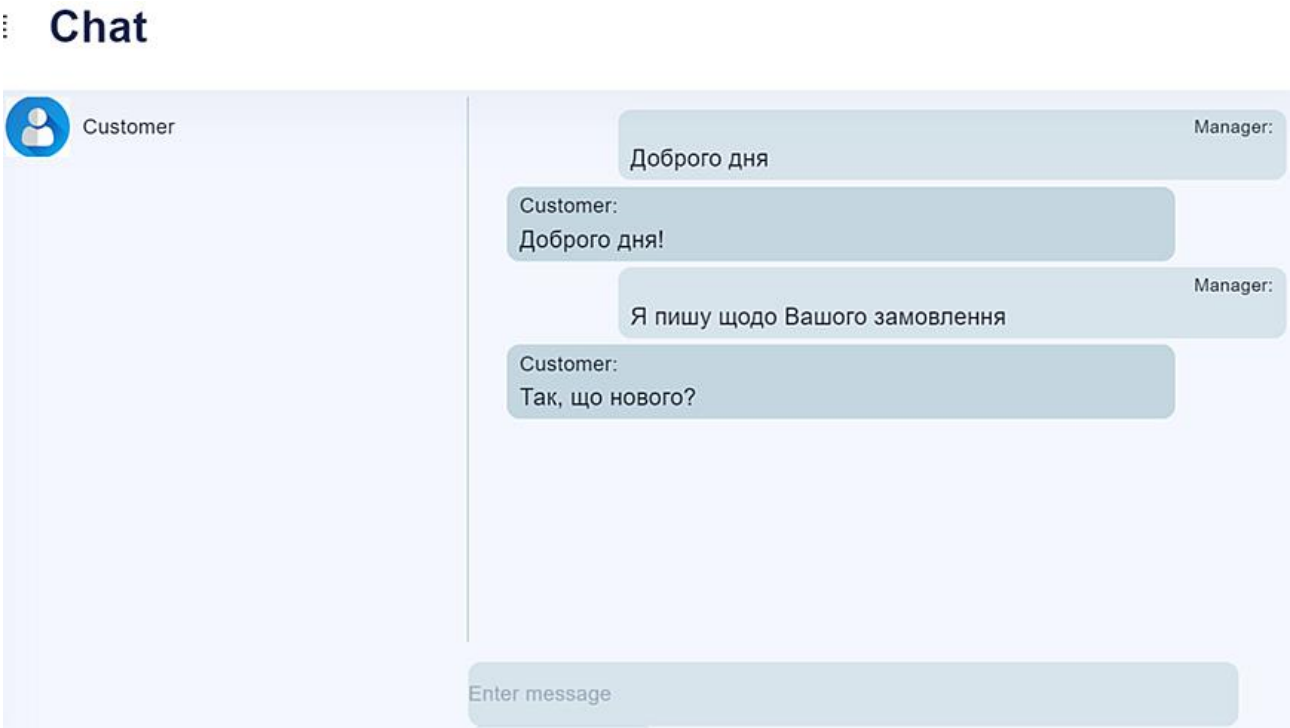

Рисунок 3.20 – Загальний вигляд інтерфейсного вікна «Chat»

На рисунку 3.21 подано «Notes», де відображаються нотатки менеджера.

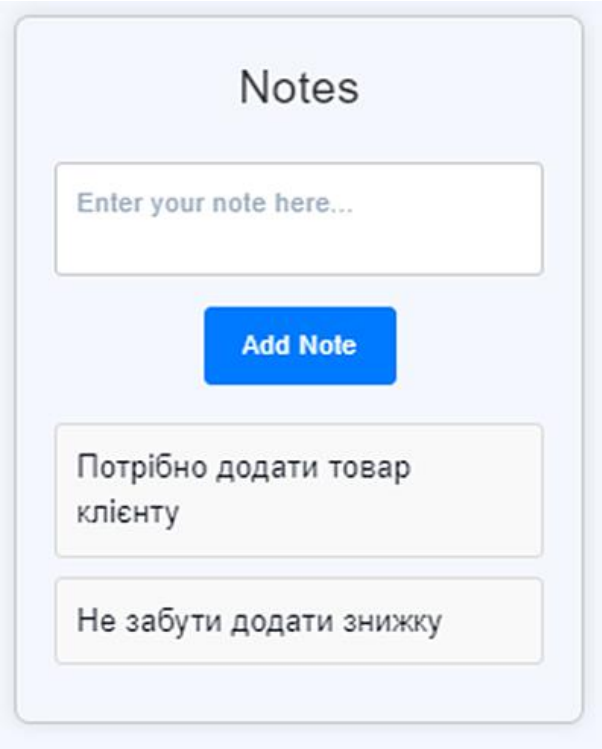

Рисунок 3.21 – Загальний вигляд інтерфейсного вікна «Notes»

Розроблений програмний продукт для управління відносинами з клієнтами успішно пройшов тестування та відповідає всім заданим вимогам. У таблиці 3.1 подано результати тестування розробленого продукту та аналогічних CRM-систем, таких як Salesforce, HubSpot, Zoho CRM, та Pipedrive.

В процесі розробки вдалося реалізувати функціонал, який був відсутній або обмежений в системах-аналогах, зокрема:

- інтегрований чат для швидкої комунікації з клієнтами;
- можливість додавання та зберігання нотаток, які на відміну від аналогів мають спрощений механізм додавання нотаток;
- оптимізація користувацького інтерфейсу, що робить роботу з програмою швидшою та інтуїтивно зрозумілою;
- доступна ціна, що робить систему привабливою для малого та середнього бізнесу.

Таблиця 3.1 − Результати тестування програмного продукту та аналогічних CRM-систем

Параметр	Розробка	Salesforce	HubSpot	Zoho CRM	Pipedrive
Інтегрований чат	Так	Ні	Ні	Так	Ні
Нотатки	Так	Так	Так	Так	Так
Оптимізація інтерфейсу	Так	Так	Так	Ні	Ні
Вартість	Нижча	Висока	Середня	Середня	Середня

Результати тестування підтвердили, що розроблена CRM-система здатна забезпечити ефективне управління відносинами з клієнтами завдяки розширеному функціоналу, зручності використання та конкурентній вартості.

Отже, проаналізувавши та протестувавши розроблений програмний продукт, можна зробити висновок, що поставлених цілей було досягнуто. З таблиці 3.1 видно, що розроблений програмний продукт має покращений функціонал у порівнянні з аналогами щонайменше за 4 ключовими можливостями: наявність інтегрованого чату, можливість додавання нотаток, оптимізований інтерфейс та доступна ціна.

Розширений функціонал забезпечить вище задоволення користувачів від використання цього програмного продукту, що сприятиме підвищенню ефективності управління відносинами з клієнтами.

Потрібно зауважити, що розроблений програмний продукт на відміну від аналогів займає менший обсяг пам’яті на сервері.

На рисунку 3.22 для кращого сприйняття результатів тестування подано оцінювання функціональних можливостей розробки та аналогів за чотири бальною шкалою.

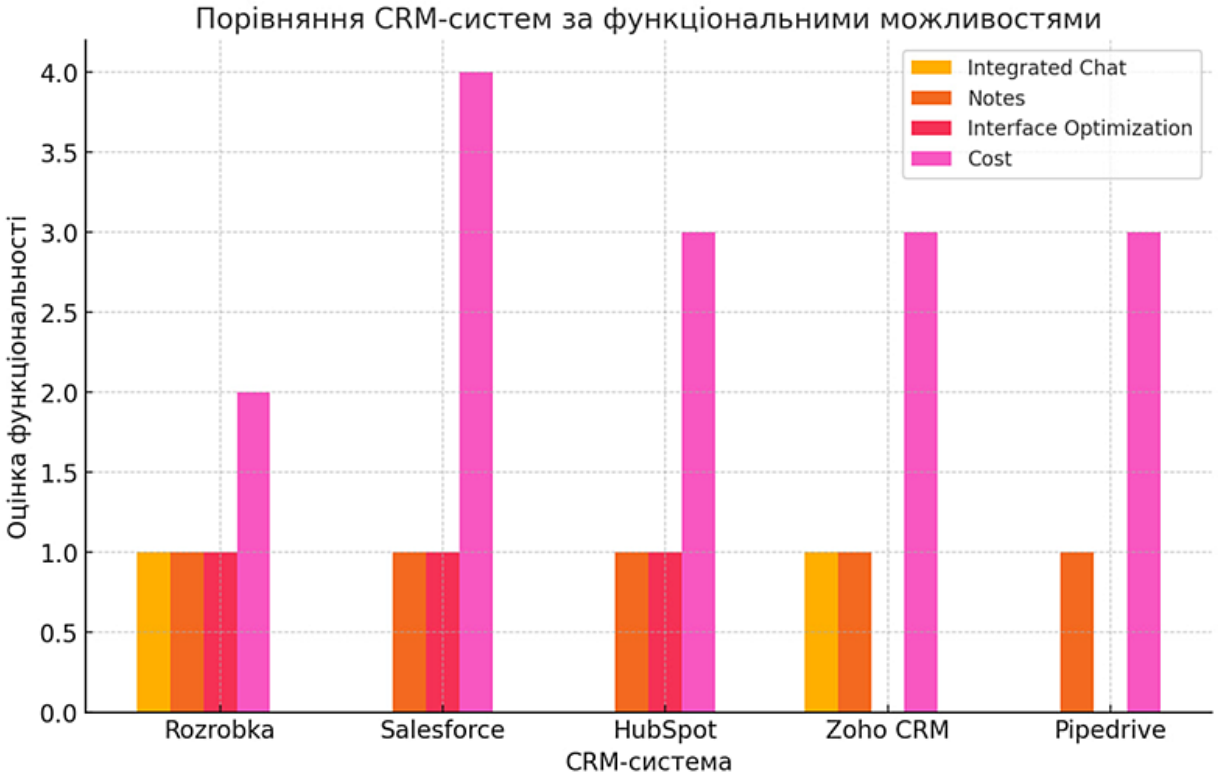

Рисунок 3.22 − Оцінювання функціональних можливостей розробки та аналогів

З рисунка 3.22 видно, що інтегрований чат є тільки в Rozrobka і Zoho CRM − 1 бал, у всіх інших системах цієї функції немає – 0 балів. Всі CRM-системи мають підтримку нотаток за однаковою оцінкою для цього параметра – 1 бал. В Zoho CRM і Pipedrive оптимізація інтерфейсу відсутня, тобто мають низьку оцінку – 0 балів. Потрібно зазначити те, що вартість є ще одним важливим параметром, яка для Rozrobka оцінена як «Низька – 2 бали», а для Salesforce − «Висока – 4 бали», інші CRM-системи мають середню вартість − 3 бали.

На рисунку 3.23 подано порівняльні результати швидкості роботи розробки та аналогів: Salesforce, HubSpot, Zoho CRM та Pipedrive за п’яти бальною шкалою.

З рисунка 3.23 видно, що серед інших CRM-систем найвищу швидкість роботи має Rozrobka – 5 балів, найменшу швидкість роботи має HubSpot та Pipedrive – 2 бали.

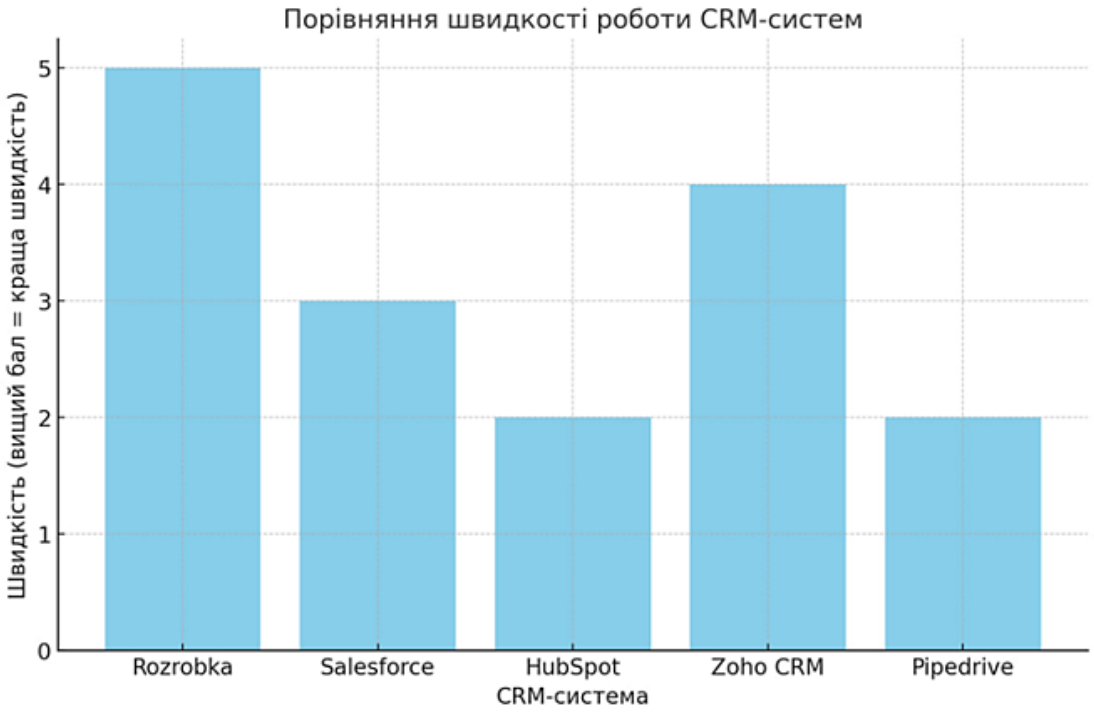

Рисунок 3.23 − Порівняння швидкості роботи розробленого програмного продукту та аналогів

Отже, на основі результатів тестування можна зробити висновок, що поставлених цілей було досягнуто.

