НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК _004.8___

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою
«Інформаційні управляючі системи та технології»

зі спеціальності 126 «Інформаційні системи та технології»

на тему: «Інформаційна система служби підтримки страхової компанії на основі великих мовних моделей»

Виконав:
студент 2 курсу, групи ІС-33мп
Поночовний Павло Сергійович _____

Керівник:
доцент кафедри ІСТ, к.т.н., доц.
Олійник Володимир Валентинович _____

Рецензент:
доцент кафедри ІПІ, к.т.н, доц.
Лісовиченко Олег Іванович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Поночовному Павлу Сергійовичу

1. Тема дисертації «Інформаційна система служби підтримки страхової компанії на основі великих мовних моделей», науковий керівник дисертації Олійник Володимир Валентинович, доцент кафедри ІСТ, к.т.н., доц., затверджені наказом по університету від «08» 11 2024 р. № 5016-с

2. Термін подання студентом дисертації «09» 12 2024 р.

3. Об’єкт дослідження: інформаційна система чат-бота (служби) підтримки клієнтів страхової компанії, яка автоматизує процес надання інформації про страхові договори, використовуючи технології обробки природної мови та векторні бази даних.

4. Вихідні дані: швидкий та безпечний доступ до інформації про страхові договори, обробка запитів користувачів із дотриманням конфіденційності та коректним аналізом текстових документів.

5. Перелік завдань, які потрібно розробити: аналіз предметної області та існуючих рішень, визначення вимог та постановка цілей проєкту, дослідження доступних технологій та вибір оптимальних рішень, проєктування архітектури інформаційної системи чат-боту, розробка та навчання великої мовної моделі, створення інтерфейсу користувача для інтеграції з мовною моделлю, тестування системи, оформлення

документації, написання пояснювальної записки та підготовка до захисту проєкту.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: структурна схема архітектури, міжрівнева схема архітектури, діаграма прецедентів, діаграма послідовності, діаграма станів, діаграма потоків даних, блок-схеми основних алгоритмів розроблюваної системи.

7. Орієнтовний перелік публікацій:

Поночовний П.С. Аналітичний огляд способів застосування великих мовних моделей (LLM) для вирішення прикладних задач / Поночовний П.С., Олійник В.В.// Інженерія програмного забезпечення і передові інформаційні технології (Soft Tech-2023): матеріали V Міжнародної науково-практичної конференції молодих вчених та студентів, 19-21 грудня 2023 року, м. Київ, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 2023. С. 272-276.

9. Дата видачі завдання 02.09.2024 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Аналіз предметної області та існуючих рішень	08.09.2024	
2.	Визначення вимог та постановка цілей проєкту	15.09.2024	
3.	Дослідження доступних технологій та вибір оптимальних рішень	22.09.2024	
4.	Проєктування архітектури інформаційної системи чат-боту	29.09.2024	
5.	Розробка та навчання великої мовної моделі	06.10.2024	
6.	Створення інтерфейсу користувача для інтеграції з мовною моделлю	20.10.2024	
7.	Тестування системи та її оптимізація для забезпечення стабільної роботи	03.11.2024	
8.	Оформлення документації, написання пояснювальної записки та підготовка до захисту проєкту	02.12.2024	

Студент Павло ПОНОЧОВНИЙ

Науковий керівник Володимир ОЛІЙНИК

РЕФЕРАТ

Інформаційна система служби підтримки страхової компанії на основі великих мовних моделей: 121 с., 29 табл., 45 рис., 9 дод., 25 джерел.

ЧАТ-БОТ, СЛУЖБА ПІДТРИМКИ, СТРАХОВА КОМПАНІЯ, ВЕЛИКА МОВНА МОДЕЛЬ, ГЕНЕРАЦІЯ З ДОПОВНЕННЯМ ПОШУКУ, ВЕКТОРНА БАЗА, ОЦІНКА ВІДПОВІДІ.

Сучасні страхові компанії стикаються з великою кількістю запитів щодо умов договорів і правил страхування, що перевантажує персонал та уповільнює відповіді. Автоматизація процесу обслуговування за допомогою чат-ботів на основі великих мовних моделей (LLM) та системи генерації з доповненим пошуком (RAG) дозволяє забезпечити точні й персоналізовані відповіді, значно покращуючи якість клієнтського обслуговування.

Метою дослідження є підвищення ефективності роботи служби підтримки страхових компаній за рахунок розробки інформаційної системи чат-бота на основі LLM, який забезпечить швидкий та зручний доступ до інформації про страхові договори.

Об’єктом дослідження є інформаційна система служби підтримки клієнтів страхової компанії, яка автоматизує процеси обслуговування запитів клієнтів.

Предметом дослідження є властивості інтеграції LLM і системи RAG для автоматизації обробки клієнтських запитів у страхуванні.

Основний підхід дослідження – RAG для інтеграції мовних моделей.

Наукова новизна роботи полягає в розробці інтегрованої системи, яка поєднує LLM і RAG для автоматизації обробки складних юридичних текстів, що значно перевищує можливості традиційних підходів. Запропоноване рішення забезпечує персоналізовану взаємодію з користувачами та може бути адаптоване до інших галузей. Чат-бот автоматизує обслуговування клієнтів, знижує навантаження на персонал і може бути адаптований до інших галузей, де потрібна обробка складних текстових документів.

ABSTRACT

Insurance company support information system based on large language models: 121 p., 29 tab., 45 draw., 9 app., 25 sources.

CHATBOT, CUSTOMER SERVICE, INSURANCE COMPANY, LARGE LANGUAGE MODEL, RETRIEVAL-AUGMENTED GENERATION, VECTOR BASE, RESPONSE EVALUATION.

Modern insurance companies face a large number of inquiries about the terms of contracts and insurance rules, which overloads staff and slows down responses. Automation of the service process with chatbots based on large language models (LLMs) and retrieval-augmented generation (RAG) system allows for accurate and personalised responses, significantly improving the quality of customer service.

The purpose of the study is to improve the efficiency of the insurance company support service by developing an information system of a chatbot based on LLM, which will provide quick and convenient access to information about insurance contracts.

The object of the study is an insurance company customer support information system that automates the processes of servicing customer requests.

The subject of the study is the properties of integration of LLM and RAG system to automate the processing of customer requests in insurance.

The main research approach is RAG for the integration of language models.

The scientific novelty of the work lies in the development of an integrated system that combines LLM and RAG to automate the processing of complex legal texts, which significantly exceeds the capabilities of traditional approaches. The proposed solution provides a personalised user experience and can be adapted to other industries. The chatbot automates customer service, reduces staff workload, and can be adapted to other industries that require processing complex text documents.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

CPU – Central Processing Unit – центральний процесор.

FAQ – Frequently Asked Question – часто поставлені, поширені питання.

FAISS – Facebook AI Similarity Search – векторне сховище від Facebook.

GDPR – General Data Protection Regulation – це загальний регламент захисту даних.

GPT – Generative Pre-trained Transformer – це велика мовна модель, розроблена компанією OpenAI.

GPU – Graphics Processing Unit – графічний процесор.

HNSW – Hierarchical navigable small world – це метод пошуку наближених найближчих сусідів на основі графів, який використовується в багатьох векторних базах даних.

LLM – Large Language Model – велика мовна модель.

Q&A – Questions and Answers – питання та відповіді.

RAG – Retrieval-Augmented Generation – це техніка, що поєднує пошук інформації з її генерацією для створення більш точних і контекстуально релевантних відповідей.

RAGAs – Retrieval-Augmented Generation Assessment – фреймворк для оцінки систем RAG.

UML – Unified Modeling Language – уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування.

ВСТУП

В сучасному світі інформаційні технології відіграють важливу роль у розвитку бізнесу та управлінні взаємодією між компаніями і їхніми клієнтами. Швидке поширення цифрових інструментів, таких як автоматизовані системи обслуговування, трансформує традиційні підходи до ведення бізнесу і створює нові можливості для підвищення ефективності та якості надання послуг. Особливо це стосується сфер, де клієнтська підтримка є одним із ключових елементів взаємодії, таких як страхування. У цій галузі, через великий обсяг документації, складність юридичних термінів та необхідність оперативного реагування на запити, автоматизація процесів взаємодії є актуальною задачею.

Проблематика автоматизації клієнтської підтримки у страхуванні активно досліджується. Часто звертається увага на необхідність вдосконалення процесів обслуговування клієнтів для забезпечення швидкого доступу до інформації про послуги та продукти компаній. Науковці наголошують, що традиційні підходи, засновані на телефонних консультаціях або електронних листах, мають низку недоліків. Вони є менш зручними для користувачів, часто не забезпечують швидкого реагування та потребують значних ресурсів з боку компанії.

Певна увага приділяється питанням інтеграції автоматизованих систем у діяльність страхових компаній. Дослідження вказують на ефективність використання автоматизованих консультантів, які можуть надавати інформацію про страхові послуги у режимі реального часу. Такий підхід дозволяє зменшити навантаження на співробітників, оптимізувати внутрішні процеси компанії та підвищити задоволеність клієнтів. Проте, попри ці успіхи, залишається низка невирішених питань.

Більшість існуючих рішень орієнтовані на стандартизовані відповіді, що обмежує їх здатність ефективно обробляти специфічні запити клієнтів. Для страхових компаній, де ключову роль відіграють деталі страхових договорів, важливою є можливість роботи з реальними документами та надання персоналізованих відповідей. Також, значною перепоною залишається недостатня

інтеграція автоматизованих рішень із популярними каналами комунікації, що знижує їх доступність для клієнтів.

Крім того, зміна потреб і очікувань клієнтів страхових компаній зумовлює необхідність більшої адаптації послуг до цифрової реальності. Клієнти дедалі більше очікують простих і зрозумілих способів взаємодії з компаніями, включаючи миттєвий доступ до потрібної інформації через мобільні пристрої. Ця тенденція вимагає не лише впровадження інноваційних технологій, але й розробки зручних та ефективних рішень, які здатні інтегруватися у вже існуючу інфраструктуру компаній, забезпечуючи при цьому надійність і безпеку даних.

Також важливим аспектом є потреба у підтримці природного діалогу між клієнтом і системою. Здатність системи розуміти контекст попередніх запитань та адаптувати відповіді у залежності від конкретного сценарію взаємодії стає вирішальною для підвищення задоволеності клієнтів. Ця проблема є особливо актуальною для страхових компаній, де кожен клієнтський запит може включати деталізовану інформацію, яка повинна враховуватися для формування точної відповіді.

Обов’язково варто звернути увагу на необхідність балансування між автоматизацією процесів та людським фактором. Хоч і автоматизовані системи можуть обробляти більшість запитів, критично важливі або нестандартні ситуації, але вони все ще потребують участі кваліфікованих співробітників. Це підкреслює значення правильної інтеграції автоматизованих рішень у загальну структуру роботи компанії, щоб забезпечити високу якість обслуговування та мінімізувати ризики втрати клієнтів.

Ефективна автоматизація здатна забезпечити значні переваги як для клієнтів, так і для компаній. Зокрема, для клієнтів це означає зручний доступ до інформації, швидкі відповіді на запити та зниження необхідності звертатися до контактного центру. Для компаній це – оптимізація витрат, зменшення навантаження на персонал, підвищення рівня задоволеності клієнтів та, як наслідок, збільшення їхньої лояльності.

Робота над вирішенням раніше вказаних проблем потребує системного підходу, який враховує як потреби клієнтів, так і особливості роботи страхових компаній. Доцільність проведення цього дослідження визначається не лише актуальністю проблеми, але й можливістю впровадження ефективного рішення, яке могло б стати прикладом для інших компаній.

Метою проєкту є підвищення ефективності роботи служби підтримки страхових компаній за рахунок розробки інформаційної системи чат-бота на основі LLM, який забезпечить швидкий та зручний доступ до інформації про страхові договори. Ця система має полегшити процес обслуговування клієнтів, надаючи автоматизовані відповіді на їхні запити, зменшити навантаження на службу підтримки та підвищити ефективність обробки звернень. Критеріями ефективності є точність, повнота, достовірність, релевантність та час отримання відповіді. Ці показники повинні бути високими для актуальності.

Об’єктом дослідження є інформаційна система служби підтримки клієнтів страхової компанії, яка автоматизує процеси обслуговування запитів клієнтів.

Предметом дослідження є властивості інтеграції великих мовних моделей і системи RAG для автоматизації обробки клієнтських запитів у страхуванні.

Для досягнення мети визначено необхідно виконати основні завдання дослідження:

– провести аналіз існуючих рішень чат-ботів підтримки та виявити їх переваги й недоліки;

– обґрунтувати вибір підходу RAG як ефективного рішення для інтеграції великих мовних моделей із пошуком контексту;

– визначити вимоги до системи, враховуючи тематику роботи;

– розробити архітектуру інформаційної системи чат-бота, охоплюючи її модулі, процеси роботи та інтеграцію з векторними базами даних;

– реалізувати систему, використовуючи сучасні технології для роботи з текстами та побудови ембедингів;

– провести тестування розробленої системи, оцінюючи її за такими критеріями, як релевантність, точність і швидкість відповіді;

– оцінити можливості впровадження системи у реальні бізнес-процеси страхових компаній.

Практичне значення розробленої системи полягає у її здатності автоматизувати обслуговування клієнтів, зменшувати навантаження на персонал страхових компаній та підвищувати ефективність їхньої роботи. Вона також може бути адаптована до інших галузей, де потрібна обробка складних текстових документів, що відкриває перспективи для її масштабування та впровадження.

Магістерська дисертація складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел із 25 найменувань, 9 додатків. Графічна частина включає 8 креслеників формату А3. Загальний обсяг 121 сторінка.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність теми

Актуальність даної теми проєкту обумовлена кількома ключовими факторами, які визначають сучасні вимоги до ефективної підтримки клієнтів у сфері страхування. Одним з факторів є стрімкий розвиток технологій штучного інтелекту та обробки природної мови відкриває нові можливості для автоматизації обслуговування клієнтів. Сучасні мовні моделі здатні розуміти й аналізувати великі обсяги текстової інформації, надавати точні відповіді на запити користувачів, що значно підвищує ефективність роботи служби підтримки [1]. Це особливо важливо для страхових компаній, які оперують значною кількістю складної інформації, включаючи страхові договори, поліси, умови й винятки, які часто потребують уточнень.

Наразі зростання кількості клієнтів і полісів збільшує навантаження на традиційні канали підтримки, такі як кол-центри чи офісні консультації, що часто призводить до затримок у наданні послуг і зниження задоволеності клієнтів. Автоматизація таких процесів за допомогою LLM дозволяє значно скоротити час обробки запитів, одночасно підвищуючи якість відповідей і зменшуючи витрати на людський ресурс. Також надає можливість працювати з великим масивом інформації та адаптуватися до нових типів запитів без необхідності ручного оновлення.

Таким чином, поєднання потреб клієнтів у швидкому та якісному обслуговуванні з новими технологічними можливостями робить дану тему особливо актуальною для страхових компаній, які прагнуть підвищити конкурентоспроможність і оптимізувати свої процеси взаємодії з клієнтами.

1.2 Аналіз існуючих рішень чат-ботів підтримки з ШІ

Чат-боти на основі штучного інтелекту активно використовуються в різних галузях, таких як фінанси, охорона здоров'я та роздрібна торгівля, і забезпечують

значні переваги для бізнесу та клієнтів. Однак, незважаючи на їхній потенціал, вони також мають свої недоліки, що можуть впливати на користувацький досвід та ефективність. Нижче було виконано огляд прикладів застосування таких ботів із зазначенням як їхніх переваг, так і недоліків.

EVA від HDFC Bank – чат-бот для банківської підтримки – рис.1.1 [2].

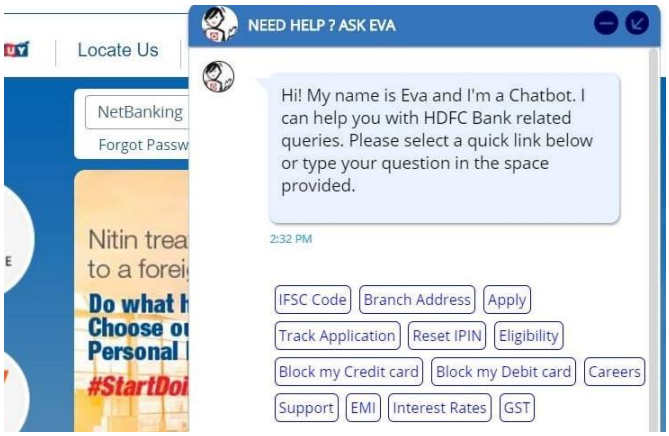


Рисунок 1.1 – Чат-бот підтримки EVA від HDFC Bank

Опис: EVA – це віртуальний асистент, який допомагає користувачам з банківськими операціями, надає інформацію про продукти банку, перевіряє залишки на рахунках, статус транзакцій тощо.

Переваги: швидкий доступ до банківських даних, скорочення часу очікування на відповіді, зручність у використанні 24/7.

Недоліки: обмежена здатність відповідати на складні або нестандартні запити, потребує додаткової персоналізації для специфічних користувачів.

EVA є ефективним прикладом використання чат-ботів у фінансовій сфері, забезпечуючи клієнтам доступ до основної інформації без необхідності звертатися до операторів. Проте її функціонал все ще обмежений, оскільки складні банківські операції вимагають людської участі. Крім того, EVA показала себе як зручний інструмент для користувачів завдяки інтеграції з мобільними додатками банку, однак вона не здатна замінити повноцінний контакт із банківським спеціалістом/

Бот від H&M – чат-бот для електронної комерції – рис.1.2 [3].

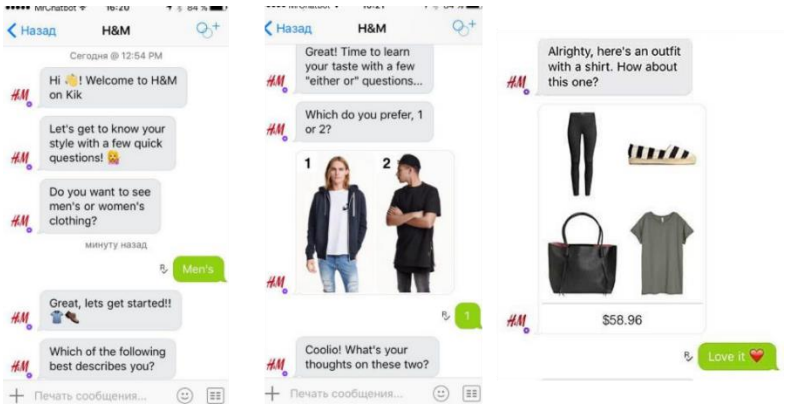


Рисунок 1.2 – Чат-бот підтримки від H&M

Опис: бот H&M допомагає клієнтам знаходити товари, отримувати рекомендації, відстежувати замовлення та перевіряти наявність продуктів у магазинах.

Переваги: персоналізовані рекомендації, швидке вирішення питань з наявністю товарів та відстеженням замовлень, інтеграція з базою даних.

Недоліки: не завжди точно відповідає на складні питання, пов’язані зі знижками чи промокодами, обмежений в обробці запитів на повернення товарів.

Бот H&M демонструє високий рівень інтеграції з базою даних товарів і замовлень, що дозволяє клієнтам швидко отримувати персоналізовану інформацію. Його перевага полягає у зручності для покупців, які можуть легко знайти потрібний товар або отримати рекомендацію.

Babylon Health – чат-бот для медичної підтримки – рис.1.3 [4].

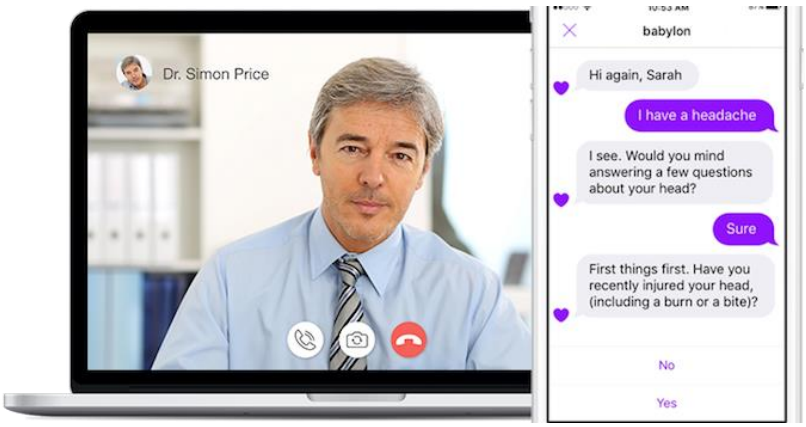


Рисунок 1.3 – Чат-бот підтримки Babylon Health

Опис: Babylon Health надає консультації на основі симптомів, допомагає знайти найближчі клініки та запланувати візити до лікаря.

Переваги: зручний доступ до базової медичної інформації, можливість попередньої діагностики симптомів, зниження навантаження на медичні служби.

Недоліки: можлива неточність у випадках складних діагнозів, потребує регулярного оновлення бази даних для врахування нових захворювань.

Babylon Health є чудовим прикладом застосування чат-ботів у сфері охорони здоров'я, дозволяючи людям швидко отримувати базові медичні консультації. Цей бот особливо корисний для попереднього аналізу симптомів та направлення до відповідних спеціалістів. Однак, він не замінює професійну медичну допомогу і може бути ненадійним для складних або рідкісних станів. Використання цього інструменту в поєднанні з кваліфікованою медичною консультацією може значно підвищити його ефективність.

KLM Royal Dutch Airlines – чат-бот для туристичної підтримки – рис. 1.4 [5]:

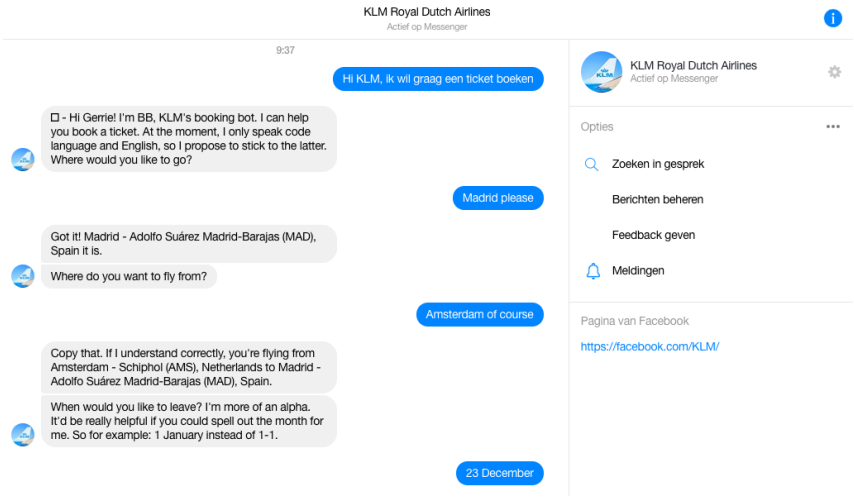


Рисунок 1.4 – Чат-бот підтримки KLM Royal Dutch Airlines

Опис: KLM допомагає бронювати квитки, відстежувати статус рейсів, надає інформацію про багаж та відповідає на питання пасажирів.

Переваги: автоматизована підтримка в режимі 24/7, швидке отримання інформації про рейси, економія часу на бронювання.

Недоліки: обмеженість у вирішенні складних ситуацій, таких як зміна квитків або повернення коштів, можливі затримки у відповідях на нетипові запити.

Чат-бот KLM ефективно допомагає пасажирам в організації їхніх подорожей, надаючи доступ до інформації про рейси та послуги компанії в будь-який час. Його інтеграція з основними сервісами KLM дозволяє значно зекономити час клієнтів, зменшуючи навантаження на контактний центр. Його функціональність обмежується стандартними запитами, і для унікальних ситуацій все ще потрібна участь людини. Удосконалення системи розуміння складних запитів може зробити його ще більш ефективним для клієнтів авіакомпанії.

1.3 Проблематика сучасних LLM

Великі мовні моделі (LLM, Large Language Models) – це системи штучного інтелекту, створені на основі нейронних мереж із мільярдами параметрів, які здатні обробляти, аналізувати й генерувати природну мову. Вони тренуються на величезних масивах текстових даних і можуть виконувати широкий спектр завдань, таких як відповіді на запитання, створення текстів, переклад, аналіз настрою та багато іншого. Завдяки своїм можливостям LLM знаходять застосування в асистентах, чат-ботах, автоматизації бізнес-процесів, дослідженнях та інших сферах, де потрібна обробка текстової інформації, особливо при використанні малопоширених мов світу (таких як українська) [6-7]. Їхньою ключовою перевагою є здатність генерувати людиноподібні відповіді, що робить їх цінним інструментом для створення інтелектуальних систем взаємодії.

Чат-боти на основі LLM значно вплинули на розвиток автоматизації обслуговування клієнтів у різних галузях. Вони здатні аналізувати запити користувачів, генерувати текстові відповіді та підтримувати діалог у зручній і природній формі. Це робить їх корисними у фінансах, охороні здоров’я, торгівлі, освіті та багатьох інших сферах. Проте, незважаючи на їхній потенціал і популярність, сучасні чат-боти на основі LLM мають низку проблем, які обмежують їх ефективність.

