Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Факультет інформатики
Кафедра мультимедійних систем

ПРОЕКТУВАННЯ ТА СТВОРЕННЯ ВЕБ-ЗАСТОСУНКУ НА ОСНОВІ ПОЄДНАННЯ МОЖЛИВОСТЕЙ LLM ТА RAG

Текстова частина до кваліфікаційної роботи
за спеціальністю «Інженерія програмного забезпечення»

Науковий керівник
к.т.н., доц. Олецький О. В.
_____
(підпис)
“____” _____ 2026 р.
Виконав студент
Українець Д. С.
“____” _____ 2026 р.

Київ 2026

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мультимедійних систем

Освітній ступінь «Бакалавр»

Спеціальність «121. Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

ЗАВДАННЯ
ДЛЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Українцю Дмитру Сергійовичу

1. Тема роботи «Проектування та створення веб-застосунку на основі поєднання можливостей LLM та RAG», керівник роботи Олецький Олексій Віталійович, кандидат технічних наук, доцент, затверджені наказом вищого навчального закладу від 3 листопада 2025 року №11.
2. Строк подання студентом роботи до 15 травня 2026 року.
3. План роботи:
   - 3.1. Проаналізувати предметну область та існуючі підходи до використання LLM і RAG у веб-застосунках для генерації текстових відповідей.
   - 3.2. Спроектувати архітектуру прототипу веб-застосунку; підібрати відповідний технологічний стек.
   - 3.3. Спроектувати узгоджені формати подання даних, правил і довідкових матеріалів для майбутнього веб-застосунку.
   - 3.4. Реалізувати прототип веб-застосунку для читання результатів аудиту з БД та/або сформованого звіту. Реалізувати API для отримання цих даних зовнішніми клієнтами та модуль формування контексту запиту до LLM на основі поєднання даних з БД, звіту й RAG, а також механізм отримання відповідей від LLM.
   - 3.5. Реалізувати логіку генерації пояснень і персоналізованих рекомендацій на основі вже сформованого аудиту та його звіту: формування додаткових текстових пояснень до виявлених проблем, узагальнення критичних ризиків, пропозицій щодо їх усунення та

пріоритизації дій на основі даних з БД/звіту, набору правил і текстових матеріалів, використаних у RAG-шарі.

3.6. Реалізувати відповідні програмні засоби (клієнтська частина).

3.7. Провести експериментальне дослідження роботи системи на тестових наборах результатів аудиту: оцінити релевантність, повноту та зрозумілість рекомендацій, дослідити вплив RAG на якість генерації рекомендацій.

3.8. Оформити текст кваліфікаційної роботи відповідно до вимог університету та підготувати всі необхідні матеріали (текстова частина, презентація, архів вихідних кодів) для захисту перед екзаменаційною комісією.

ГРАФІК ПІДГОТОВКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

№ з/п	**ПЕРЕЛІК РОБІТ**	**Термін** виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
1.	Вибір і затвердження теми кваліфікаційної роботи «Проектування та створення веб-застосунку на основі поєднання можливостей LLM та RAG», закріплення наукового керівника. Узгодження календарного графіка підготовки кваліфікаційної роботи та індивідуального завдання, подання цих документів у систему DistEdu.	до 14 грудня 2025 року			
2.	Вивчення джерел літератури та сучасних публікацій з LLM, RAG, аудиторських систем, аналіз існуючої системи аудиту та структури її звіту. Збір і узагальнення фактів, даних і практик, які будуть використані як база знань для рекомендаційної підсистеми.	жовтень – листопад 2025 року			
3.	Складання детального плану кваліфікаційної роботи (структура розділів, перелік експериментів, опис інтеграції з існуючим аудитом) та узгодження його з науковим керівником.	листопад – грудень 2025 року			
4.	Написання основних розділів роботи: виклад теоретичних основ LLM і RAG, опис предметної області аудиту, аналіз наявної реляційної БД та звіту аудиту, формулювання	листопад 2025 року – січень 2026 року			

	задачі побудови рекомендаційної підсистеми.				
5.	Проєктування та реалізація експериментального прототипу сервісу рекомендацій: модуль читання результатів аудиту з реляційної БД та/або звіту, побудова структури знань у RAG, налаштування інтеграції з LLM через відповідні програмні засоби, постановка експериментів та первинний аналіз результатів.	грудень 2025 року – лютий 2026 року			
6.	Проміжний контроль виконання роботи: підготовка й завантаження проміжного звіту про виконання кваліфікаційної роботи в систему DistEdu, обговорення проміжних результатів із науковим керівником та коригування плану за потреби.	до 31 січня 2026 року			
7.	Написання кваліфікаційної роботи в цілому: оформлення всіх розділів, опис архітектури та реалізації сервісу рекомендацій, результатів експериментів і висновків. Підготовка першого повного варіанта тексту та завантаження чорнової версії роботи до DistEdu.	до 15 березня 2026 року			
	Розділ 1 – постановка проблеми, теоретичні основи LLM і RAG, огляд літературних джерел, формалізація задачі рекомендацій за результатами аудиту.	січень 2026 року			
	Розділ 2 – аналітико-дослідницька частина: аналіз предметної області, опис	лютий 2026 року			

	існуючої системи аудиту, моделі даних у БД та структури звіту, постановка експериментів для оцінювання якості рекомендацій.				
	Розділ 3 – проєктно-експериментальна частина: опис реалізації прототипу сервісу рекомендацій (RAG, LLM), аналіз експериментальних результатів і формування висновків.	до 15 березня 2026 року			
8.	Повне завершення написання кваліфікаційної роботи, внесення правок за зауваженнями наукового керівника, підготовка остаточного тексту до подання на перевірку академічної доброчесності.	березень – квітень 2026 року			
9.	Подання кваліфікаційної роботи для перевірки письмових робіт студентів НУКМА на відповідність вимогам академічної доброчесності (завантаження фінальної версії роботи на DistEdu).	до 15 травня 2026 року			
10.	Підготовка до захисту кваліфікаційної роботи: написання доповіді, підготовка презентації та демонстраційних матеріалів для демонстрації роботи веб-застосунку з рекомендаціями на основі LLM і RAG.	березень – травень 2026 року			
11.	Попередній захист кваліфікаційної роботи на засіданні кафедри, презентація проміжних результатів та	30 березня – 01 квітня 2026 року			

	отримання зауважень щодо змісту й оформлення роботи.				
12.	Подання кваліфікаційної роботи на кафедру з усіма супровідними документами: відгуком наукового керівника, зовнішньою рецензією, результатами перевірки на академічну доброчесність.	до 20 травня 2026 року			
13.	Публічний захист кваліфікаційної роботи перед екзаменаційною комісією.	25–27 травня 2026 року			

Графік узгоджено 15 листопада 2025 року

Науковий керівник Олецький Олексій Віталійович

Виконавець кваліфікаційної роботи Українець Дмитро Сергійович

Зміст

Анотація

Кваліфікаційна робота присвячена дослідженню можливостей посилення великих мовних моделей шляхом надання їм доступу до реальних даних із використанням технологій генерації з доповненим пошуком. Метою роботи є проєктування та створення веб-застосунку, який обробляє готові результати аудиту бізнес-профілю на Google Картах, визначає виявлені проблеми та формує практичні рекомендації щодо їх усунення. Для розв’язання задачі використано базу правил, векторну базу даних, механізм добору релевантного контексту і великі мовні моделі. У межах роботи спроєктовано архітектуру системи, реалізовано серверну й клієнтську частини, модулі обробки аудиту, добору релевантного контексту, генерації відповідей, а також експериментальний модуль. Отримані результати підтверджують придатність підходу для формування зрозумілих рекомендацій на основі даних аудиту.

Ключові слова: великі мовні моделі, генерація з доповненим пошуком, векторна база даних, база правил, генерація відповідей.

Перелік прийнятих скорочень

API – Application Programming Interface, програмний інтерфейс застосунку;

LLM – Large Language Model, велика мовна модель;

RAG – Retrieval-Augmented Generation, генерація з доповненим пошуком;

JSON – JavaScript Object Notation, об'єктна нотація JavaScript;

HTTP – Hypertext Transfer Protocol, протокол передачі гіпертексту;

БД – база даних;

СКБД – система керування базами даних.

Вступ

Сучасний розвиток інформаційних технологій супроводжується активним упровадженням великих мовних моделей у прикладні програми. Такі моделі здатні аналізувати текстові дані, узагальнювати інформацію та формувати зв’язні відповіді природною мовою. Водночас їх використання у прикладних задачах має низку обмежень, пов’язаних насамперед із відсутністю прямого доступу до актуальних або спеціалізованих даних конкретної предметної області. Унаслідок цього сформовані відповіді можуть бути недостатньо точними, неповними або відірваними від фактичного контексту задачі.

Одним із сучасних підходів до подолання цих обмежень є поєднання LLM із генерацією з доповненим пошуком. Такий підхід передбачає попередній добір релевантного контексту із зовнішніх джерел інформації перед формуванням відповіді. Завдяки цьому модель може спиратися не лише на власні знання, отримані під час навчання, а й на конкретні правила, структуровані дані або довідкові матеріали, пов’язані з поставленою задачею.

Актуальність роботи зумовлена потребою дослідити можливості практичного використання LLM і RAG у веб-застосунках, які працюють із реальними прикладними даними та формують текстові рекомендації для користувача.

Як предметну область у роботі розглянуто обробку результатів аудиту бізнес-профілю на Google Картах, який є важливим інструментом цифрової присутності локального бізнесу. Проблема полягає в тому, що результати аудиту подаються у вигляді набору оцінок та окремих зауважень. Такий формат не є достатньо зрозумілим для користувача, оскільки не пояснює повною мірою зміст виявленої проблеми, її можливий вплив і послідовність дій для усунення. Виникає потреба у програмному засобі, який здатний перетворювати готові результати аудиту на структуровані, обґрунтовані та практично корисні рекомендації.

Аналіз сучасного стану розв’язання цієї проблеми показує, що великі мовні моделі широко застосовуються для генерації текстових відповідей та пояснень. Водночас у прикладних задачах їх доцільно поєднувати з механізмами добору релевантної інформації, оскільки це дає змогу зменшити ризик неточних або надто загальних відповідей. Генерація з доповненим пошуком розглядається як один із підходів, що дозволяє пов’язати LLM із конкретним контекстом предметної області та підвищити змістову обґрунтованість сформованого результату.

Наукове значення роботи полягає в розробленні прикладного підходу до поєднання можливостей LLM та RAG для формування рекомендацій на основі структурованих результатів аудиту.

Практичне значення полягає у створенні веб-застосунку, який може перетворювати результати аудиту бізнес-профілю на Google Картах на зрозумілі рекомендації щодо усунення виявлених проблем.

Джерельну основу роботи становлять наукові публікації та технічна документація, присвячені LLM, RAG, семантичному пошуку, векторному поданню текстових даних, векторним базам даних та методам оцінювання якості сформованих відповідей.

Зв’язок кваліфікаційної роботи з попередніми дослідженнями полягає у використанні відомих підходів до генерації тексту, пошуку релевантного контексту та оцінювання якості відповідей у межах конкретної прикладної задачі.

Новизна теми полягає в поєднанні підходів великих мовних моделей та генерації з доповненим пошуком у веб-застосунку, який працює з готовими результатами аудиту та формує рекомендації на основі структурованих даних і дібраного контексту.

Метою кваліфікаційної роботи є проєктування та створення веб-застосунку, який на основі готових результатів аудиту бізнес-профілю на Google Картах формує рекомендації щодо їх усунення з використанням можливостей LLM та RAG.

Для досягнення поставленої мети необхідно розв’язати такі завдання:

а) проаналізувати предметну область та існуючі підходи до використання великих мовних моделей і генерації з доповненим пошуком для формування текстових відповідей;

б) спроєктувати архітектуру веб-застосунку та визначити основні компоненти системи;

в) реалізувати механізм обробки результатів аудиту, виділення виявлених проблем і добору релевантного контексту;

г) інтегрувати LLM в процес формування рекомендацій;

ґ) провести експериментальне дослідження роботи системи та оцінити вплив добору релевантного контексту на якість сформованих відповідей.

Об’єктом дослідження є процес формування текстових рекомендацій із використанням LLM та RAG.

Предметом дослідження є програмні засоби поєднання можливостей LLM та RAG для автоматизованого формування рекомендацій на основі структурованих вхідних даних.

Методологічну основу роботи становлять методи аналізу предметної області, систематизації наукових джерел і технічної документації, проєктування програмної архітектури, моделювання структури даних, семантичного пошуку, експериментального дослідження та порівняння результатів роботи системи за різних режимів формування рекомендацій.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел і додатків. У роботі послідовно розглянуто теоретичні засади використання великих мовних моделей і генерації з доповненим пошуком, проєктування веб-застосунку, реалізацію його основних компонентів, а також експериментальне оцінювання отриманих результатів.

1 Теоретичні засади використання LLM та RAG

1.1 LLM як засіб генерації текстових відповідей

Великі мовні моделі є одним із провідних напрямів сучасного розвитку штучного інтелекту та безпосередньо пов’язані із задачами оброблення природної мови. У наукових джерелах LLM розглядаються як розширені попередньо навчені мовні моделі, які завдяки значній кількості параметрів і великому обсягу навчальних даних можуть виконувати різні задачі оброблення природної мови [1]. Основою їх роботи є статистичне моделювання мови: модель аналізує наданий контекст і визначає найбільш імовірні наступні елементи тексту.

Розвиток великих мовних моделей став можливим завдяки збільшенню обсягів текстових даних, зростанню обчислювальних потужностей і вдосконаленню методів навчання нейронних мереж. Важливу роль у цьому процесі відіграла архітектура Transformer, що використовує механізм уваги та дає змогу враховувати зв’язки між компонентами вхідної послідовності без рекурентних або згорткових шарів [2]. Завдяки цьому сучасні LLM ефективно опрацьовують довгі послідовності й можуть виконувати генерацію пояснень, узагальнення, переформулювання, переклад, класифікацію та створення структурованих відповідей.

Типовий процес формування відповіді великою мовною моделлю складається з кількох етапів. Спочатку вхідний текст перетворюється на токени – окремі одиниці, які можуть відповідати словам, частинам слів, символам або розділовим знакам [3]. Далі токени подаються у вигляді числових векторів і опрацьовуються шарами нейронної мережі. Для авторегресійних моделей характерне послідовне прогнозування наступного токена на основі попереднього контексту: кожен сформований токен додається до послідовності та використовується на наступному кроці генерації [4]. Таким чином, відповідь не зберігається в моделі наперед, а створюється під час опрацювання конкретного запиту. Узагальнену схему цього процесу наведено на рисунку 1.1.