3.7 Висновок до розділу 3

У цьому розділі було проведено аналіз мов програмування та технологій для розробки CRM-системи. Для клієнтської та серверної частини розглядалися різні мови програмування, зокрема TypeScript, але в результаті обрано JavaScript завдяки його гнучкості, популярності та широкій підтримці об’єктно-орієнтованого програмування. JavaScript забезпечує динамічну типізацію, що спрощує розробку, а також має тісну інтеграцію з сучасними середовищами розробки та бібліотеками, що дозволяє прискорити розробку та підтримувати високу продуктивність продукту.

Для управління базою даних обрано MongoDB. Це документоорієнтована NoSQL база даних, яка ідеально підходить для роботи з великими обсягами даних та забезпечує гнучкість у структурі збереження даних. MongoDB дозволяє легко масштабувати систему та адаптувати її до нових вимог, що є важливим для проекту.

Середовищем розробки обрано Visual Studio Code, який надає численні переваги, зокрема відкритий вихідний код, зручність у використанні, інтелектуальне автозавершення, вбудований відлагоджувач, інтеграцію з Git та підтримку різноманітних розширень, що значно підвищує продуктивність розробки.

Розроблені діаграми діяльності для бази даних дозволяють чітко уявити процеси та взаємозв'язки між сутностями, що допомагає в проектуванні та оптимізації системи. Вони відображають, як дані циркулюють у системі, як користувачі взаємодіють з додатком і як обробляються запити. Це дозволяє виявити потенційні проблеми та оптимізувати процеси, що забезпечує ефективну реалізацію проекту.

Тестування підтвердило, що всі основні функції було успішно реалізовано. У порівнянні з аналогічними CRM-системами, розробка має щонайменше 4 переваги: інтегрований чат, можливість залишати нотатки, оптимізований інтерфейс, а також конкурентоспроможну ціну.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Проведення комерційного та технологічного аудиту інформаційної технології управління відносинами з клієнтами.

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 4.1.

Таблиця 4.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу інформаційної технології управління відносинами з клієнтами

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	3	2	3
Ринкові переваги (наявність аналогів)	2	3	2
Ринкові переваги (ціна продукту)	3	2	3
Ринкові переваги (технічні властивості)	3	2	3
Ринкові переваги (експлуатаційні витрати)	2	3	3

Продовження таблиці 4.1

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Ринкові перспективи (розмір ринку)	3	3	2
Ринкові перспективи (конкуренція)	2	3	3
Практична здійсненність (наявність фахівців)	3	2	3
Практична здійсненність (наявність фінансів)	2	3	3
Практична здійсненність (необхідність нових матеріалів)	3	3	3
Практична здійсненність (термін реалізації)	3	2	3
Практична здійсненність (розробка документів)	3	2	2
Сума балів	32	30	31
Середньоарифметична сума балів, СБ	31		

За результатами розрахунків, наведених в таблиці 4.1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології управління відносинами з клієнтами – вище середнього.

4.2 Розрахунок витрат на здійснення науково-дослідної роботи інформаційної технології управління відносинами з клієнтами

4.2.1 Витрати на оплату праці

Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з

чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці» [30].

4.2.2 Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.1)$$

де к – кількість посад дослідників, залучених до процесу дослідження;

$М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_р$ – число робочих днів в місяці, приблизно $Т_р$ = (21…23) дні;

t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 4.2 [30].

Таблиця 4.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	40000	1905	25	47625
Розробник	32000	1524	28	42672
Консультанти	20000	952	21	19992
Всього:	110289			

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду С і можна визначити за формулою:

$$C_i = \frac{М_м \cdot К_i \cdot К_c}{Т_р \cdot t_{зм}}, \quad (4.3)$$

де $М_м$ – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), у 2024 році $М_м$=8000 грн;

$К_i$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

$К_c$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати;

$Т_р$ – середня кількість робочих днів в місяці, приблизно $Т_р$ = 21…23 дні;

$t_{зм}$ – тривалість зміни, год.

Зроблені розрахунки зводимо до таблиці 4.3 [30].

Таблиця 4.3 – Витрати на заробітну плату робітників

Найменування робіт	Трудомісткість, н-год.	Розряд роботи	Погодинна тарифна ставка	Тариф. коеф.	Величина, грн.
Аналіз предметної області	23	5	61,8	1,36	1421,4
Моделювання інформаційної технології	113	5	61,8	1,36	6983,4
Створення основної структури проєкту	18	6	65,9	1,45	1186,2
Створення та наповнення бази даних	79	6	65,9	1,45	5206,1
Тестування інформаційної технології	71	5	61,8	1,36	4387,8
Всього					19185

4.2.3 Додаткова заробітна плата

Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}11 \cdot (З_о + З_р) = 0{,}11 \cdot (110289 + 19185) = 14242 \text{ (грн.)}.$$

4.2.4 Відрахування на соціальні заходи

Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$
$$= 0{,}22 \cdot (110289 + 19185 + 14242) = 31618 \text{ (грн.)}.$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_р$ – основна заробітна плата робітників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

4.2.5 Програмне забезпечення

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_1^к Ц_{іпрг} \cdot С_{прг.і} \cdot К_і, \quad (4.4)$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.;

$С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_і$ = (1,1…1,12); к – кількість видів програмного забезпечення. Програмне забезпечення, що використовувалось, можемо побачити у таблиці 4.4 [30].

Таблиця 4.4 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
Navicat for MongoDB	19863	1	19863
VS Code Professional	1845	1	1845
Всього, з врахуванням коефіцієнта інсталяції та налагодження			23879

4.2.6 Амортизація обладнання

Амортизація обладнання, комп’ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_б}{Т_в} \cdot \frac{t}{12}, \quad (4.5)$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп’ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.;

t – термін використання основного фонду, місяці;

$Т_в$ – термін корисного використання основного фонду, роки.

Основне обладнання можемо побачити у таблиці 4.5 [30].

Таблиця 4.5 – Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Ноутбук	40000	2	1,5	2500
Стіл	8000	5	1,5	200
Стілець	4000	5	1,5	100
Мишка	1000	2	1,5	62,5
Клавіатура	1200	2	1,5	75
Настільна лампа	1250	3	1,5	52
Всього	2990			

4.2.7 Витрати на електроенергію для науково-виробничих цілей

Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

$$Ве = \sum \frac{W_i \cdot t_i \cdot Ц_е \cdot К_{впі}}{ККД} = \frac{0{,}23 \cdot 300 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} + \frac{0{,}034 \cdot 130 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} = 534\ (грн.),$$

де W_i – встановлена потужність обладнання, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год., Це – вартість 1 кВт електроенергії, грн.;

$К_{впі}$ – коефіцієнт використання потужності;

ККД – коефіцієнт корисної дії обладнання.

В таблиці 4.6 можна побачити скільки витрачається на електроенергію [30].

Таблиця 4.6 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Ноутбук	0,23	300
Освітлення	0,034	130

4.2.8 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$Ів = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%} = (110289 + 19185) \cdot \frac{55}{100} = 71211 \text{ (грн.)},$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

4.2.8 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$Внзв = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%} = (110289 + 19185) \cdot \frac{110}{100} = 142421 \text{ (грн.)},$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + \\ + І_в + В_{нзв} = \\ = 110289 + 19185 + 14242 + 31618 + 23879 + 2990 + 534 + 71211 \\ + 142421 = \ 416369 \ (грн.).$$

4.3.1 Загальні витрати

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{416369}{0{,}9} = 462632 \ (грн.),$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Оскільки, якщо науково-технічна розробка знаходиться на стадії впровадження, то η=0,9 [30].

4.4 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N_і)_і \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right), \qquad (4.6)$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки);

$\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни);

N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки;

$Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році;

$Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів;

ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки);

λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П = ((1640 - 1400) \cdot 600 - (2820 - 600) \cdot 1400) \cdot 0{,}8333 \cdot 0{,}5 \cdot \left(1 - \frac{18}{100}\right) = 1675740 \text{ (грн.)}.$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{1675740}{(1+0{,}1)^1} = 1523400 \text{ (грн.)},$$

де ΔП – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.;

Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=1 рік);

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} \cdot \text{ЗВ} = 2 \cdot 462632 = 925264 \text{ (грн.)}.$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай kінв=2...5, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$\text{E}_{\text{абс}} = \text{ПП} - PV = 1523400 - 925264 = 598136 \text{ (грн.)},$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн.;

PV – теперішня вартість початкових інвестицій, грн.

Оскільки $E_{абс}>0$, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої

норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{Еабс}{PV}} - 1 = \sqrt[1]{1 + \frac{598136}{925264}} - 1 = 0{,}28,$$

де $T_ж$ – життєвий цикл розробки, роки.

Визначимо бар'єрну ставку дисконтування $\tau_{мін}$, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть [30].

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f = 0{,}12 + 0{,}05 = 0{,}17,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,9...0,12;

f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f =0,05...0,5, але може бути і значно вищою.

Оскільки $E_в$=0,28>$\tau_{мін}$=0,17, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховуємо період окупності інвестицій T_o, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_o = \frac{1}{E_в} = \frac{1}{0{,}28} = 3{,}6 \text{ (року)}.$$

Оскільки T_o<1…5-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок [30].

4.5 Висновок до розділу 4

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія управління відносинами з клієнтами»» становить 31 бал, що свідчить про комерційну важливість проведення цих досліджень, оскільки рівень комерційного потенціалу розробки високий.

Також термін окупності становить 3,6 роки, що менше 5-ти років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок. Отже, можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія управління відносинами з клієнтами».

ВИСНОВКИ

Всі завдання, поставлені для реалізації інформаційної технології управління відносинами з клієнтами виконані в повному обсязі, а саме: проаналізовано технічній рівень існуючих систем управління відносинами з клієнтами та обґрунтувати доцільність розробки; удосконалено математичну модель інформаційної технології управління відносинами з клієнтами; розглянуто методи та технології, які використовуються для управління взаємовідносинами з клієнтами; спроектовано архітектуру та структуру програмного модуля управління відносинами з клієнтами; проаналізовано та обґрунтовано вибір інструментів для розробки; виконана програмна реалізація, проведено тестування програми та аналіз отриманих результатів; економічно обґрунтована доцільність розробки інформаційної технології управління відносинами з клієнтами.

При виконанні магістерської кваліфікаційної роботи реалізовано інформаційну технологію управління відносинами з клієнтами.

Проведено аналіз технічних рішень для систем управління відносинами з клієнтами (CRM) та доведено доцільність розробки інформаційної технології управління взаємовідносинами з клієнтами, яка включає нові функціональні можливості, зокрема чат та нотатки. Розглянуті системи-аналоги є корисними та популярними, проте вони представлені у вигляді комерційних рішень і не мають таких розширених функцій або обмежують користувача певними рамками.

На основі аналізу існуючих моделей і методів, які застосовуються для вирішення поставленої задачі була удосконалена математична модель інформаційної технології управління відносинами з клієнтами. Завдяки такій моделі підприємства отримують можливість оптимізувати свої ресурси та покращити якість обслуговування, що сприятиме збільшенню прибутку та конкурентоспроможності на ринку.

Для вирішення поставлених завдань, було обґрунтовано використання JavaScript як основної мови програмування, а також Node.js для серверної

частини. Ці технології дозволяють реалізувати інтерактивні та масштабовані рішення, що включають функціонал чату та нотаток, який забезпечує гнучкість і зручність у взаємодії з користувачем.

Була розроблена структура інформаційної технології управління відносинами з клієнтами, що включає інтеграцію чату та нотаток. Спроектовано алгоритм роботи системи та вибрано архітектуру, що дозволяє ефективно обробляти дані та забезпечувати безперебійну роботу чату і нотаток для клієнтів.

Розробка програмного продукту передбачала використання JavaScript для клієнтської частини, що дозволяє створити зручний та інтуїтивно зрозумілий інтерфейс. Серверна частина, реалізована за допомогою Node.js, забезпечує стабільну та ефективну обробку запитів, що дозволяє працювати з великими обсягами даних у реальному часі.

Проведено тестування розробленої програми, яке підтвердило її працездатність. Порівняно з існуючими системами-аналогами, розроблена інформаційна технологія має покращений функціонал у порівнянні з аналогічними продуктами щонайменше на 4 можливості: інтегрований чат, можливість залишати нотатки, оптимізований інтерфейс, а також конкурентоспроможну ціну.

Економічно обґрунтувано доцільність розробки інформаційної технології управління відносинами з клієнтами. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат, результатом стала сума 462 632 гривень. Період окупності інформаційної технології управління відносинами з клієнтами складе близько 3,6 років.

Поставлена мета, а саме розширення функціональних можливостей інформаційної технології управління відносинами з клієнтами була досягнута за рахунок використання комплексної математичної моделі. Це дозволило розробити зручний WEB-додаток для інформаційної технології управління відносинами з клієнтами, який відповідає всім запитам користувача, інтуїтивно зрозумілий та конкурує з іншими системами-аналогами на ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шаргало Д. В Крилик Л. В. Перспективи розробки інформаційної технології управління відносинами з клієнтами. Матеріали LIII науково-технічної конференції підрозділів Вінницького національного технічного університету, Вінниця, 20 − 22 березня 2024 р. [Електронний ресурс]. URL: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20386/16891 (дата звернення 05.11.2024).