Однією з ключових проблем є обмеженість у доступі до актуальної інформації. Великі мовні моделі, як-от GPT-3 чи GPT-4, навчаються на великих

наборах даних, які охоплюють знання до певного моменту часу. Наприклад, GPT-4, популярна мовна модель, не має доступу до даних, створених після вересня 2021 року (на момент її навчання). Це означає, що такі чат-боти не можуть обробляти інформацію про нові події, оновлення законодавства, зміни у ринкових умовах або інші динамічні аспекти реального світу.

Недолік актуальності стає особливо помітним у таких випадках:

– швидкозмінна інформація: чат-боти не здатні дати відповіді, які базуються на даних, створених після їх навчання. Наприклад, події, що відбулися після дати оновлення даних, залишаються для них невідомими;

– відсутність доступу до реальних джерел інформації: багато моделей працюють у «замкнутому» режимі і не інтегруються з актуальними базами даних чи веб-сервісами;

– оновлення контексту: моделі LLM не можуть самостійно оновлювати свої знання, тому вони обмежені лише тим, що було включено до їх початкового набору даних.

Сучасні чат-боти на основі LLM, хоча і демонструють значний прогрес у сфері обробки природної мови, стикаються з проблемами актуальності даних. Їхня здатність надавати точну інформацію обмежена набором даних, використаних для навчання, і відсутністю інтеграції з динамічними джерелами. Це створює перешкоди у застосуванні таких ботів для рішень, які потребують сучасної інформації, особливо у сфері страхування, охорони здоров’я та ін.

1.4 Сучасне вирішення актуальності LLM – Retrieval-Augmented Generation

Розглянувши проблеми сучасних LLM, було виявлено, що існує багато викликів при роботі з ними, таких як прогалини в галузевих знаннях, проблеми з достовірністю інформації та «галюцинації». Це обумовлено тим, що у традиційних мовних моделях відповіді генеруються виключно на основі попередньо вивчених шаблонів та інформації на етапі навчання. Підхід, що базується на доповненні генерації за допомогою пошуку (RAG), усуває ці обмеження, залучаючи зовнішні

дані за потреби під час процесу генерації [8]. Ось як це працює: при надходженні запиту система RAG спершу витягує релевантну інформацію з великого масиву даних або бази знань, а потім використовує її для формування та направлення процесу створення відповіді.

Розглянемо архітектуру системи RAG. Перше, що можна сказати, що це складна система, створена для розширення можливостей LLM шляхом поєднання їх із потужними механізмами пошуку інформації. Вона складається з двох основних компонентів: механізму пошуку (retriever) і генератора (generator). Розглянемо кожен компонент та їхні функції у загальному процесі. [9]

Компонент пошуку (Retriever). Завдання: основна функція цього компонента – знаходити релевантні документи або інформацію, яка допоможе відповісти на запит. Він отримує запит і шукає в базі даних потрібні дані, які можуть бути корисними для генерації відповіді.

Типи пошукових механізмів:

– щільні ретривери (Dense Retrievers): використовують методи на основі нейронних мереж для створення щільних векторних представлень тексту. Вони краще працюють у випадках, коли важливе значення тексту, а не точне збігання слів, оскільки такі представлення враховують семантичну подібність;

– розріджені ретривери (Sparse Retrievers): опираються на техніки порівняння термінів, є ефективними для пошуку документів із точними збігами ключових слів, що корисно, якщо запит містить унікальні або рідкісні терміни.

Вибір між щільними та розрідженими ретриверами залежить від особливостей бази даних і типів очікуваних запитів. Щільні пошуковики забезпечують виявлення глибоких семантичних зв’язків, хоча вимагають більше обчислювальних ресурсів, тоді як розріджені працюють швидше і краще підходять для точного збігу конкретних термінів.

Компонент генерації (Generator). Завдання: генератор – це мовна модель, яка створює підсумковий текстовий результат. Він отримує запит і контекст, знайдений компонентом пошуку, для формування зв’язної та релевантної відповіді.

Взаємодія з ретривером: генератор працює не окремо, а в тісній взаємодії з контекстом, наданим пошуковим механізмом. Це забезпечує, щоб відповідь була не лише правдоподібною, але й детальною та точною.

Розглянемо робочий процес системи RAG. [9]

1. Обробка запиту: процес починається із запиту — це може бути питання, підказка або будь-який інший ввід, на який мовна модель повинна відповісти.

2. Модель векторизації: Запит передається до моделі векторизації, яка перетворює його у вектор — числове представлення, зрозуміле та зручне для обробки системою.

3. Пошук у векторній базі даних: вектор запиту використовується для пошуку у векторній базі даних, яка містить попередньо обчислені вектори можливих контекстів. Система знаходить найбільш релевантні контексти, ґрунтуючись на схожості їхніх векторів із вектором запиту.

4. Отримані контексти: знайдені контексти передаються у LLM. Ці контексти містять інформацію, яку модель використовує для створення точної та інформативної відповіді.

5. Генерація відповіді LLM: мовна модель обробляє як початковий запит, так і отримані контексти, щоб створити повну та релевантну відповідь. Вона синтезує інформацію з контекстів, забезпечуючи, щоб відповідь не лише базувалася на її початкових знаннях, але й була доповнена специфічними деталями з отриманих даних.

6. Підсумкова відповідь: на завершення модель формує відповідь, яка тепер збагачена зовнішньою інформацією, що робить її більш точною та деталізованою.

RAG знаходить застосування в багатьох сферах штучного інтелекту, значно підвищуючи якість і релевантність відповідей, що генеруються мовними моделями.

Підвищення ефективності чат-ботів та розмовних агентів:

– підтримка клієнтів: чат-боти з підтримкою RAG можуть знаходити інформацію про продукти, FAQ та документацію для надання точних і детальних відповідей на запити клієнтів;

– персональні асистенти: віртуальні помічники використовують RAG для отримання актуальних даних, таких як прогноз погоди чи новини, роблячи взаємодію більш контекстуально релевантною.

Поліпшення точності в автоматизованому створенні контенту:

– створення контенту: журналістські інструменти на базі RAG витягують актуальні факти та дані, що дозволяє створювати насичені та достовірні статті;

– копірайтинг: маркетингові боти генерують креативні й точні описи продуктів та рекламні тексти, посилаючись на бази даних із характеристиками товарів та відгуками.

Використання в системах запитання-відповідь:

– освітні платформи: RAG допомагає студентам зрозуміти складні теми, надаючи додатковий контекст із освітніх баз даних;

– дослідження: системи на базі RAG знаходять відповіді на наукові питання, аналізуючи корпус академічних статей і створюючи узагальнення.

Переваги RAG у різних галузях:

– охорона здоров’я: системи RAG допомагають лікарям, витягуючи інформацію з медичних журналів і записів пацієнтів для підтримки діагностування чи лікування;

– обслуговування клієнтів: завдяки доступу до політик компанії та історії клієнтів, RAG дозволяє персоналізувати відповіді, підвищуючи задоволеність клієнтів та зменшуючи завантаження на кол-центр;

– освіта: викладачі можуть використовувати для створення індивідуальних планів занять і навчальних матеріалів, забезпечуючи студентів різноманітними точками зору.

RAG ідеально підходить для розробки системи чат-бота підтримки, оскільки дозволяє забезпечити точні, релевантні та детальні відповіді на запити користувачів. Завдяки використанню зовнішніх баз знань, RAG компенсує обмеження мовних моделей у галузевих знаннях, надаючи можливість отримувати актуальну інформацію з документів, таких як договори, політики чи бази FAQ. Крім того, цей підхід не вимагає повторного навчання моделі для кожного типу

запитів, що спрощує адаптацію чат-бота до нових сценаріїв. У результаті RAG значно підвищує якість обслуговування, забезпечуючи користувачів корисними та достовірними відповідями, адаптованими до їхніх потреб.

1.5 Вимоги інформаційної системи чат-бота підтримки

На основі опрацьованої та проаналізованої інформації про сучасні підходи до створення інформаційних систем, а також з урахуванням аналізу проблематики існуючих чат-ботів, було визначено низку вимог до розробки інформаційної системи для автоматизації служби підтримки страхової компанії.

Вимоги до системи розподілені на функціональні та нефункціональні аспекти, що дозволяє деталізувати як її основні завдання, так і умови їх ефективного виконання. Нижче подано опис кожної категорії вимог.

1.5.1 Функціональні вимоги

Функціональні вимоги визначають основну функціональність чат-бота для забезпечення потреб клієнтів та страхової компанії – наведено в таблиці 1.1.

Таблиця 1.1 – Функціональні вимоги

Id	**Назва вимоги**	**Опис**
F1	Прийом текстових запитів	Система повинна приймати текстові запити від користувачів через різні канали (месенджери, веб-інтерфейси).
F2	Інтеграція з документами	Система повинна працювати з файлами договорів у форматі PDF, розбивати їх на сегменти та зберігати у векторному вигляді для пошуку.
F3	Пошук у документах	Чат-бот має здійснювати пошук відповідей у текстах страхових договорів на основі запиту користувача.

Id	Назва вимоги	Опис
F4	Підтримка контексту діалогу	Бот повинен зберігати історію діалогу для врахування контексту при наданні відповідей.
F5	Динамічний вибір договору	Користувач повинен мати можливість обирати договір зі списку для отримання консультації або змінювати свій вибір під час роботи з ботом.
F6	Логування взаємодій	Система повинна записувати всі запити, відповіді та можливі помилки у лог-файл для моніторингу та аналізу.
F7	Відповідність умовам договорів	Чат-бот повинен формувати відповіді виключно на основі даних з договорів, уникаючи спотворення інформації.
F8	Підтримка користувацьких помилок	Система повинна розпізнавати некоректні або неповні запити та надавати рекомендації для їх виправлення.
F9	Зворотний зв'язок із користувачем	У разі виникнення складнощів бот повинен запропонувати користувачеві звернутися до служби підтримки або надати контактну інформацію.

Даний перелік функціональних вимог допоможе забезпечити виконання поставленої мети та завдань системи.

1.5.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи, необхідні для її ефективної роботи – наведено в таблиці 1.2.

Таблиця 1.2 – Нефункціональні вимоги

Id	Назва вимоги	Опис
NF1	Швидкодія	Час відповіді на запит не повинен перевищувати 3-5 секунд.
NF2	Масштабованість	Система повинна підтримувати додавання нових страхових договорів без зниження продуктивності.
NF3	Надійність	Система повинна бути доступною для користувачів у безперебійному форматі, автоматично обробляючи незначні збої або інформуючи про технічні труднощі.
NF4	Конфіденційність	Усі конфіденційні дані, такі як токени доступу та інформація користувачів, мають зберігатися у захищеному середовищі та не передаватися третім сторонам.
NF5	Простота використання	Інтерфейс системи має бути інтуїтивно зрозумілим для користувачів, без необхідності спеціальних технічних знань.
NF6	Підтримка кількох мов	Система повинна забезпечувати можливість роботи щонайменше двома мовами для зручності міжнародних користувачів.
NF7	Сумісність	Чат-бот повинен бути легко адаптованим для роботи на різних платформах (месенджери, веб-інтерфейси, мобільні додатки) та до різних типів пристроїв.
NF8	Логування і моніторинг	Взаємодії користувачів мають бути детально зафіксовані для подальшого аналізу та вдосконалення системи.
NF9	Зручне масштабування	Архітектура проєкту повинна бути гнучкою для адаптації до різних бізнес-завдань, таких як додавання нових типів запитів або інтеграція з іншими системами.

Визначені функціональні та нефункціональні вимоги забезпечують основу для створення інформаційної системи, яка відповідає сучасним запитам клієнтів страхових компаній. Впровадження цих вимог дозволяє підвищити ефективність роботи служби підтримки, забезпечити доступність послуг та підвищити рівень задоволеності клієнтів.

Висновки до розділу 1

У розділі здійснено аналіз актуальності створення інформаційної системи для автоматизації підтримки клієнтів страхової компанії з використанням LLM та технології RAG. Обґрунтовано необхідність впровадження таких рішень у страхову галузь через зростання обсягів клієнтських запитів, складність обробки інформації та вимоги до підвищення ефективності обслуговування. Розглянуто приклади існуючих чат-ботів у різних галузях, зокрема фінансах, електронній комерції та медицині, що дозволило виявити переваги й недоліки подібних систем, а також їхні обмеження у доступі до актуальних даних.

Розкрито концепцію RAG, яка інтегрує мовні моделі з механізмами пошуку релевантного контексту, забезпечуючи динамічне оновлення знань та адаптацію до галузевих запитів без необхідності перевчання. Деталізовано архітектуру RAG, що включає компоненти пошуку та генерації, і проаналізовано її переваги для автоматизації обслуговування клієнтів.

Сформульовано мету проєкту – є підвищення ефективності роботи служби підтримки страхових компаній за рахунок розробки інформаційної системи чат-бота на основі LLM, здатного надавати точну й релевантну інформацію про договори страхування. Визначено функціональні й нефункціональні вимоги до системи, що забезпечують її ефективність, масштабованість і відповідність сучасним стандартам. Таким чином, розділ закладає науково-практичну основу для реалізації проєкту, орієнтованого на оптимізацію бізнес-процесів та підвищення якості клієнтського сервісу.

2 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ РОЗРОБКИ ЧАТ-БОТУ

У даному розділі описана формалізації підходів і алгоритмів, які використовуються в процесі розробки системи, у цьому випадку – чат-бота. Він дозволяє описати основні математичні моделі, метрики та алгоритми, які використовуються при розробці чат-бота підтримки.

Цей розділ дає теоретичне обґрунтування всіх технічних рішень, забезпечуючи чіткість і прозорість розробки. Завдяки йому можна оцінити коректність використаних методів, їхню ефективність і обмеження, що важливо для подальшої оптимізації роботи системи. Він також слугує базою для вибору відповідних алгоритмів і моделей, дозволяючи узгодити технічну реалізацію із загальними цілями проєкту.

2.1 Метрика Fuzzy Matching. Алгоритм Левенштейна

Fuzzy Matching – це метод порівняння текстових рядків, який оцінює їхню схожість, враховуючи незначні відмінності, шляхом використання нормалізованої редакційної відстані (алгоритм Левенштейна) [10]. Формула порівняння рядків (2.1):

$$Fuzzy\ Similarity = 1 - \frac{D(m,n)}{\max(m,n)}, \tag{2.1}$$

де *Fuzzy Similarity* – схожість між двома рядками, що приймає значення у діапазоні [0, 1];

D(m,n) – редакційна відстань (відстань Левенштейна) між рядками довжини m і n;

max(m,n) – довжина довшого рядка.

Алгоритм Левенштейна визначає мінімальну кількість операцій (вставки, видалення або заміни), які необхідно виконати, щоб перетворити один рядок у інший. Формула розрахунку при різних сценаріях (2.2):

$$D(i,j) = \begin{cases} i, \text{ якщо } j = 0, \\ j, \text{ якщо } i = 0 \\ D(i-1, j-1), \text{ якщо } a_i = b_j, \\ min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + 1 \end{cases} \text{, якщо } a_i \neq b_j, \end{cases} \quad (2.2)$$

де *D(i,j)* – редакційна відстань між префіксами;

i – довжина префікса рядка A;

j – довжина префікса рядка B;

a_i – i-й символ рядка A;

b_j – j-й символ рядка B;

min – мінімум із трьох операцій: *D(i−1,j)+1* – операція видалення; *D(i,j−1)+1* – операція вставки; *D(i−1,j−1)+1* – операція заміни.

Метрика *Fuzzy Matching* використовує редакційну відстань *D(m,n)* нормалізуючи її шляхом ділення на максимальну можливу довжину рядків *max(m,n)*. Це дозволяє оцінювати схожість текстів у відносних одиницях, зручних для пошуку часткових збігів або приблизного порівняння.

2.2 RAGAs: метрики оцінювання системи та їх розрахунок

RAGAs (Retrieval-Augmented Generation and Assessment System) — це підхід, який поєднує пошук релевантної інформації (retrieval) із генерацією відповідей (generation) для створення більш точних і контекстуально релевантних результатів. Він включає механізми оцінки якості відповідей за допомогою спеціалізованих метрик, нижче розглянемо основі, що використовуватимуться в роботі [11].

Context Precision (точність контексту) – вимірює частку релевантних контекстів серед усіх витягнутих контекстів. Формула розрахунку метрики (2.3):

$$CP@k = \frac{\sum_{k=1}^{K} (Precision@k * v_k)}{R_K}, \quad (2.3)$$

де *CP@K* – точність контексту;

K – загальна кількість контекстів (або фрагментів тексту), які розглядаються у топ-K результатах, це межа для аналізу точності;

@K – вказує на позиційну залежність метрики: вона враховує тільки перші *K* результати в списку, ігноруючи решту;

R_K – загальна кількість релевантних елементів серед перших K;

Precision@k – точність на конкретній позиції *k* у ранжованих результатах. Формула (2.4):

$$Precision@k = \frac{true\_positives@k + false\_positives@k}{true\_positives@k}, \quad (2.4)$$

де *true positives@k* – кількість релевантних результатів до позиції k;

false positives@k – кількість нерелевантних результатів до позиції k.

Context Recall (повнота контексту) – оцінює, наскільки система змогла знайти всі релевантні контексти. Формула розрахунку метрики (2.5):

$$CR = \frac{|\mathrm{S}_{\mathrm{attrib}}|}{|\mathrm{S}_{CT}|}, \quad (2.5)$$

де *CR* – повнота контексту;

S_{attrib} – кількість речень із Ground Truth (GT), які можуть бути чітко пов'язані з контекстом, наданим системою;

S_{GT} – загальна кількість речень у Ground Truth (GT), тобто в еталонних даних, які вважаються правильними або релевантними для оцінки.

Faithfulness (достовірність) – показує, наскільки відповіді системи узгоджуються з наданим контекстом. Формула розрахунку метрики (2.6):

$$\mathrm{FS} = \frac{|\mathrm{C}_{inferred}|}{|C_C|}, \quad (2.6)$$

де *FS* – достовірність;

$C_{inferred}$ – кількість тверджень у згенерованій відповіді, які можна вивести або підтвердити на основі наданого контексту;

C_{total} – загальна кількість тверджень у згенерованій відповіді.

Answer Relevancy (релевантність відповіді) – визначає, наскільки відповідь відповідає запиту користувача. Формула розрахунку метрики (2.7):

$$AR = \frac{1}{N}\sum_{i=1}^{N} \cos(E_{g_i}, E_o) \,, \tag{2.7}$$

де AR – релевантність відповіді;

E_{gi} – векторне (ембединг) представлення згенерованого запитання;

E_o – векторне (ембединг) представлення оригінального запитання (embedding of the original question);

N – загальна кількість згенерованих запитань (the number of generated questions);

$cos(E_{gi}, E_o)$ – косинусна схожість між ембедингами згенерованого запитання та оригінального запитання. Косинусна схожість визначає, наскільки подібними є два вектори, де 1 означає повну схожість, а 0 — відсутність схожості. Формула (2.8):

$$\cos(E_{g_i}, E_o) = \frac{E_{g_i} * E_o}{\left\|E_{g_i}\right\| * \|E_o\|} \,, \tag{2.8}$$

де $E_{gi} * E_o$ – скалярний добуток (dot product) двох векторів;

$\| E_{gi} \|$ – норма (довжина) вектора E_{gi} ;

$\| E_o \|$ – норма (довжина) вектора E_o .

Answer Semantic Similarity (семантична схожість відповіді) – оцінює схожість між згенерованою відповіддю та правильною відповіддю на семантичному рівні. Формула розрахунку метрики вже була раніше записана, це косинусна схожість між ембедингами згенерованого запитання та оригінального запитання – формула (2.8).

Висновки до розділу 2

У цьому розділі було розглянуто метрики оцінювання, які дозволяють кількісно правильність згенерованих відповідей у системах на основі алгоритмів обробки природної мови. Зокрема, метрика Fuzzy Matching, заснована на алгоритмі Левенштейна, дозволяє визначати схожість текстових рядків із врахуванням незначних відмінностей, а система RAGAs використовує набір метрик для оцінки якості витягнутого контексту та згенерованих відповідей, таких як Context Precision, Context Recall, Faithfulness, Answer Relevancy, Answer Semantic Similarity, забезпечуючи всебічну оцінку роботи системи.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ МЕТОДІВ ТА ТЕХНОЛОГІЙ РЕАЛІЗАЦІЇ RAG-СИСТЕМИ

3.1 Аналітичний огляд найпопулярніших фреймворків для реалізації RAG

Отже, раніше вже було згадано, що метод RAG об'єднує можливості систем пошуку інформації з генеративними моделями, що робить його надзвичайно ефективним для вирішення завдань, таких як відповідь на запитання, узагальнення тексту та інші завдання обробки природної мови. Для реалізації RAG сьогодні найчастіше використовують два популярні фреймворки: LangChain і LlamaIndex.

Ці інструменти спеціально розроблені для роботи з документами, їх поділу, індексації та побудови послідовних кроків для забезпечення безперервного функціонування процесів RAG. В цьому підрозділі розглянуто кожен з фреймворків, їхні принципи роботи, основні компоненти, найкращі випадки використання та їхні особливості.

3.1.1 LlamaIndex

LlamaIndex – це ефективний інструмент для індексації та пошуку даних, створений для покращення доступності інформації. Він забезпечує оптимізацію процесу організації даних, дозволяючи швидше і простіше знаходити релевантну інформацію. LlamaIndex спеціалізується на використанні ембеддінгів для збереження даних, що значно підвищує точність і релевантність пошуку.

Дана платформа дозволяє налаштовувати LLM, інтегруючи ваші власні дані в пам’ять системи, що дозволяє моделі з часом надавати все точніші відповіді, враховуючи контекст.

Опис процесу надання відповіді складається з двох основних етапів:

– етап індексації: дані перетворюються у векторний індекс. На цьому етапі дані перетворюються в векторне представлення або числову форму зі смисловим наповненням;

– етап запиту: коли система отримує запит, вона визначає інформацію, яка найбільш семантично схожа на цей запит. Ця важлива інформація разом із початковим запитом надсилається LLM для отримання остаточної відповіді. Цей механізм дозволяє RAG отримувати високоточні та відповідні результати, які неможливо отримати, маючи лише базові знання LLM. [12]

Ключові особливості LlamaIndex:

– ефективна індексація: забезпечує швидку організацію та структурування великих обсягів даних;

– покращений пошук: передові алгоритми гарантують швидкий і точний пошук, покращуючи взаємодію користувача.

Основні функції:

– індексація даних : спеціалізується на швидкій організації даних у числові представлення (ембеддінги), що дозволяє миттєво знаходити потрібну інформацію для різних сценаріїв використання;

– алгоритми ранжування: оптимізує пошук, ранжуючи документи на основі їхньої семантичної схожості із запитом;

– продуктивність і ефективність: орієнтований на високу швидкість обробки даних і мінімальну затримку у пошуку інформації;

– збереження контексту: підходить для простих завдань пошуку, забезпечуючи базові функції утримання контексту;

– налаштування: LlamaIndex орієнтований на оптимізацію пошукових завдань, пропонуючи обмежені можливості кастомізації.

3.1.2 LangChain

LangChain – це багатофункціональний фреймворк, розроблений для створення додатків, які використовують мовні моделі. Його модульна архітектура дозволяє розробникам легко створювати індивідуальні рішення для різних задач. Він забезпечує інструменти для управління підказками, ланцюгами завдань, інтеграції з мовними моделями та управління пам’яттю для збереження контексту

попередніх взаємодій. Ця платформа підтримує інтеграцію з різноманітними джерелами, такими як реляційні бази даних (табличні дані), нереляційні бази даних (наприклад, документи), програмні джерела (такі як API) або навіть власні бази знань.

LangChain використовує механізм створення ланцюгів – послідовностей запитів, що надсилаються LLM разом з іншими інтегрованими інструментами. Вихідні дані одного кроку стають вхідними для наступного, забезпечуючи безперервний потік обробки інформації.

Ця платформа працює з власними даними, гарантуючи, що LLM надається необхідний контекст для створення відповідей. Незалежно від того, чи це чат-бот для запитань і відповідей на основі даних компанії, інструмент внутрішньої аналітики або AI-асистент, що працює з джерелами даних, LangChain забезпечує інтеграцію інших інструментів у додаток. Завдяки своєму механізму ланцюгів LangChain допомагає створювати більш комплексні та ефективні системи. [13]

Ключові особливості LangChain:

– модульна архітектура: пропонує гнучкість для адаптації до різних сценаріїв використання;

– різноманітність застосувань: дозволяє створювати додатки, такі як чат-боти, генерація текстів і переклади.