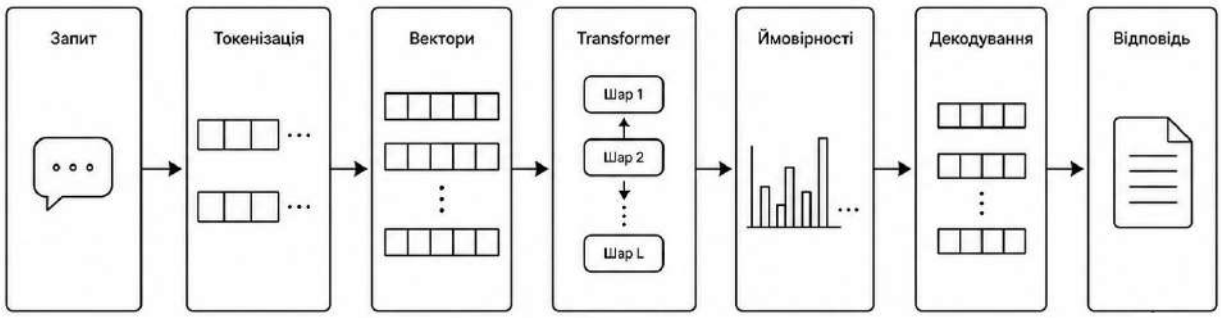

Рисунок 1.1 – Узагальнена схема формування текстової відповіді великою мовною моделлю

Важливою особливістю LLM є залежність результату від контексту вхідного запиту. Модель виконує інструкцію відповідно до її формулювання, наданих даних і очікуваної структури відповіді. У прикладних системах це дає змогу керувати генерацією через запити, що містять опис задачі, вхідні дані, обмеження та бажаний формат результату.

Важливим етапом розвитку мовних моделей стало попереднє навчання на великих корпусах текстів із подальшим пристосуванням до конкретних задач. Наприклад, модель BERT продемонструвала ефективність глибоких двонапрямлених мовних подань для різних задач оброблення природної мови [5]. Інший напрям пов’язаний з авторегресійними моделями, які формують текст шляхом послідовного передбачення токенів і можуть виконувати низку задач без окремого навчання, якщо опис задачі або приклади подано у запиті [6]. Подальше масштабування таких моделей привело до появи GPT-3, для якої було показано здатність виконувати задачі на основі кількох прикладів у запиті [7]. Це зробило LLM придатними для використання в програмних системах, де розробник може описати задачу природною мовою та задати формат відповіді.

У сучасних застосунках LLM часто використовуються як компонент, що перетворює вхідні дані на зрозумілий текстовий результат. Для цього модель отримує питання користувача, опис ролі, правила формування відповіді, фрагменти даних, вимоги до структури тексту та змістові обмеження. Такий

підхід дає змогу використовувати LLM не тільки для відкритого діалогу, а й для контрольованого формування відповідей.

У межах цієї роботи великі мовні моделі розглядаються як засіб автоматичного формування текстових пояснень і практичних рекомендацій на основі підготовлених вхідних даних. У розробленому веб-застосунку такими даними є результати аудиту бізнес-профілю на Google Картах [8]. Завдання LLM полягає в тому, щоб перетворити ці результати на зв’язний, зрозумілий і корисний для користувача текст. Перевагою такого підходу є можливість адаптувати форму відповіді до потреб користувача.

Водночас ефективність LLM залежить не лише від масштабу, а й від здатності виконувати інструкції та узгоджувати відповідь із наміром користувача. Для цього застосовують додаткове навчання з урахуванням людського зворотного зв’язку, що підвищує здатність моделі дотримуватися інструкцій, формату відповіді та очікувань користувача [9].

Отже, LLM можна розглядати як засіб генерації текстових відповідей, що поєднує статистичне моделювання мови, нейронні архітектури та керування результатом через текстовий запит. У цій роботі LLM є центральним компонентом, оскільки забезпечує формування кінцевого тексту рекомендації.

1.2 Обмеження LLM під час роботи з прикладними даними

Використання великих мовних моделей у прикладних інформаційних системах може викликати низку обмежень, які впливають на якість і надійність результатів.

Основні обмеження LLM:

а) відсутність доступу до актуальних даних – модель формує відповіді на основі інформації, отриманої під час попереднього навчання, тому може використовувати застарілі або неповні дані [10];

б) ризик генерації некоректної інформації – LLM здатні формувати правдоподібні, але фактологічно помилкові відповіді, оскільки модель прогнозує текст, а не перевіряє достовірність інформації [11];

в) недостатня точність у вузькоспеціалізованих задачах – модель може неправильно інтерпретувати специфічні дані певної предметної області, якщо вони недостатньо представлені в навчальних даних [12];

г) значні обчислювальні витрати та обмеження обсягу контексту – робота з великими обсягами вхідного тексту потребує більше пам’яті й обчислювальних ресурсів, тому система має попередньо відбирати лише найбільш релевантну інформацію [13].

Таким чином, використання LLM у прикладних веб-застосунках потребує додаткових механізмів для добору релевантного контексту. Одним із підходів до розв’язання цієї проблеми є використання RAG.

1.3 RAG як підхід до посилення LLM

Одним із підходів до подолання обмежень LLM є генерація з доповненим пошуком. Її основна ідея полягає у поєднанні механізму добору релевантної інформації з можливостями LLM щодо формування текстової відповіді [10].

У RAG-системах перед генерацією виконується пошук додаткового контексту у зовнішніх джерелах даних. Після цього знайдена інформація передається до моделі разом із початковим запитом. Завдяки цьому відповідь може спиратися на актуальні або спеціалізовані дані конкретної предметної області [12].

Загальний принцип роботи RAG-системи наведено на рисунку 1.2.

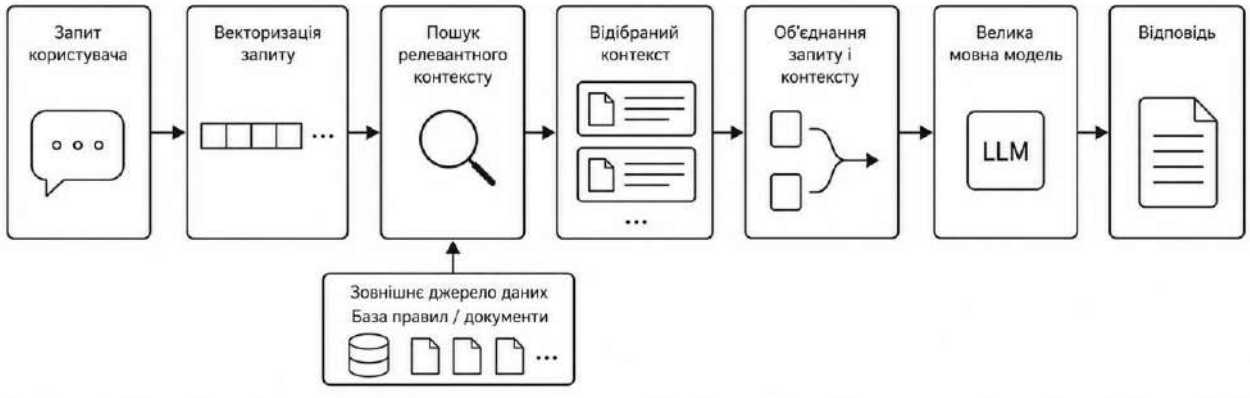

Рисунок 1.2 – Узагальнена схема роботи RAG-системи

Основними етапами роботи генерації з доповненим пошуком є: отримання запиту користувача, добір релевантної інформації із зовнішнього джерела, об’єднання знайденого контексту із запитом, передавання сформованого контексту до LLM та генерація фінальної відповіді.

У більшості сучасних RAG-систем пошук виконується за семантичною схожістю. Для цього текстові дані перетворюються у векторне подання, після чого порівнюються вектори запиту та доступних фрагментів [12].

Типова RAG-система містить модуль добору релевантного контексту, зовнішнє джерело даних і модуль генерації, представлений LLM. Використання цього підходу зменшує залежність відповіді від попереднього навчання моделі та підвищує точність і змістовність сформованого результату [10].

У межах цієї кваліфікаційної роботи RAG використовується для добору релевантних правил, пов’язаних із виявленими проблемами в аудиті бізнес-профілю, які передаються до LLM і використовуються під час формування рекомендацій для користувача.

1.4 Семантичний пошук і векторне подання текстових даних

Одним із ключових компонентів RAG-систем є механізм семантичного пошуку, який орієнтується на змістову близькість між запитом і доступними текстовими фрагментами. Це дає змогу знаходити релевантну інформацію навіть тоді, коли запит і документ сформульовані різними словами, але мають близьке значення [12].

Основою семантичного пошуку є векторне подання текстових даних. Для цього текст перетворюється на числовий вектор, який відображає його змістові характеристики. Такі вектори формуються за допомогою спеціалізованих моделей, що враховують контекст слів і зв’язки між ними [12].

Загальний принцип семантичного пошуку наведено на рисунку 1.3.

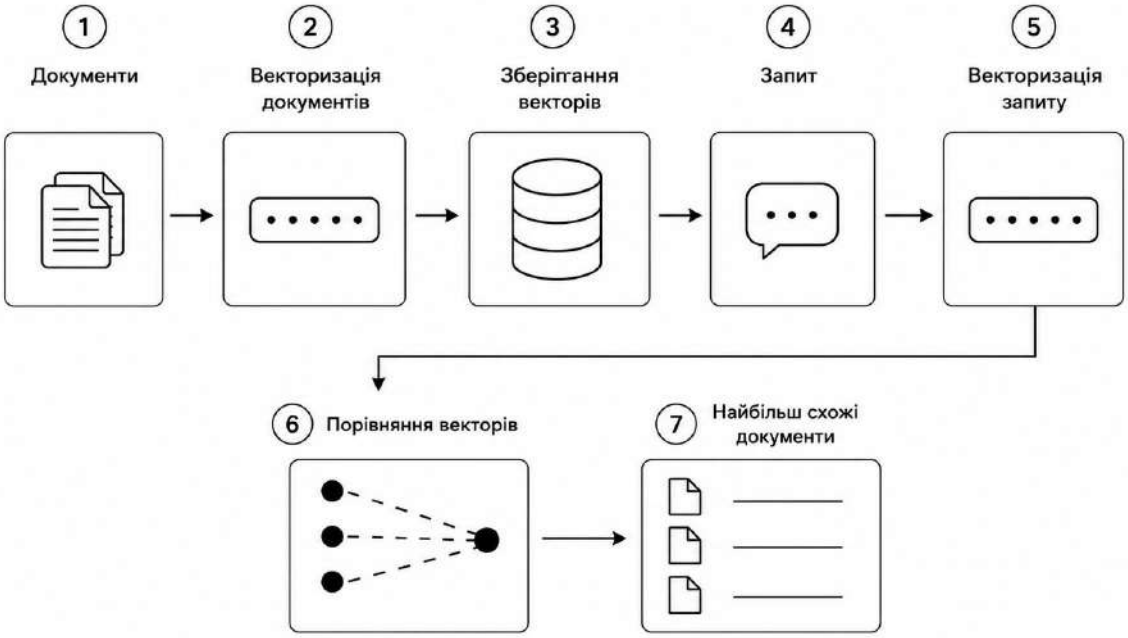

Рисунок 1.3 – Узагальнена схема семантичного пошуку у векторній БД

Процес семантичного пошуку передбачає кілька основних етапів: текстові документи або правила перетворюються у векторне подання, сформовані вектори зберігаються у векторній БД, запит користувача також перетворюється у вектор, після чого система порівнює його з векторами доступних фрагментів і повертає найбільш змістовно близькі результати.

Для визначення схожості між векторами використовуються математичні метрики. Однією з найпоширеніших є косинусна схожість, яка визначає близькість напрямків двох векторів у векторному просторі [15].

У межах цієї кваліфікаційної роботи семантичний пошук використовується для добору правил, пов’язаних із виявленими проблемами в аудиті бізнес-профілю. Завдяки цьому система отримує релевантний контекст, який надалі використовується під час формування рекомендацій.

1.5 База правил як джерело структурованої інформації

У системах із генерацією з доповненим пошуком важливу роль відіграють зовнішні джерела інформації, що використовуються для формування контексту відповіді [12; 16]. Одним із таких джерел може бути база правил –

структурований набір текстових правил, пояснень і рекомендацій, підготовлених для певної предметної області.

У межах цієї роботи база правил використовується як джерело структурованої інформації для семантичного пошуку та формування контексту, який передається LLM. Кожне правило містить опис проблеми, пояснення її значення та рекомендації щодо усунення.

У реалізованій системі правила зберігаються у форматі JSON. Їхній текстовий вміст перетворюється у векторне подання, після чого використовується під час пошуку у векторній базі даних. Такий підхід поєднує контрольованість наперед підготовленої інформації з гнучкістю генерації текстових відповідей [16].

1.6 Оцінювання якості відповідей, сформованих із використанням LLM та RAG

Однією з важливих задач під час використання LLM і RAG є оцінювання якості сформованих відповідей. Оскільки результат роботи LLM має генеративний характер, його якість визначається не лише правильністю інформації, а й релевантністю, повнотою, зрозумілістю та відповідністю контексту [17].

Оцінювання таких відповідей є складним, оскільки одна й та сама відповідь може мати кілька коректних варіантів формулювання. Через це метрики точного збігу тексту не дають змоги повною мірою оцінити результат. У RAG-системах основна увага приділяється змістовній якості відповіді та її зв’язку зі знайденим контекстом [17].

Під час оцінювання зазвичай аналізуються такі характеристики:

а) релевантність відповіді;

б) повнота охоплення важливих аспектів;

в) відповідність знайденому контексту;

г) зрозумілість і структурованість для користувача.

Для RAG-систем також важливою є обґрунтованість, тобто ступінь опори відповіді на знайдені джерела. Якщо відповідь формується без достатньої прив’язки до отриманого контексту, зростає ризик появи некоректної інформації [11].

У сучасних дослідженнях для автоматизованого оцінювання RAG і LLM застосовуються спеціалізовані інструменти, зокрема RAGAS і TruLens-Eval [17; 18]. Вони дають змогу комплексно оцінювати якість відповіді та її зв’язок із використаним контекстом. Водночас такі засоби орієнтовані переважно на універсальні сценарії та не враховують особливостей конкретної прикладної області.

У межах цієї кваліфікаційної роботи реалізовано власний модуль оцінювання метрик, що зумовлено специфікою архітектури розробленого веб-застосунку.

Готові інструменти оцінювання не забезпечують усіх необхідних можливостей для такого типу системи. Зокрема, вони не враховують роботу з наперед підготовленою базою правил, механізм вибору релевантного правила, різницю між режимами формування відповіді, службову інформацію про джерело рекомендації, використане правило, режим генерації та резервні сценарії. Крім того, універсальні метрики не враховують структуру сформованих рекомендацій.

Тому в роботі використано власний набір метрик, адаптований до архітектури системи та задачі оцінювання рекомендацій. Під час його розробки враховано загальні підходи до оцінювання RAG-систем. Реалізований модуль дає змогу оцінювати характеристики, безпосередньо пов’язані з логікою роботи створеного веб-застосунку.

2 Проєктування веб-застосунку та вибір технологічного стеку

2.1 Постановка задачі та загальні вимоги до системи

Задача кваліфікаційної роботи полягає в проєктуванні та створенні веб-застосунку, призначеного для автоматизованого формування рекомендацій на основі готових результатів аудиту бізнес-профілю на Google Картах за допомогою поєднання можливостей LLM та RAG.