2. Свідоцтво про реєстрацію авторського права на твір № 130631. Комп'ютерна програма «Інформаційна технологія управління відносинами з клієнтами» / Крилик Л. В., Шаргало Д. В., дата реєстрації 14.10.2024 р.

3. Що таке CPM система та як вона працює [Електронний ресурс]. URL: https://www.creatio.com/ua/crm/what-is-crm (дата звернення: 13.09.2024).

4. Fatouretchi Max. The Art of CRM: Proven strategies for modern customer relationship management. Packt publishing ltd: BIRMINGHAM-MUMBAI, 2019. 335 p.

5. Salesforce [Електронний ресурс]. URL: https://www.salesforce.com (дата звернення: 13.09.2024).

6. HubSpot [Електронний ресурс]. URL: https://www.hubspot.com (дата звернення: 13.09.2024).

7. Zoho CRM [Електронний ресурс]. URL: https://www.zoho.com (дата звернення: 13.09.2024).

8. Pipedrive [Електронний ресурс]. URL: https://www.pipedrive.com/uk (дата звернення: 13.09.2024).

9. Box G. E., Jenkins G. M., Reinsel G. C., Ljung G. M. Time Series Analysis: Forecasting and Control. John Wiley & Sons, 2015. 704 с.

10. CRM-система [Електронний ресурс].

URL: https://sendpulse.ua/support/glossary/crm (дата звернення: 17.09.2024).

11. CRM-система − це ключ до розвитку бізнесу [Електронний ресурс]. URL: https://remonline.ua/blog/what-is-crm/ (дата звернення: 17.09.2024).

12. Як обрати CRM-систему для служби підтримки [Електронний ресурс]. URL: https://dou.ua/forums/topic/46758/ (дата звернення: 17.09.2024).

13. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Science & Business Media, 2009. 745 с.

14. Що таке JavaScript, та як функціонує JavaScript [Електронний ресурс]. URL: http://ruszura.in.ua/uncategorized/scho-take-javascript-yakfunktsionuyu javascript.html (дата звернення 20.09.2024).

15. NodeJS [Електронний ресурс]. URL: https://nodejs.org/en (дата звернення 20.09.2024).

16. ExpressJs [Електронний ресурс]. URL: https://expressjs.com (дата звернення 20.09.2024).

17. JavaScript [Електронний ресурс]. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення 20.09.2024).

18. MacKay D. J. Information Theory, Inference and Learning Algorithms. Cambridge University Press, 2003. 618 с.

19. Wirth R., Hipp J. CRISP-DM: Towards a Standard Process Model for Data Mining. In Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining, 2000. Pp. 29-39.

20. Kumar V., Reinartz W. Customer Relationship Management: Concept, Strategy, and Tools. Springer, 2018. 572 с.

21. Ngai E. W. T., Xiu L., Chau D. C. K. Application of Data Mining Techniques in Customer Relationship Management: A Literature Review and Classification. Expert Systems with Applications, 2009, 36(2), Pp. 2592-2602.

22. UML diagrams [Електронний ресурс]. URL: https://www.lucidchart.com/blog/types-of-UML-diagrams (дата звернення 20.10.2024).

23. Software architecture [Електронний ресурс]. URL: https://sarrahpitaliya.medium.com/understanding-software-architecture-a-complete-guide-cb8f05900603 (дата звернення 20.10.2024).

24. Javascript as a programming language [Електронний ресурс]. URL: https://fortyseven47.com/blog/javascript-as-a-programming-language/ (дата звернення 20.10.2024).

25. Development framework [Електронний ресурс]. URL: https://www.3pillarglobal.com/insights/blog/key-decision-criteria-for-selecting-a-development-framework/ (дата звернення 20.10.2024).

26. Database for your application [Електронний ресурс]. URL: https://wearebrain.com/blog/how-to-choose-the-right-database-for-your-application/ (дата звернення 20.10.2024).

27. Importance of selecting programming languages [Електронний ресурс]. URL: https://iies.in/blog/importance-of-selecting-programming-languages-carefully/ (дата звернення 20.10.2024).

28. Яровий А. А., Крилик Л. В., Козловський А. В. Сучасні інформаційні технології у сфері штучного інтелекту : електронний навчальний посібник комбінованого (локального та мережного) використання [Електронний ресурс]. Вінниця : ВНТУ, 2023. 145 с.

29. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. К. Колесницький. Вінниця : ВНТУ, 2023. (PDF, 58 с.)

30. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад.: В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

Додаток А (обов'язковий)

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія управління відносинами з клієнтами

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Turnitin

Оригінальність 93,00% Схожість 7,00%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Turnitin щодо роботи.

Автор роботи _____ Шаргало Д.В.

Керівник роботи _____ Крилик Л.В.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку _____ Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
import { Icon } from "@chakra-ui/react";
import { HiUsers } from "react-icons/hi";
import {
  MdContacts,
  MdHome,
  MdInsertChartOutlined,
  MdLeaderboard,
  MdLock
} from "react-icons/md";
// icon
import React from "react";
import { AiFillFolderOpen, AiOutlineMail } from "react-icons/ai";
import { FaCalendarAlt, FaRupeeSign, FaTasks, FaWpforms } from "react-icons/fa";
import { LuBuilding2 } from "react-icons/lu";
import { PiPhoneCallBold } from "react-icons/pi";
import { FaCreativeCommonsBy } from "react-icons/fa";
import { SiGooglemeet  } from "react-icons/si";
import { ROLE_PATH } from "./roles";
import ChangeImage from "views/admin/image";
import Validation from "views/admin/validation";
import CustomField from "views/admin/customField";
import TableField from "views/admin/tableField";
import activeDeactiveModule from "views/admin/activeDeactiveModule";
import { TbExchange, TbTableColumn } from "react-icons/tb";
import { GrValidate } from "react-icons/gr";
import { VscFileSubmodule } from "react-icons/vsc";
import { HiTemplate } from "react-icons/hi";
import { TbBulb } from "react-icons/tb";
import { BsBlockquoteRight } from "react-icons/bs";
import { RiAccountCircleFill } from "react-icons/ri";
import { TbInvoice } from "react-icons/tb";
import { TbFileInvoice } from "react-icons/tb";
import { IoChatboxEllipsesOutline } from "react-icons/io5";
import { CgNotes } from "react-icons/cg";
// Admin Imports
const MainDashboard = React.lazy(() => import("views/admin/default"));
```

```
// My component
const Contact = React.lazy(() => import('views/admin/contact'));
const ContactView = React.lazy(() => import('views/admin/contact/View'));
const ContactImport = React.lazy(() =>
import("views/admin/contact/components/ContactImport"));

const Quotes = React.lazy(() => import('views/admin/quotes'));
const QuotesView = React.lazy(() => import('views/admin/quotes/View'));
const QuotesImport = React.lazy(() =>
import("views/admin/quotes/components/QuotesImport"));

const Invoices = React.lazy(() => import('views/admin/invoice'));
const InvoicesView = React.lazy(() => import('views/admin/invoice/View'));
const InvoicesImport = React.lazy(() =>
import("views/admin/invoice/components/InvoiceImport"));

const User = React.lazy(() => import("views/admin/users"));
const UserView = React.lazy(() => import("views/admin/users/View"));

const Property = React.lazy(() => import("views/admin/property"));
const PropertyView = React.lazy(() => import("views/admin/property/View"));
const PropertyImport = React.lazy(() =>
import("views/admin/property/components/PropertyImport"))

const Lead = React.lazy(() => import("views/admin/lead"));
const LeadView = React.lazy(() => import("views/admin/lead/View"));
const LeadImport = React.lazy(() =>
import("views/admin/lead/components/LeadImport"));

const Communication = React.lazy(() => import("views/admin/communication"));

const Task = React.lazy(() => import("views/admin/task"));
const TaskView = React.lazy(() =>
import("views/admin/task/components/taskView"));
const Calender = React.lazy(() => import("views/admin/calender"));
const Role = React.lazy(() => import("views/admin/role"));

const Document = React.lazy(() => import("views/admin/document"));

const EmailHistory = React.lazy(() => import("views/admin/emailHistory"));
```

```
const EmailHistoryView = React.lazy(() =>
import("views/admin/emailHistory/View"));

const Meeting = React.lazy(() => import("views/admin/meeting"));
const MettingView = React.lazy(() => import("views/admin/meeting/View"));

const PhoneCall = React.lazy(() => import("views/admin/phoneCall"));
const PhoneCallView = React.lazy(() => import("views/admin/phoneCall/View"));

const Report = React.lazy(() => import("views/admin/reports"));
const EmailTemplate = React.lazy(() => import("views/admin/emailTemplate"));
const AddEdit = React.lazy(() => import("views/admin/emailTemplate/AddEdit"));
const templateView = React.lazy(() =>
import("views/admin/emailTemplate/view.js"));
const chatView = React.lazy(() => import("views/admin/chat/Chat.jsx"));
const notesView = React.lazy(() => import("views/admin/notes/Notes.jsx"));

// Auth Imports
const SignInCentered = React.lazy(() => import("views/auth/signIn"));
// admin setting
const AdminSetting = React.lazy(() => import("views/admin/adminSetting"));
const validation = React.lazy(() => import("views/admin/validation"));
const module = React.lazy(() => import("views/admin/moduleName"));
const Opportunities = React.lazy(() => import("views/admin/opportunities"));
const OpportunitiesView = React.lazy(() =>
import("views/admin/opportunities/View"));
const OpportunitiesImport = React.lazy(() =>
import("views/admin/opportunities/components/OpprtunityImport"));
const Account = React.lazy(() => import("views/admin/account"));
const AccountView = React.lazy(() => import("views/admin/account/View"));
const AccountImport = React.lazy(() =>
import("views/admin/account/components/AccountImport"));

const routes = [
  // ========================== Dashboard
==========================
  {
    name: "Dashboard",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/default",
    icon: <Icon as={MdHome} width='20px' height='20px' color='inherit' />,
```

```
    component: MainDashboard,
  },
  // ========================== Admin Layout
==========================
  // ------------- lead Routes -----------------------
  {
    name: "Leads",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/lead",
    icon: <Icon as={MdLeaderboard} width='20px' height='20px' color='inherit' />,
    component: Lead,
  },
  {
    name: "Leads",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    under: "lead",
    parentName: "Leads",
    path: "/leadView/:id",
    component: LeadView,
  },
  {
    name: "Lead Import",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    under: "lead",
    parentName: "Leads",
    path: "/leadImport",
    component: LeadImport,
  },
  // ---------------- contact Routes --------------------
  {
    name: "Contacts",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/contacts",
    icon: <Icon as={MdContacts} width='20px' height='20px' color='inherit' />,
    component: Contact,
  },
  {
    name: "Contacts",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    under: "contacts",
    parentName: "Contacts",
```

```
  path: "/contactView/:id",
  component: ContactView,
 },
 {
  name: "Contact Import",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  both: true,
  under: "contacts",
  parentName: "Contacts",
  path: "/contactImport",
  component: ContactImport,
 },
// ------------- Property Routes -----------------------
 {
  name: "Properties",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  path: "/properties",
  icon: <Icon as={LuBuilding2} width='20px' height='20px' color='inherit' />,
  component: Property,
 },
 {
  name: "Properties",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  parentName: "Properties",
  under: "properties",
  path: "/propertyView/:id",
  component: PropertyView,
 },
 {
  name: "Property Import",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  both: true,
  under: "properties",
  parentName: "Properties",
  path: "/propertyImport",
  component: PropertyImport,
 },

// ----------------------------Opportunities-----------------------------------
 {
  name: "Opportunities",
```

```
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/opportunities",
    icon: <Icon as={TbBulb} width='20px' height='20px' color='inherit' />,
    component: Opportunities,
   },
   {
    name: "Opportunities",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/opportunitiesView/:id",
    under: "opportunities",
    parentName: "Opportunities",
    icon: <Icon as={TbBulb} width='20px' height='20px' color='inherit' />,
    component: OpportunitiesView,
   },
   {
    name: "Opportunities",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/opprotunitiesImport",
    under: "opportunities",
    parentName: "Opportunities",
    icon: <Icon as={TbBulb} width='20px' height='20px' color='inherit' />,
    component: OpportunitiesImport,
   },
   // ----------------------------Account-----------------------------------
   {
    name: "Account",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/account",
    icon: <Icon as={RiAccountCircleFill} width='20px' height='20px' color='inherit'
/>,
    component: Account,
   },
   {
    name: "Account",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/accountView/:id",
    under: "account",
    parentName: "Account",
    icon: <Icon as={RiAccountCircleFill} width='20px' height='20px' color='inherit'
/>,
    component: AccountView,
```