Основні функції:

– індексація даних : акцентує увагу на модульності, дозволяє розробникам створювати індивідуальні рішення для роботи з даними відповідно до конкретних вимог;

– алгоритми ранжування: комбінує алгоритми пошуку з мовними моделями, забезпечуючи контекстуальні та точні відповіді;

– продуктивність і ефективність: поєднує ефективність із гнучкістю, підтримуючи різні бази даних і дозволяючи інтегрувати алгоритми пошуку з мовними моделями для досягнення балансу між швидкістю і точністю;

– збереження контексту: розроблений для довготривалих та складних взаємодій, таких як чат-боти, що потребують збереження і використання контексту попередніх запитів;

– налаштування: підтримує складніші робочі процеси і дозволяє гнучко налаштовувати підказки, ланцюги завдань і робочі процеси, роблячи його ідеальним для створення індивідуальних рішень.

3.1.3 Випадки застосування та підсумок огляду фреймворків

Після огляду та опису основних підходів, методик та особливостей LlamaIndex і LangChain, підсумуємо усе проаналізоване у таблиці 3.1, виділивши основні ознаки кожного фреймворку, щоб зрозуміти, який з них буде краще використовувати при реалізації системи.

Таблиця 3.1 – Характеристики та функції LlamaIndex і LangChain

Характеристика	**LlamaIndex**	**LangChain**
Основний фокус	Ефективна організація та пошук інформації	Інтеграція різних AI-інструментів та процесів
Головний сценарій використання	Створення баз даних із можливістю пошуку	Розробка складних AI-систем, здатних виконувати кілька завдань
Робота з даними	Спеціалізується на організації різних типів даних	Може працювати з даними, але це не його основна сильна сторона
Інтеграція	Добре працює з існуючими даними	Краще інтегрує різні AI-інструменти
Складність	Зазвичай простіший для базових завдань	Пропонує більше можливостей, але вимагає більше часу на вивчення

Характеристика	LlamaIndex	LangChain
Оптимізація запитів	Має вбудовані функції для пришвидшення та покращення пошуку	Часто потребує ручної оптимізації пошуку
Налаштування	Менше можливостей для кастомізації	Дозволяє розширені налаштування

Найкращі практики застосування кожного з фреймворків.

LlamaIndex:

– чудово підходить для створення систем, які орієнтовані на запити та пошук інформації з визначеної бази знань;

– використовується для розробки чат-ботів Q&A, які надають точні та релевантні відповіді на запити користувачів;

– підтримує завдання, такі як узагальнення великих документів, автозавершення текстів і переклад мовами.

LangChain:

– ідеальний для розробки чат-ботів і просунутих AI-агентів для діалогу;

– дозволяє інтегрувати індивідуальні робочі процеси безпосередньо в LLM;

– сприяє підключенню мовних моделей до зовнішніх джерел даних, таких як API, для розширення можливостей обробки даних.

Отже, після проведеного огляду фреймворків LlamaIndex та LangChain, їх функціоналу, сфер застосування, а також аналізу вимог до розроблюваної системи, було прийнято рішення віддати перевагу LangChain. Це було обумовлено тим, що даний фреймворк володіє розширеним можливостям підтримки діалогу та інтеграції, є більш гнучким в налаштуванні, що дозволяє адаптувати систему під змінні вимоги. Також одним з основних факторів є відповідність як функціональним, так і нефункціональним вимогам, визначеним для проєкту. І LangChain надає гнучку, масштабовану й ефективну платформу для побудови комплексної системи, що відповідає завданням сучасного бізнесу.

3.2 Реалізація програми для оцінки вибору фреймворку

Після проведення детального аналізу та огляду фреймворків LlamaIndex та LangChain, перевіримо на практиці якість та швидкість відповідей кожного з них на прикладі простої системи Q&A. В наступних пунктах буде опис розробки програми, яка буде давати оцінку відповідям реалізацій Q&A. Виконана вона у Google Colaboratory – онлайн редактор коду, який підтримує необхідні бібліотеки та надає середовище розробки. Надає безкоштовний доступ до GPU та TPU для користувачів, що суттєво прискорює процес виконання коду.

3.2.1 Метрика Fuzzy matching

Для побудови універсальної програми оцінки відповідей було вирішено використати метрику Fuzzy matching.

Fuzzy matching (розмитий пошук або нечітке порівняння) – це метод, який використовується для порівняння текстових рядків з урахуванням можливих варіацій, таких як орфографічні помилки, перестановки слів або неповні збіги. Вона допомагає визначити ступінь подібності між двома текстами за допомогою різних алгоритмів. Обрахунок подібності наведено за формулою (2.1). [10]

Принцип роботи полягає в тому, що дана техніка оцінює схожість текстового рядка і повертає число, яке показує, наскільки схожі два тексти. Основна ідея полягає в тому, щоб враховувати і порівнювати текст з буквами і словами: порядок букв/слів, пропущені або додані символи, заміна символів (наприклад, помилок).

У нашому випадку було використано саме інструмент: FuzzyWuzzy (Python). Він заснований на алгоритмі Левенштейна, який вимірює кількість операцій (вставки, видалення та заміни), необхідних для перетворення одного рядка в інший. Математичний розрахунок алгоритму наведено у формулі (2.2).

А також підтримує кілька методів оцінки, наприклад:

– fuzz.ratio: базове порівняння двох рядків;

– fuzz.partial_ratio: знаходить часткові збіги;

– fuzz.token_sort_ratio: порівнює рядки, ігноруючи порядок слів;

– fuzz.token_set_ratio: враховує унікальні слова і порівнює множини.

Було використано fuzz.ratio: базове порівняння двох рядків, для реалізації методу оцінки – рис. 3.1.

```
def evaluate_answer(reference: str, response: str) -> float:
    """Оцінювання відповіді на основі схожості тексту."""
    return fuzz.ratio(reference, response)
```

Рисунок 3.1 – Методу оцінки відповідей

Як можна побачити, у методі присутній поточний запит (responce) та вже раніше підготовлений (reference). Тобто потрібно мати вже заготовлений список відповідей та запитань, для виконання оцінки.

3.2.2 Отримання еталонних відповідей

Як було вияснено з попереднього пункту, для виконання оцінки потрібно мати вже заготовленні (еталонні) відповіді на певний перелік питань.

Для вирішення цього запитання було використано OpenAI Assistants API, даний сервіс знаходиться у розробці, тому це лише beta, але він показує дуже гарні результати та ефективність роботи [14]. Асистенти, створені за допомогою API Assistants, можуть бути оснащені інструментами, які дозволяють виконувати складніші завдання або взаємодіяти з вашим застосунком. Вони надають вбудовані інструменти для асистентів, але також можна визначити власні інструменти, щоб розширити їхні можливості, використовуючи функціональні виклики.

На даний момент API Assistants підтримує такі інструменти:

– пошук файлів (File Search) – вбудований інструмент RAG для обробки та пошуку інформації у файлах;

– інтерпретатор коду (Code Interpreter) – написання та виконання Python-коду, обробка файлів та різноманітних даних;

– функціональні виклики (Function Calling) – використання власних користувацьких функцій для взаємодії із вашим застосунком.

У нашому випадку нас цікавить вбудований інструмент RAG. Використаємо цей інструмент при створені свого «помічника». Тому розпочнемо роботу з API Assistants. Коли зайшли у режим створення бота, то нас зустріне наступне вікно – рис. 3.2. В частині зліва потрібно буде вказати поля Name – назва «помічника», System Instruction, Model – LLM, яка буде використовуватися для надання відповідей. Також, якщо прогорнути поле з налаштуваннями нижче, там можна буде побачити налаштування Tools – рис. 3.3 та Model Configuration – рис. 3.4.

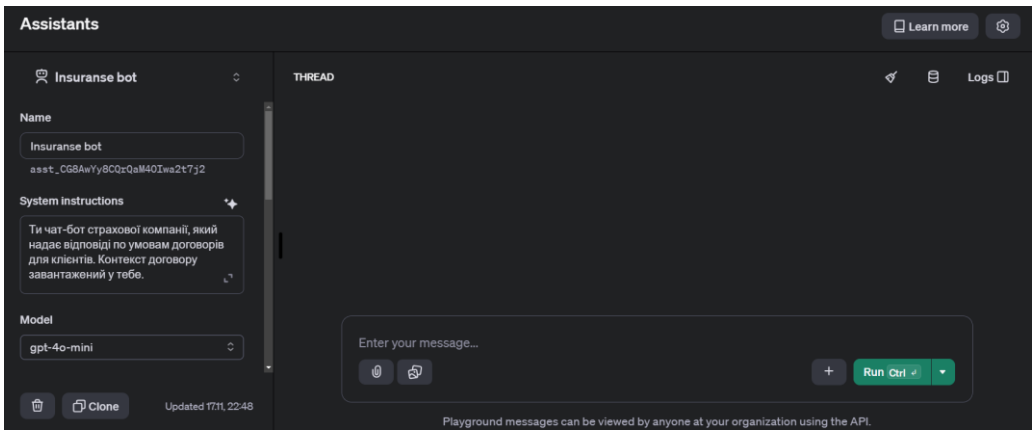
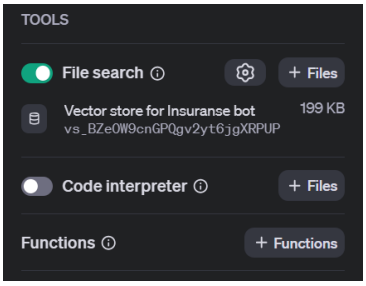
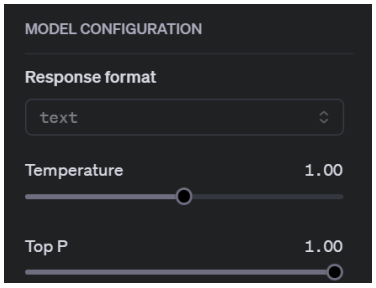


Рисунок 3.2 – Налаштування помічника через Assistants API



Рисунки 3.3 – Налаштування Tools



Рисунки 3.4 – Налаштування Model Configuration

На рисунках 3.2–3.4 можна побачити, які налаштування було обрано. А також можна помітити, що було завантажено PDF-файл, з договором. Договір було взято з сайту однієї зі страхових компаній. Посилання на договір – [15].

Після налаштування можна повноцінно користуватися помічником. Приклад роботи зображено на рисунку 3.5.

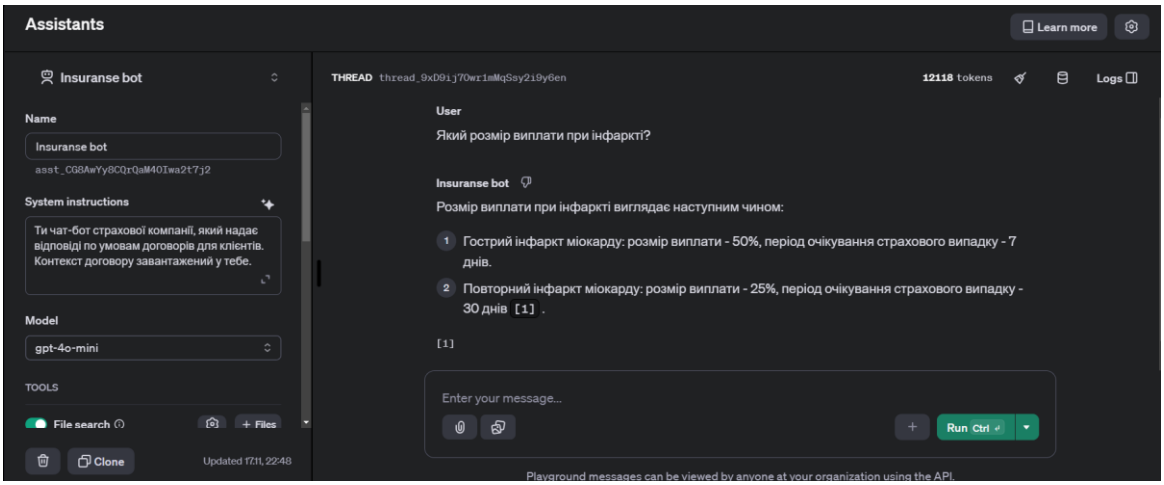


Рисунок 3.5 – Приклад запиту та відповідь у Assistants API

Тепер маючи такого даний інструмент, можна отримати еталонні відповіді по завантаженому договору, для подальших оцінок. Для цього було створено запит до налаштованого чат-бота підтримки страхової компанії, у якому він мав придумати 20 запитань по договору, який у нього завантажено, і він надав наступні питання – рис. 3.6.

```
# Список питань
questions = [
    "Що таке страховий випадок згідно з договором страхування здоров'я?",
    "Як подати заявку на відшкодування?",
    "Який розмір страхової виплати у випадку діагностування гострого інфаркту міокарду?",
    "Які виключення зі страхових випадків передбачено в договорі?",
    "Чи покриває страховка лікування хронічних захворювань?",
    "Які лікарі та медичні установи входять до мережі партнерів компанії?",
    "Напиши контактні дані для подачі заявки.",
    "Що робити, якщо сталася надзвичайна ситуація і потрібна термінова медична допомога?",
    "Чи можу я отримати страховку на свою дитину?",
    "Про що йдеться у пункті 9.3.4?",
    "Чи покриває страховка лікування повторного інфаркту міокарду?",
    "Які хвороби не підлягають страхуванню відповідно до виняткiв у договорі?",
    "Сепсис це страховий випадок?",
    "Чи покривається обстеження на цукровий діабет у разі його вперше діагностування?",
    "Чи є у страхового покриття обмеження щодо віку застрахованої особи?",
    "Які наслідки настання страхової події, якщо захворювання було виявлено до укладання договору?",
    "Чи є можливість отримати виплату за лікування гайморит за цим договором?",
    "Що таке Субкомпенсація?",
    "Які хвороби крові, кровотворних органів є страховими згідно Додаток №1?",
    "Який розмір виплати при Злоякісні новоутворення тіла матки та придатків матки, якщо мені 37 років?"
]
```

Рисунок 3.6 – Згенеровані питання по договору

Після отримання запитань, було зроблено запит на те, що він дав на них відповіді, він також це виконав, і ці відповіді теж було записано у змінну reference_answers.

Тепер маючи еталонні відповіді можна переходити до реалізації систем Q&A.

3.2.3 Реалізація функцій отримання відповідей від кожного фреймворку

Для виконання оцінки фреймворків було прийнято рішення взяти базові прості підходи для систем Q&A. Взято рішення було з сайтів використаних платформ. В даному розділі не буде акцентуватися увага на реалізації цих підходів, оскільки він потрібен просто для проведення експерименту.

Реалізації методів для отримання відповіді від системи на основі LlamaIndex і LangChain зображенні відповідно на рис. 3.7 та 3.8.

```python
def query_llamaindex(file_path: str):
    """Отримання відповіді від LlamaIndex системи."""
    documents = SimpleDirectoryReader(input_files=[file_path]).load_data()
    index = VectorStoreIndex.from_documents(documents)
    query_engine = index.as_query_engine()
    return query_engine
```

Рисунок 3.7 – Система Q&A на LlamaIndex

```python
def query_langchain(file_path: str):
    """Отримання відповіді від LangChain системи."""
    loader = PyPDFLoader(file_path)
    docs = loader.load()
    text_splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
    splits = text_splitter.split_documents(docs)
    vectorstore = InMemoryVectorStore.from_documents(documents=splits, embedding=OpenAIEmbeddings())
    retriever = vectorstore.as_retriever()

    llm = ChatOpenAI(model="gpt-4o-mini")
    system_prompt = (
        "Ви - асистент для завдань з відповідями на запитання страхової компанії. "
        "Використовуйте наступні фрагменти знайденого контексту, щоб відповісти на "
        "на запитання. І не пиши, що немає такої інформації в контексті "
        "\n\n"
        "{context}"
    )

    prompt = ChatPromptTemplate.from_messages(
        [
            ("system", system_prompt),
            ("human", "{input}"),
        ]
    )
    question_answer_chain = create_stuff_documents_chain(llm, prompt)
    rag_chain = create_retrieval_chain(retriever, question_answer_chain)
    return rag_chain
```

Рисунок 3.8 – Система Q&A на LangChain

Отже, після реалізації даних методів, можна перейти до виконання самої оцінки та підсумку результатів

3.2.4 Оцінювання фреймворків та підсумок результатів

На основі реалізованих методів, описаних у підрозділі, було реалізовано програму для оцінки відповідей систем Q&A на поставлені для них запитання.

Після завершення реалізації виконаємо оцінку. На рисунку 3.9 зображено приклад оцінки та часу відповіді на запитання 18.

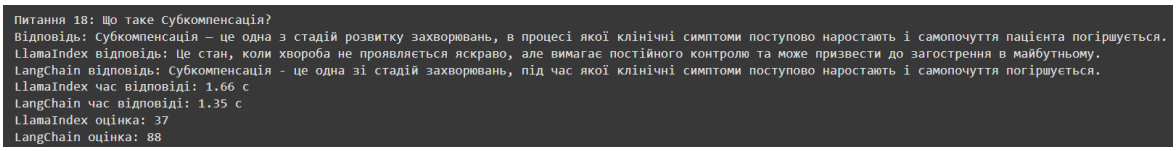


Рисунок 3.9 – Оцінка відповідей на запитання 18

Після закінчення оцінювання відповідей на всі запитання виводиться інформація про середній час відповіді та оцінку кожної системи – рис. 3.10.

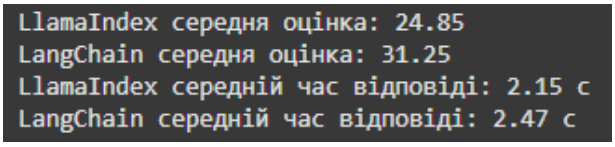


Рисунок 3.10 – Загальна оцінка відповідей на запитання

Як можна побачити з отриманих результатів, якість відповідей системи на основі LangChain має трохи кращі показники по оцінці відповідей на поставлені запитання – 31.25%, проти 24.85% від LlamaIndex. Проте LlamaIndex трішки швидше справлявся із запитами, на 0,32 с . Різниця в часі дуже мала, тому за цим критерієм прийматися рішення не буде, отже залишається лише оцінка відповідей, згідно якої LangChain є кращою. Тому наш аналіз та огляд у підрозділі 3.1 був вірним, це підтверджують отримані результати. На основі цього реалізація чат-боту вже буде остаточно побудована із допомогою платформи LangChain.

3.3 Реалізація оцінки відповідей в залежності від векторного сховища та оптимального розміру розділення фрагментів у RAG-системі

Після вибору платформи для розробки RAG-системи потрібно визначитись з її параметрами та методами реалізації компонентів. Для цього було прийнято рішення провести експерименти для визначення найкращих складових системи.

3.3.1 RAGAs – фреймворк для оцінки системи RAG та її компонентів

RAGAs – це фреймворк, який надає необхідні інструменти для оцінки RAG-системи, як на рівні окремих компонентів, так і для всього процесу загалом.

Особливість RAGAs у тому, що він розроблявся як інструмент для «оцінки без референсів». Це означає, що замість покладання на датасети з людською анотацією, RAGAs використовує LLMs для проведення оцінки. [16]

Для оцінки RAG-системи фреймворк потребує такі дані:

– пошук файлів (File Search) – вбудований інструмент RAG для обробки та пошуку інформації у файлах;

– question: запит користувача, що є вхідними даними для RAG;

– answer: згенерована відповідь від RAG-системи (вихідні дані);

– contexts: контексти, отримані з зовнішнього джерела знань, які використовувалися для відповіді на запит;

– ground_truths: правильна відповідь на запит, створена людиною.

Використання LLM для оцінки без референсів є активною сферою досліджень. Зменшення потреби в даних, створених людиною, робить цей підхід дешевшим і швидшим, проте обговорюються його недоліки, такі як упередженість. Водночас деякі дослідження вже показали багатообіцяючі результати, що свідчить про перспективу такого напряму оцінювання.

Цей фреймворк також розширено для використання метрик, які потребують анотованих даних. Крім того, RAGAs пропонує інструменти для автоматичної

генерації тестових даних. У таблиці 3.2 наведено метрики та їхній опис, а також який компонент вони оцінюють.

Таблиця 3.2 – Метрики оцінки RAGAs

Метрика	**Опис**
Context Precision	Вимірює співвідношення сигналу до шуму в отриманих контекстах. Оцінює, наскільки релевантні контексти були знайдені для відповіді на запит користувача. Формула обрахунку (2.3).
Context Recall	Визначає, чи були отримані всі необхідні контексти для відповіді. Розраховується на основі еталонної відповіді (ground truth). Формула обрахунку (2.5).
Faithfulness	Оцінює фактичну точність відповіді, аналізуючи, скільки тверджень з контекстів є правильними у згенерованій відповіді. Формула обрахунку (2.6).
Answer Relevancy	Вимірює релевантність згенерованої відповіді запиту користувача. Оцінює, наскільки повно та точно відповідь відповідає запиту. Формула обрахунку (2.7).
Answer Semantic Similarity	Визначає семантичну схожість між згенерованою відповіддю та еталонною, аналізуючи їх зміст. Вищий бал означає кращу відповідність за змістом. Формула обрахунку (2.8).

Усі показники масштабуються до діапазону [0, 1], при цьому вищі значення вказують на кращу продуктивність. Для проведення нашого експериментального дослідження будуть використані всі описані раніше метрики.

Перейдемо до побудови програми для подальших досліджень. Першим кроком встановимо бібліотеку RAGAs та імпортуємо перелік метрик, які будемо використовувати, функцію оцінки і бібліотеку для генерації дата-сету – рис. 3.11. Наступним кроком вкажемо всі необхідні дані виконання оцінки – рис. 3.12.

```python
from ragas.metrics import (
    answer_relevancy,
    faithfulness,
    context_recall,
    context_precision,
    answer_similarity
)

from ragas import evaluate
```

Рисунок 3.11 – Імпорт модулів для методу оцінки

```python
data = {
    "question": questions,
    "answer": [str(ans) for ans in answers],  # Перетворюємо відповіді на рядки
    "contexts": contexts,  # Список списків рядків
    "ground_truth": [str(ref) for ref in reference_answers]  # Перетворюємо reference_answers на рядки
}
```

Рисунок 3.12 – Визначення даних для методу оцінки

В даному випадку:

– question: запитання, отримані за допомогою Assistance API, у попередньому підрозділі;

– answer: згенерована відповідь від RAG-системи (вихідні дані);

– contexts: контексти, отримані з PDF-файлу договору, які використовувалися для відповіді на запит;

– ground_truths: еталонні відповіді, отримані за допомогою Assistance API, у попередньому підрозділі.

Пошукова система (File search) буде далі описана, тому що вона буде відрізнятися для оцінки в залежності від компонентів.

3.3.2 Оцінка та вибір векторного сховища

Для порівняння було обрано три популярні, для цього рішення, векторні сховища. Нижче коротко оглянемо кожну з них.

Chroma – це сучасна векторна база даних, спеціально створена для роботи з embeddings. Вона орієнтована на потреби розробників AI-застосунків, таких як пошукові системи, чат-боти та інші додатки, які використовують векторні репрезентації даних. [17]

Основні характеристики:

– простота інтеграції: Chroma надає простий API для роботи з embeddings, що дозволяє легко зберігати та виконувати запити до бази;

– локальне зберігання: база розроблена для роботи в межах одного вузла, що обмежує її масштабованість, але спрощує налаштування і використання для невеликих проєктів;

– робота з мультимодальними даними: підтримує роботу з текстами, зображеннями та іншими форматами даних, що робить її універсальним вибором для багатьох AI-застосунків;

– інтеграція з фреймворками: інтегрується з популярними бібліотеками та інструментами машинного навчання, такими як PyTorch і TensorFlow.

До сфери застосування можна віднести: розробку AI-додатків; прототипування пошукових систем або рекомендаційних моделей; використання у невеликих або локальних проектах, де масштабованість не є критичною.

FAISS (Facebook AI Similarity Search) – це високопродуктивна бібліотека, створена Facebook для ефективного пошуку схожості у великих наборах векторів. Вона спеціально оптимізована для роботи з величезними обсягами даних і активно використовується в наукових і комерційних проєктах. [18]

Основні характеристики:

– методи індексації: FAISS пропонує різні алгоритми індексації, такі як: Flat Index, Inverted Index, HNSW;

– висока продуктивність: підтримує оптимізацію для CPU та GPU, що дозволяє обробляти мільярди векторів значно швидше;

– масштабованість: FAISS розроблена для великих проектів, таких як пошукові системи, аналіз схожості документів і великих масивів зображень;

– гнучкість: підтримує як точний, так і приблизний пошук векторів, дозволяючи балансувати між швидкістю і точністю.

Сфера застосування: пошукові системи великих баз даних; системи рекомендацій для великих платформ; чат-боти підтримки; аналіз схожості текстів або зображень в умовах великих обсягів даних.