Вхідними даними є результати аудиту, отримані із зовнішнього сервісу. Приклад візуального подання результату аудиту наведено в додатку А, а приклад JSON-структури одного питання аудиту – у додатку Б.

Кожен тематичний блок аудиту містить набір питань, що описують окремі перевірки бізнес-профілю. Для кожного питання визначається набір характеристик. На основі цих даних система повинна виділяти проблемні пункти аудиту, добирати для них релевантний контекст на основі правил і формувати підсумкові рекомендації. Веб-застосунок при цьому працює з уже сформованими результатами зовнішнього аудиторського сервісу [8].

Під час проєктування системи було визначено такі основні функціональні вимоги:

а) отримання результатів аудиту із зовнішнього джерела, їх нормалізація та перетворення у внутрішній формат системи;

б) автоматичне виділення проблемних пунктів аудиту на основі статусів і значень окремих перевірок;

в) добір релевантного контексту з бази правил із використанням семантичного пошуку;

г) формування рекомендацій у кількох режимах: на основі бази правил, із використанням RAG та лише за допомогою LLM;

ґ) використання резервного сценарію у випадках, коли релевантне правило не знайдено або генерація відповіді недоступна;

д) збереження результатів аналізу, сформованих рекомендацій та історії запусків;

е) надання веб-інтерфейсу для запуску аналізу, перегляду результатів, рекомендацій, історії запусків і налаштувань системи;

є) забезпечення адміністративного доступу до службових функцій;

ж) надання можливості змінювати параметри роботи системи через веб-інтерфейс без редагування програмного коду;

з) відображення детальної інформації про окремий запуск аналізу;

и) створення експериментальних сесій для набору аудитів та розрахунок метрик якості.

З огляду на визначені вимоги систему спроєктовано як веб-застосунок із поділом на серверну та клієнтську частини.

2.2 Загальна архітектура веб-застосунку

З урахуванням поставленої задачі веб-застосунок було спроєктовано за клієнт-серверним принципом. Загальну архітектуру веб-застосунку наведено на рисунку 2.1.

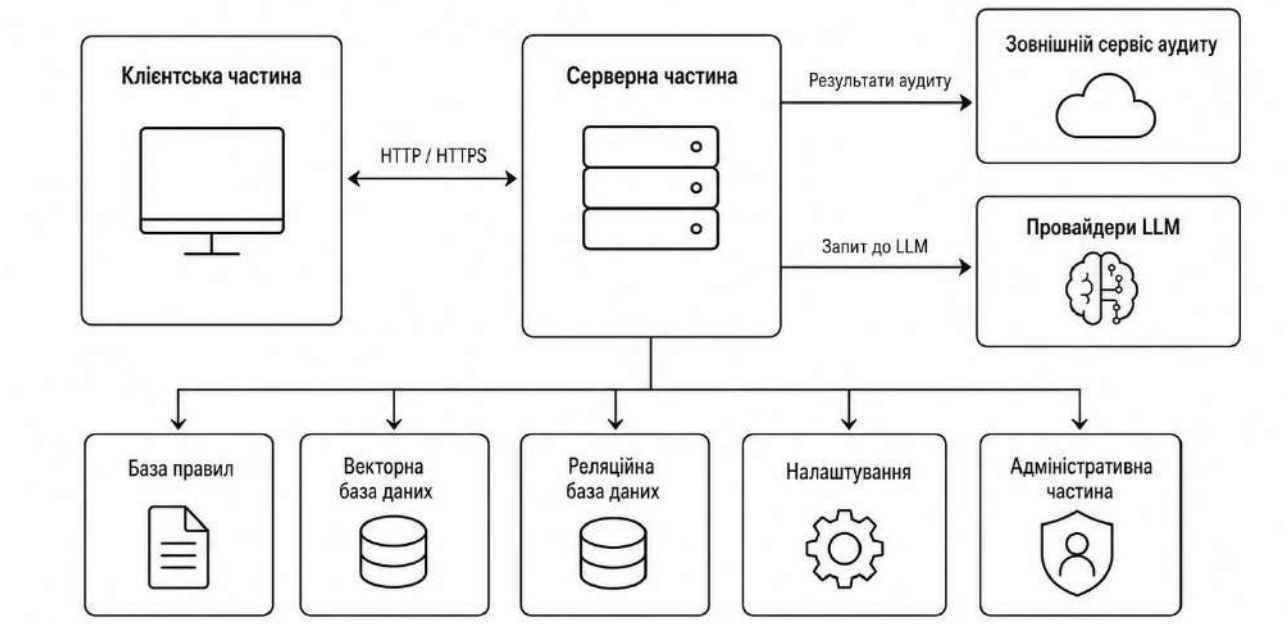

Рисунок 2.1 – Загальна архітектура веб-застосунку

Основними компонентами розробленої системи є:

а) клієнтська частина – забезпечує взаємодію користувача із системою;

б) серверна частина – реалізує основну логіку роботи застосунку;

в) зовнішній сервіс аудиту – надає готові результати аудиту бізнес-профілю на Google Картах;

г) база правил – структурований JSON-файл із підготовленими правилами, описами проблем, рекомендаціями та додатковими характеристиками;

г) векторна БД – використовується для збереження векторних представлень правил і виконання семантичного пошуку;

д) реляційна БД – зберігає результати аналізу, історію запусків, експериментальні сесії, налаштування системи, адміністративних користувачів і сесії авторизації;

е) провайдери LLM – зовнішні сервіси, що використовуються для генерації текстових рекомендацій.

Робота системи починається з того, що користувач вводить ідентифікатор аудиту та запускає аналіз у клієнтській частині. Після цього запит передається до серверної частини. Сервер перевіряє наявність збереженого результату в кеші або реляційній базі даних. Якщо результат відсутній або користувач запускає повторний аналіз, система звертається до зовнішнього сервісу аудиту й отримує вхідні дані.

Після отримання результатів аудиту серверна частина перетворює їх у внутрішній формат, виконує нормалізацію структури даних і виділяє проблемні пункти. Далі для кожної виявленої проблеми формується рекомендація відповідно до вибраного режиму роботи.

Адміністративна частина забезпечує доступ до налаштувань системи, експериментальних сесій, журналу запусків, метрик і керування користувачами. Для захисту цих можливостей передбачено механізм автентифікації адміністративних користувачів.

Після формування рекомендацій система зберігає дані про запуск, використані налаштування, сформовані рекомендації та обчислені метрики. Це

дає змогу переглядати історію запусків, аналізувати окремі результати й порівнювати якість роботи різних режимів.

В поточній реалізації архітектура веб-застосунку поєднує клієнт-серверну модель, базу правил, реляційну та векторну БД та інтеграцію з LLM.

2.3 Вибір технологій для реалізації серверної частини

Для реалізації серверної частини веб-застосунку було обрано набір таких технологій:

а) Python – мова програмування, обрана завдяки розвиненій екосистемі для роботи з API та інструментами штучного інтелекту;

б) FastAPI – фреймворк для створення API [19];

в) Pydantic – бібліотека для опису структур даних, валідації об’єктів та формалізації моделей [20];

г) MySQL – реляційна СКБД для збереження результатів роботи системи та адміністративних даних [21];

ґ) ChromaDB – векторна база даних для збереження векторних подань правил і добору релевантного контексту [22];

д) OpenAI API та Google Gen AI SDK – засоби інтеграції з великими мовними моделями для генерації текстових рекомендацій [23; 24];

е) pytest – інструмент автоматизованого тестування основних модулів серверної частини [25].

Обраний набір технологій відповідає архітектурі веб-застосунку та забезпечує можливість подальшого розширення системи.

2.4 Вибір технологій для реалізації клієнтської частини

Для реалізації клієнтської частини веб-застосунку було обрано наступні технології:

а) JavaScript – мова програмування клієнтської частини, що використовується для реалізації логіки взаємодії користувача з веб-інтерфейсом;

б) Vue.js – фреймворк для побудови інтерфейсу на основі компонентного підходу, реактивного оновлення даних і зручної організації логіки відображення [26];

в) Vue Router – засіб маршрутизації, який забезпечує перехід між сторінками та поділ клієнтської частини на окремі представлення [27];

г) Vite – інструмент збирання та запуску клієнтської частини, що забезпечує швидке середовище розробки й формування фінальної версії застосунку [28];

ґ) Tailwind CSS – інструмент стилізації інтерфейсу за допомогою службових класів, що спрощує підтримку єдиного оформлення сторінок і компонентів [29];

д) HTTP і JSON – засоби обміну даними між клієнтською та серверною частинами за клієнт-серверною моделлю.

Обраний набір технологій забезпечує створення зручного веб-інтерфейсу, логічне розділення сторінок і компонентів та можливість подальшого розширення функціональності застосунку.

2.5 Структура бази даних

Для зберігання даних у веб-застосунку використано реляційну базу даних MySQL, яка містить шість основних таблиць. Загальну структуру БД наведено на рисунку 2.2.

Для збереження читабельності діаграми на рисунку не деталізовано всі однотипні службові поля та поля метрик. Зокрема, у таблиці analysis_run_journal групу полів для збереження окремих метрик якості узагальнено як metric_columns.

Таблиця audit_analysis використовується для збереження останнього сформованого результату аналізу для конкретного аудиту. Основним ключем є audit_id, а результат зберігається у полі result_json.

Таблиця system_settings містить поточні параметри роботи системи.

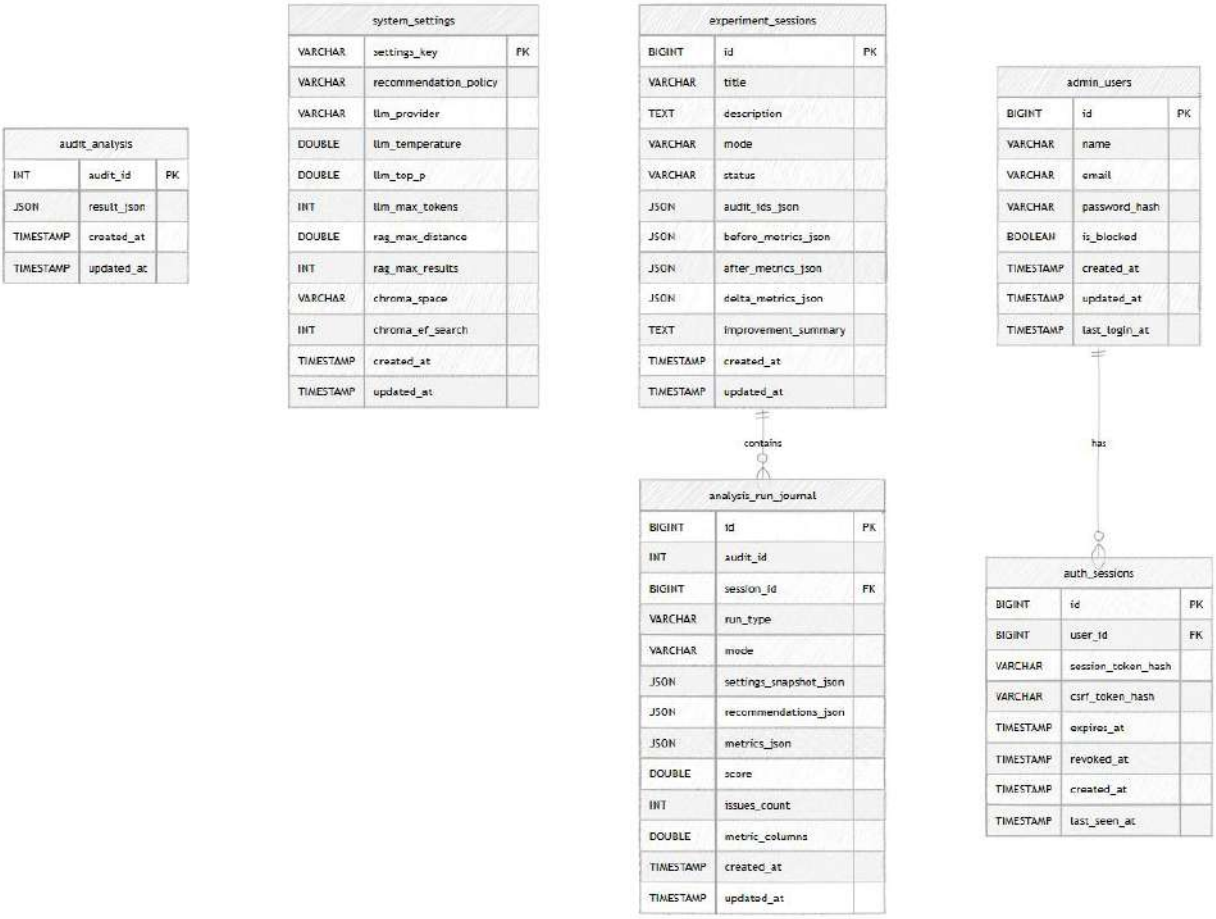

Рисунок 2.2 – Структура реляційної БД

Таблиця experiment_sessions призначена для групування експериментальних запусків.

Таблиця analysis_run_journal зберігає історію окремих запусків аналізу. Якщо запуск належить до експериментальної сесії, він пов’язується з таблицею experiment_sessions через поле session_id.

Для адміністративного доступу використовуються таблиці admin_users, яка містить дані адміністративних користувачів, і auth_sessions, яка зберігає активні сесії доступу, пов’язані з відповідним користувачем.

У схемі передбачено два основні зв’язки: analysis_run_journal.session_id пов’язується з experiment_sessions.id, а auth_sessions.user_id – з admin_users.id. Перший зв’язок дає змогу віднести окремий запуск до експериментальної сесії, другий – пов’язати адміністративного користувача з його сесіями доступу.

В результаті база даних поєднує реляційне подання основних сутностей із гнучким збереженням вкладених даних і забезпечує основу для збереження результатів роботи системи.

2.6 Організація бази правил

У розробленому веб-застосунку база правил зберігається у вигляді JSON-файлу. Кожне правило відповідає певній проблемі, яка може бути виявлена під час аналізу результатів аудиту. Приклад такого правила наведено в лістингу 2.1.

Лістинг 2.1

Приклад структури правила

```
{
  "id": "Фотографії:have_photos_owner",
  "title": "Чи має компанія фотографії від власника у профілі компанії в Google?",
  "description": "Фото від власника показують, що профіль ведеться і контролюється бізнесом. Для Google і клієнтів це сигнал достовірності: реальна локація/команда/процес, а не випадковий набір фото від відвідувачів.",
  "recommendation": "Завантажте базовий набір 10–20 власних фото (реальні зйомки саме вашої локації): 1) фасад/вивіска (2–3 ракурси), 2) вхід і орієнтири (1–2), 3) інтер’єр/робоча зона (2–4), 4) команда в роботі (2–3), 5) процес надання послуги або виробництво (2–4), 6) товари/послуги крупним планом (2–4), 7) приклади результату/до-після (за доречності). Оновлюйте профіль регулярно: 2–4 нових фото на місяць або після змін у приміщенні/послугах. Не використовуйте стокові, чужі або згенеровані зображення, колажі з великим текстом, «банери» чи нерелевантні картинки.",
  "category": "content",
  "priority": "low",
  "impact": "medium",
  "difficulty": "low"
}
```

Поле id використовується для однозначного позначення правила та його зв’язку з відповідною перевіркою аудиту. Поле title містить текст перевірки, description – пояснення суті проблеми, а recommendation – підготовлений текст рекомендації.