```
  },
  {
    name: "Account",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/accountImport",
    under: "account",
    parentName: "Account",
    icon: <Icon as={RiAccountCircleFill} width='20px' height='20px' color='inherit'
/>,
    component: AccountImport,
  },
  // ---------- Quotes Routes ----------
  {
    name: "Quotes",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/quotes",
    icon: <Icon as={BsBlockquoteRight} width='20px' height='20px' color='inherit' />,
    component: Quotes,
  },
  {
    name: "Quotes",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    under: "quotes",
    parentName: "Quotes",
    path: "/quotesView/:id",
    component: QuotesView,
  },
  {
    name: "Quotes Import",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    both: true,
    under: "quotes",
    parentName: "Quotes",
    path: "/quotesImport",
    component: QuotesImport,
  },
  // ---------- Invoices Routes ----------
  {

    name: "Invoices",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
```

```
    path: "/invoices",
    icon: <Icon as={TbFileInvoice} width='20px' height='20px' color='inherit' />,
    component: Invoices,
  },
  {
    name: "Invoices",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    under: "invoices",
    parentName: "Invoices",
    path: "/invoicesView/:id",
    component: InvoicesView,
  },
  {
    name: "Invoices Import",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    both: true,
    under: "invoices",
    parentName: "Invoices",
    path: "/invoicesImport",
    component: InvoicesImport,
  },

  // // ------------- Communication Integration Routes -----------------------
  // {
  //   name: "Communication Integration",
  //   layout: [ROLE_PATH.admin, ROLE_PATH.user],

  //   path: "/communication-integration",
  //   icon: <Icon as={GiSatelliteCommunication} width='20px' height='20px'
color='inherit' />,
  //   component: Communication,
  // },
  // ------------- Task Routes -----------------------
  {
    name: "Tasks",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/task",
    icon: <Icon as={FaTasks} width='20px' height='20px' color='inherit' />,
    component: Task,
  },
  {
```

```
    name: "Tasks",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    under: "task",
    parentName: "Tasks",
    path: "/view/:id",
    component: TaskView,
  },
  // ------------- Meeting Routes ------------------------
  {
    name: "Meetings",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/metting",
    icon: <Icon as={SiGooglemeet} width='20px' height='20px' color='inherit' />,
    component: Meeting,
  },
  {
    name: "Meetings ",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    under: "Meetings",
    parentName: "Meetings",
    path: "/metting/:id",
    component: MettingView,
  },
  // ------------- Phone Routes ------------------------
  {
    name: "Calls",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/phone-call",
    icon: <Icon as={PiPhoneCallBold} width='20px' height='20px' color='inherit' />,
    component: PhoneCall,
  },
  {
    name: "Calls",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    under: "phone-call",
    parentName: "Calls",
    path: "/phone-call/:id",
    component: PhoneCallView,
  },
// ------------- Email Routes------------------------
  {
```

```
  // separator: 'History',
  name: "Emails",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  path: "/email",
  icon: <Icon as={AiOutlineMail} width='20px' height='20px' color='inherit' />,
  component: EmailHistory,
 },
 {
  name: "Emails ",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  under: "Emails",
  parentName: "Emails",
  path: "/Email/:id",
  component: EmailHistoryView,
 },
// ----------------------------Email Template-----------------------------------
 {
  name: "Email Template",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  path: "/email-template",
  icon: <Icon as={HiTemplate} width='20px' height='20px' color='inherit' />,
  component: EmailTemplate,
 },
 {
  name: "Add Email Template",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  under: "email-template",
  parentName: "Email Template",
  path: "/email-template/email-template-addEdit",
  icon: <Icon as={HiTemplate} width='20px' height='20px' color='inherit' />,
  component: AddEdit,
 },
 {
  name: "Email Template",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  under: "email-template",
  parentName: "Email Template",
  path: "/email-template/:id",
  icon: <Icon as={HiTemplate} width='20px' height='20px' color='inherit' />,
  component: templateView,
 },
```

```
  // ------------ Calender Routes -----------------------
  {
    name: "Calender",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/calender",
    icon: <Icon as={FaCalendarAlt} width='20px' height='20px' color='inherit' />,
    component: Calender,
  },

  // ----------------------------Admin setting-----------------------------------
  {
    name: "Admin Setting",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    // parentName: "admin",
    under: "admin",
    path: "/admin-setting",
    component: AdminSetting,
  },
  {
    name: "Roles",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/role",
    under: "role",
    icon: <Icon as={FaCreativeCommonsBy} width='20px' height='20px'
color='inherit' />,
    component: Role,
  },
  {
    name: "Custom Fields",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/custom-Fields",
    under: "customField",
    icon: <Icon as={FaWpforms} width='20px' height='20px' color='inherit' />,
    component: CustomField,
  },
  {
    name: "Change Images",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/change-images",
    under: "image",
    icon: <Icon as={TbExchange} width='20px' height='20px' color='inherit' />,
```

```
  component: ChangeImage,
},
{
  name: "Validation",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  path: "/validations",
  under: "Validation",
  icon: <Icon as={GrValidate} width='20px' height='20px' color='inherit' />,
  component: Validation,
},
{
  name: "Table Fields",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  path: "/table-field",
  under: "tableField",
  icon: <Icon as={TbTableColumn} width='20px' height='20px' color='inherit' />,
  component: TableField,
},
{
  name: "Active Deactive Module",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  path: "/active-deactive-module",
  under: "activeDeactiveModule",
  icon: <Icon as={TbTableColumn} width='20px' height='20px' color='inherit' />,
  component: activeDeactiveModule,
},
{
  name: "Module",
  layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
  path: "/module",
  under: "module",
  icon: <Icon as={VscFileSubmodule} width='20px' height='20px' color='inherit' />,
  component: module,
},
// // ------------- Text message Routes ----------------------
// {
//   name: "Text Msg",
//   layout: [ROLE_PATH.admin, ROLE_PATH.user],
//
//   path: "/text-msg",
```

```
  //   icon: <Icon as={MdOutlineMessage} width='20px' height='20px' color='inherit'
/>,
  //   component: TextMsg,
  // },
  // {
  //   name: "Text Msg View",
  //   layout: [ROLE_PATH.admin, ROLE_PATH.user],
  //
  //   under: "text-msg",
  //   path:  text-msg/:id",
  //   component: TextMsgView,
  // },
  // ------------- Document Routes ------------------------
  {
    name: "Documents",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/documents",
    icon: <Icon as={AiFillFolderOpen} width='20px' height='20px' color='inherit' />,
    component: Document,
  },
  // ----------------- Reporting Layout -----------------
  {
    name: "Reporting and Analytics",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/reporting-analytics",
    icon: <Icon as={MdInsertChartOutlined} width='20px' height='20px'
color='inherit' />,
    component: Report,
  },
  {
    name: "Chat",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/chat",
    icon: <Icon as={IoChatboxEllipsesOutline} width='20px' height='20px'
color='inherit' />,
    component: chatView,
  },
  {
    name: "Notes",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/notes",
```

```
    icon: <Icon as={CgNotes} width='20px' height='20px' color='inherit' />,
    component: notesView,
  },
  // ------------ user Routes ----------------------
  {
    name: "Users",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    path: "/user",
    under: "user",
    icon: <Icon as={HiUsers} width='20px' height='20px' color='inherit' />,
    component: User,
  },
  {
    name: "User View",
    layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
    parentName: "Email",
    under: "user",
    path: "/userView/:id",
    component: UserView,
  },
  // ========================== User layout
==========================

  // ========================== auth layout
==========================
  {
    name: "Sign In",
    layout: "/auth",
    path: "/sign-in",
    icon: <Icon as={MdLock} width='20px' height='20px' color='inherit' />,
    component: SignInCentered,
  },
];

export default routes;

import React, { useEffect, useState } from 'react';
import ReactDOM from 'react-dom';
import 'assets/css/App.css';
import { BrowserRouter as Router, Routes, Route, useNavigate } from 'react-router-
dom';
```

```
import AuthLayout from './layouts/auth';
import AdminLayout from 'layouts/admin';
import UserLayout from 'layouts/user';
import { ChakraProvider } from '@chakra-ui/react';
import theme from 'theme/theme';
import { ThemeEditorProvider } from '@hypertheme-editor/chakra-ui';
import { ToastContainer } from 'react-toastify';
import 'react-toastify/dist/ReactToastify.css';
import { Provider } from 'react-redux';
import { store, persistor } from './redux/store';
import { PersistGate } from 'redux-persist/integration/react';

function App() {
  const token = localStorage.getItem("token") || sessionStorage.getItem("token");

  const user = JSON.parse(localStorage.getItem("user"))
  useNavigate()

  return (
    <>
      <ToastContainer />
      <Routes>
        {token && user?.role ? (
          user?.role === 'user' ?
            <Route path="/*" element={<UserLayout />} />
            : user?.role === 'superAdmin' ?
              <Route path="/*" element={<AdminLayout />} />
              : "
        ) : (
          <Route path="/*" element={<AuthLayout />} />
        )}
      </Routes>
    </>
  );
}

ReactDOM.render(
  <Provider store={store}>
    <PersistGate loading={null} persistor={persistor}>
      <ChakraProvider theme={theme}>
        <React.StrictMode>
```

```
                <ThemeEditorProvider>
                  <Router>
                    <App />
                  </Router>
                </ThemeEditorProvider>
              </React.StrictMode>
            </ChakraProvider>
          </PersistGate>
        </Provider>
        , document.getElementById('root')
);

import axios from "axios";
import React, { useState } from "react";
import ReactDOM from "react-dom";

const Notes = () => {
  const [notes, setNotes] = useState([]);
  const [note, setNote] = useState("");

  const handleAddNote = () => {
    if (note.trim() !== "") {
      setNotes([...notes, note]);
      setNote("");
      axios.post('/notes', note)
    }
  };

  return (
    <div style={styles.container}>
      <h1 style={styles.header}>Notes</h1>
      <div style={styles.inputContainer}>
        <textarea
          style={styles.textarea}
          value={note}
          onChange={(e) => setNote(e.target.value)}
          placeholder="Enter your note here..."
        />
        <button style={styles.button} onClick={handleAddNote}>
          Add Note
        </button>
```

```
      </div>
      <div style={styles.notesContainer}>
        {notes.map((note, index) => (
          <div key={index} style={styles.note}>
            {note}
          </div>
        ))}
      </div>
    </div>
  );
};

const styles = {
  container: {
    fontFamily: "Arial, sans-serif",
    width: "300px",
    margin: "50px auto",
    textAlign: "center",
    border: "1px solid #ccc",
    borderRadius: "8px",
    padding: "20px",
    boxShadow: "0 0 10px rgba(0, 0, 0, 0.1)",
  },
  header: {
    fontSize: "24px",
    marginBottom: "20px",
    color: "#333",
  },
  inputContainer: {
    marginBottom: "20px",
  },
  textarea: {
    width: "100%",
    height: "60px",
    padding: "10px",
    fontSize: "14px",
    borderRadius: "4px",
    border: "1px solid #ccc",
    marginBottom: "10px",
    resize: "none",
  },
```

```
  button: {
    padding: "10px 20px",
    fontSize: "14px",
    color: "#fff",
    backgroundColor: "#007BFF",
    border: "none",
    borderRadius: "4px",
    cursor: "pointer",
  },
  notesContainer: {
    textAlign: "left",
  },
  note: {
    backgroundColor: "#f9f9f9",
    padding: "10px",
    marginBottom: "10px",
    borderRadius: "4px",
    border: "1px solid #ddd",
  },
};

export default Notes
import { useState } from "react"
import s from "./Chat.module.css"
import React from "react"
import { Link } from "react-router-dom"
import { useRef, useEffect } from "react";
import { mockedChats, mockedMessages, mockedUsers } from "./mock.chats.js"

const Chat = () => {
  const [chats, setChats] = useState([]);
  const [message, setMessage] = useState('');
  const [chatId, setChatId] = useState('');
  const [chatMessages, setChatMessages] = useState(mockedMessages)

  // const { userData } = useSelector(state => state.auth.data)

  const getChats = async () => {
    // const chatsObj = await axios.get('/chat')
     // setChats(chatsObj.data)
    setChats(mockedChats)
```

```
}
const subscribeGetMessages = async (id) => {

  try {
    console.log(id)
    setChatId(id)
    // const messages = await axios.get(`/message/${id}`);
    setChatMessages(mockedMessages)
    subscribeGetMessages(id)
  }
  catch (err) {
    setTimeout(() => {
      subscribeGetMessages(id)
    }, 500)
  }
}

React.useEffect(() => {
  getChats();

}, [])

const sendMessage = async () => {
  // await axios.post('/message', { message, chatId })
  setMessage(' ')
  setChatMessages([...chatMessages, {
    sender: mockedUsers.customer,
    message: message
  }])
}

const getMessages = async (id) => {
  // const messages = await axios.get(`/messages/${id}`);
  // setChatMessages(messages.data)
  setChatMessages(mockedMessages)
}
console.log(chatId)
console.log({ chats })
```

```
 return (
   <div className={s.chat__wrapper}>
    <ul className={s.sidebar} style={{ width: '100%', maxWidth: 360, bgcolor:
'background.paper' }}>
      {
       chats.map(chat => {
        let writer;
        if (1 === chat.customer._id) {
         writer = 'performer'
        } else {
         writer = 'customer'
        }
        return (
         <li alignItems="flex-start" style={{ display: "flex", gap: "10px", alignItems:
"center" }} key={chat.customer._id}>
          <div>
           <img alt={chat[writer].fullName} src={`${chat[writer].avatar}`}
width={"50px"} />
          </div>
          <div onClick={async () => {
           setChatId(chat._id)
           getMessages(chat._id)
           // subscribeGetMessages(chat._id)
          }} className={s.sidebar__chat}>
           {chat[writer].fullName}
          </div>
         </li>