LanceDB – це новітня відкрита векторна база даних, яка пропонує високу продуктивність і гнучкість у роботі з мультимодальними даними (тексти, зображення, відео тощо). Вона розроблена для горизонтального масштабування, що дозволяє ефективно обробляти великі набори даних. [19]

Основні характеристики:

– горизонтальне масштабування: LanceDB підтримує масштабування, що дозволяє обробляти дані в розподілених середовищах;

– швидкість роботи: використовує індексацію, що забезпечує високу швидкість пошуку навіть у великих наборах даних;

– підтримка мультимодальних даних: може зберігати embeddings, створені з текстів, зображень, аудіо тощо, і виконувати пошук у різних доменах;

– відкрита архітектура: база даних є відкритою і має широкий спектр інструментів для інтеграції з сучасними AI-платформами;

– простий синтаксис запитів: дозволяє швидко виконувати запити до бази, що робить її зручною для розробників.

До сфери застосування можна віднести: складні AI-застосунки, які потребують роботи з мультимодальними даними; побудова великих, розподілених пошукових систем; обробка та управління embeddings у проєктах, що вимагають масштабованості.

Першим чином було завантажено PDF-файл договору та виконано його розбиття на фрагменти (розмір фрагменту і розмір перекриття взято стандартний для більшості задач). Це необхідно для створення більш точних embeddings і подальшого пошуку. Далі текстові чанги перетворюються на векторні представлення. Векторні представлення використовуються для пошуку схожих фрагментів тексту. За основу генерації відповідей по запиту використано модель gpt-4o-mini. Потім створюється ланцюг для об’єднання знайдених документів і генерації відповіді на основі LLM. Фрагмент коду зображений на рис. 3.13.

Для оцінки кожної векторної бази створено універсальну функцію створення RAG-ланцюга – рис. 3.14.

```
pdf_path = "/content/gdrive/MyDrive/Rag/R1-health_insurance-.pdf"

loader = PyPDFLoader(file_path=pdf_path)
documents = loader.load()

text_splitter = CharacterTextSplitter(
    chunk_size=1000, chunk_overlap=50, separator="\n"
)
docs = text_splitter.split_documents(documents)

embeddings = OpenAIEmbeddings()
llm = ChatOpenAI(model="gpt-4o-mini")

retrieval_qa_chat_prompt = hub.pull("langchain-ai/retrieval-qa-chat")

combine_docs_chain = create_stuff_documents_chain(llm, retrieval_qa_chat_prompt)
```

Рисунок 3.13 – Підготовка даних для RAG-системи

```
def rag_chain(retriever):
  rag_chain = create_retrieval_chain(retriever, combine_docs_chain)
  return rag_chain
```

Рисунок 3.14 – Функція для створення RAG-ланцюга

Функція приймає retriever (інструмент для пошуку в базі векторів) і створює повноцінний RAG-ланцюг. Ланцюг комбінує пошук релевантних фрагментів тексту з генерацією відповіді.

Далі було реалізовано функцію create_ragas_dataset , у якій йде цикл обробки для кожного запиту з questions. Обчислюється час, витрачений на обробку запиту. Виконується збереження часу виконання та відповіді у відповідні списки. З використанням retriever знаходяться документи, релевантні запиту. Витягується текст із знайдених документів та зберігається у вигляді списків текстів для кожного запиту. Формується словник з даними questions, answers, contexts та ground_truth. Їх описано на рис. 3.12. Потім створюється окремий DataFrame для збереження часу виконання запитів. І створюється датасет у форматі, який легко інтегрується з інструментами машинного навчання та аналізу даних. Як можна побачити, було створено окремий DataFrame з часом обробки запиту, щоб надалі теж можна було проаналізувати дану інформацію.

Тепер можна переходити створення векторних сховищ для трьох різних баз даних: Chroma, FAISS та LanceDB. Далі ці сховища використовуються для

створення механізму пошуку (retriever) і ланцюга (chain) для обробки запитів у рамках RAG-системи – рис. 3.15.

```
# Chroma
vectorstore = Chroma.from_documents(docs, embeddings)
chroma_retriever = vectorstore.as_retriever()
chroma_qa_chain = rag_chain(chroma_retriever)

# FAISS
vectorstore = FAISS.from_documents(docs, embeddings)
faiss_retriever = vectorstore.as_retriever()
faiss_qa_chain = rag_chain(faiss_retriever)

# LanceDB
vectorstore = LanceDB.from_documents(docs, embeddings)
lancedb_retriever = vectorstore.as_retriever()
lancedb_qa_chain = rag_chain(lancedb_retriever)
```

Рисунок 3.15 – Створення векторних сховищ

Після виконання всіх попередніх дій, виконується оцінка трьох векторних баз даних шляхом створення тестових дата-сетів, аналізу відповідей на запити користувачів та збору метрик ефективності роботи – рис. 3.16.

```
# Chroma
chroma_dataset, elapsed_time_df = create_ragas_dataset(chroma_qa_chain, chroma_retriever)
chroma_result = evaluate_ragas_dataset(chroma_dataset)
chroma_df = chroma_result.to_pandas()
chroma_df = pd.merge(chroma_df, elapsed_time_df, on="user_input", how="left")

# FAISS
faiss_dataset, elapsed_time_df = create_ragas_dataset(faiss_qa_chain, faiss_retriever)
faiss_result = evaluate_ragas_dataset(faiss_dataset)
faiss_df = faiss_result.to_pandas()
faiss_df = pd.merge(faiss_df, elapsed_time_df, on="user_input", how="left")

# LanceDB
lancedb_dataset, elapsed_time_df = create_ragas_dataset(lancedb_qa_chain, lancedb_retriever)
lancedb_result = evaluate_ragas_dataset(lancedb_dataset)
lancedb_df = lancedb_result.to_pandas()
lancedb_df = pd.merge(lancedb_df, elapsed_time_df, on="user_input", how="left")
```

Рисунок 3.16 – Оцінка трьох векторних баз

Після оцінки векторних баз було отримано датафрейми з раніше вказаними нами метриками та часом відповіді на кожне питання.

Тепер зробимо аналіз отриманих даних, для цього створимо датафрейм зі стовпцями, які нас цікавлять, це будуть наступні стовпці: user_input, reference, answer_relevancy, faithfulness, context_recall, context_precision, answer_correctness,

semantic_similarity, elapsed_time. Для прикладу виведемо дані оцінок з векторною базою Chroma із сформованого датафрейму– рис. 3.17. Такий самий вигляд мають інші сформовані датафрейми.

	user_input	answer_relevancy	faithfulness	context_recall	context_precision	semantic_similarity	elapsed_time
0	Що таке страховий випадок згідно з договором страхування здоров'я?	0.999999	1	0.333333	1	0.896868	2.15149
1	Як подати заявку на відшкодування?	1	0.857143	1	0.916667	0.866087	4.84351
2	Який розмір страхової виплати у випадку діагностування гострого інфаркту міокарду?	0.844852	1	1	0.916667	0.863114	1.14801
3	Які виключення зі страхових випадків передбачено в договорі?	1	0.8	0.571429	0.805556	0.856857	5.55359
4	Чи покриває страховка лікування хронічних захворювань?	0	0.571429	0.666667	0	0.889646	5.73976
5	Що таке Субкомпенсація?	0.993249	1	1	1	0.861256	1.41363
6	Напиши контактні дані для подачі заявки.	0.809338	0.9375	1	0.916667	0.852581	3.03919
7	Що робити, якщо сталася надзвичайна ситуація і потрібна термінова медична допомога?	0.855368	0.666667	0.428571	0	0.904549	4.34913
8	Чи можу я отримати страховку на свою дитину?	0	0.75	0	1	0.877605	1.67825
9	Чи покриває страховка лікування повторного інфаркту міокарду?	1	1	1	1	0.892354	1.18529

Рисунок 3.17 – Сформований датафрейм оцінок для Chroma

Щоб було зручно порівняти результати, було прийнято рішення знайти середні значення отриманих даних і об’єднати їх у один датафрейм.

Отримані результати відображені на рисунку 3.18.

dataset	answer_relevancy	faithfulness	context_recall	context_precision	semantic_similarity	elapsed_time
chroma_df	0.750281	0.858274	0.7	0.755556	0.876092	3.11018
faiss_df	0.965183	0.866667	0.9	0.8	0.890555	2.76104
lancedb_df	0.933294	0.79	0.9	0.816667	0.893194	3.00584

Рисунок 3.18 – Середні значення метрик по кожній з векторних баз

Підсумок. Серед векторних баз найкращі результати демонструє FAISS. Вона забезпечує найвищу точність відповідей, максимальну повноту контексту і високу достовірність, а також має найшвидший час виконання. Це робить її оптимальним вибором для задач, де важливі якість відповідей і продуктивність.

LanceDB також показала хороші результати завдяки високим показникам context recall і semantic similarity . Однак її faithfulness і answer relevancy трохи нижчі, а час виконання довший, що обмежує її ефективність у порівнянні з FAISS.

Chroma має найнижчі показники за ключовими метриками, включаючи answer relevancy і context recall, що свідчить про недостатню точність і повноту відповідей. Водночас вона демонструє пристойний показник faithfulness, але поступається іншим базам за загальною ефективністю.

3.3.3 Оцінка та вибір оптимального розміру розділення на фрагменти тексту

Наступним параметром для дослідження буде chunk size та chunk overlap. Це важливі параметрами в процесі розбиття тексту на фрагменти, що особливо важливо для розробки систем RAG і чат-ботів підтримки з використанням великих мовних моделей.

Chunk size визначає максимальну кількість символів для кожного фрагмента тексту. Цей параметр визначає розмір кожного фрагмента. Менше значення створює менший фрагмент, який допомагає локалізувати певну інформацію. Збільшення значення зменшує кількість фрагментів і зберігає більше контексту.

Chunk overlap визначає кількість символів, які повторюються між двома сусідніми фрагментами. Це дозволяє зберегти зміст і контекст між частинами тексту. [20]

Ці параметри є необхідними для RAG-систем, де основне завдання полягає в отриманні релевантної інформації з бази даних (retriever) і формуванні точних відповідей за допомогою генеративної моделі (generator). Надто малі чанки можуть призвести до втрати зв’язності між частинами тексту, ускладнюючи витяг контексту, тоді як надто великі фрагменти можуть включати зайву інформацію, яка знижує релевантність. Аналогічно, недостатнє перекриття між чанками створює розриви в логічному зв’язку між частинами тексту, а надмірне перекриття збільшує обчислювальні витрати.

Після аналізу векторних баз для RAG-системи, було експериментально визначено, що кращим варіантом у нашому випадку є база FAISS. Тому для оцінки буде використовуватись саме вона.

Програма для оцінки відповідей моделей з попереднього пункту буде дещо схожою, основною відмінністю буде те, що в даному випадку буде вже наявна векторна база FAISS, а буде змінюватися лише розмір chunk size та chunk overlap. Для цього теж було реалізовано функцію evaluate_chunk_size, яка буде приймати параметри змінні параметри, над якими проводиться експеримент. Вона включає в себе структуру точно таку саму, як попередня RAG-система, тільки в дану функцію

було додано одразу реалізації векторного сховища та оцінка системи за тими самими метриками, що і раніше – рис. 3.19. Функція evaluate_ragas_dataset та create_ragas_dataset з попереднього пункту 3.4.2 залишилися незмінними.

```python
def evaluate_chunk_size(chunk_size, chunk_overlap):
    text_splitter = CharacterTextSplitter(
        chunk_size=chunk_size, chunk_overlap=chunk_overlap, separator="\n"
    )
    docs = text_splitter.split_documents(documents)

    embeddings = OpenAIEmbeddings()
    llm = ChatOpenAI(model="gpt-4o-mini")

    retrieval_qa_chat_prompt = hub.pull("langchain-ai/retrieval-qa-chat")

    combine_docs_chain = create_stuff_documents_chain(llm, retrieval_qa_chat_prompt)

    vectorstore = FAISS.from_documents(docs, embeddings)
    retriever = vectorstore.as_retriever()
    rag_chain = create_retrieval_chain(retriever, combine_docs_chain)

    faiss_dataset, elapsed_time_df = create_ragas_dataset(faiss_qa_chain, faiss_retriever)
    faiss_result = evaluate_ragas_dataset(faiss_dataset)
    faiss_df = faiss_result.to_pandas()
    faiss_df = pd.merge(faiss_df, elapsed_time_df, on="user_input", how="left")

    return faiss_df
```

Рисунок 3.19 – Функція evaluate_chunk_size

Тепер розглянемо наступні комбінації chunk size та chunk overlap для експерименту, та пояснення даного тесту – таблиця 3.3.

Таблиця 3.3 – Комбінації chunk size та chunk overlap

№	Chunk size	Chunk overlap	Пояснення
1	500	50	Малий розмір чанків із мінімальним перекриттям. Тестуємо втрату контексту через занадто короткі фрагменти тексту.
2	1000	50	Середній розмір чанків із мінімальним перекриттям. Досліджуємо, чи достатньо контексту при низькому перекритті.
3	1000	200	Середній розмір чанків із помірним перекриттям. Тестуємо баланс між контекстом і швидкістю обробки.

№	Chunk size	Chunk overlap	Пояснення
4	1500	500	Великі чанки з високим перекриттям. Тестуємо, чи підходить для текстів із довгими пунктами договорів, де важливий контекст.

Виконаємо експерименти із таблиці 3.3. Знову для зручності знайдемо середні значення отриманих даних і об’єднаємо їх у один датафрейм. Отримаємо наступний результат – рис. 3.20.

dataset	answer_relevancy	faithfulness	context_recall	context_precision	semantic_similarity	elapsed_time
faiss_500_50_chunk_size_and_overlap_df	0.951175	0.818182	0.8	0.816667	0.892624	3.59871
faiss_1000_50_chunk_size_and_overlap_df	0.950767	0.819972	0.9	0.816667	0.891994	2.96236
faiss_1000_200_chunk_size_and_overlap_df	0.950937	0.790417	0.9	0.780556	0.891495	2.95429
faiss_1500_500_chunk_size_and_overlap_df	0.951006	0.848333	0.8	0.816667	0.892098	3.8678

Рисунок 3.20 – Середні значення метрик оцінки параметрів поділу

Щоб краще було видно результати, транспонуємо отриману таблицю середніх значень і запишемо отримані результати у таблицю 3.4.

Таблиця 3.4 – Середні значення метрик оцінки параметрів поділу

Комбінація	faiss_500_50	faiss_1000_50	faiss_1000_200	faiss_1500_500
Answer Relevancy	0.7259	0.7175	0.7416	0.7367
Faithfulness	0.8956	0.8211	0.8601	0.8702
Context Recall	0.0000	0.0000	0.0000	0.0000
Context Precision	0.1000	0.1000	0.1000	0.1000
Semantic Similarity	0.7167	0.7161	0.7157	0.7165
Elapsed Time(c)	1.8985	2.1781	1.9624	4.1275

На основі отриманих результатів зробимо висновок по кожній з комбінацій:

1. faiss_500_50: дрібні чанки з мінімальним перекриттям забезпечують чудову достовірність і коректність відповідей; швидкість обробки найкраща серед

усіх варіантів; недолік контексту призводить до меншої релевантності, що може бути критичним для роботи зі структурованими документами, як договори;

2. faiss_1000_50: середній розмір чанків із мінімальним перекриттям швидко працює, але втрата контексту суттєво знижує релевантність і достовірність відповідей; ця комбінація не оптимальна для вашого сценарію, оскільки не задовольняє функціональні вимоги до якості відповідей;

3. faiss_1000_200: балансує між швидкістю, релевантністю та достовірністю; забезпечує найвищу релевантність із високою достовірністю; легко відповідає нефункціональним вимогам до швидкодії й забезпечує оптимальне збереження контексту, що особливо важливо для страхових договорів;

4. faiss_1500_500: великі чанки із значним перекриттям добре підходять для забезпечення контексту, але це значно збільшує час обробки; ця комбінація може бути корисною для глибоких запитів, але не є оптимальною для системи, де час відповіді має бути мінімальним.

Підсумок. На основі аналізу, було визначено, що комбінація chunk_size=1000 та chunk_overlap=200 є найкращим вибором для розроблюваної системи RAG для побудови чат-бота. Це обумовлено тим, такі параметри надають змогу забезпечити найвищу релевантність, що в свою чергу дозволить чат-ботові відповідати на запити максимально точно й відповідно до запитів користувачів. Він показує високу достовірність, а це забезпечує відповіді, які базуються на фактичних даних із документів, що є важливим для розроблюваної системи. І один з основних факторів, що така комбінація має швидкий час обробки, що забезпечуючи відповідь менш ніж за 2 секунди. Не можна не виділити також гарне збереження контексту, це надає змогу збереження необхідного контексту без надмірного дублювання, що особливо важливо для страхових договорів із чіткою структурою. Отже ця комбінація найкраще балансує між якістю відповідей, швидкістю обробки та відповідністю вимогам системи.

Висновки до розділу 3

У цьому розділі було зосереджено увагу на дослідженні та виборі оптимальних методів і технологій для подальшої реалізації RAG-системи, що використовується в чат-боті. Основною метою стало визначення найкращого фреймворку, векторної бази даних та параметрів поділу тексту, які забезпечать високу якість, релевантність відповідей і швидкість обробки.

Було проведено аналіз фреймворків LlamaIndex та LangChain. На основі порівняльної таблиці та практичної перевірки через створення базової системи Q&A визначено, що LangChain є більш підходящим завдяки своїй модульній архітектурі, підтримці складних робочих процесів та інтеграції з іншими AI-інструментами.

Далі оцінено три популярні векторні сховища – Chroma, FAISS та LanceDB. Результати показали, що FAISS забезпечує найкращі результати завдяки високій продуктивності та точності, що робить його оптимальним вибором для RAG-системи.

Також було протестовано параметри поділу тексту на фрагменти (chunk size і chunk overlap). Результати експерименту показали, що параметри chunk_size=1000 і chunk_overlap=200 забезпечують оптимальний баланс між збереженням контексту, швидкістю роботи та якістю відповідей.

Таким чином, у цьому розділі визначено ключові компоненти для реалізації RAG-системи: фреймворк LangChain, векторну базу FAISS і оптимальні параметри поділу тексту. Отримані результати закладають основу для створення ефективної системи підтримки для страхової компанії.

4 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЧАТ-БОТА ПІДТРИМКИ

4.1 Схема архітектури системи RAG у чат-боті підтримки

Для розробки інформаційної системи підтримки (чат-бота) було використано наступну схему архітектури системи RAG – рис. 4.1 (ресурс – [21]).

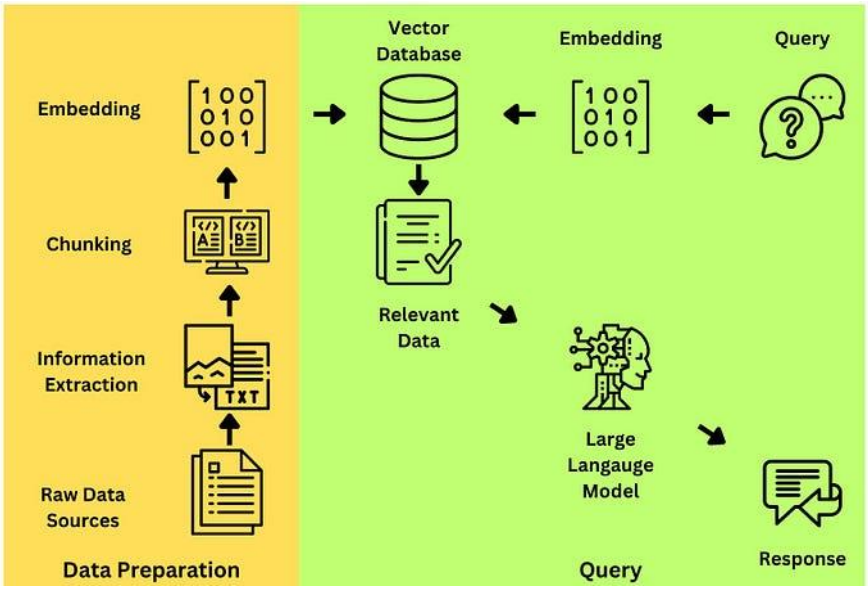


Рисунок 4.1 – Обрана схема архітектури системи RAG

Схема роботи системи RAG, яка представлена на рисунку 4.1, демонструє процес обробки даних і генерації відповідей у системі, що використовує поєднання LLM та баз даних із векторними представленнями.

Жовта частина – підготовка даних (Data Preparation):

– Raw Data Sources (джерела даних): дані надходять із різних джерел, таких як текстові файли, зображення або інші документи;

– Information Extraction (виділення інформації): з цих джерел вилучається релевантна інформація для подальшої обробки;

– Chunking (розбиття даних): виділені дані розбиваються на невеликі логічні частини, зручні для аналізу;

– Embedding (перетворення у векторні представлення): кожна частина даних кодується у векторну форму для зберігання у векторній базі даних. Цей процес дозволяє знаходити подібності між запитами і збереженими даними.

Зелена частина – обробка запиту (Query Processing):

– Query (запит): користувач надсилає запит до системи;

– Embedding (перетворення запиту): запит перетворюється у векторну форму, аналогічну до представлених даних;

– Vector Database (векторна база даних): використовуючи векторну форму запиту, система шукає релевантні частини даних у базі, де як результат – набір релевантних даних;

– Relevant Data (релевантні дані): Знайдені дані передаються для подальшої обробки;

– Large Language Model (велика мовна модель): LLM отримує релевантні дані та використовує їх разом із запитом для генерації точної та персоналізованої відповіді;

– Response (відповідь): згенерована моделлю відповідь повертається користувачеві.

Для розробки системи підтримки чат-бота страхової компанії була обрана архітектура RAG (рис. 4.1), оскільки вона ідеально відповідає потребам галузі. Основною перевагою є здатність швидко й ефективно обробляти великі обсяги даних, такі як страхові договори, поліси та умови обслуговування. Використання векторної бази для пошуку релевантної інформації дозволяє значно скоротити час обробки запитів, забезпечуючи точність відповідей на основі реальних документів. Крім того, поєднання цих даних із можливостями генерації відповіді за допомогою LLM гарантує персоналізований підхід і високу якість комунікації, що є критичним для клієнтів страхових компаній.

Ця архітектура також відзначається гнучкістю та масштабованістю, що дозволяє інтегрувати систему з іншими корпоративними інструментами, такими як CRM або ERP, а також адаптувати її до змінних обсягів даних і запитів. Відповідність регуляторним вимогам, таким як GDPR, додає рівень довіри клієнтів,

а автоматизація типових запитів значно знижує операційні витрати та навантаження на співробітників служби підтримки. Таким чином, використання такої структури RAG забезпечує створення інноваційного рішення, яке відповідає потребам клієнтів і підвищує ефективність роботи страхової компанії.

4.2 Архітектура проєкту розроблюваної системи служки підтримки страхової компанії

Для розробки даного проєкту, було вирішено використати багатощарову архітектуру з реалізацією у модульному стилі.

У даному випадку, *багатошарова архітектура* дозволяє розділити функціональні частини системи на окремі логічні шари, кожен з яких виконує свої завдання. Це забезпечує ряд переваг, коротко розглянемо кожну з них. [22]

Розподіл обов'язків. Кожен шар системи виконує чітко визначені функції, що зменшує взаємозалежність між компонентами.

Масштабованість. Завдяки розділенню системи на шари, можна додавати або змінювати функціональність у певному шарі без впливу на інші. Наприклад, оновлення алгоритмів для обробки текстів договорів не впливає на роботу інтерфейсу чи бази даних.

Можливість розподіленої реалізації. Кожен шар може працювати на окремому сервері чи використовуватися в хмарному середовищі, що забезпечує високу продуктивність і стійкість до навантажень.

Спрощення тестування. Окреме тестування кожного шару спрощує виявлення помилок і знижує витрати часу на усунення проблем.

Підтримка стандартів. Багатошаровий підхід підтримує реалізацію сучасних стандартів захисту даних (наприклад, відповідність GDPR у шарі даних).

Схема міжрівневої архітектури відображає структуру програмної системи, яка поділена на декілька логічних рівнів, кожен із яких виконує чітко визначені завдання. Ця схема візуально демонструє, як дані, логіка та функціональні

компоненти взаємодіють між різними рівнями. Міжрівневу ахітектуру розроблюваного проєкту наведено на схемі у Додаток Б.

Схема міжрівневої архітектури чат-бота підтримки, зображена на кресленні, демонструє три основних рівні:

1. Presentation Layer (PL) – рівень представлення: Цей рівень відповідає за взаємодію користувачів із системою.

2. Business Logic Layer (BLL) – рівень бізнес-логіки: на цьому рівні реалізуються основні алгоритми і правила обробки даних. BLL відповідає за виконання ключових функцій системи, забезпечуючи зв’язок між рівнем представлення та рівнем доступу до даних.

3. Data Access Layer (DAL) – рівень доступу до даних: DAL здійснює безпосередню роботу з базами даних і забезпечує доступ до таких даних, як: список договорів і файли договорів, лог-файли та історія взаємодій, векторні залежності текстів і релевантні фрагменти документів. Цей рівень відповідає за безпечний, ефективний і структурований доступ до інформації, яка зберігається в базах даних.

Ця архітектура добре підходить для розробки системи підтримки чат-бота, оскільки дозволяє ефективно обробляти великі обсяги даних, забезпечувати інтеграцію з різними сервісами й адаптуватися до змін вимог бізнесу.