Поля category, priority, impact і difficulty виконують допоміжну роль і використовуються для додаткової класифікації правила. Така структура дає змогу зберігати правила в уніфікованому форматі, спрощує їх подальше опрацювання та забезпечує узгоджене подання рекомендацій у системі.

2.7 Використання векторної бази даних для добору релевантного контексту

У межах розробленого веб-застосунку для добору релевантного контексту використовується ChromaDB. Цей компонент виконує роль векторного індексу для підготовлених правил і дає змогу знаходити записи, змістово близькі до виявленої проблеми аудиту [22].

Під час ініціалізації системи правила зчитуються з JSON-файлу, після чого з їхніх основних текстових полів формується текст для індексації. Далі цей текст перетворюється у векторне подання та зберігається в колекції ChromaDB разом із метаданими правила. Для формування векторних подань тексту використано модель all-MiniLM-L6-v2 [30].

Під час аналізу аудиту система формує текстовий запит на основі виявленої проблеми. Цей запит також перетворюється у векторне подання, після чого виконується пошук найближчих правил. Отримані результати надалі використовуються відповідно до вибраного режиму формування рекомендацій.

Для керування добором релевантного контексту в системі передбачено такі параметри:

а) rag_max_results – визначає максимальну кількість правил, які можуть бути повернуті під час пошуку;

б) rag_max_distance – задає допустиму межу відстані між вектором запиту та знайденим правилом;

в) chroma_space – визначає тип векторного простору подібності: cosine для косинусної схожості, l2 – евклідової відстані, ip – внутрішнього добутку;

г) chroma_ef_search – параметр наближеного пошуку, який впливає на кількість кандидатів, що аналізуються під час пошуку [14].

Параметри добору контексту не є жорстко закріпленими в програмному коді й можуть змінюватися через налаштування системи. Це дає змогу експериментально перевіряти вплив різних конфігурацій на якість сформованих рекомендацій.

3 Реалізація системи формування рекомендацій

3.1 Обробка результатів аудиту та виділення виявлених проблем

Першим етапом роботи системи є обробка результатів аудиту. Оскільки вхідні дані мають вкладену JSON-структуру, перед подальшим опрацюванням вони приводяться до єдиного внутрішнього формату.

Отримання даних виконується за ідентифікатором аудиту. Після цього отриманий JSON передається на етап нормалізації. Для кожної перевірки формується внутрішній об’єкт, що містить ідентифікатор, назву, статус проходження, вагу, пояснення та інші службові дані.

Статус перевірки визначається на основі вибраної відповіді та кількості балів, пов’язаних із нею. Якщо відповідь має додатне або нульове значення балів, перевірка вважається пройденою. Якщо кількість балів є від’ємною, така перевірка розглядається як проблемна. Вага проблеми формується на основі абсолютного значення бала.

Після нормалізації система відбирає лише перевірки з негативним результатом. На їх основі формуються об’єкти проблем, що містять назву, опис, ідентифікатор, вагу та рівень важливості. Ці дані використовуються для подальшого формування рекомендацій і відображення результатів користувачу.

3.2 Реалізація механізму добору релевантного контексту

Механізм добору релевантного контексту використовується після виділення проблем аудиту. Його призначення полягає в пошуку правил, які найбільш точно відповідають змісту виявленої проблеми. На цьому етапі система визначає інформаційну основу для подальшої обробки.

Для пошуку релевантних правил система формує текстовий запит на основі назви проблеми та її опису. Після цього виконується пошук у векторній БД. Результатом є перелік найближчих за змістом правил із відповідними значеннями відстані між запитом і знайденими записами.

У системі використовуються два типи зіставлення. Перший передбачає точний збіг між ідентифікатором проблеми та ідентифікатором правила. Він застосовується тоді, коли для конкретної перевірки аудиту вже підготовлено пряме правило. Другий тип має пріоритет над першим, оскільки ґрунтується на семантичній близькості та дає змогу знаходити змістовно подібні правила.

Відібрані правила використовуються залежно від режиму роботи системи. У режимі формування на основі бази правил знайдене правило може бути основою відповіді. У режимі RAG знайдені правила передаються до LLM як контекст. У режимі LLM добір правил не виконується.

В розробленому застосунку механізм добору контексту забезпечує зв’язок між проблемами аудиту та підготовленими правилами, поєднуючи точне зіставлення з пошуком за змістовою близькістю.

3.3 Інтеграція LLM у процес формування рекомендацій

Інтеграція LLM у системі реалізована як окремий етап формування текстової відповіді. На цьому етапі система передає до моделі підготовлені дані про виявлену проблему, а модель формує структуровану рекомендацію. Такий підхід розмежовує попередню обробку аудиту та добір контексту від безпосередньої генерації відповіді.

Зміст запиту до LLM залежить від обраного режиму роботи. У режимі використання лише LLM до запиту передаються назва проблеми, її опис, рівень важливості та службові характеристики. У режимі RAG до цих даних додатково додаються релевантні правила, знайдені на попередньому етапі, що забезпечує формування відповіді з урахуванням підготовленого контексту. Приклади шаблонів запитів для цих режимів наведено в додатках В і Г.

Поведінка генерації регулюється такими параметрами:

а) llm_provider – визначає постачальника LLM;

б) llm_temperature – впливає на варіативність відповіді;

в) llm_top_p – обмежує набір імовірних варіантів продовження тексту;

г) llm_max_tokens – визначає максимальний обсяг відповіді.

Важливою вимогою є структурований формат результату. Відповідь моделі повинна містити текст рекомендації та службові характеристики. Це потрібно для подальшого програмного опрацювання, відображення результату в клієнтській частині та оцінювання якості рекомендацій.

Оскільки звернення до LLM залежить від зовнішнього сервісу, у системі передбачено обробку помилок. Якщо модель недоступна, повертає некоректний результат або відповідь не відповідає очікуваній структурі, застосовується резервний сценарій формування відповіді.

3.4 Режими формування рекомендацій у системі

У розробленому веб-застосунку передбачено кілька режимів формування рекомендацій. Вибір режиму визначається параметром recommendation_policy.

У системі реалізовано такі режими роботи:

а) режим бази правил – рекомендація створюється без звернення до LLM. Для виявленої проблеми система шукає відповідне правило та використовує його як основу фінальної відповіді;

б) режим RAG – система спочатку добирає релевантні правила, після чого передає їх до LLM як контекст. У цьому режимі відповідь формується на основі опису проблеми та знайдених правил;

в) режим LLM – рекомендація формується без добору контексту. До моделі передаються назва проблеми, її опис, рівень важливості та службові характеристики.

Для забезпечення стабільності роботи передбачено резервний сценарій. У такому випадку система послідовно переходить до альтернативних варіантів:

1) використання відповіді LLM;
2) використання найбільш релевантного правила з переліку кандидатів;
3) використання точного правила за ідентифікатором перевірки аудиту;
4) формування базової поради.

Такий механізм дає змогу продовжити обробку аудиту навіть у разі збою окремого компонента та забезпечує отримання результату з урахуванням доступних джерел інформації.

3.5 Реалізація серверної частини веб-застосунку

Серверна частина веб-застосунку реалізована як центральний рівень обробки запитів і координації основних компонентів системи. Вона приймає HTTP-запити від клієнтської частини, виконує необхідну прикладну логіку та повертає структуровані відповіді у форматі JSON.

У межах серверної частини виокремлено кілька логічних напрямів:

а) обробка запитів на аналіз аудиту та повернення результатів;

б) взаємодія із зовнішнім сервісом аудиту для отримання вхідних даних;

в) передавання результатів аудиту до модулів нормалізації, виділення проблем і формування рекомендацій;

г) робота з реляційною БД для зберігання результатів, історії запусків, налаштувань і службових даних;

ґ) обробка адміністративних запитів.

Функціональність застосунку поділена на публічну і службову частини. Публічна частина забезпечує запуск аналізу та перегляд результатів. Адміністративна частина доступна лише після автентифікації та використовується для керування налаштуваннями системи та користувачами, перегляду історії запусків, проведення експериментів.

3.6 Реалізація клієнтської частини веб-застосунку

Клієнтська частина веб-застосунку реалізована як інтерфейс, що приховує складність серверної обробки та забезпечує зрозумілий сценарій роботи користувача із системою.

Основна сторінка, яка зображена на рисунку 3.1, містить форму введення ідентифікатора аудиту. Після надсилання запиту клієнтська частина отримує структуровану відповідь від сервера й оновлює інтерфейс без перезавантаження

сторінки. Також передбачено відображення стану завантаження та повідомлень про помилки.

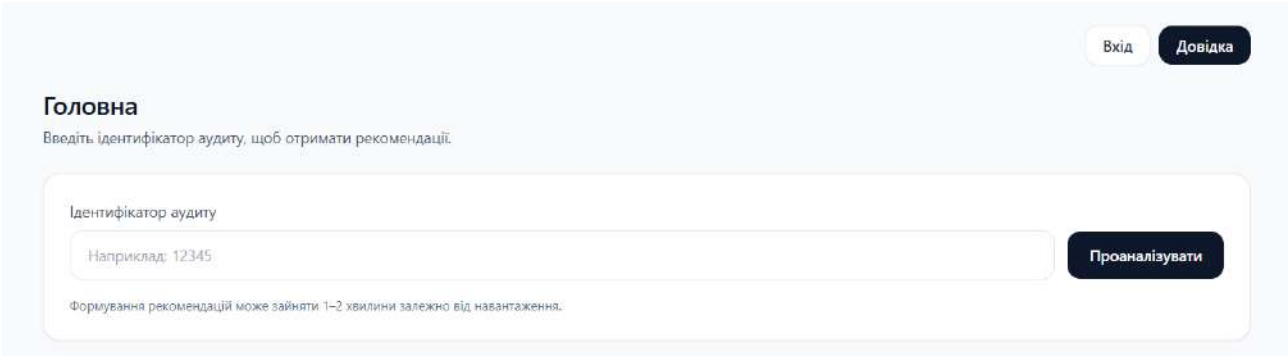

Рисунок 3.1 – Головна сторінка запуску аналізу аудиту

Після завершення аналізу результат подається у вигляді окремих блоків: загальна інформація про аудит, перелік виявлених проблем зі сформованими рекомендаціями. Сторінка з рекомендаціями зображена на рисунку 3.2.

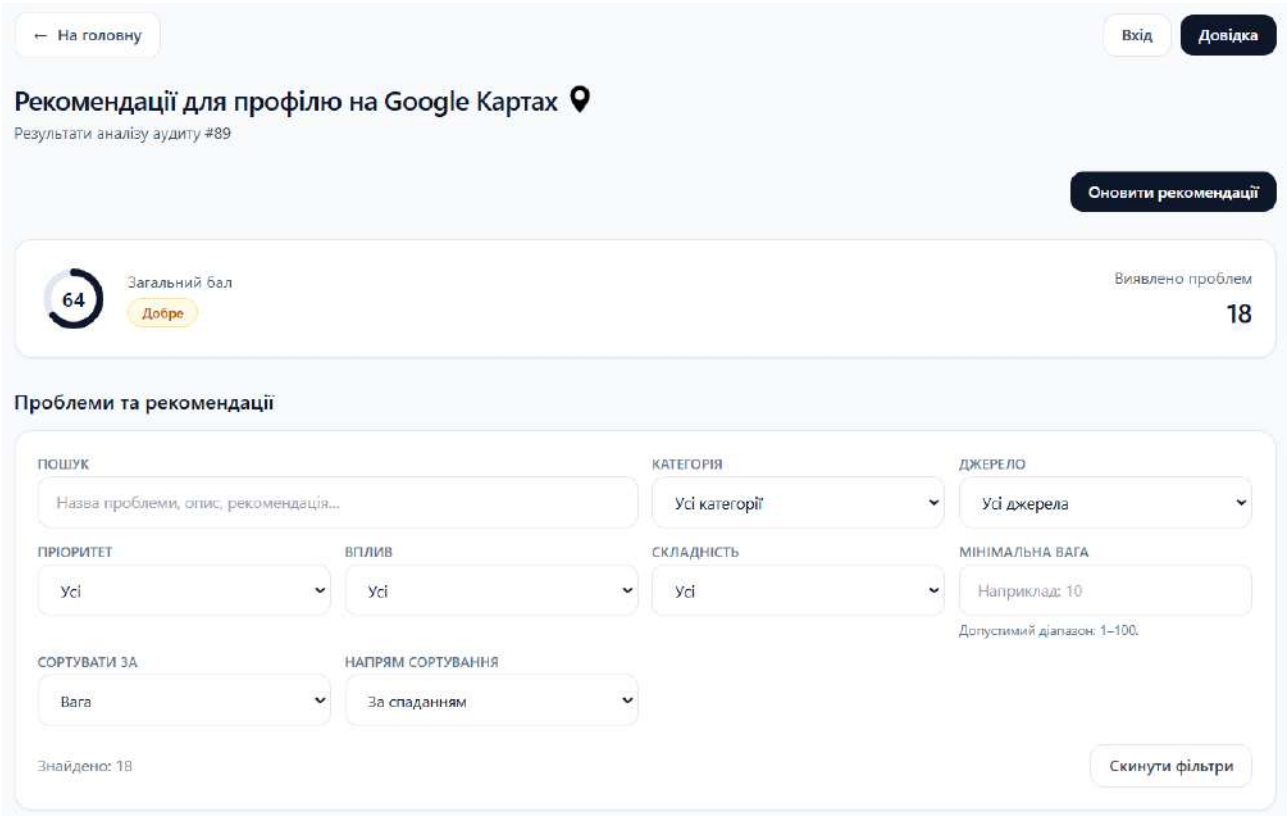

Рисунок 3.2 – Відображення сформованих рекомендацій

Окрема частина інтерфейсу призначена для службових можливостей. Після автентифікації адміністративний користувач отримує доступ до налаштувань системи, інтерфейс яких наведено на рисунку 3.3, до журналу запусків, експериментальних сесій, сторінок детального перегляду результатів та панелі керування користувачами.