        )
       }

      )

     }
    </ul>
    <div className={s.content__wrapper}>
     <div className={s.chat}>
      {
       !chatMessages
        ?
        <p>Select chat</p>
```

```
            :
            chatMessages.map(message => {
              let owner = false;
              if (message.sender?._id === 1) {
                owner = true;
              }
              return (
                // <div >
                <div className={owner ? s.owner__message : s.message}>

                  <div className={s.message__wrapper}>
                    <p className={s.owner}>{message.sender?.fullName}:</p>
                    <p className={s.message__text}>{message.message}</p>
                  </div>
                </div>
                // </div>
              )
            })
        }
      </div>
      <input placeholder="Enter message" className={s.input} value={message}
onChange={e => setMessage(e.target.value)} type="text" />
      <button className={s.button} onClick={() => sendMessage()}>Send
message</button>
    </div>
  </div >
 )
}

export default Chat

// Chakra imports
import { Portal, Box, useDisclosure, Text, Button, Link, Flex, Icon } from '@chakra-
ui/react';
import Footer from 'components/footer/FooterAdmin.js';
// Layout components
import Navbar from 'components/navbar/NavbarAdmin.js';
import Sidebar from 'components/sidebar/Sidebar.js';
import { SidebarContext } from 'contexts/SidebarContext';
import React, { Suspense, useEffect, useState } from 'react';
import { Navigate, Route, Routes } from 'react-router-dom';
```

```
import { ROLE_PATH } from '../../roles';
import newRoute from 'routes.js';
import { MdHome, MdLock } from 'react-icons/md';
import Spinner from 'components/spinner/Spinner';
import { useDispatch, useSelector } from 'react-redux';
import { fetchImage } from '../../redux/slices/imageSlice';
import { getApi } from 'services/api';
import DynamicPage from 'views/admin/dynamicPage';
import { LuChevronRightCircle } from 'react-icons/lu';
import { FaCalendarAlt } from 'react-icons/fa';
import { fetchModules } from '../../redux/slices/moduleSlice';

const MainDashboard = React.lazy(() => import("views/admin/default"));
const SignInCentered = React.lazy(() => import("views/auth/signIn"));
const Calender = React.lazy(() => import("views/admin/calender"));
const UserView = React.lazy(() => import("views/admin/users/View"));

// Custom Chakra theme
export default function User(props) {
   const { ...rest } = props;
   // states and functions
   const [fixed] = useState(false);
   const [toggleSidebar, setToggleSidebar] = useState(false);
   const [route, setRoute] = useState();
   const [openSidebar, setOpenSidebar] = useState(true)
   const user = JSON.parse(localStorage.getItem("user"))
   const modules = useSelector((state) => state?.modules?.data)
   // functions for changing the states from components
   const getRoute = () => {
      return window.location.pathname !== '/admin/full-screen-maps';
   };

   const fetchRoute = async () => {
      let response = await getApi("api/route/");
      setRoute(response?.data);
   };

   const pathName = (name) => {
      return `/${name.toLowerCase().replace(/ /g, '-')}`;
   }
```

```
useEffect(() => {
   fetchRoute();
}, []);

const layoutName = user?.roles?.map(item => `/${item.roleName}`)

const filterAccess = (rolesData) => {
   return rolesData?.map(role => {
      role.access = role?.access?.filter(access => access.view);
      return role;
   });
};

// Example usage:
const updatedRolesData = filterAccess(user?.roles);
let access = []
updatedRolesData?.map((item) => {
   item?.access?.map((data) => access.push(data))
})

let mergedPermissions = {};

access.forEach((permission) => {
   const { title, ...rest } = permission;

   if (!mergedPermissions[title]) {
      mergedPermissions[title] = { ...rest };
   } else {
      // Merge with priority to true values
      Object.keys(rest).forEach((key) => {
         if (mergedPermissions[title][key] !== true) {
            mergedPermissions[title][key] = rest[key];
         }
      });
   }
});

let routes =
   [
      {
         name: "Dashboard",
```

```
            layout: [ROLE_PATH.user],
            path: "/default",
            icon: <Icon as={MdHome} width='20px' height='20px' color='inherit' />,
            component: MainDashboard,
          }, {
            name: "Sign In",
            layout: "/auth",
            path: "/sign-in",
            icon: <Icon as={MdLock} width='20px' height='20px' color='inherit' />,
            component: SignInCentered,
          },
          {
            name: "Calender",
            layout: [ROLE_PATH.user],
            path: "/calender",
            icon: <Icon as={FaCalendarAlt} width='20px' height='20px' color='inherit'
/>,
            component: Calender,
          },
          {
            name: "User View",
            layout: [ROLE_PATH.superAdmin, ROLE_PATH.user],
            parentName: "Email",
            under: "user",
            path: "/userView/:id",
            component: UserView,
          },
        ]

    route?.map((item, i) => {
      if (!newRoute.some(route => route.name === item.moduleName)) {
        return (
          newRoute.push({
            name: item?.moduleName,
            layout: [ROLE_PATH.user],
            path: pathName(item.moduleName),
            icon: <Icon as={LuChevronRightCircle} width='20px' height='20px'
color='inherit' />,
            component: DynamicPage,
          })
        )
```

```
        }
    })
    const accessRoute = newRoute?.filter(item =>
Object.keys(mergedPermissions)?.find(data => (data?.toLowerCase() ===
item?.name?.toLowerCase()) || (data?.toLowerCase() ===
item.parentName?.toLowerCase())))

    // routes.push(...accessRoute)
    let filterData = [...accessRoute]

    const activeModel = modules?.filter(module => module?.isActive)?.map(module
=> module?.moduleName);

    const activeRoutes = filterData?.filter(
        (data) =>
            activeModel?.includes(data?.name) ||
            activeModel?.includes(data?.parentName) ||
            !modules?.some(
                (module) =>
                    module?.moduleName === data?.name ||
                    module?.moduleName === data?.parentName
            )
    );
    routes.push(...activeRoutes)

    const getActiveRoute = (routes) => {
        let activeRoute = 'Eldisto';
        for (let i = 0; i < routes.length; i++) {
            if (routes[i].collapse) {
                let collapseActiveRoute = getActiveRoute(routes[i].items);
                if (collapseActiveRoute !== activeRoute) {
                    return collapseActiveRoute;
                }
            } else if (routes[i].category) {
                let categoryActiveRoute = getActiveRoute(routes[i].items);
                if (categoryActiveRoute !== activeRoute) {
                    return categoryActiveRoute;
                }
            } else {
                if (window.location.href.indexOf(routes[i].path.replace("/:id", "")) !== -1)
{
```

```
                return routes[i].name;
            }
        }
    }
    return activeRoute;
};
const under = (routes) => {
    let activeRoute = false
    for (let i = 0; i < routes?.length; i++) {
        if (routes[i]?.collapse) {
            let collapseActiveRoute = getActiveRoute(routes[i]?.items);
            if (collapseActiveRoute !== activeRoute) {
                return collapseActiveRoute;
            }
        } else if (routes[i]?.category) {
            let categoryActiveRoute = getActiveRoute(routes[i]?.items);
            if (categoryActiveRoute !== activeRoute) {
                return categoryActiveRoute;
            }
        } else {
            if (window.location.href?.indexOf(routes[i]?.path?.replace("/:id", "")) !== -
1) {
                return routes[i];
            }
        }
    }
    return activeRoute;
};

const getActiveNavbar = (routes) => {
    let activeNavbar = false;
    for (let i = 0; i < routes.length; i++) {
        if (routes[i].collapse) {
            let collapseActiveNavbar = getActiveNavbar(routes[i].items);
            if (collapseActiveNavbar !== activeNavbar) {
                return collapseActiveNavbar;
            }
        } else if (routes[i].category) {
            let categoryActiveNavbar = getActiveNavbar(routes[i].items);
            if (categoryActiveNavbar !== activeNavbar) {
                return categoryActiveNavbar;
```

```
            }
          } else {
            if (window.location.href.indexOf(routes[i].path) !== -1) {
              return routes[i].secondary;
            }
          }
        }
        return activeNavbar;
      };
      const getActiveNavbarText = (routes) => {
        let activeNavbar = false;
        for (let i = 0; i < routes.length; i++) {
          if (routes[i].collapse) {
            let collapseActiveNavbar = getActiveNavbarText(routes[i].items);
            if (collapseActiveNavbar !== activeNavbar) {
              return collapseActiveNavbar;
            }
          } else if (routes[i].category) {
            let categoryActiveNavbar = getActiveNavbarText(routes[i].items);
            if (categoryActiveNavbar !== activeNavbar) {
              return categoryActiveNavbar;
            }
          } else {
            if (window.location.href.indexOf(routes[i].path) !== -1) {
              return routes[i].messageNavbar;
            }
          }
        }
        return activeNavbar;
      };

      const getRoutes = (routes) => {
        return routes?.map((prop, key) => {
          // if (!prop.under && prop.layout === '/admin') {
          if (!prop?.under && prop?.layout !== '/auth') {
            return <Route path={prop?.path} element={prop && <prop.component />}
key={key} />;
          } else if (prop?.under) {
            return <Route path={prop?.path} element={prop && <prop.component />}
key={key} />
          }
```

```
            if (prop?.collapse) {
               return getRoutes(prop?.items);
            }
            if (prop?.category) {
               return getRoutes(prop?.items);
            } else {
               return null;
            }
         });
      };
      document.documentElement.dir = 'ltr';
      const { onOpen } = useDisclosure();
      document.documentElement.dir = 'ltr';

      const dispatch = useDispatch();

      useEffect(() => {
         // Dispatch the fetchRoles action on component mount
         dispatch(fetchImage());
         dispatch(fetchModules())

      }, [dispatch]);

      const largeLogo = useSelector((state) => state?.images?.images?.filter(item =>
item?.isActive === true));

      return (
         <Box>
            <Box>
               <SidebarContext.Provider
                  value={{
                     toggleSidebar,
                     setToggleSidebar
                  }}>
                  <Sidebar routes={routes} display='none' {...rest}
openSidebar={openSidebar} setOpenSidebar={setOpenSidebar}
largeLogo={largeLogo} />
                  <Box
                     float='right'
                     minHeight='100vh'
                     height='100%'
```

```
                overflow='auto'
                position='relative'
                maxHeight='100%'
                // w={{ base: '100%', xl: 'calc( 100% - 290px )' }}
                w={{ base: '100%', xl: openSidebar === true ? 'calc( 100% - 300px )' :
'calc( 100% - 88px )' }}
                maxWidth={{ base: '100%', xl: openSidebar === true ? 'calc( 100% -
300px )' : 'calc( 100% - 88px )' }}
                transition='all 0.33s cubic-bezier(0.685, 0.0473, 0.346, 1)'
                transitionDuration='.2s, .2s, .35s'
                transitionProperty='top, bottom, width'
                transitionTimingFunction='linear, linear, ease'>
                <Portal>
                  <Box className='header'>
                    <Navbar
                      onOpen={onOpen}
                      logoText={'Horizon UI Dashboard PRO'}
                      brandText={getActiveRoute(routes)}
                      secondary={getActiveNavbar(routes)}
                      message={getActiveNavbarText(routes)}
                      fixed={fixed}
                      routes={routes}
                      under={under(routes)}
                      largeLogo={largeLogo}
                      openSidebar={openSidebar}
setOpenSidebar={setOpenSidebar}
                      {...rest}
                    />
                  </Box>
                </Portal>
                <Box pt={{ base: "150px", md: "95px", xl: "95px" }}>
                  {getRoute() ? (
                    <Box mx='auto' pe='20px' minH='84vh' pt='50px' style={{
padding: openSidebar ? '8px 20px 8px 20px' : '8px 20px' }}>
                      <Suspense fallback={
                        <Flex justifyContent={'center'} alignItems={'center'}
width="100%" >
                          <Spinner />
                        </Flex>
                      }>
                        <Routes>
```

```
                                {getRoutes(routes)}
                                <Route path="/*" element={<Navigate to="/default" />}
/>
                            </Routes>
                        </Suspense>
                    </Box>
                ) : null}
            </Box>
            <Box>
                <Footer />
            </Box>
        </Box>
    </SidebarContext.Provider>
</Box>
</Box>
);
}
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ УПРАВЛІННЯ ВІДНОСИНАМИ З КЛІЄНТАМИ»

Виконав: студент 2-го курсу, групи 2КН-23м
спеціальності 122 – «Комп'ютерні науки»

Шаргало Д. В.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН

Крилик Л. В.
(прізвище та ініціали)

«05» 12 2024 р.