Модульність забезпечує можливість поділу системи на незалежні компоненти, що значно спрощує розробку, розширення та обслуговування. Це критично важливо для системи чат-бота підтримки, де можливі численні оновлення та інтеграції. Основні переваги модульності теж описані нижче. [23]

Гнучкість: легкість у додаванні нових функцій, таких як підтримка додаткових форматів файлів або нових каналів комунікації.

Повторне використання компонентів: модулі, такі як обробка текстів договорів або інтеграція з LLM, можуть бути використані в інших проєктах компанії.

Локалізація змін: зміни в одному модулі не впливають на інші.

Сприяння розподіленій розробці: можливість одночасної роботи над різними модулями кількома командами розробників.

Швидкість усунення помилок: окремі модулі легше тестувати та оновлювати, що пришвидшує розробку і підтримку системи.

Для відображення модульності у архітектурі розроблюваної ситеми було обрано структурну схему архітектури.

Структурна схема архітектури чат-бота підтримки складається з кількох компонентів і модулів, кожен із яких виконує певну функціональну роль у забезпеченні роботи системи. Основні компоненти та їх модулі взаємодіють між собою, утворюючи багатошарову, модульну структуру. Нижче наведено опис основних компонентів і модулів:

1. компонент: Interface: відповідає за взаємодію з користувачем;

2. компонент: Contracts Process: обробляє інформацію про договори;

3. компонент: Files Process: відповідає за обробку файлів договорів;

4. компонент: Vector Process: забезпечує роботу з векторним представленням тексту;

5. компонент: LLM Process: працює із системою генерації відповідей на основі LLM;

6. бази даних: DB – зберігає текст договорів, списки договорів і метаінформацію; LogDB – містить інформацію про взаємодію користувачів, логи запитів і відповіді; VectorDB – відповідає за зберігання векторних представлень тексту договорів.

Більш детальний опис, відображено на побудованій структурній схемі архітектури у Додаток В.

Структура системи базується на модульному підході з чітким розподілом функціональності між компонентами. Такий підхід забезпечує масштабованість, гнучкість і можливість незалежного оновлення окремих частин системи. Завдяки взаємодії модулів через стандартизовані інтерфейси, система може легко адаптуватися до нових потреб бізнесу.

Отже на основі переваг багатошарової архітектури та модульності, можна виділити те, що в комбінації вони надіють ще більше переваг, і доповнюють один одного. Вони надають переваги та можливості, які описані нижче.

Зручність інтеграції: модульний підхід дозволяє легко інтегрувати чат-бот із CRM, векторними сховищами та іншими IT-системами, які вже використовуються в страхових компаніях.

Підтримка складних функцій: обробка великих текстів договорів, пошук інформації, інтеграція з LLM – це все реалізується через окремі модулі, що спрощує управління складними процесами.

Швидка адаптація до змін: к страхових компаніях часто змінюються законодавчі вимоги. Завдяки модульності можна швидко адаптувати систему до нових умов, наприклад, змінити функції обробки персональних даних.

Надійність і масштабованість: високе навантаження, викликане кількома одночасними запитами від користувачів, може оброблятися за рахунок розподілу функцій між різними шарами. Це забезпечує стабільність системи.

Розвиток у майбутньому: багатошаровість і модульність дозволяють масштабувати продукт, додаючи нові функції або адаптуючи систему для інших галузей (наприклад, фінансів чи юриспруденції).

Як підсумок, багатошарова архітектура з модульною реалізацією забезпечує зручність, масштабованість і гнучкість системи чат-бота для підтримки страхової компанії. Вона ідеально підходить для задач, пов'язаних із обробкою великої кількості даних, інтеграцією з іншими сервісами та адаптацією до мінливих умов ринку. Такий підхід гарантує стійкість, високу продуктивність та довгострокову конкурентоспроможність системи.

4.3 Діаграма прецедентів (Use Case Diagram)

Діаграма прецедентів (Use Case Diagram) – це графічне представлення функціональних вимог до системи, яке показує взаємодію між користувачами (акторами) та її основними функціями (прецедентами). Вона є складовою методології об'єктно-орієнтованого проєктування, зокрема мови моделювання UML. Діаграма зображує:

– акторів – зовнішні системи, користувачів або інші суб'єкти, які взаємодіють із системою;

– прецеденти – функції або дії, які система виконує на запит акторів;

– зв'язки між акторами і прецедентами – ілюструють, які дії виконуються і ким.

Діаграма прецедентів фокусується на тому, «що» повинна робити система, а не «як» це буде реалізовано. Дана схема для інформаційної системи служби підтримки страхової компанії з використанням LLM зображена на Додаток Г.

Зображена діаграма представляє інформаційну систему підтримки, яка функціонує через чат-бот для обслуговування клієнтів страхової компанії. На діаграмі зображені основні актори, їхні ролі та взаємодії з системою через прецеденти (виконувані дії).

Актори:

1. користувач – клієнт страхової компанії, який взаємодіє з чат-ботом для отримання інформації чи послуг;

2. служба підтримки – співробітники, які забезпечують додаткову консультацію та моніторинг роботи системи;

3. адмін (адміністратор) – адміністратор системи, відповідальний за управління функціоналом, перевірку логів та внесення змін у параметри.

Даний тип діаграм є важливим при проєктуванні інформаційних систем, оскільки він визначає функціональні вимоги, взаємодії користувачів із системою та ключові бізнес-процеси. Це допомагає уникнути пропусків у вимогах, полегшує комунікацію між розробниками і зацікавленими сторонами, а також сприяє плануванню, тестуванню та масштабуванню системи.

4.4 Діаграма послідовностей (Sequence Diagram)

Діаграма послідовності (Sequence Diagram) – це один із типів діаграм UML, який відображає послідовність взаємодій між об'єктами або компонентами системи у часовій перспективі. Вона показує, як повідомлення передаються між різними

учасниками (акторів, модулів чи компонентів) для виконання певного процесу. Основними елементами діаграми є учасники (актори чи компоненти), повідомлення між ними та часові лінії.

Діаграма послідовності для інформаційної системи служби підтримки страхової компанії з використанням LLM зображена на Додаток Д.

Вона ілюструє взаємодії між користувачем, адміністратором і внутрішніми компонентами системи для обробки запитів. Включає наступні ключові етапи:

1. початок роботи з ботом: користувач ініціює роботу з ботом через інтерфейс; чат-бот запитує дані з конфігурації для завантаження списку доступних договорів; дані повертаються, і користувач отримує список договорів для вибору;

2. вибір договору: користувач вибирає договір, який передається до чат-бота; система перевіряє правильність вибору договору, звертаючись до конфігурації та сховища даних; договір записується в лог-файл, і підтвердження передається користувачеві;

3. запит інформації по договору: користувач надсилає запит щодо інформації, пов'язаної з обраним договором; чат-бот ініціює пошук релевантних фрагментів тексту договору через систему векторного пошуку; фрагменти тексту передаються до мовної моделі (LLM), яка генерує відповідь;

4. вивід відповіді: згенерована відповідь повертається користувачу; відповідь записується в лог-файл для подальшого аналізу активності;

5. адміністративна взаємодія: адміністратор оновлює параметри конфігурації або завантажує нові договори у PDF-форматі; договори проходять через процесор контрактів, векторне сховище та завантажуються в систему; усі зміни логуються для подальшого моніторингу.

При проєктуванні інформаційних систем, таких як чат-боти, діаграма послідовності дозволяє зрозуміти, як саме будуть оброблятися запити користувачів, як компоненти системи (API, бази даних, модулі обробки) взаємодіють між собою та як забезпечується правильність і швидкість виконання запитів. Це важливо для створення ефективної, надійної та масштабованої системи.

4.5 Діаграма станів (State Diagram)

Діаграма станів (State Diagram) – це один із видів UML-діаграм, що використовується для моделювання динамічної поведінки системи. Вона ілюструє, у яких станах може перебувати об'єкт системи, як він переходить з одного стану до іншого залежно від подій, що відбуваються, і які дії виконуються під час цих переходів.

Дана діаграма для розроблюваної системи чат-бота відображена у Додаток Е.

Вона показує життєвий цикл функціонування чат-бота у системі підтримки страхових компаній. Діаграма ілюструє, як система переходить між різними станами залежно від отриманих подій чи виконаних дій.

Вона зображає ключові аспекти:

– перевірка валідності: на кожному етапі перевіряється коректність введених даних (наприклад, вибір договору);

– логування: усі критичні дії (вибір договору, запити, відповіді) логуються для забезпечення аудиту та аналізу;

– асинхронна взаємодія: робота з векторними сховищами, LLM і запитами виконуються у фоновому режимі, забезпечуючи швидкий відгук користувачеві;

– модульна архітектура: кожен стан відповідає за конкретну частину функціоналу системи, що дозволяє розширювати або модифікувати її без порушення роботи інших частин.

Діаграми стану є важливим інструментом при розробці таких проектів, як інформаційні системи підтримки чат-ботів. Це пов'язано з тим, що вони дають вам чітке уявлення про те, як система реагує на різні події та як відбуваються переходи між станами під час роботи. Це допомагає візуалізувати динаміку системи та показати всі можливі стани об'єктів та дії, які будуть виконуватися під час зміни. Крім того, ця діаграма може бути використана для приведення функцій у відповідність з вимогами системи, щоб врахувати всі важливі етапи, такі як обробка запитів і перевірка даних. Це значно полегшує планування, реалізацію та подальшу підтримку проекту.

4.6 Діаграма потоків даних (Data Flow Diagram)

Діаграма потоків даних (Data Flow Diagram, DFD) – це графічне представлення потоків інформації всередині системи. Вона показує, як дані переміщуються між компонентами системи, звідки вони надходять, як обробляються, куди передаються і зберігаються. Основними елементами DFD є зовнішні сутності (користувачі або інші системи), процеси, сховища даних і самі потоки даних.

DFD для інформаційної системи підтримки зображено на Додаток Ж.

Діаграма демонструє ключові потоки даних між користувачем, чат-ботом, сховищами даних, LLM та адміністративними модулями, підкреслюючи високий рівень автоматизації і точності системи. Вона забезпечує наочність основних етапів роботи, дозволяючи ефективно аналізувати і вдосконалювати систему.

Такий тип діаграм забезпечує цілісне розуміння роботи системи, дозволяючи виявити, як дані проходять через всі етапи її функціонування. Вона допомагає розробникам, аналітикам ідентифікувати слабкі місця, залежності, дублювання процесів і потенційні ризики. Завдяки DFD можна легко узгодити вимоги користувачів і технічні аспекти реалізації. У проєктах, які включають складну взаємодію між компонентами, як-от інтеграція чат-бота з базами даних і штучним інтелектом, DFD стає критично важливим для забезпечення ефективності, точності та оптимізації системи.

Висновки до розділу 4

У цьому розділі розглянуто проектування інформаційної системи підтримки для чат-бота страхової компанії. Було описано обрану архітектуру RAG, яка поєднує можливості векторних баз даних і LLM. Цей підхід забезпечує ефективний пошук, персоналізовані відповіді, гнучкість та відповідність сучасним регуляторним вимогам.

Система побудована за багатошаровою архітектурою, що включає рівні представлення, бізнес-логіки та доступу до даних. Така структура забезпечує чіткий розподіл функцій, зручність масштабування та адаптацію до змін. Модульний підхід дозволяє легко оновлювати функціональність, інтегрувати нові можливості та підтримувати систему. Основні модулі займаються обробкою договорів, інтеграцією з векторними базами, генерацією відповідей LLM та взаємодією з користувачами.

За допомогою діаграм UML детально змодельовано функціонал системи. Діаграми прецедентів показують основні взаємодії користувачів із системою. Діаграми послідовностей ілюструють процеси обробки запитів, починаючи з вибору договору і завершуючи генерацією відповіді. Діаграми станів демонструють життєвий цикл роботи системи, а потоки даних описують рух інформації між компонентами, такими як сховища даних, модулі обробки та LLM.

Результатом цього розділу є обґрунтованість обраної архітектури та її відповідність потребам страхової компанії. Багатошаровий і модульний підходи дозволяють створити гнучку, масштабовану і надійну систему. Вона автоматизує обробку великої кількості даних, забезпечує персоналізовану взаємодію з клієнтами та відповідає бізнес-вимогам, гарантуючи високу продуктивність і стабільність.

5 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЧАТ-БОТА ПІДТРИМКИ

5.1 Опис технологій та засобів розробки ПЗ системи

В даному підрозділі розглянемо, що потрібно для розробки чат-бота страхової підтримки з використанням системи RAG.

Для розробки даної системи було обрано мову програмування ***Python***. Такий вибір було зроблено через його простоту, універсальність і багатий вибір бібліотек. Завдяки зрозумілому синтаксису та легкій інтеграції з різними API, Python дозволяє швидко реалізувати всі ключові функції бота для обробки команд і запитів, а також для генерації відповідей. Крім того, мова має потужні інструменти для обробки тексту, як-от бібліотека для роботи з PDF-файлами, що робить можливим швидкий пошук і аналіз текстів договорів.

Також Python забезпечує розвинену екосистему бібліотек та активну спільноту, що надає змогу спростити вирішення технічних завдань і дозволити зосередитися на функціоналі бота. Універсальність Python, його сумісність із різними платформами та можливість швидкого прототипування зробили його оптимальним вибором для реалізації цього проєкту.

Середовищем розробки для створення чат-бота підтримки було обрано ***Visual Studio Code (VS Code)*** через його зручність, функціональність та адаптованість до потреб Python-розробки. VS Code має інтуїтивно зрозумілий інтерфейс, який спрощує роботу з великими проєктами, дозволяючи легко орієнтуватися в структурі файлів і модулів. Легка інтеграція з віртуальним середовищем (venv) дає змогу ізолювати залежності проєкту та забезпечити стабільну роботу з бібліотеками. Завдяки потужним інструментам відладки та інтерактивним терміналам VS Code значно спростив процес розробки, тестування й оптимізації коду, роблячи його ідеальним вибором для реалізації цього проєкту.

Для реалізації методу RAG було обрано фреймворк ***LangChain***. Чому прийнято саме таке рішення, описано у 3 розділі, у підрозділі 3.2.

Для завантаження тексту договорів у систему було обрано бібліотеку ***pdfplumber*** завдяки її здатності точно вилучати текст із PDF-документів, включаючи документи зі складною структурою, такими як багатосторінкові договори, що містять таблиці [24]. Вона забезпечує збереження форматування та ефективно обробляє текст у вигляді таблиць, що часто зустрічаються у договорах. Простий API бібліотеки дозволяє інтегрувати її з іншими компонентами системи, такими як LangChain, для подальшої обробки тексту. Завдяки стабільності, гнучкості та активній підтримці, pdfplumber є оптимальним вибором для забезпечення точного й швидкого завантаження тексту договорів у систему.

Інструментом поділу документу на фрагменти для подальшого векторного представлення було використано інструмент ***CharacterTextSplitter***. Він здатен ефективно розбивати великі текстові документи на менші частини заданого розміру, зберігаючи контекст через накладання тексту. Ця властивість особливо важлива для обробки текстів страхових договорів, які часто мають складну структуру з розділами, пунктами та підпунктами.

Векторна база у чат-боті використовується для зберігання, індексації та пошуку текстової інформації, яка представлена у вигляді векторів – числових значень, що відображають семантичний зміст тексту. На основі дослідів з різними базами у розділі 3, пункті 3.3.2, було визначено, що кращим рішенням як векторне сховище буде ***FAISS*** .

Для використання великих мовних моделей використано ***OpenAI API***. Це API було обрано для розробки чат-бота завдяки його здатності працювати з природною мовою, генерувати контекстуальні відповіді та створювати векторні представлення тексту (embeddings) [25]. Моделі OpenAI дозволяють не лише аналізувати текстові запити, але й перетворювати текстові фрагменти на вектори, які відображають їхній семантичний зміст. Це забезпечує тісну інтеграцію з векторною базою, дозволяючи боту швидко знаходити релевантну інформацію в текстах договорів і створювати персоналізовані відповіді, навіть якщо формулювання запитів не співпадають з текстами у документах.

Окрім того, використання OpenAI API для embeddings підвищує точність пошуку, адже модель враховує семантичний зміст тексту, а не лише ключові слова. Важливо зазначити, що API є платним, і вартість залежить від обсягу запитів та використання ресурсів моделі.

У роботі обрано модель ***GPT-4o-mini*** як рішення для проєкту завдяки її швидкості, точності та економічності. Вона забезпечує якісну обробку запитів, зберігаючи стабільність у роботі з контекстом і мінімізуючи витрати ресурсів. Завдяки балансу продуктивності та ефективності GPT-4o-mini є оптимальним вибором для системи підтримки на основі RAG.

У якості інтерфейсу взаємодії користувача та розробленої системи для даного проєкту було обрано ***Telegram API***, які дозволяють створити ефективний інструмент для автоматизованої підтримки клієнтів. Telegram став обраним рішенням для демонстрації функціоналу завдяки своїй доступності, зручності інтеграції та популярності серед користувачів.

Водночас важливо підкреслити, що Telegram є лише одним із можливих інтерфейсів, і реалізована система залишається достатньо гнучкою для інтеграції в інші платформи, такі як веб-додатки, мобільні застосунки чи інші месенджери. Основний акцент роботи було зроблено не на деталях реалізації інтерфейсу, а на створенні ефективної системи, здатної працювати з текстами страхових договорів, швидко знаходити релевантну інформацію та формулювати зрозумілі відповіді на запити клієнтів. Це забезпечує відповідність системи поставленим функціональним і нефункціональним вимогам.

Telegram як платформа для чат-бота надає значні переваги, зокрема доступ до великої аудиторії, підтримку миттєвого обміну повідомленнями, інтеграцію з Telegram Bot API, а також можливість використання текстових команд, кнопок та інтерактивних елементів для зручності користувачів. Крім того, Telegram забезпечує високий рівень безпеки через шифрування даних, що є важливим при роботі з конфіденційною інформацією.

Використання описаних інструментів і технологій забезпечило зручність, ефективність і гнучкість в процесі розробки. Вони дозволили автоматизувати

рутинні завдання, спростити інтеграцію компонентів системи і забезпечити швидке впровадження змін і масштабування проекту. Це дозволило оптимізувати процес розробки і сфокусуватися на створенні високоякісних функцій, які могли б вирішувати завдання і відповідати вимогам користувачів.

5.2 Структура проєкту системи чат-боту підтримки

Розроблена система чат-боту підтримки страхової компанії, як вже було описано у розділі 4, має багатошарову архітектуру, реалізовану через модульну організацію коду.

Для реалізації було використано «пакетно-модульну структуру» (Package-Module Structure). Вона є стандартним підходом для організації середніх і великих Python-проєктів. Структура проєкту системи відображена на рис.5.1.

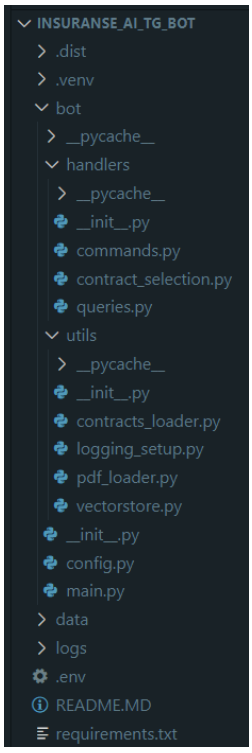


Рисунок 5.1 – Структура проєкту чат-боту підтримки

Тепер опишемо всі елементи структури, та їхнє призначення:

1. Директорія *bot/* : це основний пакет проєкту, який містить усі ключові компоненти логіки чат-бота.

Файли:

– *__init__.py*: ініціалізує пакет bot;

– *config.py*: зберігає глобальні налаштування проєкту, такі як: токен Telegram API, шляхи до даних, ліміти кешу, ключі OpenAI API; дозволяє централізовано змінювати конфігурацію проєкту;

– *main.py*: основна точка входу в програму; виконує ініціалізацію Telegram-бота, налаштування логування та реєстрацію обробників запитів; організовує взаємодію між компонентами.

2. Директорія *bot/handlers/* : містить модулі, які обробляють запити користувачів.

Файли:

– *__init__.py*: ініціалізує пакет handlers;

– *commands.py*: обробляє базові команди бота, такі як /start і /help ; реалізує логіку привітання користувачів, надання загальної інформації про бота;

– *queries.py*: обробляє текстові запити користувачів до системи; інтегрується з векторною базою та OpenAI API для пошуку інформації й генерації відповідей; враховує контекст попередніх запитів для забезпечення безперервного діалогу;

– *contract_selection.py*: обробляє вибір договору користувачем із запропонованого списку; зберігає вибраний договір у контексті взаємодії для подальшої роботи.

3. Директорія *bot/utils/* : містить утилітарні модулі, які забезпечують допоміжні функції для основної роботи системи.

Файли:

– *__init__.py*: ініціалізує пакет utils;

– *contracts_loader.py*: завантажує список договорів із JSON-файлу; дозволяє легко оновлювати список договорів без змін у коді;

– *logging_setup.py*: встановлює налаштування логування: логи зберігаються у файлі й виводяться в термінал; фіксуються всі дії бота, включаючи запити та відповіді.

– *pdf_loader.py*: відповідає за завантаження тексту з PDF-файлів договорів; розбиває текст на частини для подальшого оброблення;

– *vectorstore.py*: створює та керує векторною базою даних; використовується для пошуку релевантної інформації в текстах договорів; контролює кешування векторних сховищ.

4. Директорія *data/* : містить дані, які використовує бот.

Файли:

– *contracts.json*: список договорів із назвами та шляхами до відповідних PDF-файлів; забезпечує зручне налаштування та оновлення договорів;

– *PDF-файли договорів*: містять тексти страхових договорів, які обробляє бот.; використовуються як джерело даних для пошуку інформації.

5. Директорія *logs/* : містить логи роботи бота.

Файли:

– *bot_activity.log*: зберігає записи про всі дії бота, такі як: підключення користувачів, вибір договорів, запити та відповіді; використовується для моніторингу та діагностики.

6. Інші файли:

– *.env*: зберігає конфіденційні дані, такі як: токен Telegram API, ключ OpenAI API; забезпечує безпеку, ізолюючи ці дані від основного коду;

– *requirements.txt*: містить перелік бібліотек, необхідних для роботи проєкту; використовується для швидкого розгортання середовища розробки;

– *README.md*: документація проєкту; містить інструкції з налаштування, запуску та використання системи.

Організації структури таким чином допомагає забезпечити: модульність (кожен елемент виконує окрему роль, що спрощує розробку та підтримку), гнучкість (легко додавати нові функції або оновлювати існуючі), зручність (чіткий розподіл файлів і директорій дозволяє швидко орієнтуватися в коді), масштабованість (можливість розширювати систему без порушення її основної логіки).

5.3 Опис даних для розширення контексту LLM

Джерелом нових знань у наявній системі стали договори страхування. Дані договори було взято із сайту однієї з провідних страхових компаній в Україні – СК «UNIVERSALNA». На їхньому сайті у вкладці Страхові продукти [15] є наявна інформація про всі договори компанії. Для реалізації чат-боту підтримки, було обрано 11 договорів (що не є межею для системи, але цієї кількості буде достатньо для демонстрації ефективності роботи розробленої системи підтримки), які є одними з найпоширеніших серед клієнтів не тільки у цій страховій компанії.

Було обрано наступні договори:

1. страхування здоров'я: покриття різноманітних захворювань;
2. захист на кожен день: покриття від нещасних випадків;
3. захист тварин: страхування домашніх улюбленців від хвороб чи травм;
4. захист від шахрайства: компенсація збитків від фінансового шахрайства;
5. автоцивілка-ОЦВ: обов'язкове страхування відповідальності водіїв;
6. КАСКО-ПРИВАТ: захист автомобіля від пошкоджень чи крадіжки;
7. TOP DRIVER: КАСКО, але має менше покриття;
8. страхування майна від військових ризиків: захист від військових загроз;
9. страхування вантажів: захист вантажів під час транспортування;
10. страхування предметів іпотеки: страхування заставного майна;
11. страхування майна фізичних осіб: захист приватного майна від ризиків.

Ці договори було поміщено у директорію *data/* . Для подальшої зручності завантаження та оновлення дані договори та шлях в проєкті до них було записано у форматі JSON у файл *contracts.json* .

5.4 Опис реалізації основних компонентів та функціоналу системи

У даному підрозділі зробимо огляд реалізації основних компонентів системи в тому порядку, в якому, які забезпечують функціонування системи. Також відобразимо блок-схеми роботи цих елементів.