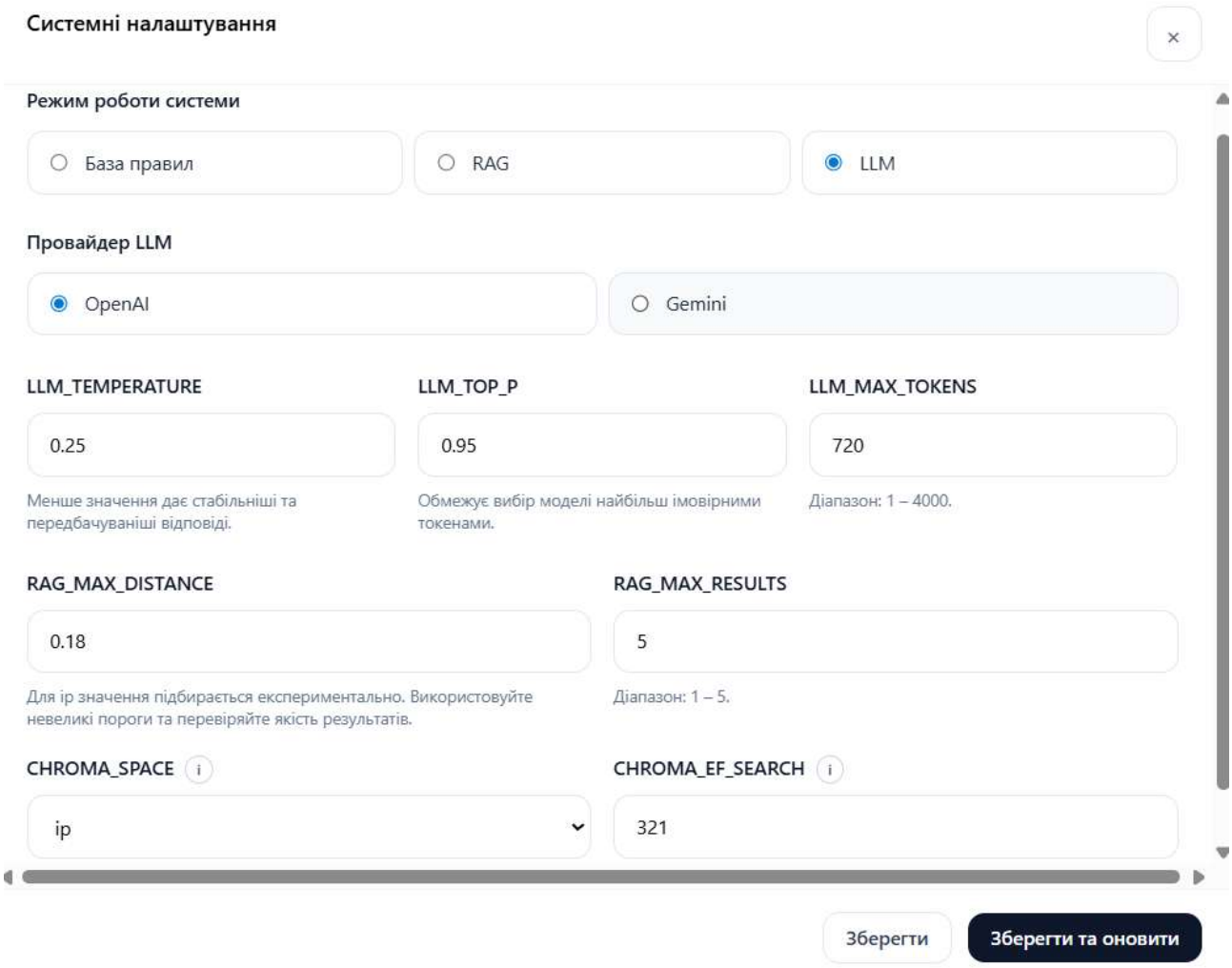

Рисунок 3.3 – Інтерфейс налаштування параметрів роботи системи

Для аналізу збережених результатів реалізовано сторінку історії запусків, представлену на рисунку 3.4. Вона дає змогу переглядати попередні аналізи, відкривати детальну інформацію про окремий запуск, бачити використані параметри, сформовані рекомендації та метрики якості.

Клієнтська частина забезпечує повний цикл взаємодії із системою: запуск аналізу, перегляд результатів, зміну параметрів і роботу з історією запусків.

Рисунок 3.4 – Перегляд історії запусків аналізу та детальних результатів

3.7 Налаштування параметрів роботи системи

У веб-застосунку передбачено централізований механізм налаштування параметрів роботи системи. Його призначення полягає в тому, щоб змінювати поведінку основних компонентів без редагування програмного коду та повторного розгортання застосунку.

Під час запуску серверна частина завантажує збережену конфігурацію з БД та використовує її під час подальшої обробки аудитів. Якщо окремі значення відсутні, система може застосувати початкові параметри, визначені в конфігурації середовища.

Після збереження нові значення передаються на сервер, проходять перевірку коректності та записуються до БД.

В системі реалізовано можливість повторного аналізу вже обробленого аудиту з оновленими параметрами. Це дає змогу порівнювати результати, отримані за різних варіантів конфігурації системи.

3.8 Збереження результатів аналізу та журнал запусків

Реалізація механізму збереження результатів аналізу аудиту та журналу запусків дає змогу повторно переглядати сформовані результати, фіксувати умови їх отримання та використовувати ці дані для порівняння різних режимів роботи системи.

Збереження організовано у двох напрямах. Перший передбачає збереження останнього результату аналізу для конкретного аудиту, що дозволяє повторно отримати актуальні дані. Другий напрям пов’язаний із журналом запусків, у якому кожен аналіз фіксується як окремий запис незалежно від того, чи оброблявся цей аудит раніше.

Якщо аналіз виконується в межах експериментальної сесії, запис журналу пов’язується з відповідною сесією. Це дає змогу групувати результати кількох запусків і надалі порівнювати їх під час експериментального дослідження.

Складні об’єкти, зокрема рекомендації, знімки параметрів і набори метрик, зберігаються у форматі JSON.

Таким чином, журнал запусків забезпечує відтворюваність аналізу та створює основу для подальшого оцінювання якості роботи системи.

4 Експериментальне дослідження та оцінювання результатів

4.1 Методика експериментального дослідження

Експериментальне дослідження спрямоване на перевірку якості рекомендацій, сформованих за різних режимів роботи веб-застосунку. Основна увага приділяється тому, наскільки відповідь відповідає виявленій проблемі аудиту, містить достатнє пояснення, є практично корисною для користувача та спирається на релевантний контекст.

Вхідними даними є готові результати аудиту. Вони містять перелік перевірок, статус проходження кожної перевірки, вагу окремих пунктів і пояснювальні дані. Після виділення проблемних пунктів кожен із них обробляється в одному з трьох режимів формування рекомендацій.

Загальна послідовність експериментального дослідження передбачає:

1) отримання результату аудиту із зовнішнього джерела;
2) нормалізацію структури отриманих даних;
3) виділення проблемних пунктів аудиту;
4) запуск формування рекомендацій у заданому режимі;
5) збереження результату запуску;
6) обчислення метрик якості;
7) аналіз отриманих значень і порівняння результатів між режимами.

Для забезпечення відтворюваності експериментів у системі фіксуються параметри кожного запуску: ідентифікатор аудиту, обраний режим формування рекомендацій, налаштування RAG та LLM, сформовані рекомендації, службові дані та значення обчислених метрик.

Окремим етапом є автоматичний підбір параметрів для кожного режиму роботи. Для цього використано бібліотеку Optuna, призначену для автоматичного підбору параметрів за підходом define-by-run [31]. У роботі застосовано TPESampler, який пропонує нові конфігурації на основі результатів попередніх запусків [32]. Для дострокового припинення неперспективних спроб

використано HyperbandPruner, що дає змогу оцінювати проміжні результати на частині аудитів і зупиняти конфігурації з низьким проміжним балом [33].

Алгоритм підбору параметрів реалізовано за такою схемою:

1) задається фіксована вибірка аудитів;
2) для кожного режиму визначається власний простір параметрів;
3) Optuna створює нову спробу та пропонує значення параметрів;
4) система виконує проміжне оцінювання конфігурації;
5) неперспективні спроби можуть бути зупинені достроково;
6) решта конфігурацій оцінюється на повній вибірці;
7) для кожного запуску обчислюються метрики й підсумковий оптимізаційний бал;
8) конфігурація з найбільшим балом визначається як найкраща для відповідного режиму.

Для режимів LLM та RAG в якості провайдера великої мовної моделі використовувався OpenAI, оскільки модель Google Gemini не впоралась з експериментальним навантаженням і викликала збій роботи.

Оцінювання виконується окремо для кожного режиму. Для режиму на основі бази правил аналізуються покриття проблем, стабільність і структурованість відповіді. Для режиму LLM оцінюється здатність моделі сформувати змістовну рекомендацію без додаткового контексту. Для режиму RAG додатково перевіряється, чи використання дібраних правил підвищує релевантність, повноту та обґрунтованість відповіді.

Головним етапом дослідження є зіставлення результатів режимів LLM і RAG для однакових проблемних пунктів аудиту. Це дає змогу визначити, чи покращує використання дібраного контексту якість сформованих рекомендацій порівняно з генерацією без такого контексту.

4.2 Метрики оцінювання якості сформованих рекомендацій

Для оцінювання результатів роботи веб-застосунку використано набір метрик, який характеризує роботу всіх трьох режимів роботи системи.

У подальших формулах через I позначено множину виявлених проблем аудиту, а через $N = |I|$ – кількість таких проблем. Для метрик, які обчислюються лише за відповідями, сформованими LLM, використовується підмножина $L \subseteq I$. Через p_i позначено текст проблеми з номером i, через a_i – текст сформованої відповіді, через r_i – текст правила, використаного для цієї проблеми.

Для оцінювання семантичної близькості між двома текстами використовується косинусна подібність між їхніми векторними поданнями.

$$sim(x, y) = \frac{v(x) \cdot v(y)}{\|v(x)\| \cdot \|v(y)\|}$$

де x і y – текстові фрагменти, $v(x)$ і $v(y)$ – їхні векторні подання. Значення $sim(x, y)$ є більшим тоді, коли тексти мають ближчий зміст.

4.2.1 Метрики режиму формування рекомендацій на основі бази правил

Частка проблем із знайденими кандидатами показує, для якої частки проблем система знайшла хоча б одне потенційно релевантне правило. Позначимо цю метрику як RCR.

$$RCR = \frac{|\{i \in I : c_i > 0\}|}{N}$$

де c_i – кількість знайдених кандидатних правил для проблеми i. Що вище значення RCR, то краще база правил покриває проблеми, які виникають у результатах аудиту.

Покриття обґрунтованими відповідями показує, для якої частки проблем фінальна рекомендація була сформована на основі правила, а не через резервну відповідь. Позначимо цю метрику як GC.

$$GC = \frac{\left|\{i \in I: source_i \in \{rule_{base}, rules, rag\}\}\right|}{N}$$

де $source_i$ – джерело фінальної відповіді для проблеми i. У цьому режимі метрика GC показує, наскільки часто система змогла використати змістовне правило замість загального резервного повідомлення.

Частка резервних відповідей показує, як часто система не змогла використати відповідне правило і повернула базову відповідь. Позначимо цю метрику як FR.

$$FR = \frac{|\{i \in I: source_i = fallback\}|}{N}$$

Низьке значення FR свідчить про те, що для більшості проблем було знайдено придатне правило.

Середній обернений ранг точного правила показує, наскільки високо серед знайдених кандидатів розташовується точне правило, якщо воно існує. Позначимо цю метрику як MRR.

$$MRR = \left(\frac{1}{|E|}\right) \cdot \sum_{i \in E} \left(\frac{1}{rank_i}\right)$$

де E – множина проблем, для яких у базі правил існує точне правило, а $rank_i$ – позиція точного правила серед знайдених кандидатів. Якщо точне правило існує, але не було знайдене серед кандидатів, його внесок у суму вважається нульовим.

Доступність точного правила показує, для якої частки проблем у базі правил є правило з відповідним ідентифікатором перевірки. Позначимо цю метрику як $EMAR$.

$$EMAR = \frac{|\{i \in I: e_i = 1\}|}{N}$$

де $e_i = 1$ означає, що для проблеми i у базі правил існує точне правило.

Частка використання точного правила показує, для якої частки проблем фінально було використано саме точне правило. Позначимо цю метрику як $ERUR$.

$$ERUR = \frac{\left|\left\{i \in I: e_i = 1 \wedge r_i^{final} = r_i^{exact}\right\}\right|}{N}$$

де r_i^{final} – правило, використане у фінальній відповіді, а r_i^{exact} – точне правило, пов'язане з відповідною перевіркою аудиту.

Середня релевантність правила до проблеми визначає семантичну відповідність між текстом проблеми та текстом використаного правила. Позначимо цю метрику як $MRRel$.

$$MRRel = \left(\frac{1}{|U|}\right) \cdot \sum_{i \in U} sim(p_i, r_i)$$

де U – множина проблем, для яких фінально було використано правило. Значення цієї метрики показує, наскільки зміст правила відповідає виявленій проблемі.

Середня зважена повнота правила оцінює, чи містить повернуте правило основні структурні елементи. Для окремого правила повнота обчислюється так:

$$RC_i = 0.25x_i + 0.35s_i + 0.20v_i + 0.20m_i$$

де x_i – наявність пояснення, s_i – наявність практичних кроків, v_i – наявність перевірки результату, m_i – наявність типової помилки або

застереження. Кожен із цих показників набуває значення 1, якщо відповідний елемент присутній, і 0 – якщо відсутній.

Середнє значення цієї метрики для всіх використаних правил позначимо як $MWRC$.

$$MWRC = \left(\frac{1}{|U|}\right) \cdot \sum_{i \in U} RC_i$$

Для узагальнення результатів запуску в режимі бази правил використовується підсумковий оптимізаційний бал S_{rule}. Він поєднує кілька основних метрик і враховує штраф за резервні відповіді. Оскільки значення семантичної подібності теоретично може виходити за межі додатного інтервалу, у підсумковому балі використовується нормалізоване значення $MRRel$, обмежене діапазоном від 0 до 1.

$$MRRel^{+} = \min(1, \max(0, MRRel))$$

$$S_{rule} = \min(1, \max(0, 0.30GC + 0.25MRRel^{+} + 0.20RCR + 0.15MWRC + 0.10MRR - 0.20FR))$$

Показник S_{rule} використовується як узагальнений індикатор якості конкретного запуску режиму бази правил.

4.2.2 Метрики режиму формування рекомендацій на основі LLM

Для режиму на основі LLM використовуються метрики двох типів. Перша група характеризує технічну стабільність звернення до моделі, а друга – якість сформованого тексту.

Частка доступності LLM показує, для якої частки проблем модель була доступна для генерації відповіді. Позначимо цю метрику як LAR.

$$LAR = \frac{|\{i \in I: q_i \neq unavailable\}|}{N}$$

де q_{i} – службовий статус роботи LLM для проблеми i.

Частка спроб виклику LLM показує, для якої частки проблем система фактично виконала звернення до моделі. Позначимо цю метрику як $LATR$.

$$LATR = \frac{|\{i \in I: t_i = 1\}|}{N}$$

де $t_{\text{i}} = 1$ означає, що для проблеми i було здійснено спробу виклику LLM.