Вінниця ВНТУ - 2024 рік

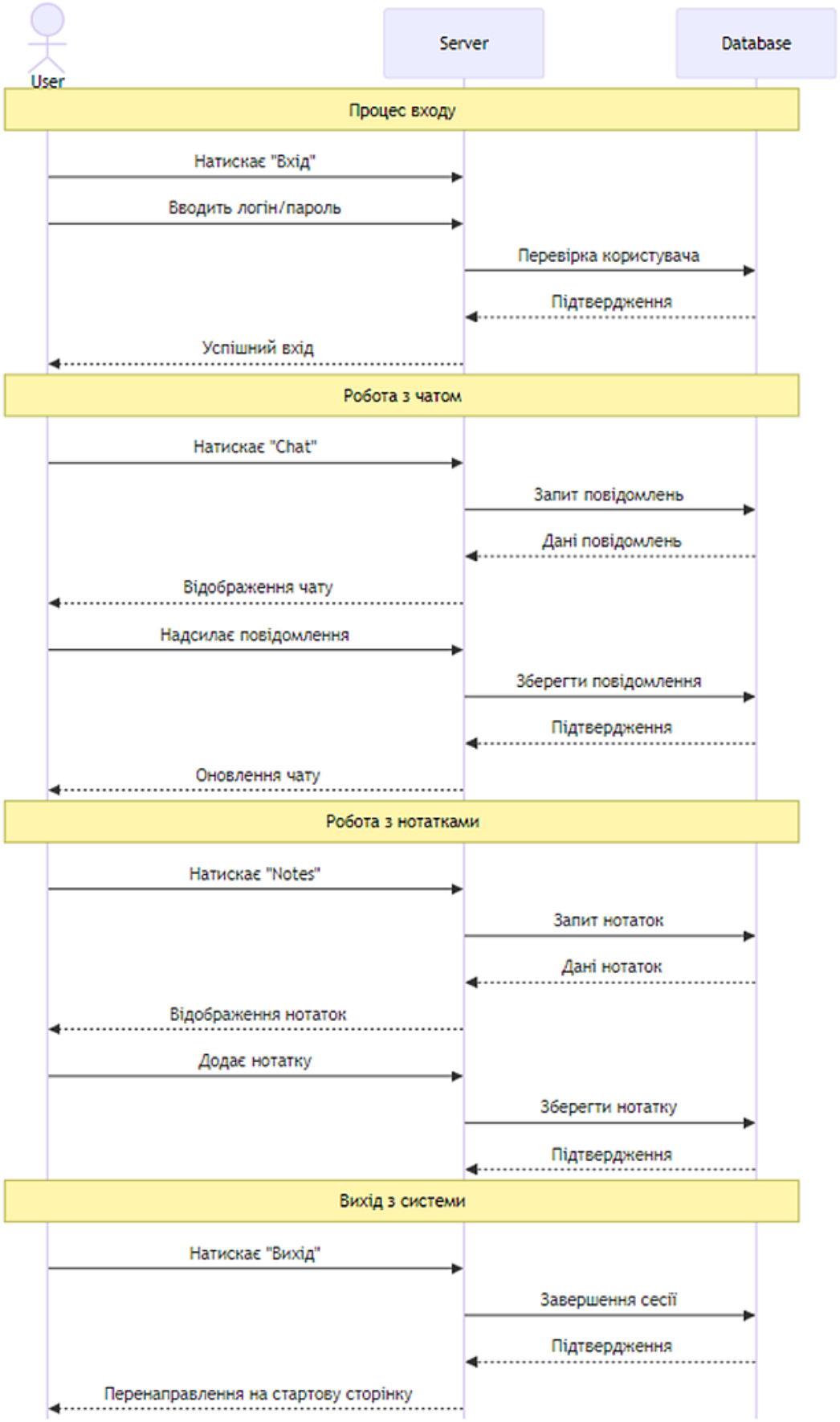

Рисунок В.1 – UML-діаграма послідовності взаємодії модулів інформаційної технології управління відносинами з клієнтами

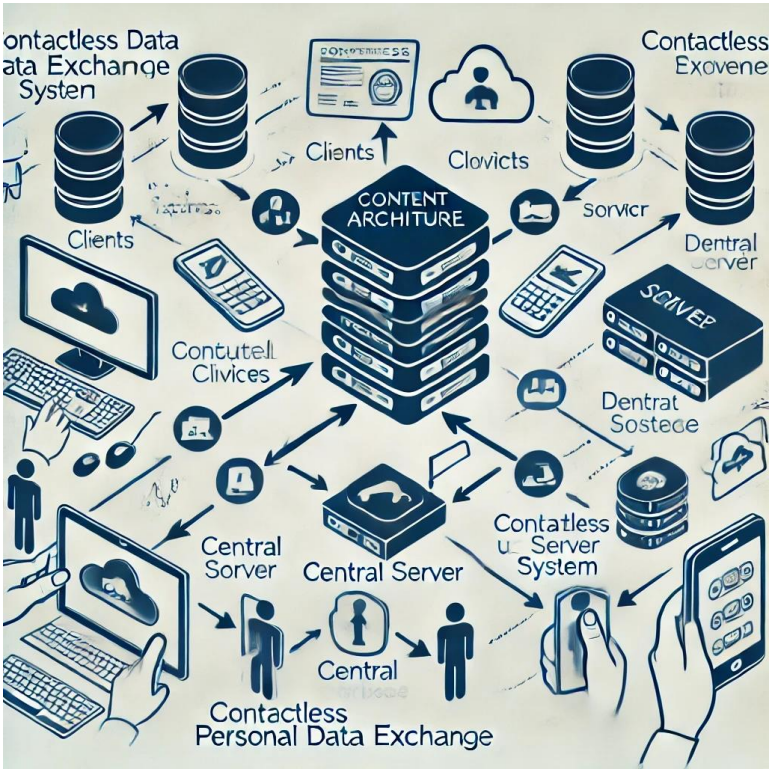

Рисунок В.2 – Схема системи клієнт-сервер

Рисунок В.3 – Загальна структурна модель програмного модуля

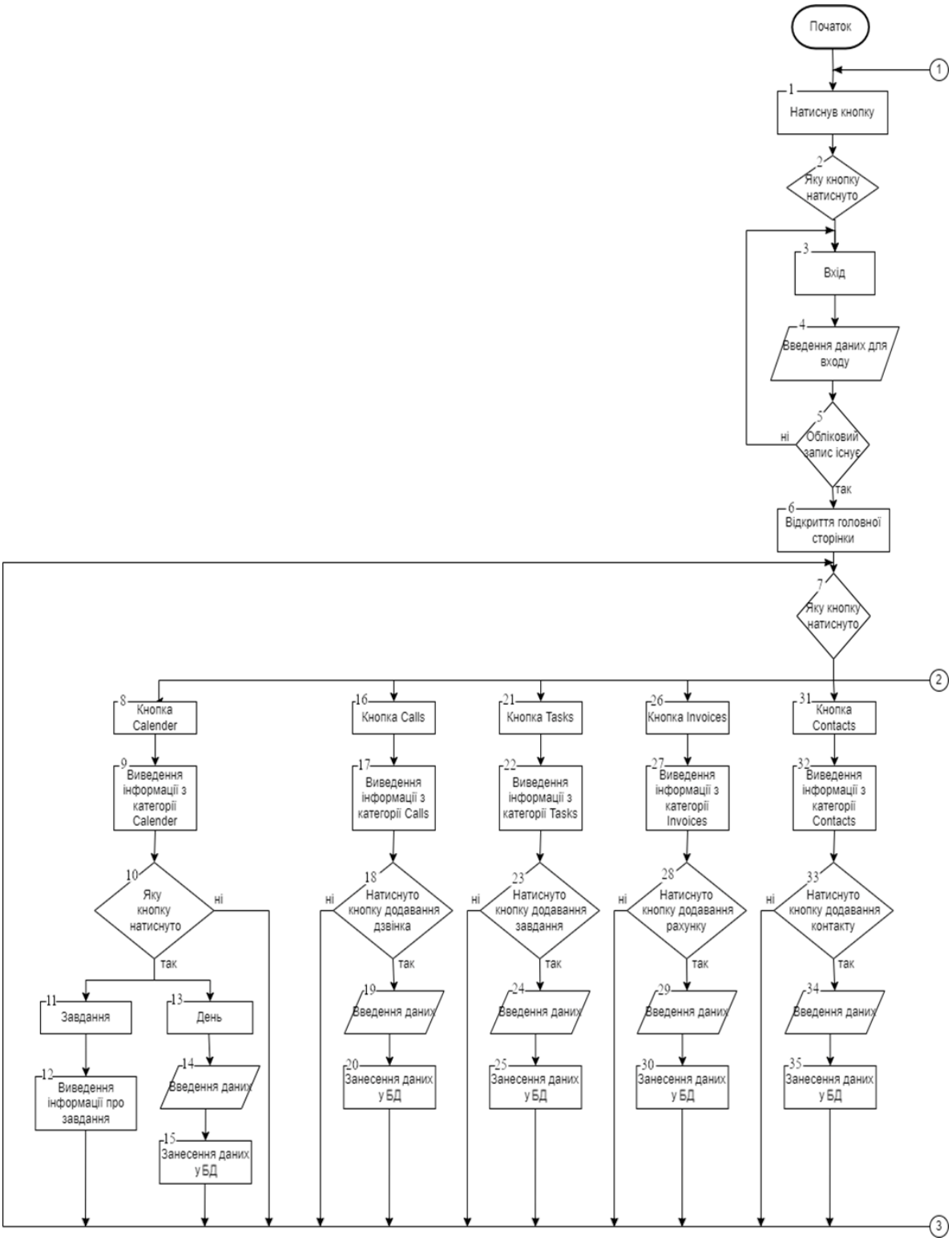

Рисунок В.4 – Схема алгоритму роботи інформаційної технології управління відносинами з клієнтами

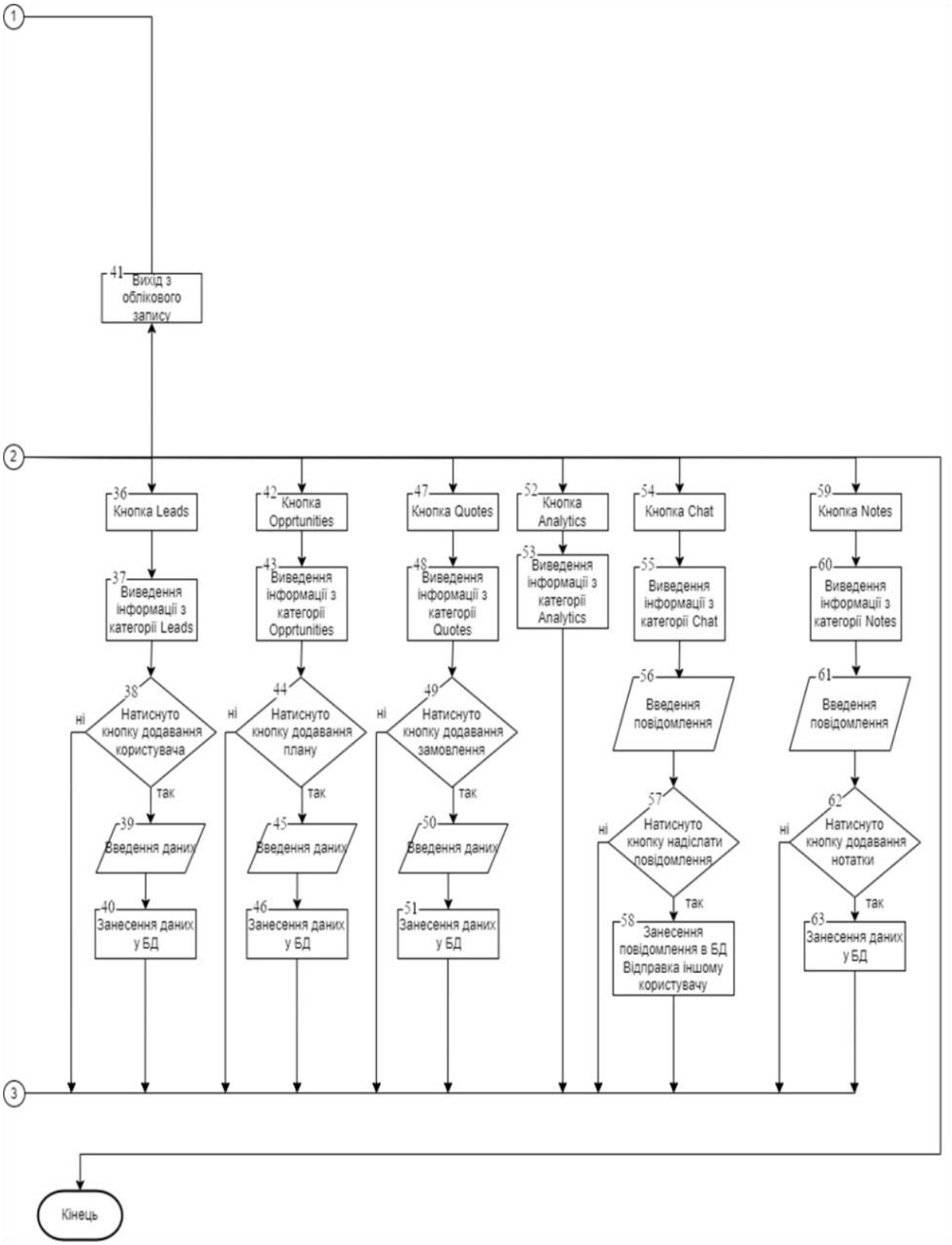

Рисунок В.4, аркуш 2

Рисунок В.5 – Діаграма діяльності серверу під час процесу автентифікації користувача