5.4.1 Модуль вибору договору

Модуль вибору договору визначає, який саме договір було обрано користувачем із запропонованого списку, для подальшої роботи з ним. Він зберігає інформацію про вибраний договір у контексті взаємодії, що дозволяє подальші дії (наприклад, обробку запитів) виконувати з урахуванням цього вибору. Його реалізація представлена на рис. 5.2.

```python
async def handle_contract_selection(update: Update, context: ContextTypes.DEFAULT_TYPE):
    user = update.effective_user
    contracts = load_contracts()
    selected_contract = update.message.text
    if selected_contract in contracts:
        context.user_data["selected_contract"] = contracts[selected_contract]
        context.user_data["dialog_history"] = []
        context.user_data["first_query"] = True
        logger.info(f"{user.username} обрав договір: {selected_contract}.")
        await update.message.reply_text(f"Ви обрали '{selected_contract}'. Запитайте, що вас цікавить.")
    else:
        await update.message.reply_text("Будь ласка, оберіть правильний договір.")

def register_contract_handlers(application):
    application.add_handler(MessageHandler(filters.Regex(f"^({'|'.join(load_contracts().keys())})$"), handle_contract_selection
```

Рисунок 5.2 – Модуль вибору договору

Функція обробки вибору договору – handle_contract_selection, викликається, коли користувач надсилає текстовий запит, який відповідає одному з договорів.

Загальний опис роботи функції обробки:

1. Отримується інформація про користувача для логів.

2. Завантажуються всі назви та шляхи договорів у систему.

3. Користувач надсилає назву договору.

4. Якщо назва відповідає списку договорів: договір зберігається в контексті користувача; історія діалогу очищується; користувач отримує повідомлення про успішний вибір договору.

5. Якщо вибір неправильний, бот повідомляє про помилку вибору.

6. Усі дії фіксуються в логах для подальшого аналізу.

Блок-схема алгоритму дії модуля вибору договору наведена у Додаток И, блок-схема (А).

5.4.2 Функція завантаження та попередньої обробки договору

Функція load_pdf_as_documents (рис. 5.3) призначена для завантаження тексту з PDF-файлу, обробки цього тексту та його підготовки для подальшої роботи.

```
def load_pdf_as_documents(file_path):
    """Завантажує текст із PDF і розбиває його на частини."""
    initial_text = "Нижче наведено договір, який складається з розділів та пунктів...\n\n"
    with pdfplumber.open(file_path) as pdf:
        all_text = initial_text + "".join(page.extract_text() + "\n" for page in pdf.pages)
    text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=50)
    split_texts = text_splitter.split_text(all_text)
    return [Document(page_content=text) for text in split_texts]
```

Рисунок 5.3 – Функція завантаження та попередньої обробки договору

Дана функція виконує наступне:

1. Завантажує весь текст із PDF-файлу.

2. Додає вступний текст.

3. Розбиває текст на частини певного розміру з накладанням для збереження контексту.

4. Повертає список об’єктів Document, які представляють кожну частину тексту.

Блок-схема роботи функції завантаження та попередньої обробки договору наведена у Додаток И, блок-схема (Б).

5.4.3 Функція створення та забезпечення векторного сховища

Створення векторного сховище відбувається у функції get_vectorstore, яка забезпечує створення та отримання векторного сховища для заданого PDF-документа – рис. 5.4.

```python
vectorstore_cache = {}
logger = logging.getLogger(__name__)

def get_vectorstore(file_path):
    """Отримує або створює векторне сховище."""
    if file_path not in vectorstore_cache:
        documents = load_pdf_as_documents(file_path)
        # logger.info(documents)
        embeddings = OpenAIEmbeddings()
        vectorstore = LanceDB.from_documents(documents, embeddings)
        vectorstore_cache[file_path] = vectorstore
        logger.info(f"Векторне сховище створено для {file_path}.")

        # Контроль розміру кешу
        if len(vectorstore_cache) > VECTORSTORE_CACHE_SIZE:
            vectorstore_cache.pop(next(iter(vectorstore_cache)))

    return vectorstore_cache[file_path]
```

Рисунок 5.4 – Функція створення та отримання векторного сховища

Дана функція виконує наступне:

1. Коли викликається функція get_vectorstore, вона перевіряє, чи вже існує векторне сховище для заданого PDF-файлу.

2. Якщо сховище вже створене, воно повертається з кешу, зменшуючи витрати часу та ресурсів.

3. Якщо сховища немає:

– текст із PDF-файлу завантажується та розбивається на частини;

– для кожної частини тексту створюються векторні представлення (embeddings) за допомогою OpenAI API;

– на основі векторів створюється векторне сховище з використанням FAISS;

– сховище додається до кешу, а в логах фіксується його створення.

4. Якщо розмір кешу перевищує встановлений ліміт, найстаріше сховище видаляється для оптимізації використання пам’яті.

5. У результаті повертається векторне сховище для заданого PDF-файлу, готове до використання для швидкого пошуку релевантної інформації.

Блок-схема роботи функції створення та отримання векторного сховища наведена у Додаток И, блок-схема (В).

5.4.4 Модуль обробки запитів користувача

Цей модуль реалізує обробку запитів користувачів до чат-бота підтримки, забезпечуючи генерацію відповідей на основі вибраного страхового договору. Його основна мета – об'єднати механізми пошуку релевантної інформації в тексті договорів із генерацією контекстуальних відповідей за допомогою мовної моделі.

Даний модуль працює в наступному порядку:

1. Отримує текстовий запит від користувача: функція bot_reply отримує повідомлення від користувача через Telegram API.

2. Перевіряє, чи вибрано договір: якщо користувач не вибрав договір, функція повідомляє про необхідність зробити вибір і завершує виконання.

3. Створює або отримує векторне сховище: завантажує текст договору і перетворює його на вектори для ефективного пошуку релевантних фрагментів.

4. Створює RAG-ланцюжок обробки:

– ініціалізує мовну модель (OpenAI GPT);

– завантажує шаблон для взаємодії у форматі «запитання-відповідь»;

– формує пошуковий механізм для отримання тексту з векторного сховища;

– з’єднує ці компоненти в єдиний ланцюжок.

5. Обробляє історію діалогу:

– зберігає попередні повідомлення користувача для контексту;

– лімітує історію до останніх п'яти повідомлень, щоб уникнути перевантаження моделі.

5. Генерує відповідь: ланцюжок обробки формує відповідь на основі тексту договору та історії діалогу.

6. Логує дії: записує у файл журналу запити користувача та згенеровані відповіді, що дозволяє відстежувати роботу бота.

7. Надсилає відповідь користувачеві: повертає контекстуальну відповідь у Telegram.

8. Реєструє обробник запитів: функція register_query_handlers додає bot_reply як обробник для всіх текстових повідомлень, що надсилає користувач.

Блок-схема роботи модуля обробки запитів клієнта наведена у Додаток К.

Даний модуль забезпечує інтерактивну обробку запитів клієнтів, використовуючи мовну модель для пошуку та формулювання відповідей на основі тексту страхових договорів. Він є ключовою складовою чат-бота, яка дозволяє автоматизувати підтримку клієнтів.

5.4.5 Логування у чат-боті підтримки

Функції setup_logging (рис. 5.5.) забезпечує централізоване налаштування логування для всіх компонентів проєкту. Логи зберігаються у визначеному файлі (LOG_FILE) і дублюються в консоль для моніторингу під час роботи. Логування дозволяє розробникам відстежувати події в системі, аналізувати помилки та забезпечувати прозорість роботи чат-бота.

```python
def setup_logging():
    """Налаштовує логування для проєкту."""
    logging.basicConfig(
        format="%(asctime)s - %(name)s - %(levelname)s - %(message)s",
        level=logging.INFO,
        handlers=[
            logging.FileHandler(LOG_FILE),
            logging.StreamHandler(),
        ],
    )
    logging.getLogger("httpx").setLevel(logging.WARNING)
```

Рисунок 5.5 – Налаштування логування у системі

Налаштування формату логів: формат логів задається як "%(asctime)s - %(name)s - %(levelname)s - %(message)s", що забезпечує збереження наступної інформації: час події (asctime); ім'я модуля або компонента, що створив лог (name); рівень важливості повідомлення (levelname); текст повідомлення (message).

Рівень логування: встановлено рівень INFO, що означає, що в логах зберігатимуться інформаційні повідомлення, попередження, помилки тощо (але не відлагоджувальні повідомлення DEBUG).

Обробники логів: логи записуються одночасно у файл, шлях до якого задається змінною LOG_FILE (logs/bot_activity.log) та у термінал (або консоль), завдяки StreamHandler.

Налаштування логування для бібліотеки httpx: для бібліотеки httpx встановлено рівень логування WARNING, щоб зменшити кількість непотрібних повідомлень.

У системі логується інформація про всі ключові події, включаючи запити користувачів, згенеровані відповіді бота, вибір договорів, створення векторних сховищ для обробки текстів, а також підключення користувачів до бота. Це дозволяє аналізувати взаємодію користувачів із системою, відстежувати помилки та оптимізувати її роботу.

5.5 Керівництво користувача по використанню розробленого чат-бота підтримки

1. Загальний опис.

Чат-бот підтримки страхової компанії розроблено для автоматизації процесу обслуговування клієнтів. Бот допомагає отримувати інформацію про страхові договори, роз’яснює їхні умови та відповідає на запитання користувачів у зручному текстовому форматі.

2. Початок роботи.

Запуск бота:

– знайдіть бота в Telegram (@Insurance_assist_bot) та натисніть кнопку «Розпочати» або введіть команду /start;

– бот привітає вас і надасть список доступних страхових договорів у вигляді кнопок.

Вибір договору:

– натисніть на кнопку з назвою потрібного договору або введіть назву договору вручну;

– бот підтвердить вибір і запропонує поставити запитання.

3. Використання функціоналу.

Запитання щодо договору:

– введіть ваше запитання (наприклад, "Що включено в страховий поліс?" або "Які винятки застосовуються?");

– бот аналізує ваш запит і надає відповідь на основі вибраного договору.

Зміна договору:

– якщо вам потрібно змінити договір, виберіть інший договір із доступного списку.

Отримання допомоги:

– для отримання короткої довідки введіть команду /help;

– бот пояснить, як працює, і які запити можна до нього ставити.

4. Основні команди.

/start – почати або перезапустити взаємодію з ботом; виводить список доступних договорів.

/help – показує інструкцію для користувача.

5. Обмеження.

– бот відповідає лише на запитання, які стосуються вибраного договору. Якщо договір не вибрано, бот попросить спочатку зробити вибір;

– для складних запитів, які виходять за рамки умов договору, бот може рекомендувати звернутися до служби підтримки.

6. Особливості роботи.

Історія діалогу:

– бот зберігає останні 5 повідомлень у контексті діалогу, щоб забезпечити відповіді, що враховують попередню розмову;

– історія очищується автоматично при виборі нового договору.

Відповіді на основі договору:

– бот шукає релевантну інформацію безпосередньо в текстах договорів, зберігаючи точність відповідей.

7. Рекомендації для ефективного використання

Сформулюйте запитання чітко і конкретно.

Наприклад: «Які умови страхового відшкодування?», а не «Що це таке?».

Якщо бот не зміг відповісти на запитання, спробуйте переформулювати його або зверніться до служби підтримки.

8. Безпека та конфіденційність.

– усі ваші запитання та відповіді бот обробляє конфіденційно;

– дані не зберігаються після завершення діалогу.

9. Поширені запитання (FAQ).

– Що робити, якщо бот не відповідає?

Відповідь: переконайтеся, що у вас стабільне інтернет-з'єднання. Спробуйте перезапустити бота за допомогою команди /start.

– Як дізнатися, які договори підтримує бот?

Відповідь: список договорів надається після запуску команди /start.

– Чи можу я використовувати бот у мобільній версії Telegram?

Відповідь: Так, бот працює як у мобільному застосунку, так і в десктопній версії Telegram.

10. Демонстрація початку роботи та використання функціоналу бота.

Приклад початку роботи та використання функціоналу бота наведено у Додаток А.

Чат-бот підтримки створений для полегшення комунікації зі страховою компанією та забезпечення швидкого доступу до інформації. Користуйтеся ботом для отримання потрібної інформації в зручному форматі!

Висновки до розділу 5

Розробка чат-бота підтримки для страхової компанії стала успішним прикладом створення сучасної інформаційної системи для автоматизації обслуговування клієнтів. У процесі реалізації було використано метод RAG, бібліотеку pdfplumber для обробки PDF-документів, векторні бази даних FAISS для зберігання та пошуку інформації, а також OpenAI API для генерації

контекстуальних відповідей. Telegram API було обрано як платформу взаємодії з користувачами завдяки її зручності, доступності та інтеграційним можливостям.

Особливу увагу було приділено модульній структурі системи та опису її основних компонентів. Розроблені модулі забезпечують злагоджену роботу бота: вибір договору, завантаження та обробку текстів, створення векторного сховища, генерацію відповідей і логування дій. Така організація дозволяє легко підтримувати й розширювати функціонал, зберігаючи стабільність системи. Крім того, інтеграція історії діалогу забезпечує контекстуальність відповідей, що значно підвищує якість обслуговування клієнтів.

Для зручності користувачів було створено керівництво, яке допомагає швидко освоїти роботу з ботом, зокрема процес вибору договорів, формулювання запитів і використання команд. Це спрямовано на спрощення взаємодії з системою та підвищення її ефективності. Таким чином, розроблений чат-бот об'єднав передові технології обробки тексту, модульну архітектуру та зручність використання, створивши платформу, яка легко масштабується й відповідає сучасним вимогам до автоматизації підтримки клієнтів.

6 ОЦІНКА ТА ТЕСТУВАННЯ РОЗРОБЛЕНОГО ЧАТ-БОТА ПІДТРИМКИ

6.1 Оцінка відповідей розробленого чат-бота підтримки

Далі було здійснено її оцінку. Для цього використано функцію оцінки з підрозділу 3.3.1. В цій функції було використано RAG-систему із чат-бота підтримки. Після виконання оцінки RAG-системи, було отримано наступні результати – рис. 6.1.

user_input	reference	answer_relevancy	faithfulness	context_recall	context_precision	answer_correctness	semantic_similarity	elapsed_time
Що таке страховий випадок згідно з договором страхування здоров'я?	1	1	1	0	1	0.181277	0.763281	1.22074
Як подати заявку на відшкодування?	2	0.824894	1	0	1	0.184194	0.762618	2.8711
Який розмір страхової виплати у випадку діагностування гострого інфаркту міокарду?	3	1	1	0.25	1	0.243802	0.764907	2.1006
Які виключення зі страхових випадків передбачено в договорі?	4	1	1	1	1	0.177777	0.750148	4.7502
Чи покриває страховка лікування хронічних захворювань?	5	0.535343	1	0	1	0.176276	0.753005	3.20297
Що таке субкомпенсація?	6	1	1	0.5	1	0.176276	0.762147	1.27127
Напиши контактні дані для подачі заявки.	7	0.806407	0.913043	0.833333	1	0.180313	0.75867	2.37036
Що робити, якщо сталася надзвичайна ситуація і потрібна термінова медична допомога?	8	0.858291	0.768932	0	1	0.179251	0.748733	3.14087
Чи можу я отримати страховку на свою дитину?	9	1	1	0	1	0.187791	0.750874	1.2414
Чи покриває страховка лікування повторного інфаркту міокарду?	10	1	1	1	1	0.187363	0.749466	1.50951

Рисунок 6.1 – Результати оцінки системи RAG у чат-боті

Зробимо розрахунок середніх значень результатів, і запишемо їх у таблицю 6.1 для зручності аналізу.

Таблиця 6.1 – Середні значення отриманих результатів

df	answer_ relevancy	faithfulness	context_ recall	context_ precision	semantic_ similarity	elapsed_ time
evalute_df	0.959828	0.931169	0.933333	0.763889	0.891834	3.03265

Аналіз результатів оцінки системи RAG наведено нижче.

Переваги системи.

Система RAG у розробленій інформаційній системі служби підтримки страхової компанії демонструє значні переваги, які підкреслюють її ефективність. Висока релевантність відповідей (95.98%) свідчить про точне розпізнавання запитів користувачів і надання інформації, яка найбільше відповідає їхнім потребам. Це критично важливо у сфері страхування, де клієнти часто потребують доступу до специфічних деталей договорів.

Достовірність відповідей системи також знаходиться на високому рівні (93.12%), що вказує на точність у формуванні відповідей, базованих виключно на інформації з документів, без викривлення фактів. Це робить систему надійним

інструментом для роботи зі складними текстами. Показник context recall (93.33%) підкреслює здатність системи враховувати важливі елементи контексту, забезпечуючи цілісність відповідей. Семантична подібність (89.18%) підтверджує ефективність інтеграції LLM із RAG для роботи зі складними текстами. Час відповіді в 3.03 секунди є прийнятним показником, що забезпечує комфортну взаємодію користувачів із системою.

Слабші показники системи.

Незважаючи на переваги, система демонструє порівняно нижчий показник context precision (76.39%), що свідчить про часткову обробку зайвої інформації в процесі формування відповіді. Це може впливати на якість відповідей у складних запитах, коли важливо уникати включення нерелевантних даних. Також, хоча швидкість відповіді відповідає встановленим вимогам, її можна оптимізувати для ще більш швидкої обробки запитів у майбутніх версіях.

Варіанти покращення системи у майбутньому.

Для покращення системи у майбутньому слід зосередитися на підвищенні точності обробки контексту шляхом оптимізації пошукових алгоритмів у векторній базі, що дозволить зменшити включення нерелевантної інформації у відповіді. Також доцільно знизити час відповіді за рахунок оптимізації архітектури моделі та вдосконалення індексації текстів страхових договорів. Розширення навчальної бази за допомогою більш різноманітних запитів і сценаріїв взаємодії допоможе системі краще обробляти складні та нетипові запити користувачів. Окрім того, варто вдосконалити обробку довших діалогів, враховуючи складніший контекст взаємодії, що дозволить забезпечити ще більшу персоналізацію та точність відповідей.

Загальний висновок по оцінці системи.

Система RAG демонструє високі показники ефективності, релевантності та достовірності відповідей, що дозволяє вважати її ефективним інструментом для автоматизації підтримки клієнтів у страхуванні. Незважаючи на деякі недоліки, такі як необхідність покращення context precision, система відповідає основним функціональним і нефункціональним вимогам проєкту. Вона надає клієнтам

зручний доступ до інформації, автоматизує обробку запитів та забезпечує високу якість обслуговування, відповідаючи меті дослідження.

6.2 Тестування розробленої системи

Після оцінки систем підтримки, проведемо її тестування. Для цього зробимо декілька тестів, спрямованих на перевірку функціональності, правильності відповідей, виконання команд, вибору різних договорів, ведення діалогу, відповіді на некоректні запити, відповідь на великі запити у системі чат-бота.

Тест №1.

1. Назва тесту: запуск телеграм-бота.

2. Мета: перевірити, чи правильно запускається телеграм -бот.

3. Дія: знайти телеграм-бота за іменем у пошуку, і натиснути кнопку «Розпочати».

4. Очікуваний результат: бот запуститься і запропонує обрати договір.

5. Результат тесту: результат виконаних дій зображений на рисунку 6.2.

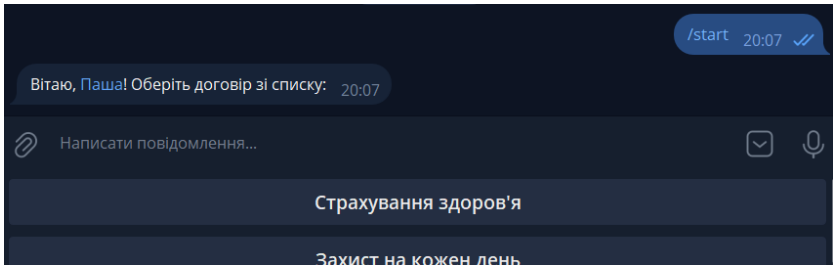


Рисунок 6.2 – Результат запуску бота

Висновок по тесту: результат тесту збігається з очікуваним результатом, тобто телеграм-бот запускається правильно і пропонує вибір договору.

Тест №2.

1. Назва тесту: ігнорування вибору договору.

2. Мета: перевірити, чи можна задавати питання не обравши договір.

3. Дія: задати будь яке питання, у чаті, не обираючи попередньо договір.

4. Очікуваний результат: бот попросить спочатку обрати договір.

5. Результат тесту: результат виконаних дій зображений на рисунку 6.3.

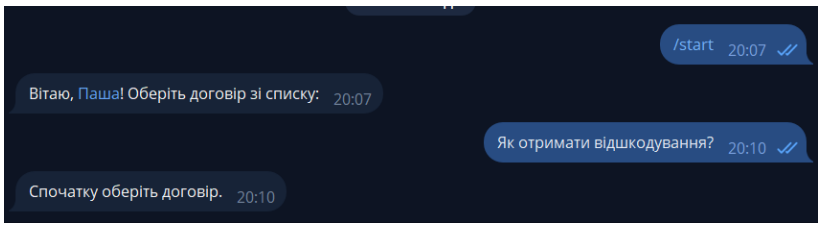


Рисунок 6.3 – Результат ігнорування вибору договору

Висновок по тесту: результат тесту збігається з очікуваним результатом, тобто телеграм-бот не починає роботи, допоки не буде обрано договір.

Тест №3.

1. Назва тесту: підтвердження вибору договору.

2. Мета: перевірити, чи правильно обраний договір.

3. Дія: обрати будь-який договір зі списку.

4. Очікуваний результат: бот підтвердить вибір договору і запропонує задати питання.

5. Результат тесту: результат виконаних дій зображений на рисунку 6.4.

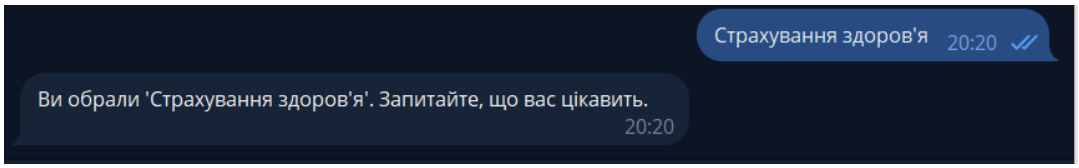


Рисунок 6.4 – Результат вибору договору

Висновок по тесту: результат тесту збігається з очікуваним результатом, тобто вибір обирається правильно.

Тест №4.

1. Назва тесту: коректність відповіді на звичайне запитання по договору (не по таблицям).

2. Мета: перевірити, чи правильно відповідає телеграм-бот на запитання по обраному договору.

3. Дія: обрати договір, задати питання по ньому.

4. Очікуваний результат: очікуваний результат коректності відповіді взятий із договору і підкреслений на рис. 6.5.

142	Інфекційний монокулеоз	10	30 днів
143	Малярія	15	30 днів
144	Сепсис	30	30 днів

Рисунок 6.5 – Результат запуску бота

5. Результат тесту: результат виконаних дій зображений на рисунку 6.6.

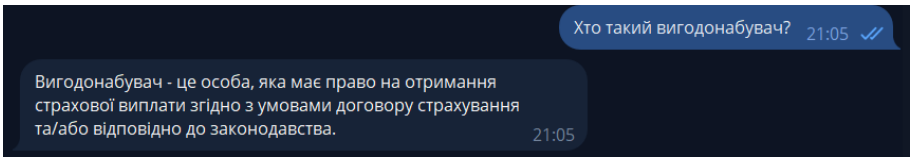


Рисунок 6.6 – Результат запуску бота

Висновок по тесту: результат тесту збігається з очікуваним результатом, тобто застосунок запускається правильно.

Тест №5.

1. Назва тесту: коректність відповіді на звичайне запитання по договору (не по таблицям).

2. Мета: перевірити, чи правильно відповідає телеграм-бот на запитання по обраному договору.

3. Дія: обрати договір, задати питання по ньому.

4. Очікуваний результат: очікуваний результат коректності відповіді взятий із договору, визначення відображено на рис. 6.5.

1.1. **Вигодонабувач** - особа, яка має право на отримання страхової виплати згідно з умовами договору страхування та/або відповідно до законодавства;

Рисунок 6.5 – Правильна відповідь на запитання

5. Результат тесту: результат виконаних дій зображений на рисунку 6.6.

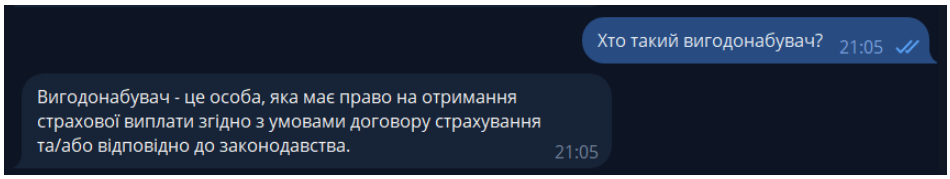


Рисунок 6.6 – Відповідь бота на запитання по договору

Висновок по тесту: результат тесту збігається з очікуваним результатом, тобто відповідь бота є правильною та коректною.

Тест №6.

1. Назва тесту: коректність відповіді на запитання по таблицям з договору .

2. Мета: перевірити, чи правильно відповідає телеграм-бот на запитання із таблиць по обраному договору.

3. Дія: обрати договір, задати питання по таблиці з нього.

4. Очікуваний результат: очікуваний результат коректності відповіді взятий із договору і підкреслений на рис. 6.7.