Частка успішних відповідей LLM обчислюється серед усіх виконаних спроб звернення до моделі. Позначимо її як LSR.

$$LSR = \frac{|\{i \in I: t_i = 1 \wedge q_i = success\}|}{|\{i \in I: t_i = 1\}|}$$

Частка фінальних LLM-відповідей показує, для якої частки проблем фінальна рекомендація була сформована саме великою мовною моделлю. Позначимо її як $LFUR$.

$$LFUR = \frac{|\{i \in I: source_i = llm\}|}{N}$$

Частка резервних відповідей після LLM показує, як часто після невдалої генерації система була змушена перейти до резервної відповіді. Позначимо цю метрику як FAL.

$$FAL = \frac{\left|\{i \in I: f_i^{LLM} = 1\}\right|}{N}$$

де $f_i^{LLM} = 1$ означає, що після спроби роботи з LLM для проблеми i було використано резервну відповідь.

Середня релевантність відповіді визначає семантичну відповідність між текстом виявленої проблеми та сформованою відповіддю. Позначимо цю метрику як MAR.

$$MAR = \left(\frac{1}{|L|}\right) \cdot \sum_{i \in L} sim(p_i, a_i)$$

де L – множина проблем, для яких відповідь була сформована LLM.

Покриття проблем достатньо релевантними відповідями показує, для якої частки всіх проблем було сформовано відповідь, релевантність якої не нижча за встановлений поріг τ. Позначимо цю метрику як PCR.

$$PCR = \frac{|\{i \in L: sim(p_i, a_i) \geq \tau\}|}{N}$$

де τ – порогове значення релевантності.

Середня зважена повнота відповіді оцінює, чи містить відповідь основні структурні елементи. Для окремої відповіді повнота обчислюється так:

$$AC_i = 0.25x_i + 0.35s_i + 0.20v_i + 0.20m_i$$

де x_i – наявність пояснення, s_i – наявність практичних кроків, v_i – наявність елемента перевірки результату, m_i – наявність застереження щодо типової помилки.

Середнє значення цієї метрики для всіх фінальних LLM-відповідей позначимо як $MWAC$.

$$MWAC = \left(\frac{1}{|L|}\right) \cdot \sum_{i \in L} AC_i$$

Частка обґрунтованих відповідей показує, для якої частки фінальних відповідей знайдено достатню семантичну опору в базі правил. Для цього спочатку обчислюється бал обґрунтованості окремої відповіді GS_i.

$$GS_i = 0.40sim\left(p_i, r_i^p\right) + 0.60sim(a_i, r_i^s)$$

де r_i^p – частина правила, що описує проблему, а r_i^s – частина правила, яка містить пояснення або рекомендаційний зміст.

Частка обґрунтованих відповідей позначається як GAR.

$$GAR = \frac{|\{i \in L: GS_i \geq \tau\}|}{|L|}$$

Середній бал обґрунтованості показує середній рівень семантичної опори відповідей на правила. Позначимо цю метрику як MGS.

$$MGS = \left(\frac{1}{|L|}\right) \cdot \sum_{i \in L} GS_i$$

Для узагальнення результатів запуску LLM-режиму використовується підсумковий оптимізаційний бал S_{LLM}.

$$S_{LLM} = \min\big(1, \max\big(0, 0.25GAR + 0.20MGS + 0.15MAR + 0.10PCR + 0.10MWAC + 0.10LSR + 0.05LFUR + 0.05(1 - FAL)\big)\big)$$

Цей показник слід розглядати разом з окремими метриками, оскільки однаковий бал S_{LLM} може відповідати різному співвідношенню їхніх значень.

4.2.3 Метрики режиму формування рекомендацій на основі RAG

У режимі RAG частина метрик обчислюється за тим самим принципом, що й у LLM-режимі, оскільки фінальна відповідь також формується великою мовною моделлю. До таких метрик належать: MAR, PCR, $MWAC$, GAR і MGS. Їхні формули наведено в описі LLM-режиму, однак у режимі RAG вони застосовуються до множини відповідей, сформованих після попереднього добору релевантного контексту.

Специфічною для RAG є метрика частки обґрунтованих RAG-відповідей. Вона показує, для якої частки фінальних відповідей використане правило входило до множини кандидатів, попередньо знайдених механізмом добору релевантного контексту. Позначимо цю метрику як $GRAR$.

$$GRAR = \frac{\left|\left\{i \in L_R : r_i^{final} \in C_i\right\}\right|}{|L_R|}$$

де L_R – множина фінальних LLM-відповідей у режимі RAG, C_i – множина RAG-кандидатів для проблеми i, а r_i^{final} – правило, з яким пов’язана фінальна відповідь.

Частка резервних відповідей після наявного RAG-контексту показує, як часто система переходила до резервної відповіді навіть тоді, коли для проблеми було знайдено хоча б одне кандидатне правило. Позначимо цю метрику як $FARG$.

$$FARG = \frac{|\{i \in I : source_i = fallback \wedge c_i > 0\}|}{N}$$

де $c_\text{i} > 0$ означає, що для проблеми i було знайдено хоча б один RAG-кандидат.

Таким чином, набір метрик побудовано так, щоб окремо оцінювати контрольований режим на основі бази правил, режим формування відповідей на

основі LLM та режим RAG. Ці метрики надалі використовуються для аналізу результатів експериментальних запусків і визначення впливу RAG на якість сформованих рекомендацій.

4.3 Оцінювання режиму формування рекомендацій на основі бази правил

Для оцінювання режиму формування рекомендацій на основі бази правил було використано вибірку з 20 аудитів. У межах експерименту виконано 17 завершених запусків, 13 запусків було зупинено достроково. Найкращу конфігурацію параметрів наведено в таблиці 4.1.

Таблиця 4.1 – Найкраща конфігурація режиму формування рекомендацій на основі бази правил

Параметр	**Значення**
Максимальна відстань пошуку	0,4821
Максимальна кількість відібраних правил	5
Простір векторного пошуку	cosine
Параметр пошуку ef search	25
Підсумковий оптимізаційний бал	88,89 %

На рисунку 4.1 показано динаміку основних метрик за всіма завершеними запусками експерименту. Горизонтальна вісь відповідає порядковому номеру завершеного запуску, а вертикальна вісь відображає значення метрик у нормалізованій шкалі від 0 до 1, де 1 відповідає 100 %.

Як видно з рисунка 4.1, більшість метрик залишалася стабільною протягом усіх завершених запусків. Частка проблем із знайденими кандидатними правилами, покриття обґрунтованими відповідями та доступність точного правила мали високі значення, тоді як частка резервних відповідей і частка використання точного правила залишалися на нульовому рівні. Незначні коливання спостерігалися переважно для середнього оберненого рангу точного правила та підсумкового оптимізаційного бала, що свідчить про обмежений вплив зміни параметрів пошуку на результат у цьому режимі.

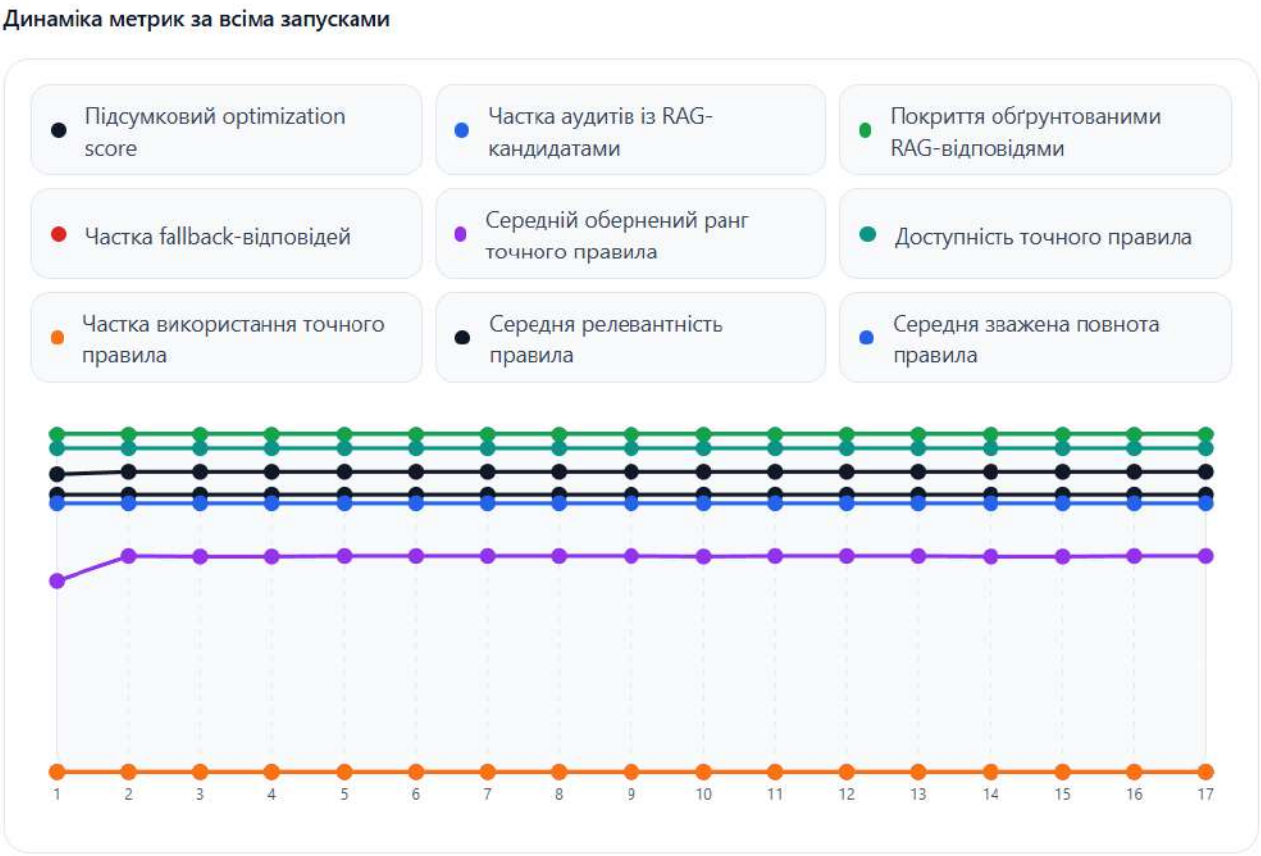

Рисунок 4.1 – Динаміка метрик режиму формування рекомендацій на основі бази правил за всіма завершеними запусками

Зведені результати оцінювання наведено в таблиці 4.2.

Таблиця 4.2 – Результати оцінювання режиму формування рекомендацій на основі бази правил

Метрика	**Мінімум**	**Середнє**	**Максимум**	**Найкращий запуск**
Частка проблем із знайденими кандидатними правилами	100,00 %	100,00 %	100,00 %	100,00 %
Покриття обґрунтованими відповідями	100,00 %	100,00 %	100,00 %	100,00 %
Частка резервних відповідей	0,00 %	0,00 %	0,00 %	0,00 %
Середній обернений ранг точного правила	0,5659	0,6342	0,6390	0,6390

Продовження таблиці 4.2

Метрика	Мінімум	Середнє	Максимум	Найкращий запуск
Доступність точного правила	95,78 %	95,78 %	95,78 %	95,78 %
Частка використання точного правила	0,00 %	0,00 %	0,00 %	0,00 %
Середня релевантність правила до проблеми	82,20 %	82,20 %	82,20 %	82,20 %
Середня зважена повнота правила	79,67 %	79,67 %	79,67 %	79,67 %
Підсумковий оптимізаційний бал	88,16 %	88,84 %	88,89 %	88,89 %

Отримані результати свідчать, що режим формування рекомендацій на основі бази правил працює стабільно та передбачувано. Для всіх виявлених проблем система знаходила кандидатні правила, тому резервні відповіді не використовувалися. Це підтверджує достатнє покриття перевірок аудиту підготовленими правилами.

Нульова частка використання точного правила пояснюється логікою реалізації системи, у якій фінальне правило визначається через механізм добору кандидатів. Водночас значення середнього оберненого рангу показує, що точні правила здебільшого потрапляли на достатньо високі позиції серед знайдених кандидатів.

Середня релевантність підтверджує змістову відповідність дібраних правил виявленим проблемам, а середня зважена повнота свідчить про наявність основних структурних елементів у більшості правил. Подальше покращення цього режиму доцільно пов’язувати з розширенням, уніфікацією та деталізацією бази правил.

4.4 Оцінювання режиму формування рекомендацій на основі LLM

Для оцінювання режиму формування рекомендацій на основі LLM було використано ту саму вибірку з 20 аудитів, що й у попередньому експерименті.

У межах оптимізації завершено 4 повні запуски, 11 запусків було зупинено достроково. Найкращу конфігурацію параметрів наведено в таблиці 4.3.

Таблиця 4.3 – Найкраща конфігурація режиму формування рекомендацій на основі LLM

Параметр	Значення
Провайдер LLM	OpenAI
Температура LLM	0,0344
Параметр Top p	0,9547
Максимальна кількість токенів	850
Підсумковий оптимізаційний бал	85,90 %

На рисунку 4.2 наведено динаміку основних метрик за всіма завершеними запусками LLM-режиму.

Рисунок 4.2 – Динаміка метрик режиму формування рекомендацій на основі LLM за всіма завершеними запусками

Зведені результати оцінювання наведено в таблиці 4.4.

Таблиця 4.4 – Результати оцінювання режиму формування рекомендацій на основі LLM

Метрика	Мінімум	Середнє	Максимум	Найкращий запуск
Доступність LLM	100,00 %	100,00 %	100,00 %	100,00 %
Частка спроб виклику LLM	100,00 %	100,00 %	100,00 %	100,00 %
Частка успішних відповідей LLM	98,43 %	98,77 %	99,01 %	98,43 %
Частка фінальних LLM-відповідей	98,43 %	98,77 %	99,01 %	98,43 %
Частка резервних відповідей після LLM	0,99 %	1,23 %	1,57 %	1,57 %
Середня релевантність відповіді	75,83 %	76,06 %	76,25 %	76,25 %
Покриття проблем	53,71 %	55,01 %	56,35 %	54,92 %
Середня зважена повнота відповіді	100,00 %	100,00 %	100,00 %	100,00 %
Частка обґрунтованих відповідей	88,58 %	89,57 %	90,74 %	90,74 %
Середній бал обґрунтованості	82,81 %	82,98 %	83,06 %	83,02 %
Підсумковий оптимізаційний бал	85,26 %	85,65 %	85,90 %	85,90 %

Отримані результати показують, що модель була доступною в усіх запусках, а частка успішних відповідей залишалася високою. Резервні відповіді використовувалися рідко, що свідчить про незначну кількість випадків, коли згенеровану відповідь не можна було використати як фінальну.

Середня релевантність відповідей залишалася близькою в усіх завершених запусках, а середня зважена повнота досягала максимального значення. Водночас покриття проблем було нижчим за середню релевантність.

Частка обґрунтованих відповідей і середній бал обґрунтованості показують, що більшість відповідей мала семантичну опору на підготовлені правила навіть без попереднього добору контексту. Отже, LLM-режим є гнучким і технічно стабільним, але менш контрольованим, оскільки частина відповідей не досягає потрібного рівня релевантності до виявленої проблеми.

4.5 Оцінювання режиму формування рекомендацій на основі RAG

Для оцінювання режиму формування рекомендацій на основі RAG було використано вибірку з 20 аудитів, що й в попередніх експериментах. У межах експерименту завершено 4 повні запуски, 10 запусків було зупинено достроково, один запуск завершився помилкою. Найкращу конфігурацію параметрів наведено в таблиці 4.5.