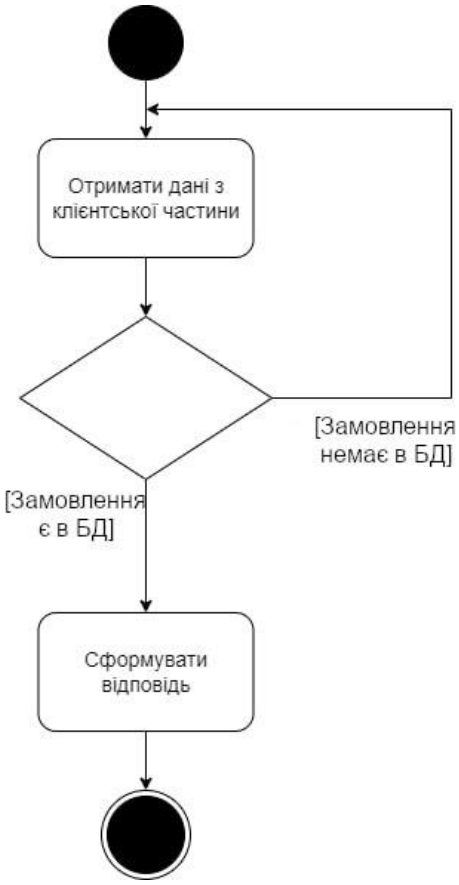

Рисунок В.6 – Діаграма діяльності серверу під час процесу перегляду замовлення користувача

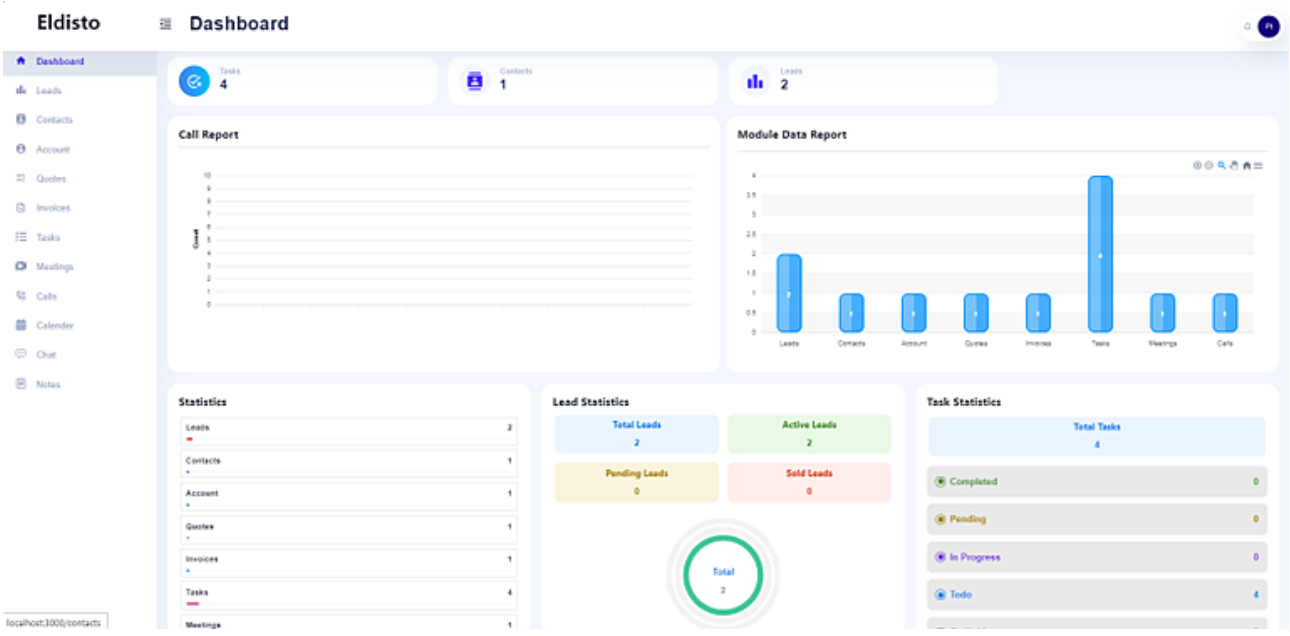

Рисунок В.7 – Загальний вигляд інтерфейсного вікна роботи програми

Додаток Г (довідниковий)

Інструкція користувача

Для того, щоб почати роботу з інформаційною технологією, потрібно мати персональний комп'ютер. Головною умовою є стабільне підключення до мобільної мережі інтернету або Wifi (рис. Г.1).

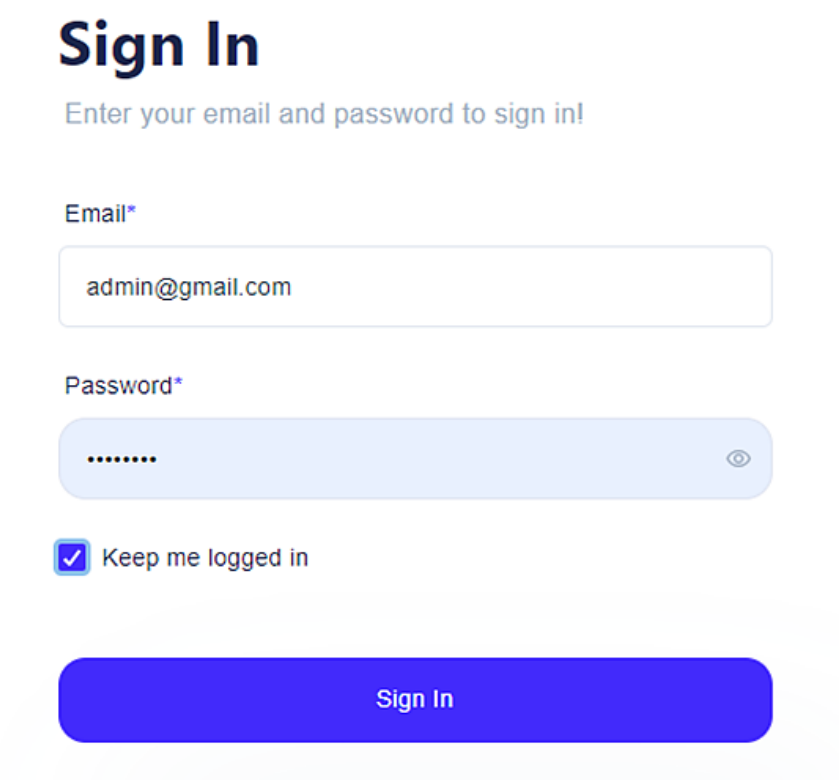

Рисунок Г.1 – Загальний вигляд інтерфейсного вікна авторизації користувача

Після успішної авторизації користувач переадресовується на вікно головного меню програми. Це вікно містить всю необхідну користувачу інформацію. Наприклад у цьому вікні відображається статистика клієнтів та інформація про них, деяка інформація про активність менеджерів, а саме: відображає працює даний менеджер чи ні. Також у цьому вікні знаходиться коротка таблиця історії останніх замовлень користувача. Ця таблиця містить колонки: Completed, Pending, In Progress, Todo та On Hold. Також можна побачити на графіку звіт про дані модуля в системі (рис. Г.2 – Г.8).

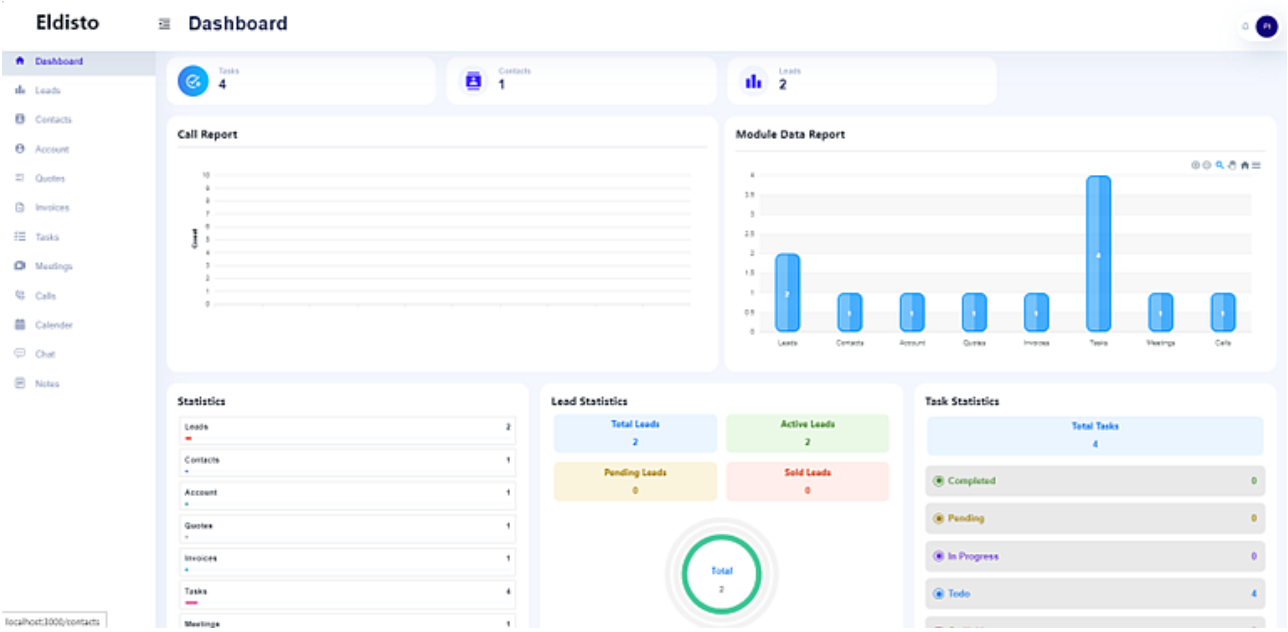

Рисунок Г.2 – Загальний вигляд інтерфейсного вікна головного меню з графіком

Рисунок Г.3 – Загальний вигляд інтерфейсного вікна «Leads» з інформацією та кнопкою активності

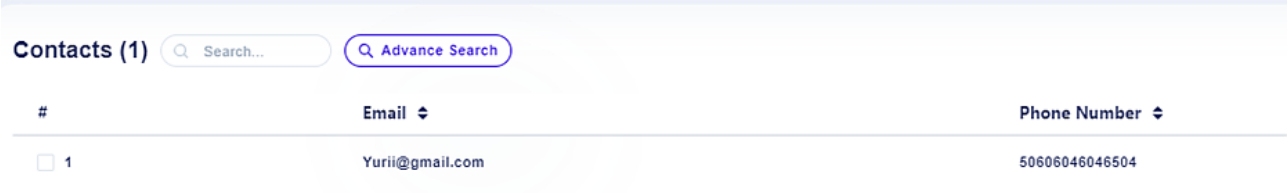

Рисунок Г.4 – Загальний вигляд інтерфейсного вікна «Contacts» з інформацією про користувача

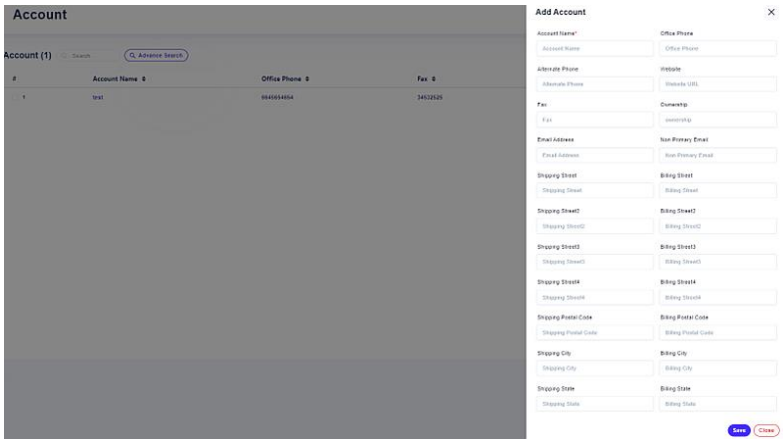

Рисунок Г.5 – Загальний вигляд інтерфейсного вікна «Account»

Рисунок Г.6 – Загальний вигляд інтерфейсного вікна «Quotes»

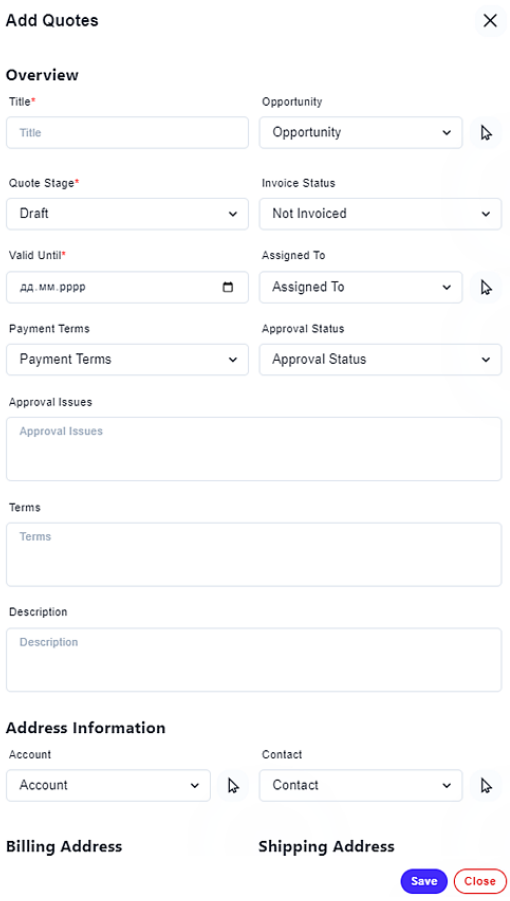

Рисунок Г.7 – Загальний вигляд інтерфейсного вікна додавання інформації про замовлення

Рисунок Г.8 – Загальний вигляд інтерфейсного вікна інформації про замовлення

На рисунку Г.9 та рисунку Г.10 подано «Tasks», де відображається інформація завдань для менеджерів, а також кнопка додавання нового завдання.