142	Інфекційний монокулеоз	10	30 днів
143	Малярія	15	30 днів
144	Сепсис	30	30 днів

Рисунок 6.7 – Дані із таблиці

5. Результат тесту: результат виконаних дій зображений на рисунку 6.8.

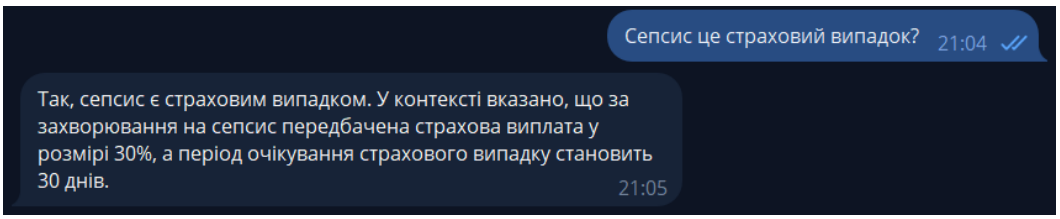


Рисунок 6.8 – Відповідь бота на запитання по договору

Висновок по тесту: результат тесту збігається з очікуваним результатом, тобто відповідь бота є правильною та коректною.

Тест №7.

1. Назва тесту: вибір іншого договору.

2. Мета: перевірити, чи можливо обрати інший договір.

3. Дія: відкрити список договорів та обрати інший договір.

4. Очікуваний результат: бот підтвердить вибір обраного договору і попросить задати питання по іншому договору.

5. Результат тесту: результат виконаних дій зображений на рисунку 6.9.

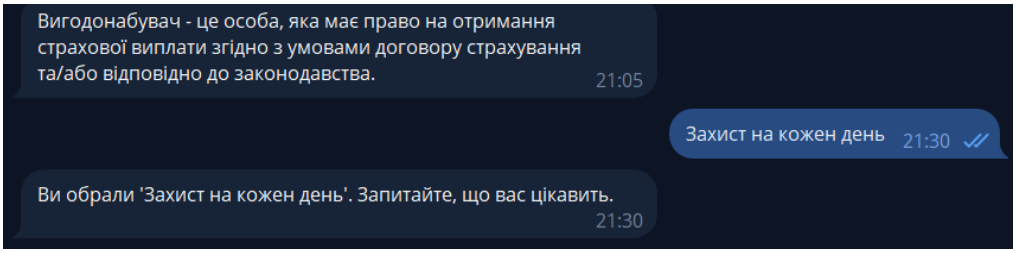


Рисунок 6.9 – Вибір іншого договору

Висновок по тесту: результат тесту збігається з очікуваним результатом, тобто бот обирає інший договір.

Тест №8.

1. Назва тесту: запитання не по темі договору.

2. Мета: перевірити реакцію бота на запитання не по темі договору та страхування.

3. Дія: обрати договір, задати питання не пов’язане з договором.

4. Очікуваний результат: бот відповість, що це запитання не стосується контексту, тому він не знає відповіді.

5. Результат тесту: результат виконаних дій зображений на рисунку 6.10.

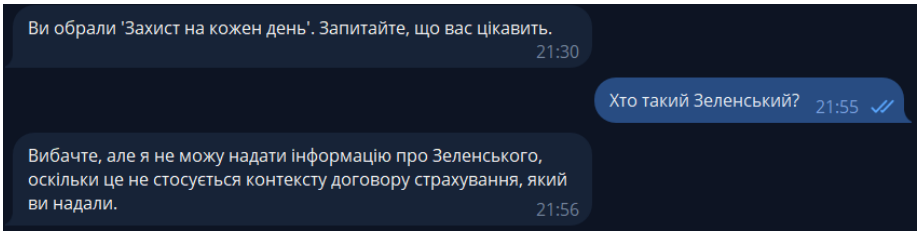


Рисунок 6.10 – Відповідь бота на запитання не по темі договору

Висновок по тесту: результат тесту збігається з очікуваним результатом, тобто бот говорить, що запитання не стосується контексту.

Тест №9.

1. Назва тесту: перевірка команди бота /start.

2. Мета: перевірити, чи правильно бот виконує функцію команди /start.

3. Дія: надіслати команду /start для бота.

4.Очікуваний результат: чат-бот скине поточний договір і попросить обрати новий договір.

5. Результат тесту: результат виконаних дій зображений на рисунку 6.11.

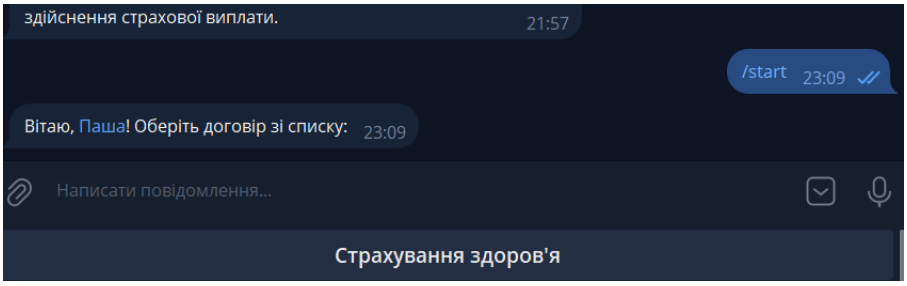


Рисунок 6.11 – Виконання команди /start

Висновок по тесту: результат тесту збігається з очікуваним результатом, тобто бот виконує правильну функцію при запуску команди.

Тест №10.

1. Назва тесту: перевірка команди бота /help.

2. Мета: перевірити, чи правильно бот виконує функцію команди /help.

3. Дія: надіслати команду /help для бота.

4. Очікуваний результат: чат-бот відправить повідомлення з інструкціями.

5. Результат тесту: результат виконаних дій зображений на рисунку 6.12.

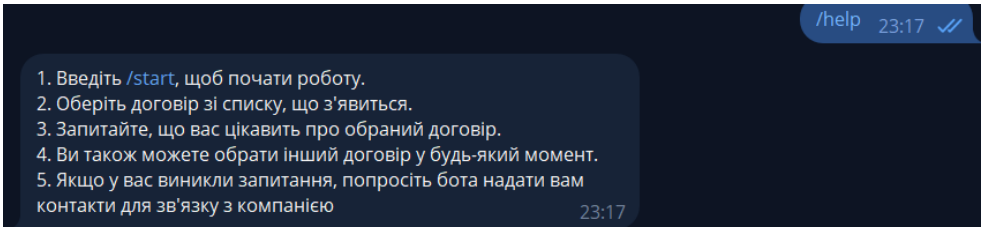


Рисунок 6.12 – Виконання команди /help

Висновок по тесту: результат тесту збігається з очікуваним результатом, тобто бот виконує правильну функцію при запуску команди.

Тест №11:

1. Назва тесту: пам’ять запитів.

2. Мета: перевірити, чи бот запам’ятовує попередні запити.

3. Дія: надіслати декілька запитів, і спитати щось про один з перших.

4. Очікуваний результат: чат-бот чітко відповість, яким був запит раніше.

5. Результат тесту: результат виконаних дій зображений на рисунку 6.13.

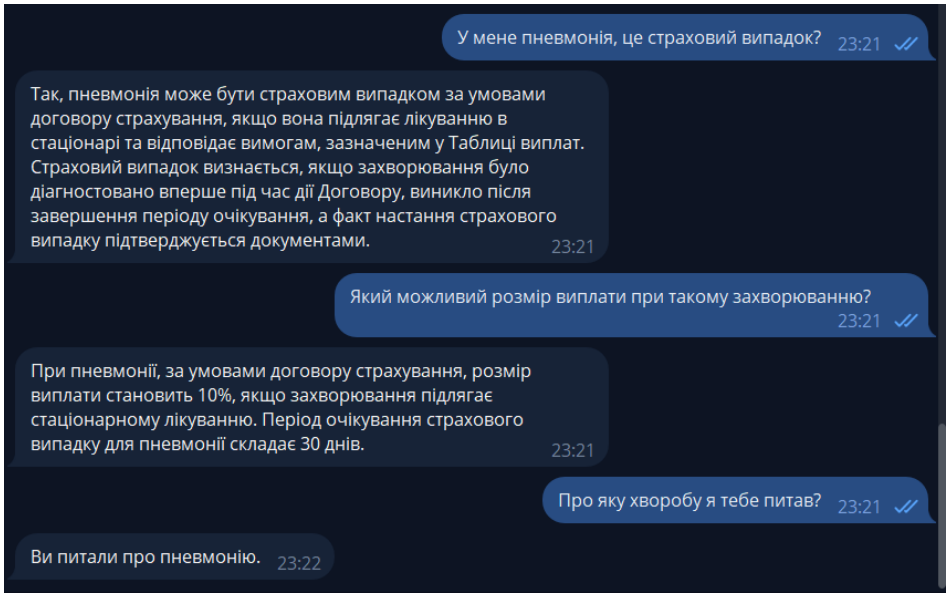


Рисунок 6.13 – Виконання команди /help

Висновок по тесту: результат тесту збігається з очікуваним результатом, тобто бот чітко пам’ятає, що запитував користувач і чітко розуміє контекст.

Отже, як можна побачити, чат-бот підтримки страхової компанії вдало пройшов всі тести і повноцінно виконує свої функції та відіграє свою роль «консультанта» по умовам договору.

Висновки до розділу 6

У цьому розділі було проведено оцінку та тестування розробленої системи чат-бота підтримки клієнтів страхової компанії. Результати оцінки показали, що система здатна ефективно забезпечувати точність і коректність надання відповідей, враховуючи запити користувачів. Чат-бот продемонстрував здатність зберігати контекст, надавати персоналізовані відповіді та працювати з різними командами,

такими як вибір договорів, запити поза контекстом і реагування на команди /start та /help.

Тестування підтвердило функціональність усіх заявлених можливостей системи. Вона успішно обробляє текстові запити, надає доступ до необхідної інформації про договори та забезпечує зручну взаємодію з користувачем. Система відповідає всім функціональним і нефункціональним вимогам, демонструючи стабільність роботи, надійність та зручність використання.

Хоча система працює ефективно, є можливість для подальшого вдосконалення, зокрема оптимізації пошукових алгоритмів для підвищення точності відповідей. Загалом, чат-бот підтвердив свою готовність до використання у страхових компаніях, забезпечуючи автоматизацію клієнтського обслуговування та підвищення його ефективності.

7 РОЗРОБКА СТАРТАП-ПРОЄКТУ

7.1 Опис ідеї стартап-проєкту

Ідея розроблюваного проєкту полягає у створенні системи підтримки (інтелектуального чат-бота) для страхових компаній, яка надає клієнтам швидкий доступ до інформації про страхові договори та відповідає на їхні запити через інтеграцію з відповідним інтерфейсом. У таблиці 7.1 описано зміст ідеї проєкту, його напрямки застосування та переваги для користувачів.

Таблиця 7.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення інтелектуального чат-бота для автоматизації підтримки клієнтів страхової компанії з	1. Обслуговування клієнтів у страхових компаніях.	Швидкий доступ до інформації про страхові договори без участі оператора.
	2. Взаємодія з клієнтами через месенджери.	Зручність у використанні завдяки доступу через досупний месенджер.
	3. Автоматизація відповідей на запити клієнтів.	Скорочення часу очікування відповіді та підвищення якості обслуговування.
	4. Використання NLP для аналізу текстів договорів.	Отримання персоналізованих та точних відповідей на складні запити.
	5. Застосування для аналізу та пошуку інформації у договорах.	Легкий доступ до складних розділів договорів і пошук необхідних пунктів.

Далі проведемо аналіз техніко-економічні характеристики стартап-проєкту. Для проведення аналізу було взято чат-боти підтримки з використанням ШІ, які зараз є на ринку: EVA (HDFC Bank), H&M Bot, Babylon Health. Умовні позначення:

W (слабка сторона) N (нейтральна сторона) S (сильна сторона). Результати аналізу наведені у таблиці 7.2.

Таблиця 7.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W	N	S
		Мій проект	EVA (HDFC Bank)	H&M Bot	Babylon Health			
1.	Зручність доступу через месенджери	+	-	-	-			+
2.	Автоматизація запитів клієнтів	+	+	+	+		+	
3.	Аналіз текстів договорів	+	-	-	-			+
4.	Швидкість обробки запитів	+	+	+	+		+	
5.	Можливість інтеграції з іншими системами	+	-	-	+			+
6.	Скорочення часу очікування клієнтів	+	+	+	-			+
7.	Індивідуальний підхід до користувачів	+	+	-	+		+	
8.	Підтримка голосових запитів	-	+	-	+	+		

За отриманим аналізом, можна побачити, що система має потенціал на ринку.

7.2 Технологічний аудит ідеї проєкту

Для системи підтримки страхової компанії на основі LLM було проведено визначення технологічних здібностей проєкту – таблиця 7.3.

Таблиця 7.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Взаємодія з користувачами через Telegram	Telegram Bot API	Наявна	Безкоштовний доступ
2.	Аналіз текстів договорів у форматі PDF	LangChain	Наявна	Безкоштовний доступ
3.	Генерація відповідей на основі запитів клієнтів	OpenAI API	Наявна	Від 0.002$ за 1k токенів
4.	Зберігання та обробка текстів договорів	Python-бібліотеки: pdfplumber, pandas	Наявна	Безкоштовний доступ
5.	Векторне представлення текстів для пошуку релевантних даних	FAISS	Наявна	Безкоштовний доступ
6.	Розробка програмного коду	Visual Studio Code	Наявна	Безкоштовний доступ
7.	Зберігання структурованих даних про договори	JSON	Наявна	Безкоштовний доступ

Обрана технологія реалізації ідеї проекту: для реалізаціх такої ідеї проєкту вже існують всі вказані технології, і більшість з них є безкоштовними, за винятко м використання ресурсу OpenAI API, його токени є платними. А загалом проєкт є загальнодоступним з вже наявними технологіями, які є переважно безкоштовними.

7.3 Аналіз ринкових можливостей запуску стартап-проєкту

Першим було проведено аналіз попиту на ринку – таблиця 7.4.

Таблиця 7.4 – Попередня характеристика потенційного ринку

№ п/п	Показники стану ринку (найменування)	Характеристика
1.	Кількість головних гравців, од	4
2.	Загальний обсяг продаж, грн/ум.од	200–400 млн. грн / рік
3.	Динаміка ринку (якісна оцінка)	Високий рівень зростання (~20% на рік)
4.	Наявність обмежень для входу (вказати характер обмежень)	Наявність технічних вимог до інтеграції з існуючими CRM-системами; високий рівень конкуренції серед загальних платформ для чат-ботів
5.	Специфічні вимоги до стандартизації та сертифікації	Відсутність жорстких стандартів, але важливо забезпечити відповідність нормам GDPR та стандартам безпеки даних для роботи з особистою інформацією
6.	Середня норма рентабельності в галузі (або по ринку), %	від 20% до 35%

Ринок чат-ботів для автоматизації обслуговування клієнтів демонструє високий потенціал зростання та динамікою розвитку близько 20% щорічно. Основні обмеження пов’язані з технічними вимогами інтеграції та дотриманням стандартів безпеки даних, однак середня рентабельність у 20-35% робить галузь привабливою для інноваційних рішень. Проєкт чат-бота має всі шанси для успішної реалізації, враховуючи зручність доступу, автоматизацію процесів та можливість аналізу страхових договорів.

Далі здійснено аналіз проаналізовані потенційних клієнтів – таблиця 7.5.

Таблиця 7.5 – Характеристика потенційних клієнтів стартап-проєкту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Забезпечення клієнтів швидким доступом до інформації про страхові договори, автоматизація підтримки.	1. Страхові компанії середнього і великого бізнесу. 2. Клієнти страхових компаній, які часто звертаються до підтримки.	1. Страхові компанії потребують інтеграції з CRM-системами та високої точності роботи. 2.Кінцеві користувачі цінують простоту використання та швидкість відповіді.	1. Інтеграція з існуючими системами компанії. 2. Простота інтерфейсу та доступність через Telegram. 3. Захист персональних даних клієнтів.
2	Скорочення часу очікування відповіді та автоматизація типових запитів клієнтів.	Великі корпорації, які прагнуть зменшити навантаження на операторів служби підтримки.	Бажання зменшити витрати на підтримку клієнтів, водночас забезпечуючи високий рівень обслуговування.	1. Масштабованість рішення. 2. Висока стійкість до збоїв. 3. Наявність звітів про оброблені запити.
3	Аналіз текстів договорів і пошук	1. Юридичні департаменти страхових	1. Юридичні департаменти потребують	1. Висока точність пошуку інформації. 2. Можливість роботи

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
	релевантної інформації без залучення людини.	компаній. 2. Брокери та консультанти у сфері страхування.	точності обробки текстів. 2. Брокери цінують швидкий доступ до важливих даних у договорах.	з документами в різних форматах (PDF, DOC).

Наступним кроком було виконано аналіз різних факторів загроз для проєкту – таблиця 7.6.

Таблиця 7.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Висока конкуренція	Сильна присутність на ринку інших чат-ботів підтримки, таких як EVA та Babylon Health.	Унікалізація продукту: додавання функцій аналізу текстів договорів та інтеграції з CRM.
2.	Технічні обмеження	Складність інтеграції з існуючими системами страхових компаній.	Забезпечення технічної підтримки інтеграції та адаптації продукту під специфіку клієнтів.
3.	Низька готовність ринку	Компанії не завжди готові інвестувати в нові технології через бюджетні обмеження.	Демонстрація економічної вигоди та швидкої окупності продукту.

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
4.	Регуляторні вимоги	Вимоги щодо захисту даних та відповідність нормам GDPR або місцевого законодавства.	Забезпечення відповідності продукту стандартам захисту даних і проведення аудитів.

Також виділено фактори можливостей проєкту – таблиця 7.7.

Таблиця 7.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Зростання попиту	Підвищений попит на автоматизацію обслуговування клієнтів у страхових компаніях.	Активний маркетинг, спрямований на демонстрацію вигод продукту.
2.	Унікальність функціоналу	Аналіз текстів договорів і пошук інформації є рідкісною функцією на ринку.	Розробка маркетингових матеріалів, що підкреслюють унікальність функцій продукту.
3.	Розширення ринку	Можливість масштабування продукту на інші галузі, наприклад, фінансову чи юридичну.	Розробкамодулів, адаптованих для інших галузей.
4.	Технологічна доступність	Використання відкритих API та доступних технологій для розробки продукту.	Зниження витрат на розробку та оптимізація витрат на підтримку.

Далі було проведено аналіз конкуренції ринку за ступенями, виділено особливості та заходи до конкурентоспроможності – таблиця 7.8.

Таблиця 7.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика.	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Монополістична конкуренція	На ринку присутні кілька великих гравців, які пропонують схожі продукти, але кожен має свої унікальні риси.	Унікалізація продукту через функції аналізу текстів договорів та інтеграції з CRM-системами.
2. Локальний і міжнародний рівень	Конкуренція зосереджена як на локальних страхових компаніях, так і на міжнародному ринку з великими платформами.	Зосередження на локальному ринку з подальшою адаптацією продукту для виходу на міжнародний рівень.
3.Галузева конкуренція	Конкуренти спеціалізуються у страхуванні, фінансах та медичних послугах.	Розширення функціоналу для інтеграції у суміжні галузі, такі як фінанси чи юриспруденція.
4. Товарно-видова конкуренція	Пряме змагання з іншими чат-ботами, що виконують схожі функції, але з різним рівнем автоматизації.	Підвищення якості послуг і функціоналу для залучення клієнтів, орієнтація на аналітичні можливості.
5. Цінова і нецінова конкуренція	Деякі конкуренти пропонують нижчі ціни, інші — унікальні функції без акценту на доступності.	Оптимізація витрат для зниження ціни та впровадження унікальних функцій для підвищення цінності продукту.
6. Марочна конкуренція	У великих конкурентів уже є впізнавані бренди, що викликають довіру клієнтів.	Формування бренду через активний маркетинг, створення репутації через позитивний клієнтський досвід.

Таблиця 7.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Опис
Прямі конкуренти в галузі	EVA (HDFC Bank), H&M Bot, Babylon Health
Потенційні конкуренти	Нові страхові чат-боти, інтегровані з CRM та платформами для аналізу даних
Постача-льники	API-провайдери (Telegram, OpenAI), постачальники IT-сервісів
Клієнти	Страхові компанії, які прагнуть автоматизації, їх клієнти
Товари-замінники	Традиційні канали підтримки: телефонні дзвінки, email

Деталізований висновок по складовим аналізу з таблиці 7.9.

1. Прямі конкуренти в галузі: конкуренція інтенсивна через наявність великих гравців, таких як EVA, H&M Bot, і Babylon Health. Вони мають схожі можливості, але функція аналізу договорів і тісна інтеграція з CRM можуть стати конкурентною перевагою вашого проєкту.

2. Потенційні конкуренти: високий рівень доступності технологій (API, хмарні сервіси) знижує бар’єри для нових конкурентів. Потенційні гравці можуть з’явитися на ринку протягом 1-2 років, запропонувавши інноваційні функції або нижчі ціни.

3. Постачальники: вплив постачальників високий. Telegram API, OpenAI, і FAISS є критичними постачальниками, що диктують умови роботи (вартість і технічні параметри). Рекомендується передбачити альтернативні джерела для ключових послуг.

4. Клієнти: клієнти (страхові компанії) вимагають індивідуального підходу, швидких відповідей, доступності через популярні месенджери (Telegram) і точності обробки запитів. Високі вимоги до автоматизації підкреслюють важливість відповідності очікуванням клієнтів.

5. Товари-замінники: традиційні методи підтримки (дзвінки, email) все ще популярні серед клієнтів, оскільки вони не потребують навчання новим інструментам і легко доступні. Проте їхні недоліки, такі як обмеження швидкості й ефективності, є перевагою для чат-ботів.

За результатами аналізу ринку можна зробити висновок, що проєкт має принципову можливість успішно працювати навіть в умовах високої конкуренції. Попит на автоматизовані рішення у сфері страхування стабільно зростає, а використання сучасних технологій, таких як аналіз текстів договорів та інтеграція з CRM-системами, забезпечує суттєві переваги для залучення клієнтів. Хоча на ринку вже присутні сильні конкуренти, такі як EVA, H&M Bot і Babylon Health, а також існує ймовірність появи нових конкурентів, ваш проєкт може зайняти свою нішу завдяки унікальним функціям, високій швидкості та якості обслуговування. Для досягнення успіху важливо зосередитися на зручності для користувачів через інтеграцію з популярними платформами, такими як Telegram, забезпеченні доступної вартості продукту, а також активному маркетингу, що підкреслює унікальні переваги проєкту. Ці характеристики дозволять проєкту стати конкурентоспроможним і зайняти стійке місце на ринку.

На основі раніше описаної інформації було виділено та обґрунтовано фактори конкурентоспроможності – таблиця 7.10.

Таблиця 7.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Унікальність функціоналу	Наявність функції аналізу текстів договорів і інтеграції з CRM-системами забезпечує перевагу перед конкурентами, які такого функціоналу не мають.
2.	Простота інтеграції та використання	Легкість доступу через Telegram і мінімальні технічні вимоги знижують бар’єри для клієнтів.

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
3.	Висока швидкість та точність роботи	Забезпечує задоволення ключових вимог клієнтів – швидкі відповіді та точність інформації.
4.	Вартість обслуговування	Конкурентна ціна на ринку через оптимізацію витрат і використання доступних технологій (Telegram Bot API, OpenAI).
5.	Відповідність регуляторним вимогам	Дотримання норм захисту даних (GDPR) підвищує довіру клієнтів та забезпечує безпеку.

За визначеними факторами конкурентоспроможності (таблиця 7.10) проведено аналіз сильних та слабких сторін стартап-проекту – таблиця 7.11.

Таблиця 7.11 – Порівняльний аналіз сильних та слабких сторін стартапу

№ п/ п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розроблюваним проєктом						
			–3	–2	–1	0	1	2	3
1.	Унікальність функціоналу	18						+	
2.	Простота інтеграції та використання	16					+		
3.	Висока швидкість та точність роботи	17				+			
4.	Вартість обслуговування	15					+		
5.	Відповідність регуляторним вимогам	14				+			

Останнім етапом ринкового аналізу можливостей впровадження проекту було складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких

(Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін (таблиці 7.11). Результати SWOT-аналізу наведені у таблиці 7.12.

Таблиця 7.12 – SWOT- аналіз стартап-проекту

Сильні сторони: – унікальний функціонал: аналіз текстів договорів і інтеграція з CRM-системами; – зручність доступу через Telegram і простота використання; – конкурентна ціна завдяки оптимізації витрат; – дотримання норм GDPR та високий рівень безпеки даних.	Слабкі сторони: – висока залежність від сторонніх API (Telegram, OpenAI), що може впливати на стабільність роботи; – відсутність брендової впізнаваності, що може ускладнювати вихід на ринок; – немає підтримки голосових запитів, що обмежує категорію користувачів.
Можливості: – зростання попиту на автоматизацію обслуговування у страхових компаніях; – масштабування рішення для інших галузей, наприклад, фінансів або юриспруденції; – підвищення впізнаваності продукту через активний маркетинг і демонстрацію переваг; – використання нових технологій (наприклад, штучного інтелекту) для вдосконалення функціоналу.	Загрози: – інтенсивна конкуренція з боку великих гравців, таких як EVA та Babylon Health; – легкість входу на ринок для нових конкурентів через доступність технологій; – залежність від традиційних методів підтримки клієнтів (телефон, email), які залишаються популярними; – ризики, пов’язані зі змінами у регуляторних вимогах (захист даних та відповідність законодавству).