Таблиця 4.5 – Найкраща конфігурація режиму формування рекомендацій на основі RAG

Параметр	**Значення**
Максимальна відстань пошуку	0,4894
Максимальна кількість відібраних правил	5
Простір векторного пошуку	cosine
Параметр ef search	25
Провайдер LLM	OpenAI
Температура LLM	0,9219
Параметр Top p	0,5442
Максимальна кількість токенів	1100
Підсумковий оптимізаційний бал	86,06 %

На рисунку 4.3 наведено динаміку ключових метрик RAG-режиму за всіма завершеними запусками.

Зведені результати оцінювання наведено в таблиці 4.6.

Отримані результати показують, що RAG-режим забезпечив стабільне формування відповідей із високим рівнем технічної доступності LLM. Модель була доступною в усіх завершених запусках, а частка успішних фінальних відповідей залишалася високою. Частка резервних відповідей після LLM була незначною, а резервні відповіді після етапу добору RAG-контексту не використовувалися.

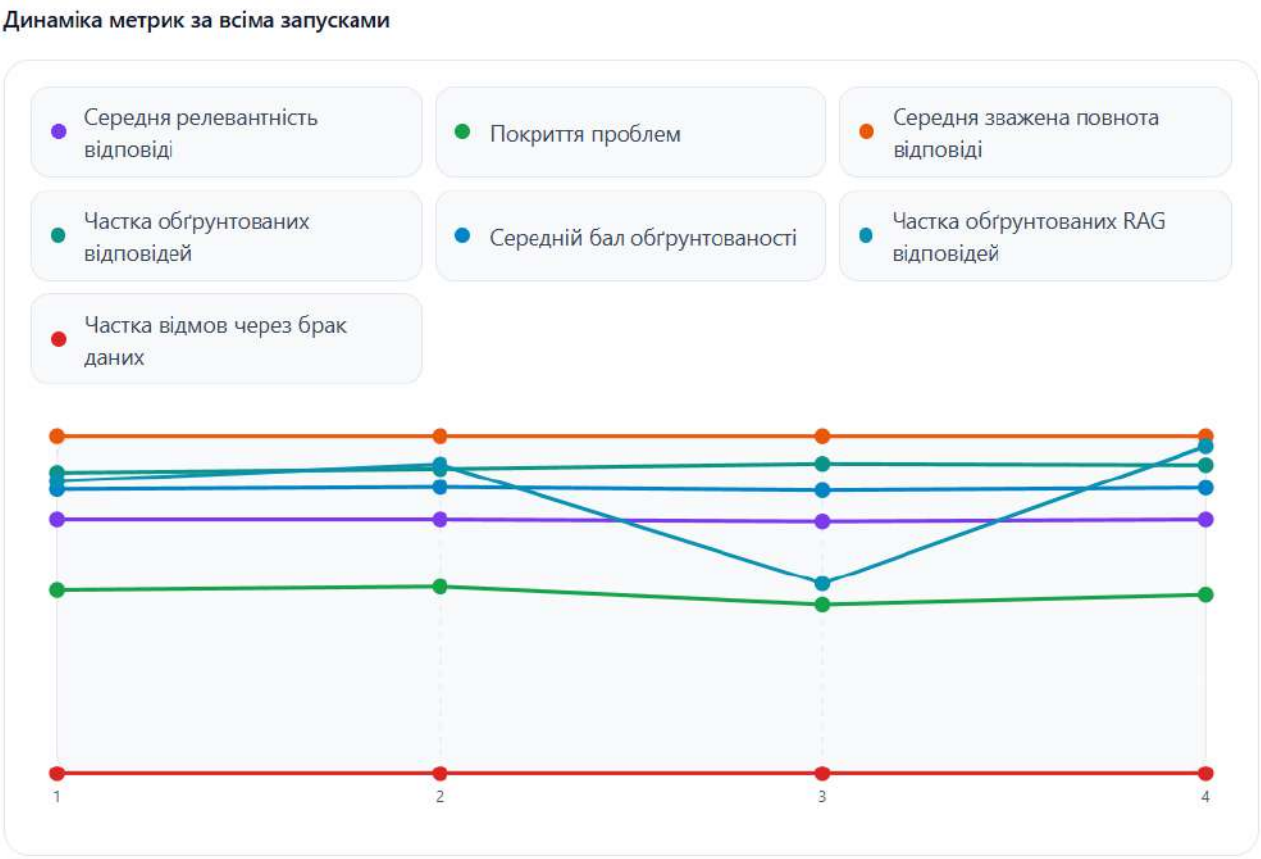

Рисунок 4.3 – Динаміка метрик режиму формування рекомендацій на основі RAG за всіма завершеними запусками

Таблиця 4.6 – Результати оцінювання режиму формування рекомендацій на основі RAG

Метрика	Мінімум	Середнє	Максимум	Найкращий запуск
Частка проблем із знайденими RAG-кандидатами	58,84 %	87,67 %	100,00 %	100,00 %
Доступність RAG-контексту	58,84 %	87,67 %	100,00 %	100,00 %
Частка обґрунтованих RAG-відповідей	56,24 %	82,92 %	97,08 %	97,08 %
Частка резервних відповідей після RAG	0,00 %	0,00 %	0,00 %	0,00 %
Середній обернений ранг точного правила	0,3804	0,5585	0,6390	0,6390
Доступність точного правила	95,78 %	95,78 %	95,78 %	95,78 %

Продовження таблиці 4.6

Метрика	Мінімум	Середнє	Максимум	Найкращий запуск
Частка використання точного правила	0,00 %	0,00 %	0,00 %	0,00 %
Доступність LLM	100,00 %	100,00 %	100,00 %	100,00 %
Частка спроб виклику LLM	100,00 %	100,00 %	100,00 %	100,00 %
Частка успішних відповідей LLM	97,87 %	98,56 %	99,50 %	98,39 %
Частка фінальних LLM-відповідей	97,87 %	98,56 %	99,50 %	98,39 %
Частка резервних відповідей після LLM	0,50 %	1,44 %	2,13 %	1,61 %
Середня релевантність відповіді	74,69 %	75,14 %	75,29 %	75,28 %
Покриття проблем	50,03 %	53,15 %	55,33 %	52,91 %
Середня зважена повнота відповіді	100,00 %	100,00 %	100,00 %	100,00 %
Частка обґрунтованих відповідей	89,24 %	90,74 %	91,88 %	91,50 %
Середній бал обґрунтованості	83,87 %	84,38 %	84,84 %	84,62 %
Підсумковий оптимізаційний бал	85,65 %	85,86 %	86,06 %	86,06 %

Змістові метрики показали, що сформовані відповіді мали повну структуру відповідно до заданих критеріїв. Водночас покриття проблем залишалося нижчим за частку обґрунтованих відповідей, хоча більшість відповідей мала достатню семантичну опору на правила.

Таким чином, перевагами режиму RAG є висока частка обґрунтованих відповідей, відсутність резервних відповідей після добору контексту та повна структурна наповненість сформованих рекомендацій.

4.6 Аналіз впливу RAG на якість рекомендацій

Для визначення впливу генерації з доповненим пошуком на якість сформованих рекомендацій доцільно порівняти результати двох режимів: формування відповідей лише на основі LLM та на основі RAG. Порівняння

виконано за найкращими завершеними запусками відповідних експериментальних сесій. Для LLM-режиму найкращий підсумковий бал становив 85,90 %, а для RAG-режиму – 86,06 %. Різниця є незначною, однак вона показує, що додавання етапу добору контексту не погіршило загальну якість роботи системи, а за окремими показниками дало кращий результат. Порівняння спільних метрик LLM- та RAG-режиму наведено в таблиці 4.7.

Таблиця 4.7 – Порівняння спільних метрик LLM- та RAG-режиму

Метрика	**LLM**	**RAG**
Доступність LLM	100,00 %	100,00 %
Частка спроб виклику LLM	100,00 %	100,00 %
Частка успішних відповідей LLM	98,43 %	98,39 %
Частка фінальних LLM-відповідей	98,43 %	98,39 %
Частка резервних відповідей після LLM	1,57 %	1,61 %
Середня релевантність відповіді	76,25 %	75,28 %
Покриття проблем	54,92 %	52,91 %
Середня зважена повнота відповіді	100,00 %	100,00 %
Частка обґрунтованих відповідей	90,74 %	91,50 %
Середній бал обґрунтованості	83,02 %	84,62 %
Підсумковий оптимізаційний бал	85,90 %	86,06 %

Порівняння метрик показує, що обидва режими працювали стабільно. В обох випадках LLM була доступною в усіх запусках, а частка успішних і фінальних відповідей відрізнялася незначно. Частка резервних відповідей після LLM також залишалася низькою, тому використання RAG не погіршило стабільність генерації.

За метриками прямої відповідності тексту проблеми LLM-режим показав дещо вищу середню релевантність і покриття проблем. Це означає, що відповіді без попереднього добору контексту інколи формулювалися ближче до опису самої проблеми. Однак така перевага не гарантує кращої прикладної якості, оскільки не показує, наскільки відповідь спирається на підготовлене правило.

RAG-режим мав кращі значення метрик обґрунтованості. На рисунку 4.4 видно, що для проблеми щодо товарів у профілі RAG-відповідь точніше відтворює конкретні вказівки правила. Натомість LLM-відповідь є змістовно

правильною, але загальнішою. У результаті RAG формує рекомендацію ближчою до підготовленої нормативної основи.

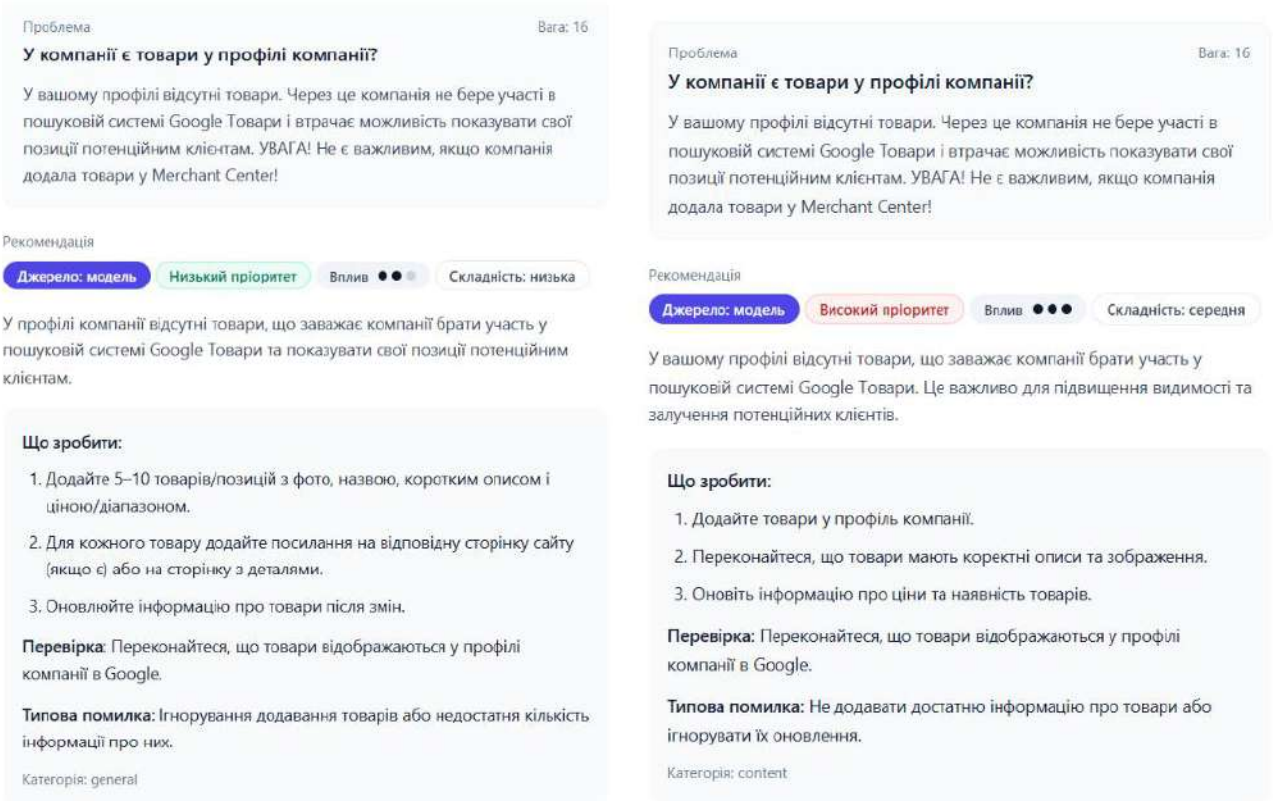

Рисунок 4.4 – Приклад порівняння рекомендацій, сформованих у LLM-режимі (праворуч) та RAG-режимі (ліворуч)

Позитивний вплив RAG проявляється насамперед у підвищенні контрольованості та обґрунтованості рекомендацій. LLM-режим є простішим і демонструє дещо вищу пряму відповідність тексту проблеми, однак має меншу керованість джерелом змісту відповіді.

RAG-режим краще забезпечує зв’язок сформованої рекомендації з підготовленими правилами, що підвищує її пояснюваність і практичну користь. Його основним обмеженням залишається залежність від якості добору контексту: якщо релевантне правило не буде знайдене достатньо точно, це може вплинути на фінальну відповідь.

З цього випливає, що RAG доцільно розглядати як підхід, що покращує не стільки формальну релевантність тексту, скільки керованість, зрозумілість

пояснення і опору рекомендацій на підготовлені правила. Тому для реалізованого веб-застосунку RAG-режим є більш придатним як основний варіант формування рекомендацій.

4.7 Перспективи подальшого вдосконалення

Реалізований веб-застосунок підтверджує можливість формування структурованих рекомендацій на основі зовнішніх вхідних даних. Подальше вдосконалення доцільно спрямувати на підвищення якості відповідей і підготовку системи до використання в реальному експлуатаційному середовищі.

Основними перспективами подальшого розвитку є:

а) інтеграція із системою проведення аудиту Mappers GEO Tools [8]. Це дасть змогу включити розроблений застосунок в екосистему Mappers GEO та застосувати його в реальному середовищі;

б) розширення та уніфікація бази правил. Якість режимів на основі правил і RAG залежить від повноти, деталізації та структурної узгодженості правил, тому їх доцільно доповнювати новими сценаріями й уточнювати для різних типів проблем аудиту;

в) удосконалення механізму добору релевантного контексту. Подальші дослідження можуть охоплювати тестування інших моделей векторного подання тексту, налаштування параметрів пошуку та порівняння способів ранжування кандидатних правил;

г) покращення клієнтської частини. Доцільно додати порівняння рекомендацій, сформованих у різних режимах, експорт результатів і можливість оцінювання рекомендацій користувачем;

ґ) посилення надійності та безпеки системи. У разі інтеграції з реальним сервісом аудиту потрібно забезпечити обробку помилок зовнішнього сервісу, контроль адміністративного доступу, журналювання важливих дій і захист даних користувачів.