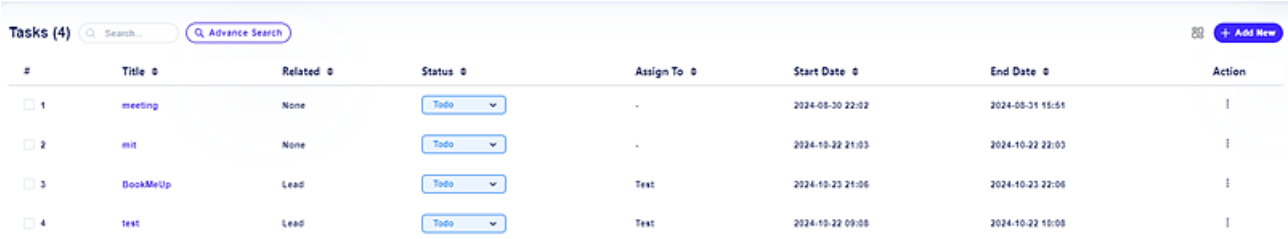

Рисунок Г.9 – Загальний вигляд інтерфейсного вікна інформації про завдання

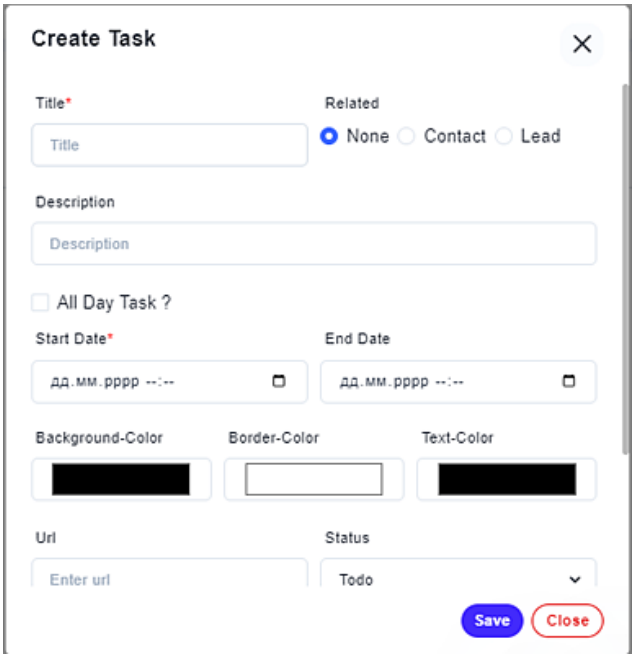

Рисунок Г.10 – Загальний вигляд інтерфейсного вікна додавання нового завдання

На рисунку Г.11 подано «Meetings», де відображається інформація про заплановані мітинги, на яких буде оговорюватись загальна інформація про

продажі та підвищення продуктивності менеджерів, також на цій сторінці можна додати новий мітинг, що і зображено на рисунку Г.12.

Рисунок Г.11 – Загальний вигляд інтерфейсного вікна «Meeting»

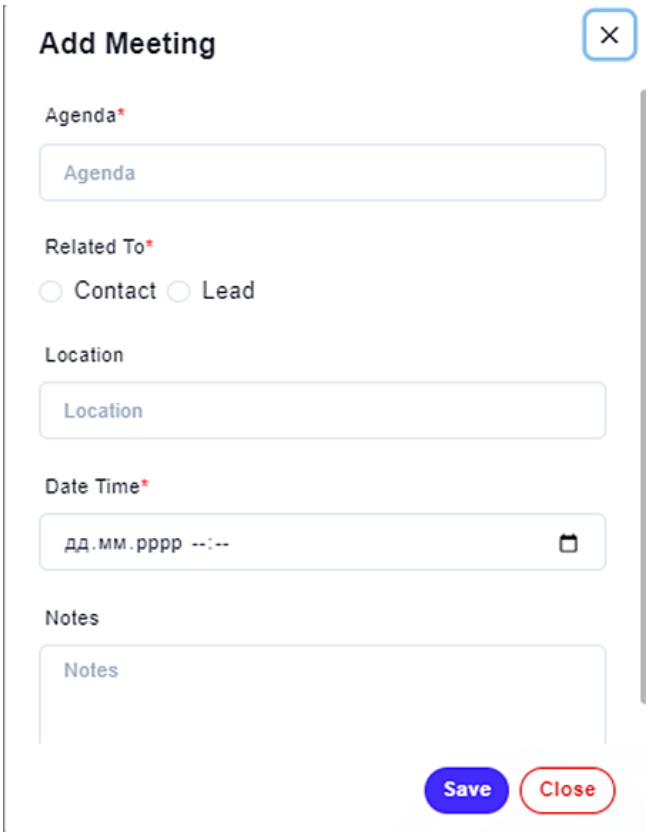

Рисунок Г.12 – Загальний вигляд інтерфейсного вікна додавання «Meeting»

На рисунку Г.13 подано «Calls», де відображається інформація про дзвінки до клієнтів або від клієнтів для узгодження замовлення.

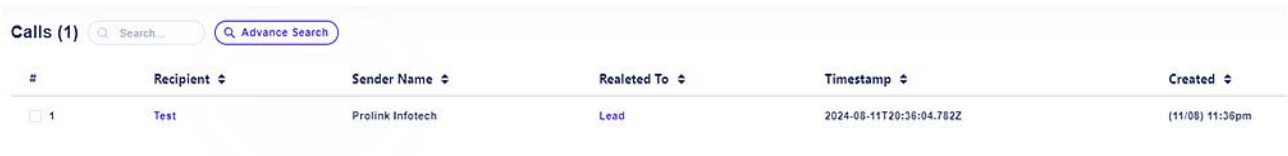

Рисунок Г.13 – Загальний вигляд інтерфейсного вікна Calls

На рисунку Г.14 подано «Calender», де відображається інформація про завдання для менеджерів.

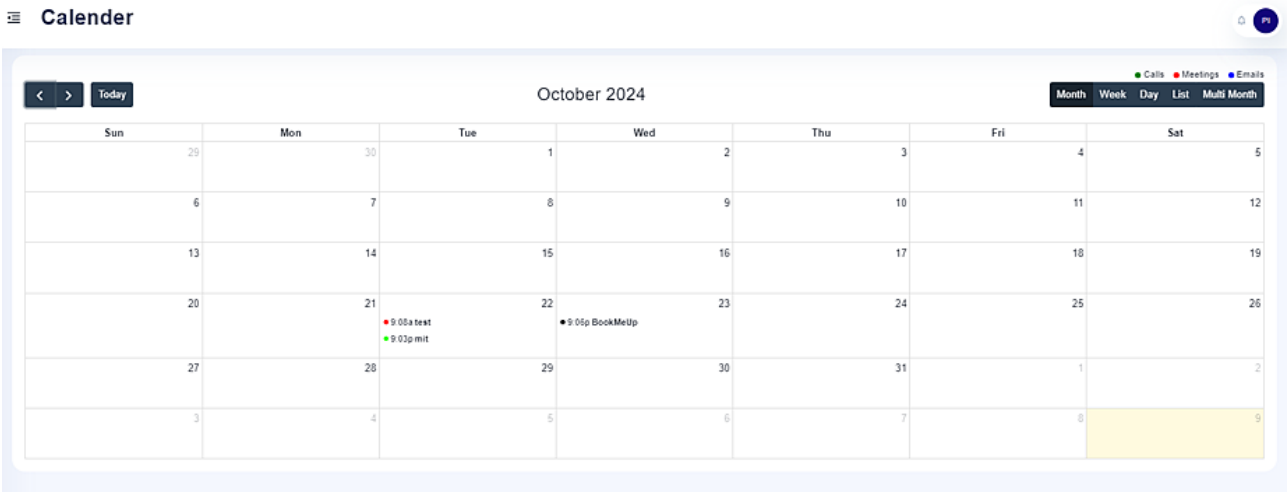

Рисунок Г.14 – Загальний вигляд інтерфейсного вікна «Calender»

На рисунку Г.15 подано «Chat», де відображається чат з клієнтом.

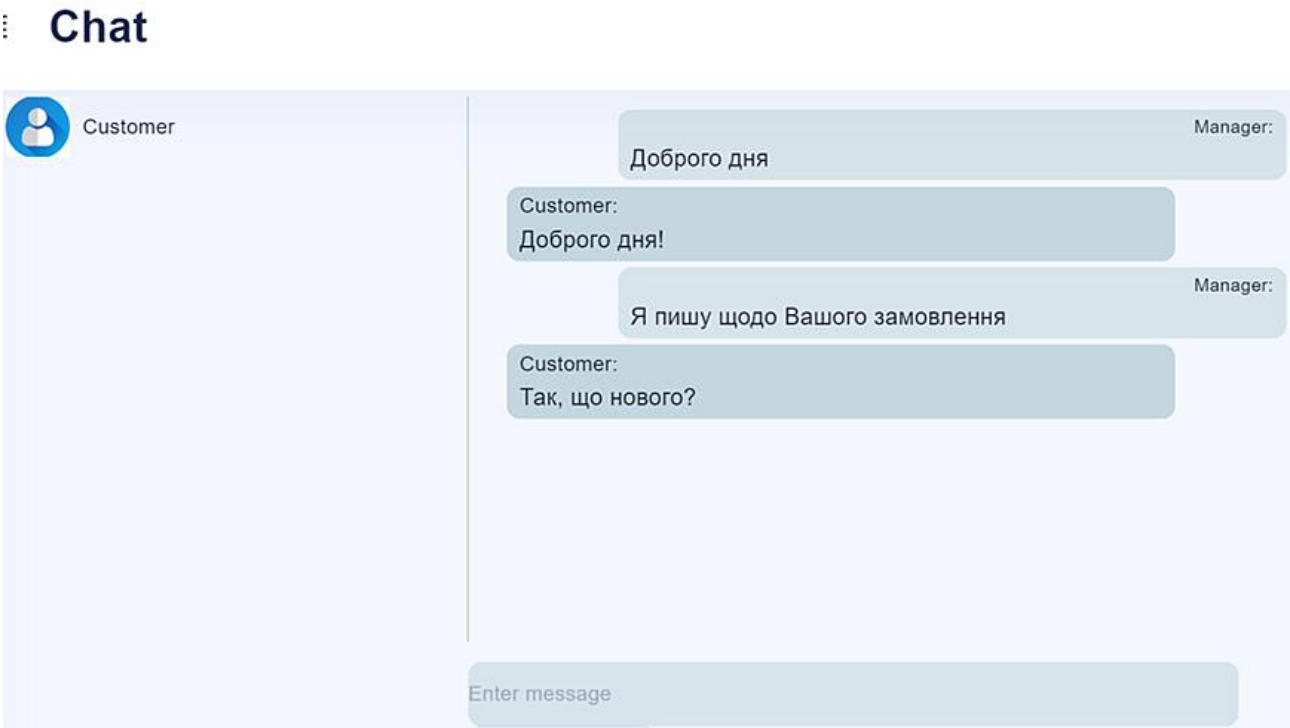

Рисунок Г.15 – Загальний вигляд інтерфейсного вікна «Chat»

На рисунку Г.16 подано «Notes», де відображаються нотатки менеджера.

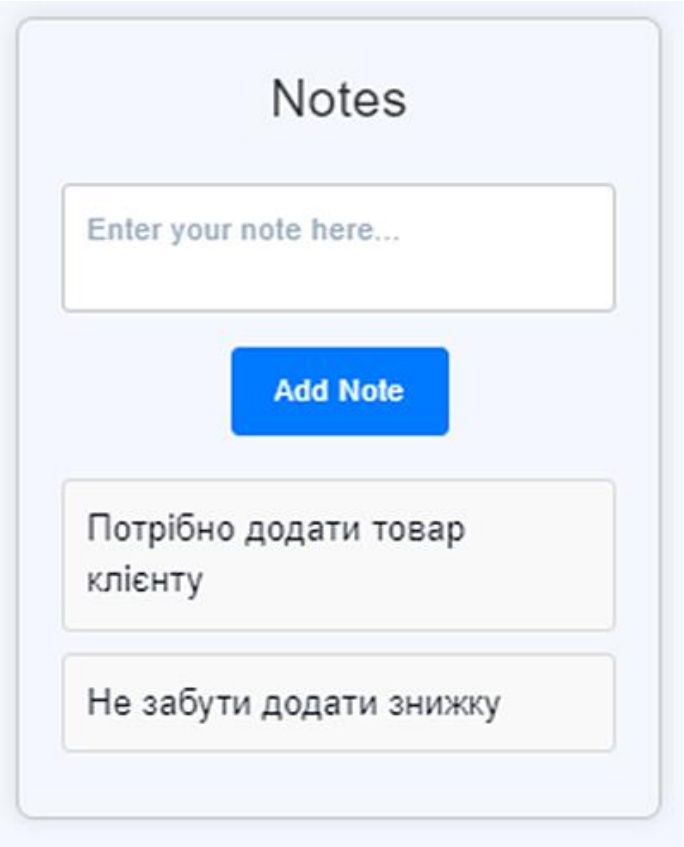

Рисунок Г.16 – Загальний вигляд інтерфейсного вікна «Notes»