Додатково розроблено альтернативні варіанти впровадження ринкового впровадження розроблюваної системи. Вони відображені у таблиці 7.13.

Таблиця 7.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Проведення активної маркетингової кампанії з акцентом на унікальний функціонал (аналіз текстів договорів)	Висока	3–6 місяців
2.	Розширення функціоналу, включаючи голосову інтерактивність, для охоплення ширшої аудиторії.	Середня	6–12 місяців
3.	Масштабування продукту для роботи у фінансовому та юридичному секторах.	Низька	12–18 місяців

Обраною альтернативою є проведення активної маркетингової кампанії з акцентом на унікальний функціонал. Цей варіант має найвищу ймовірність отримання ресурсів, стислий строк реалізації (3–6 місяців) та дозволяє максимально швидко вивести продукт на ринок, забезпечуючи перевагу над конкурентами. Інші альтернативи вимагають більших ресурсів та триваліших строків реалізації. Розширення функціоналу або масштабування до інших галузей можуть бути реалізовані на наступних етапах розвитку проєкту після успішного входу на ринок.

7.4 Розроблення ринкової стратегії проєкту

Для розроблення ринкової стратегії спочатку необхідно визначити стратегії охоплення ринку, а саме опису цільових груп потенційних споживачів. Опис здійснено у таблиці 7.14.

Таблиця 7.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Страхові компанії середнього і великого бізнесу	Висока	Помірний	Середня	Середня
2.	Клієнти страхових компаній, які потребують інформаційної підтримки	Висока	Високий	Низька	Висока
3.	Юридичні департаменти страхових компаній	Середня	Низький	Середня	Середня
4.	Брокери та консультанти у сфері страхування	Середня	Середній	Середня	Середня
Які цільові групи обрано: для початкового впровадження обрано страхові компанії середнього і великого бізнесу та їхніх клієнтів, оскільки вони демонструють високу готовність сприйняти продукт, значний орієнтовний попит та низький або середній рівень конкуренції. Ці сегменти також характеризуються простотою входу, що дозволить швидше впровадити продукт і закріпити його на ринку. Юридичні департаменти та брокери можуть бути охоплені на подальших етапах розвитку.					

За результатами аналізу потенційних груп споживачів було обрано страхові компанії середнього і великого бізнесу та їхніх клієнтів як пріоритетні цільові групи для впровадження продукту. Ці сегменти мають високу готовність до сприйняття продукту, значний орієнтовний попит та відносно низький рівень конкуренції.

Для роботи з цими цільовими групами буде використовуватися стратегія диференційованого маркетингу, оскільки продукт орієнтований на кілька сегментів ринку, для кожного з яких будуть розроблені окремі маркетингові програми. Для страхових компаній акцент буде зроблено на функціоналі аналізу договорів та інтеграції з CRM-системами. Для клієнтів страхових компаній основний акцент буде зроблено на швидкості та зручності використання через месенджери.

Цей підхід дозволяє максимально ефективно врахувати потреби різних сегментів, підвищуючи конкурентоспроможність продукту на ринку.

Далі для роботи в обраних сегментах ринку сформуємо базову стратегію розвитку – таблиці 7.15.

Таблиця 7.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	Акцент на аналізі текстів договорів та інтеграції з CRM.	Диференційований маркетинг	Унікальність функціоналу, простота інтеграції.	Стратегія диференціації
2.	Активний маркетинг для формування бренду.	Масовий маркетинг	Позиціонування як сучасного рішення для автоматизації.	Стратегія диференціації

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
3.	Масштабування на фінансовий та юридичний ринки.	Диференційований маркетинг	Адаптація функціоналу для нових галузей.	Стратегія спеціалізації

Наступним кроком було визначено базову стратегію конкурентної поведінки – таблиця 7.16.

Таблиця 7.16 – Визначення базової стратегії конкурентної поведінки

Запитання	Відповідь
Чи є проект «першопрохідцем» на ринку?	Ні, на ринку вже присутні конкуренти, такі як EVA, H&M Bot, Babylon Health.
Чи буде компанія шукати нових споживачів?	Так, основний фокус – страхові компанії середнього бізнесу та їхні клієнти, а також нові сектори.
Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Ні, компанія зосередиться на унікальних функціях, таких як аналіз текстів договорів і CRM-інтеграція.
Стратегія конкурентної поведінки	Стратегія заняття конкурентної ніші: фокус на страхових компаніях і їхніх клієнтах.

Розроблюваний проєкт не є першим на ринку, але він може зайняти унікальну конкурентну нішу завдяки функціоналу, що вирізняється серед існуючих рішень. Основна увага буде приділена роботі зі страховими компаніями, їхніми клієнтами та поступовому розширенню на інші ринкові сегменти. Стратегія заняття ніші дозволить ефективно закріпитися на ринку завдяки фокусуванню на конкретних потребах обраного сегмента.

І на останок, на основі попередніх таблицю відбувається визначення стратегії позиціонування у стартап-проєкті.

Таблиця 7.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Висока швидкість обробки запитів, точність інформації, зручний доступ через месенджер	Стратегія диференціації	Унікальний функціонал аналізу договорів. Інтеграція з CRM-системами. Простота використання.	1. Технологічність. 2. Швидкість обслуговування. 3. Надійність і безпека даних.
2.	Персоналізація послуг, адаптивність під конкретні потреби клієнта	Стратегія спеціалізації	Відповідність специфічним потребам ринку. Орієнтація на страхові компанії. Підтримка унікальних вимог клієнтів	1. Індивідуальний підхід. 2. Гнучкість у налаштуванні. 3. Відповідність потребам бізнесу.
3.	Висока якість підтримки, відповідність законодавчим вимогам	Стратегія диференціації	Дотримання норм GDPR і захист даних. Високий рівень обслуговування. Конкурентна ціна	1. Безпека даних. 2. Професіоналізм. 3. Доступність через конкурентну ціну.

Проєкт буде позиціонуватися як технологічно передове рішення, що забезпечує високу швидкість обслуговування, надійність і безпеку. Основними асоціаціями, які споживачі мають ідентифікувати з проєктом, є: технологічність, індивідуальний підхід і професіоналізм. Це дозволить сформувати сильну ринкову позицію та відповідати ключовим вимогам цільової аудиторії.

7.5 Розроблення маркетингової програми стартап-проєкту

Першим кроком розроблення маркетингової компанії є формування маркетингової концепції, який отримає споживач – таблиця 7.18.

Таблиця 7.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1.	Зручний і швидкий доступ до інформації	Швидкість обслуговування, доступ через Telegram	Вища швидкість обробки запитів порівняно з конкурентами Простота інтеграції без складних налаштувань
2.	Автоматизація рутинних запитів клієнтів	Зменшення витрат часу і ресурсів на обробку стандартних запитів	Унікальний функціонал аналізу договорів. Інтеграція з CRM-системами
3.	Надійний захист персональних даних	Відповідність GDPR і високий рівень безпеки	Відповідність законодавчим вимогам. Надійність і довіра клієнтів
4.	Конкурентна вартість і економія ресурсів	Оптимальна ціна на ринку за високої якості послуг	Оптимізація витрат. Вища рентабельність для клієнтів

Надалі була розроблена трирівнева маркетингова модель товару. Результат наведено у таблиці 7.19.

Таблиця 7.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Опис базової потреби споживача: автоматизація процесів підтримки клієнтів у страхових компаніях.
Основні функціональні вигоди: швидкий доступ до інформації, точність у відповідях, захист персональних даних.</td></tr>
<tr><td rowspan="6">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Висока швидкість обробки запитів.</td><td>Нм</td><td>Тх</td></tr>
<tr><td>2. Інтеграція з CRM-системами.</td><td>Нм</td><td>Тх</td></tr>
<tr><td colspan="3">Якість: відповідність міжнародним стандартам, таким як GDPR.</td></tr>
<tr><td colspan="3">Пакування: програмний інтерфейс із доступом через месенджери.</td></tr>
<tr><td colspan="3">Марка: Insurance AI Assistance Bot.</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: розробка маркетингових матеріалів, що демонструють переваги продукту.</td></tr>
<tr><td colspan="3">Після продажу: технічна підтримка 24/7, оновлення функціоналу відповідно до потреб клієнтів.</td></tr>
<tr><td colspan="4">Захист від копіювання:
1. Використання унікального алгоритму аналізу договорів і текстових даних, заснованого на спеціально розроблених моделях. 2. Регулярні оновлення функціоналу, що враховують сучасні вимоги ринку та відгуки клієнтів. 3. Запатентовані інструменти інтеграції з CRM-системами, які не мають аналогів серед конкурентів. 4. Дотримання стандартів безпеки, що робить продукт привабливим для страхових компаній та важким для копіювання конкурентами.</td></tr>
</table>

Наступним етапом є визначення цінових меж. Ними потрібно керуватися при встановленні цін на потенційний товар. Цінові межі вказані у таблиці 7.20.

Таблиця 7.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1.	$5–15/місяць (базові чат-боти з мінімальним функціоналом)	$30–80/місяць (чат-боти зі складним функціоналом, інтеграціями)	$15,000- $90,000/рік (страхові компанії малого та середнього бізнесу)	Нижня межа: $20/місяць

Далі визначимо оптимальну та ефективну систему збуту для прийняття рішень по проєкту – таблиця 7.21.

Таблиця 7.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Страхові компанії малого та середнього бізнесу здійснюють закупівлю IT-продуктів на основі рекомендацій, аналізу ринку та консультацій.	Підготовка комерційних пропозицій, презентації продукту, персоналізовані консультації та підтримка	Прямий збут клієнтам	Власна система збуту через відділ продажу та онлайн-платформу для підписки
2.	Частина клієнтів може покладатися на зовнішніх	Надання технічної підтримки партнерам,	Один рівень	Залучена система збуту через

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
	консультантів та IT-партнерів для впровадження продуктів.	підготовка маркетингових матеріалів для партнерської мережі	(через партнерів)	сертифікованих IT-партнерів для збільшення охоплення ринку

І останньою складової маркетингової програми є розробка концепції маркетингових комунікацій – таблиця 7.22.

Таблиця 7.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуютьс я цільові клієнти	Ключові позиції, обрані для позиціонува ння	Завдання рекламного повідомлення	Концепція рекламного звернення
1.	Страхові компанії шукають рішення, які зменшують витрати, підвищують ефективність та автоматизую ть процеси	Цільова реклама в соцмережах (LinkedIn, Facebook), конференції, email-розсилки	Надійність, інтеграція з існуючими системами, простота впроваджен ня	Демонструват и економію часу та підвищення продуктивнос ті завдяки чат-боту	«Автоматизація обслуговування клієнтів – мінімум витрат, максимум результатів. Приєднайтеся до інноваційного рішення!»

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуютьс я цільові клієнти	Ключові позиції, обрані для позиціонува ння	Завдання рекламного повідомлення	Концепція рекламного звернення
2.	Схильність клієнтів до отримання консультацій перед прийняттям рішення	Онлайн-семінари, демонстрації продукту, персоналізов ані дзвінки відділу продажів	Експертніст ь компанії, підтримка на всіх етапах впроваджен ня	Показати професійний підхід до клієнтів і високий рівень технічної підтримки	«Ваш партнер у світі технологій: впровадження нашого чат-бота – це ваша перевага на ринку!»
3.	Клієнти звертають увагу на відгуки та кейси успішного впровадженн я	Відгуки на вебсайті, публікації в медіа, кейси клієнтів	Довіра, практичніст ь, перевіреніс ть на ринку	Побудувати довіру та впевненість у виборі	«Обирайте технологію, якій довіряють ваші колеги по ринку. Чат-бот, який змінює підхід до обслуговування клієнтів!»

Цільова аудиторія прагне знижувати витрати, впроваджувати легко інтегровані рішення та орієнтується на перевірені кейси. Для їх охоплення використовуються ефективні канали: соцмережі, конференції, вебінари та медіа-публікації. Позиціонування базується на надійності, підтримці та економії часу, що відповідає їхнім потребам. Рекламне звернення акцентує увагу на практичній користі, професійній підтримці та реальних результатах, створюючи комплексну програму для просування чат-бота.

Висновки до розділу 7

За результатами проведеного аналізу ринку, техніко-економічних характеристик проекту, конкурентного середовища та потенційних клієнтів, зроблено висновок, що проєкт має значний потенціал для ринкової комерціалізації. Ринок чат-ботів для страхових компаній демонструє стабільне зростання (~20% щороку), наявний попит з боку страхових компаній середнього і великого бізнесу, а також клієнтів, які потребують автоматизації процесів і швидкого доступу до інформації. При середній рентабельності галузі 20-35% і низьких витратах на реалізацію, проєкт має економічну привабливість.

Аналіз конкурентів і потенційних бар'єрів входження показав, що, незважаючи на високу конкуренцію з боку таких гравців, як EVA, H&M Bot і Babylon Health, проєкт має конкурентоспроможні переваги: унікальний функціонал (аналіз текстів договорів, інтеграція з CRM), простота впровадження та відповідність регуляторним вимогам. Бар'єри входу є середніми, але їх можна подолати завдяки технічній підтримці та демонстрації економічних вигод.

Для ринкової реалізації обрано альтернативу проведення активної маркетингової кампанії з акцентом на унікальні функції проєкту, що дозволить швидко зайняти конкурентну нішу на ринку. Ця стратегія має високу ймовірність отримання ресурсів та короткий термін реалізації (3–6 місяців). Для охоплення клієнтів використовується стратегія диференційованого маркетингу, що передбачає індивідуальний підхід до кожного сегмента аудиторії.

Узагальнюючи, впровадження проекту є доцільним, адже він відповідає потребам ринку, має конкурентні переваги та перспективи масштабування. Подальша імплементація проекту дозволить закріпитися на ринку, створити сильну ринкову позицію та забезпечити економічну ефективність.

ВИСНОВКИ

У даному дослідженні було розроблено та впроваджено інформаційну систему підтримки клієнтів страхової компанії на основі великих мовних моделей. Дана система створена з використанням методології RAG. Система забезпечує швидкий і зручний доступ до інформації про страхові договори, автоматизуючи процес обслуговування клієнтів та зменшуючи навантаження на людський персонал.

Розроблена система підтвердила свою ефективність за допомогою оцінки та тестування. Чат-бот забезпечує високий рівень релевантності відповідей (95.98%) та достовірності (93.12%), що відповідає вимогам до точності й коректності обробки запитів користувачів. Система ефективно інтегрує векторні бази для зберігання і пошуку текстів договорів, що дозволяє забезпечити динамічну і точну обробку інформації. Показники context recall (93.33%) та семантичної подібності (89.18%) свідчать про здатність системи враховувати контекст та надавати відповідну інформацію. Час відповіді в середньому становить 3.03 секунди, що забезпечує комфортну взаємодію користувачів із системою. Тестування підтвердило, що бот коректно виконує свої функції, включаючи обробку контексту, роботу з командами, вибір договору, надання відповідей на складні запити та обробку запитань поза контекстом.

Система забезпечує виконання функціональних вимог наступним чином:

– прийом текстових запитів (F1): чат-бот підтримує текстові запити через платформу Telegram, демонструючи зручний інтерфейс для користувачів. Запити обробляються у реальному часі, що гарантує оперативність взаємодії;

– інтеграція з документами (F2): завдяки використанню бібліотеки pdfplumber для обробки PDF-документів, система ефективно зчитує текст договорів, включаючи таблиці, та інтегрує їх у векторну базу, забезпечуючи швидкий пошук інформації;

– пошук у документах (F3): використання методології RAG та векторних баз (FAISS) дозволяє системі здійснювати семантичний пошук інформації в договорах, навіть якщо формулювання запиту не збігається із текстом документів;

– підтримка контексту діалогу (F4): бот зберігає історію діалогу, що дозволяє враховувати попередні запити при формуванні нових відповідей, забезпечуючи більш персоналізований досвід взаємодії;

– динамічний вибір договору (F5): користувачі можуть обирати або змінювати договір під час роботи з ботом, що дає змогу отримувати інформацію в контексті конкретного документа;

– логування взаємодій (F6): усі дії користувачів, запити та відповіді чат-бота детально фіксуються у лог-файлі для моніторингу, аналізу продуктивності та вдосконалення системи;

– відповідність умовам договорів (F7): відповіді бота базуються виключно на текстах страхових договорів, що виключає можливість викривлення чи неточності наданої інформації;

– підтримка користувацьких помилок (F8): система аналізує некоректні чи незрозумілі запити, пропонуючи користувачам уточнити свій запит або обрати інший варіант;

– зворотний зв'язок із користувачем (F9): у складних випадках бот надає інструкції щодо подальших дій або пропонує звернутися до служби підтримки.

Система також відповідає нефункціональним вимогам:

– швидкодія (NF1): час відповіді не перевищує 3-5 секунд, що забезпечує комфортну взаємодію;

– масштабованість (NF2): архітектура системи дозволяє легко додавати нові договори чи інші типи документів без значного впливу на продуктивність;

– надійність (NF3): бот працює стабільно, автоматично обробляючи дрібні збої та забезпечуючи доступність у будь-який час;

– конфіденційність (NF4): усі конфіденційні дані, такі як токени доступу, зберігаються у захищеному середовищі, що запобігає їх витоку;

– простота використання (NF5): інтерфейс Telegram забезпечує інтуїтивну взаємодію, не потребуючи додаткових технічних знань від користувачів;

– підтримка кількох мов (NF6): використання LLM надає змогу спілкуватися з чат-ботом всіма найпопулярнішими мовами світу, які знає використана мовна модель;

– сумісність (NF7): система побудована з урахуванням можливості інтеграції з іншими платформами, такими як вебдодатки чи мобільні застосунки;

– логування і моніторинг (NF8): детальна фіксація взаємодій дозволяє аналізувати ефективність роботи .

У результаті, система не лише відповідає поставленим завданням, але й демонструє високу ефективність, масштабованість та відповідність сучасним стандартам розробки інформаційних систем. Це забезпечує її готовність до впровадження у страхову галузь, а також можливість подальшого вдосконалення.

Незважаючи на високу ефективність, є аспекти для вдосконалення. Зокрема, необхідно покращити показник context precision (76.39%) для зменшення включення нерелевантної інформації у відповіді та прискорити обробку запитів. Подальший розвиток може включати розширення навчальної бази, оптимізацію пошукових алгоритмів та вдосконалення моделей генерації відповідей.

Наукова значущість роботи полягає у розробці системи підтримки, яка поєднує сучасні мовні моделі з пошуковими механізмами RAG, адаптованими до специфіки страхування. Практична цінність полягає у впровадженні ефективного рішення, яке може бути застосоване для автоматизації клієнтського сервісу у страховій галузі, а також адаптоване до інших сфер, де необхідна обробка текстових документів і запитів.

Розроблений чат-бот підтримки клієнтів страхової компанії виконує поставлені вимоги та функціонує відповідно до завдань. Він забезпечує автоматизацію роботи служби підтримки, надає користувачам зручний доступ до інформації про страхові договори та підвищує ефективність обслуговування. Система підтвердила свою надійність і функціональність, що свідчить про її відповідність сучасним стандартам розробки інформаційних систем.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Aslanova V. Psychological support assistant based on fine-tuned LLaMA 3 model / Aslanova V., Oliinyk V. // The International Conference on Security, Fault Tolerance, Intelligence ICSFTI2024. 2024, Kyiv, Ukraine, – P. 1-13. URL: https://icsfti-proc.kpi.ua/article/view/309532 (дата звернення: 25.11.2024).

2. HDFC Bank launches chatbot Eva for customer services. DNA India. URL: https://www.dnaindia.com/technology/report-hdfc-bank-launches-chatbot-eva-for-customer-services-2343370 (дата звернення: 25.11.2024).

3. H&M Bot – chatbotguide.org. ChatbotGuide.org. URL: https://www.chatbotguide.org/h-m-bot (дата звернення: 25.11.2024).

4. UK practitioner organization, Twitter influencers dispute Babylon Health's strong performance claims. *MobiHealthNews*. URL: https://www.mobihealthnews.com/content/uk-practitioner-organization-twitter-influencers-dispute-babylon-healths-strong-performance (дата звернення: 25.11.2024).

5. KLM Bot – chatbotguide.org. ChatbotGuide.org. URL: https://www.chatbotguide.org/klm-bot (дата звернення: 25.11.2024).

6. Oliinyk V. Low-resource text classification using cross-lingual models for bullying detection in the Ukrainian language / Oliinyk V., Matviichuk I. // Adaptive systems of automatic control, 2023. Vol. 1, №42. – P. 87-100.

7. Oliinyk V. Data augmentation with foreign language content in text classification using machine learning / Oliinyk V., Osadcha K. // Adaptive systems of automatic control, 2020. Vol. 1, №36. – P. 51-59.

8. Поночовний П.С. Аналітичний огляд способів застосування великих мовних моделей (LLM) для вирішення прикладних задач / Поночовний П.С., Олійник В.В.// Інженерія програмного забезпечення і передові інформаційні технології (Soft Tech-2023): матеріали V Міжнародної науково-практичної конференції молодих вчених та студентів, 19-21 грудня 2023 року, м. Київ, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», 2023. С. 272-276. URL:

https://drive.google.com/file/d/1racc22TBKkFFNzBRSzOrePKpFfbnGDJ1/view (дата звернення: 25.11.2024).

9. Sahota H. Practical retrieval augmented generation. Wiley & Sons, Incorporated, John, 2024, – P. 100-111.

10. Şen Z. Fuzzy String Matching Procedure. The Open Bioinformatics Journal. 2020. Vol. 13, no. 1. P. 50–56.

11. Metrics | Ragas. Ragas. URL: https://docs.ragas.io/en/v0.1.21/concepts/metrics/index.html (дата звернення: 25.11.2024).

12. Malviya R. K., Javalkar V., Malviya R. Scalability and Performance Benchmarking of LangChain, LlamaIndex, and Haystack for Enterprise AI Customer Support Systems. IJGIS Fall of 2024 Conference. URL: https://doi.org/10.21428/e90189c8.43aeb06e (дата звернення: 25.11.2024).

13. Korat A. S. AI-Driven Multi-PDF Chatbot: Integrating LangChain and GPT-3 for Enhanced Data Processing. Journal of Artificial Intelligence & Cloud Computing. 2024. Vol. 3, no. 4. P. 1-6.

14. OpenAI. Assistants API overview. URL: https://platform.openai.com/docs/assistants/overview (дата звернення: 25.11.2024).

15. UNIVERSALNA. Страхова компанія UNIVERSALNA. Страховка онлайн. Купити страховку в Страховій компанії UNIVERSALNA | Київ, Львів, Дніпро і вся Україна. URL: https://universalna.com/insurance-products/ (дата звернення: 25.11.2024).

16. Introduction | Ragas. Ragas. URL: https://docs.ragas.io/en/v0.1.21/index.html (дата звернення: 25.11.2024).

17. Chroma. Chroma. URL: https://www.trychroma.com/ (дата звернення: 25.11.2024).

18. Faiss: A library for efficient similarity search. Engineering at Meta. URL: https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/ (дата звернення: 25.11.2024).

19. LanceDB - LanceDB. URL: https://lancedb.github.io/lancedb/ (date of access: 25.11.2024).

20. Beaman C. P. The size and nature of a chunk. Behavioral and Brain Sciences. 2017. Т. 24, № 1. С. 118. URL: https://doi.org/10.1017/s0140525x01263924 (дата звернення: 25.11.2024).

21. Medium. Implementing RAG in LangChain with Chroma: A Step-by-Step Guide. 2024. URL: https://medium.com/@callumjmac/implementing-rag-in-langchain-with-chroma-a-step-by-step-guide-16fc21815339 (дата звернення: 25.11.2024).

22. Isaías González, Antonio José Calderón and José María Portalo. Innovative Multi-Layered Architecture for Heterogeneous Automation and Monitoring Systems: Application Case of a Photovoltaic Smart Microgrid. 2021. P. 2-5.

23. Sarthak Mittal, Yoshua Bengio, Guillaume Lajoie. Is a Modular Architecture Enough. 2022. P. 2-10.

24. Sayyad A. A Step-by-Step Guide to Parsing PDFs using the pdfplumber Library In Python. Medium. URL: https://azhar-sayyad.medium.com/a-step-by-step-guide-to-parsing-pdfs-using-the-pdfplumber-library-in-python-c12d94ae9f07 (дата звернення: 25.11.2024).

25. OpenAI. OpenAI API. URL: https://openai.com/index/openai-api/ (дата звернення: 25.11.2024).