Отже, подальший розвиток має бути спрямований на перехід від експериментального прототипу до повноцінного прикладного інструмента. Найважливішими напрямами є інтеграція з Mappers GEO, покращення бази правил, удосконалення добору контексту та розширення оцінювання якості рекомендацій.

Висновки

У межах кваліфікаційної роботи було спроєктовано та реалізовано веб-застосунок з поєднанням можливостей LLM та RAG для формування структурованих рекомендацій на основі готових результатів аудиту бізнес-профілю на Google Картах. Розроблена система виконує обробку вхідних даних аудиту, виділяє виявлені проблеми, добирає релевантні правила та формує відповідь для користувача відповідно до обраного режиму роботи.

Під час виконання роботи було проаналізовано теоретичні засади використання великих мовних моделей і генерації з доповненим пошуком, визначено основні обмеження LLM у прикладних задачах та обґрунтовано доцільність поєднання мовної моделі з механізмом добору релевантного контексту. Також розглянуто семантичний пошук, векторне подання текстових даних і організацію бази правил як джерела структурованої предметної інформації.

У процесі проєктування було визначено архітектуру веб-застосунку, структуру серверної та клієнтської частин, модель збереження даних, формат подання правил і логіку взаємодії основних компонентів. Серверну частину реалізовано з використанням FastAPI, клієнтську частину – з використанням Vue.js, а для збереження даних застосовано MySQL. Для добору релевантного контексту використано ChromaDB, що забезпечило пошук правил за семантичною близькістю.

У системі реалізовано три режими формування рекомендацій: на основі бази правил, лише за допомогою LLM та з використанням RAG. Режим на основі бази правил забезпечує найбільш стабільний і передбачуваний результат. LLM-режим дає змогу формувати гнучкі текстові відповіді без попереднього добору контексту. RAG-режим поєднує добір релевантних правил із генерацією відповіді великою мовною моделлю, що підвищує керованість і предметну обґрунтованість результату.

У межах експериментального дослідження було визначено набір метрик для оцінювання якості сформованих рекомендацій. Вони охоплюють технічну стабільність роботи системи, релевантність відповіді до проблеми, структурну повноту рекомендації та ступінь її опори на підготовлені правила. Це дало змогу оцінити не лише факт успішного формування відповіді, а й її практичну придатність.

Результати експериментів показали, що LLM-режим має високий рівень технічної стабільності та повну структурну наповненість відповідей, а його найкращий підсумковий бал становив 85,90 %. RAG-режим досяг підсумкового бала 86,06 % і продемонстрував вищі показники обґрунтованості порівняно з LLM-режимом. Порівняння цих режимів показало, що додавання етапу добору релевантного контексту не обов’язково суттєво підвищує формальну релевантність відповіді до тексту проблеми, однак посилює її зв’язок із підготовленими правилами.

Практичне значення роботи полягає у створенні веб-застосунку, який перетворює результати аудиту на зрозумілі рекомендації щодо усунення виявлених проблем. Подальше вдосконалення доцільно пов’язувати з інтеграцією із системою Mappers GEO, розширенням і уніфікацією бази правил, покращенням добору релевантного контексту та розвитком клієнтської частини. Це дасть змогу перейти від експериментального прототипу до повноцінного прикладного інструмента для підтримки аналізу й покращення бізнес-профілів на Google Картах.

Список використаних джерел

1. Zhao W. X. A Survey of Large Language Models [Електронний ресурс] / W. X. Zhao, K. Zhou, J. Li [et al.] // arXiv. – 2023. – Режим доступу: https://arxiv.org/abs/2303.18223
2. Vaswani A. Attention Is All You Need [Електронний ресурс] / A. Vaswani, N. Shazeer, N. Parmar [et al.] // arXiv. – 2017. – Режим доступу: https://arxiv.org/abs/1706.03762
3. OpenAI. What are tokens and how to count them? [Електронний ресурс] // OpenAI Help Center. – Режим доступу: https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them
4. Hugging Face. Causal language modeling [Електронний ресурс] // Hugging Face. – Режим доступу: https://huggingface.co/docs/transformers/en/tasks/language_modeling
5. Devlin J. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Електронний ресурс] / J. Devlin, M.-W. Chang, K. Lee, K. Toutanova // arXiv. – 2018. – Режим доступу: https://arxiv.org/abs/1810.04805
6. Radford A. Language Models are Unsupervised Multitask Learners [Електронний ресурс] / A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever. – OpenAI Technical Report. – 2019. – Режим доступу: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
7. Brown T. B. Language Models are Few-Shot Learners [Електронний ресурс] / T. B. Brown, B. Mann, N. Ryder [et al.] // arXiv. – 2020. – Режим доступу: https://arxiv.org/abs/2005.14165
8. Mappers GEO Tools. Аудит: Профілю компанії в Google [Електронний ресурс]. – Режим доступу: https://tools.mappersgeo.com/gbp-audit/

9. Ouyang L. Training language models to follow instructions with human feedback [Електронний ресурс] / L. Ouyang, J. Wu, X. Jiang [et al.] // arXiv. – 2022. – Режим доступу: https://arxiv.org/abs/2203.02155
10. Lewis P. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks [Електронний ресурс] / P. Lewis, E. Perez, A. Piktus [et al.] // arXiv. – 2020. – Режим доступу: https://arxiv.org/pdf/2005.11401
11. Wallat J. Correctness is not Faithfulness in RAG Attributions [Електронний ресурс] / J. Wallat, M. Heuss, M. de Rijke, A. Anand // arXiv. – 2024. – Режим доступу: https://arxiv.org/pdf/2412.18004
12. Gao Y. Retrieval-Augmented Generation for Large Language Models: A Survey [Електронний ресурс] / Y. Gao, Y. Xiong, X. Gao [et al.] // arXiv. – 2023. – Режим доступу: https://arxiv.org/pdf/2312.10997
13. Wang X. Beyond the Limits: A Survey of Techniques to Extend the Context Length in Large Language Models [Електронний ресурс] / X. Wang, M. Salmani, P. Omidi, X. Ren, M. Rezagholizadeh, A. Eshaghi // arXiv. – 2024. – Режим доступу: https://arxiv.org/pdf/2402.02244
14. Chroma Cookbook. Configuration [Електронний ресурс]. – Режим доступу: https://cookbook.chromadb.dev/core/configuration/
15. Han J. Data Preprocessing [Електронний ресурс] / J. Han, M. Kamber, J. Pei // Data Mining: Concepts and Techniques. – 3rd ed. – Waltham : Morgan Kaufmann, 2011. – Розд. 3. – Режим доступу: https://hanj.cs.illinois.edu/cs412/bk3/03.pdf
16. Cheng M. A Survey on Knowledge-Oriented Retrieval-Augmented Generation [Електронний ресурс] / M. Cheng, Y. Luo, J. Ouyang [et al.] // arXiv. – 2025. – Режим доступу: https://arxiv.org/pdf/2503.10677
17. Es S. RAGAs: Automated Evaluation of Retrieval Augmented Generation [Електронний ресурс] / S. Es, J. James, L. Espinosa Anke, S. Schockaert // Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations. – 2024. – P. 150–158. – Режим доступу: https://aclanthology.org/2024.eacl-demo.16.pdf

18. TruLens: Evals and Tracing for Agents [Електронний ресурс]. – Режим доступу: https://www.trulens.org/
19. FastAPI Documentation [Електронний ресурс]. – Режим доступу: https://fastapi.tiangolo.com/
20. Pydantic Docs [Електронний ресурс]. – Режим доступу: https://pydantic.dev/docs/
21. MySQL Connector/Python Developer Guide [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/connector-python/en/
22. Chroma Docs. Introduction [Електронний ресурс]. – Режим доступу: https://docs.trychroma.com/docs/overview/introduction
23. OpenAI Python API Library [Електронний ресурс]. – Режим доступу: https://github.com/openai/openai-python
24. Google Gen AI SDK Documentation [Електронний ресурс]. – Режим доступу: https://googleapis.github.io/python-genai/
25. pytest Documentation [Електронний ресурс]. – Режим доступу: https://docs.pytest.org/en/stable/
26. Vue.js. Introduction [Електронний ресурс]. – Режим доступу: https://vuejs.org/guide/introduction.html
27. Vue Router. The official Router for Vue.js [Електронний ресурс]. – Режим доступу: https://router.vuejs.org/
28. Vite. Getting Started [Електронний ресурс]. – Режим доступу: https://vite.dev/guide/
29. Tailwind CSS. Installing Tailwind CSS with Vite [Електронний ресурс]. – Режим доступу: https://tailwindcss.com/docs/installation/using-vite
30. Sentence Transformers. sentence-transformers/all-MiniLM-L6-v2 [Електронний ресурс] // Hugging Face. – Режим доступу: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
31. Optuna: A hyperparameter optimization framework [Електронний ресурс]. – Режим доступу: https://optuna.readthedocs.io/en/stable/

32. Optuna. optuna.samplers.TPESampler [Електронний ресурс]. – Режим доступу: https://optuna.readthedocs.io/en/stable/reference/samplers/generated/optuna.samplers.TPESampler.html
33. Optuna. optuna.pruners.HyperbandPruner [Електронний ресурс]. – Режим доступу: https://optuna.readthedocs.io/en/stable/reference/generated/optuna.pruners.HyperbandPruner.html

Додаток А

(довідниковий)

Приклад результату аудиту бізнес-профілю на Google Картах за допомогою сервісу Mappers GEO Tools. Аудит: Профілю компанії в Google

Додаток Б

(довідниковий)

Фрагмент JSON-структури одного питання аудиту за допомогою сервісу Mappers GEO Tools. Аудит: Профілю компанії в Google


```
{
  "section": {
    "id": 44,
    "title": "Фотографії",
    "name": "photos"
  },
  "question": {
    "question": "Чи є у профілі фотографії горизонтальної орієнтації?",
    "name": "horizontal_photos",
    "input_type": "radio",
    "answers": [
      {
        "answer": "Так",
        "val": "yes",
        "points": 3,
        "report": "У компанії є кілька горизонтальних фотографій — чудово! Такі фото краще відображаються у пошуку.",
        "report_color": "default",
        "sub_questions": []
      },
      {
        "answer": "Ні",
        "val": "no",
        "points": -3,
        "report": "У компанії немає фото горизонтальної орієнтації. Рекомендуємо додати кілька таких фото, адже алгоритми поки що віддають перевагу цьому формату.",
        "report_color": "default",
        "sub_questions": []
```

```
      }
    ],
    "do": "",
    "auditor_note": "<p>Формат сервісу Google Карти наразі орієнтований переважно на перегляд з ПК, тому бажано мати хоча б кілька фотографій у горизонтальній орієнтації.</p>",
    "desc": "<p>Горизонтальні фото забезпечують кращу видимість та виглядають професійніше при показі у результатах пошуку Google.</p>",
    "condition": ""
  }
}
```

Додаток В

(обов’язковий)

Приклад шаблону запиту до великої мовної моделі в режимі LLM

```
Ти — консультант з оптимізації Google Business Profile.

Вхідні дані:
ISSUE_JSON:
$issue_json

Завдання:
1) Поясни проблему у 1–2 реченнях:
   - що саме не так;
   - чому це важливо для коректності та якості бізнес-профілю.
   - Не вигадуй фактів, яких немає в ISSUE_JSON.
   - Не посилайся на правила, базу правил, RAG або зовнішні
джерела.

2) Склади короткий і практичний план виправлення у 3–5 кроків:
   - План має прямо випливати з ISSUE_JSON.
   - Якщо ISSUE_JSON містить явну вимогу дії, зокрема "видалити",
"прибрати", "замінити", "додати", "оновити", "виправити" —
обов’язково включи цю дію окремим кроком.
   - Додай один рядок "Перевірка:" — як переконатися, що проблему
усунено.
   - Додай один рядок "Типова помилка:" — що найчастіше роблять
неправильно.

3) Сформуй meta-поля рекомендації:
   - category: коротка узагальнена категорія проблеми латиницею в
нижньому регістрі, наприклад "general", "content", "contact_info",
"media", "reviews", "hours".
   - priority: одне зі значень "low" | "medium" | "high".
   - impact: одне зі значень "low" | "medium" | "high".
   - difficulty: одне зі значень "low" | "medium" | "high".
```

```
Вихід:
Поверни лише валідний JSON (без markdown), такого вигляду:
{
  "explanation": "…",
  "action": {
    "category": "…",
    "priority": "…",
    "impact": "…",
    "difficulty": "…",
    "action": "1) …\n2) …\n3) …\nПеревірка: …\nТипова помилка: …"
  }
}

Обмеження:
- Не вигадуй даних про компанію.
- Не використовуй інформацію, якої немає в ISSUE_JSON.
- Не згадуй правила, базу правил, candidate rules,
selected_rule_id або fallback.
- Не додавай жодного тексту поза JSON.
```

Додаток Г

(обов’язковий)

Приклад шаблону запиту до великої мовної моделі в режимі RAG

```
Ти — консультант з оптимізації Google Business Profile.

Вхідні дані:
ISSUE_JSON:
$issue_json

CANDIDATE_RULES_JSON:
$rules_json

ALLOWED_RULE_IDS_JSON:
$allowed_rule_ids_json

Завдання:
1) Обери одне правило зі списку CANDIDATE_RULES_JSON, яке найкраще відповідає проблемі.
   - selected_rule_id має бути одним із ALLOWED_RULE_IDS_JSON АБО "default_fallback".
   - Якщо жодне правило не підходить або правила суперечать ISSUE_JSON — обери "default_fallback".

2) Поясни проблему (1–2 речення): що не так і чому це важливо.
   - Не додавай залякувань санкціями, якщо цього немає в ISSUE_JSON.
   - Не вигадуй причин, яких немає в ISSUE_JSON/правилах.

3) Склади короткий план виправлення у 3–5 кроків:
   - Крок 1 має базуватись на recommendation обраного правила (можна перефразувати без втрати змісту).
   - Якщо ISSUE_JSON містить явну дію/вимогу (наприклад: "видалити", "негайно видалити", "прибрати", "замінити", "додати",
```

"оновити") — ОБОВ’ЯЗКОВО включи цю дію окремим кроком. План не може ігнорувати такі вимоги або замінювати їх загальними порадами.
   - Додай 1 рядок "Перевірка:" (як перевірити, що проблему усунено).
   - Додай 1 рядок "Типова помилка:" (коротко, що роблять неправильно найчастіше).

4) Значення category/priority/impact/difficulty потрібно копіювати з обраного правила без змін.

Вихід:
Поверни лише валідний JSON (без markdown), такого вигляду:

```
{
  "selected_rule_id": "…",
  "explanation": "…",
  "action": {
    "category": "…",
    "priority": "…",
    "impact": "…",
    "difficulty": "…",
    "action": "1) …\n2) …\n3) …\nПеревірка: …\nТипова помилка: …"
  }
}
```

Обмеження:
- Не вигадуй даних про компанію.
- Не використовуй інформацію, якої немає в ISSUE_JSON або правилах.
- Якщо правила суперечать ISSUE_JSON або очевидно не відповідають питанню — обирай "default_fallback".