ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота

на тему: «**ХМАРНА СИСТЕМА АНАЛІЗУ ДОКУМЕНТІВ НА ОСНОВІ МУЛЬТИМОДАЛЬНИХ МОВНИХ МОДЕЛЕЙ»**

на здобуття освітнього ступеня магістра
зі спеціальності __122 Комп'ютерні науки_____ *(код, найменування спеціальності)*
освітньо-професійної програми ___Комп'ютерні науки_____ *(назва)*

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ ___ Антон ПАРХОМЕНКО_____
(підпис) Ім'я, ПРІЗВИЩЕ здобувача

Виконав: здобувач вищої освіти гр. КНДМ-61
_Антон ПАРХОМЕНКО_____
(Ім'я, ПРІЗВИЩЕ)

Керівник: к.т.н., доцент Микола ГНІДЕНКО__
Науковий ступінь, (Ім'я, ПРІЗВИЩЕ)
вчене звання

Рецензент: _____
Науковий ступінь, (Ім'я, ПРІЗВИЩЕ)
вчене звання

Київ – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук
Ступінь вищої освіти Магістр
Спеціальність 122 Комп'ютерні науки
Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Пархоменку Антону Володимировичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Хмарна система аналізу документів на основі мультимодальних мовних моделей»

керівник кваліфікаційної роботи Микола ГНІДЕНКО к.т.н., професор,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024р. №320

2. Строк подання кваліфікаційної роботи «15» грудня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, технічна документація мовних моделей та бібліотеки роботи з ними (Langchain), документація хмарних сервісів (AWS, Vercel), приклади технічних звітів нерухомості, методи розробки програмного забезпечення.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1. Аналіз можливостей мультимодальних мовних моделей та їх застосування для автоматизованої обробки технічної документації.

4.2. Проектування модулів обробки та валідації документів.

4.3. Реалізація аналітичного модуля на основі AI.

4.4. Тестування та оцінка ефективності системи

5. Перелік графічного матеріалу: *презентація*

5.1 Архітектура хмарної системи.

5.2 Схема процесу обробки документів.

5.3 Інтерфейс користувача системи

5.3 Приклади програмного коду.

5.4 Приклади результатів аналізу документів.

5.5 Інтерфейс для аналізу ефективності роботи системи.

6. Дата видачі завдання «16» вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	16.09-27.09.24	виконано
2	Дослідження методів обробки PDF та OCR	30.09-11.10.24	виконано
3	Розробка модуля обробки документів	14.10-25.10.24	виконано
4	Реалізація аналітичного модуля	28.10-08.11.24	виконано
5	Розробка клієнтської частини	11.11-22.12.24	виконано
6	Інтеграція компонентів системи	25.11-29.11.24	виконано
7	Оформлення роботи: вступ, висновки, реферат	02.12-06.12.24	виконано
8	Розробка демонстраційних матеріалів	09.12-13.12.24	виконано

Здобувач вищої освіти _____ Антон ПАРХОМЕНКО

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник

кваліфікаційної роботи _____ Микола ГНІДЕНКО

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 74 стор., 28 рис., 33 джерела.

Наукове завдання – розробка хмарної системи для автоматизованого аналізу технічної документації нерухомості з використанням мультимодальних мовних моделей.

Мета роботи – підвищення ефективності процесу аналізу технічної документації шляхом створення хмарної системи, що автоматично виявляє проблеми та надає рекомендації щодо їх усунення.

Об'єкт дослідження – процес аналізу технічної документації нерухомості та формування звітів про виявлені проблеми.

Предмет дослідження – методи та засоби автоматизованого аналізу технічної документації з використанням мультимодальних мовних моделей.

Методи дослідження – системний аналіз, методи обробки природної мови, технології хмарних обчислень, методи машинного навчання.

Короткий зміст роботи: Проведено аналіз сучасних технологій обробки документів та можливостей мультимодальних мовних моделей. Досліджено методи інтеграції OCR-технологій з AI-моделями для ефективного аналізу технічних звітів.

Розроблено хмарну систему на основі сучасних технологій (Next.js, Flask, AWS). Система забезпечує аналіз звітів технічного огляду нерухомості, виявлення проблемних моментів та оцінку вартості ремонтних робіт.

Ключові слова: мультимодальні мовні моделі, аналіз документів, хмарні технології, обробка природної мови, екстрагування структурованих даних, обробка технічної документації, OCR, LLM

ABSTRACT

Text part of the master's qualification work: 74 pages, 28 pictures, 33 sources.

Scientific task is to develop a cloud system for automated analysis of real estate technical documentation using multimodal language models.

Goal of the work is to improve the efficiency of technical documentation analysis by creating a cloud system that automatically identifies problems and provides recommendations for their resolution.

Object of research – the process of analyzing real estate technical documentation and generating reports on identified problems.

Subject of research – methods and tools for automated analysis of technical documentation using multimodal language models.

Research methods – system analysis, natural language processing methods, cloud computing technologies, machine learning methods.

Summary of the work: An analysis of modern document processing technologies and capabilities of multimodal language models was conducted. Methods of integrating OCR technologies with AI models for effective analysis of technical reports were investigated.

A cloud system has been developed based on modern technologies (Next.js, Flask, AWS). The system provides analysis of real estate inspection reports, identification of problem areas, and estimation of repair costs.

Keywords: multimodal language models, document analysis, cloud technologies, natural language processing, structured data extraction, technical documentation processing, OCR, LLM.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми і її актуальність. Тема "Хмарна система аналізу документів на основі мультимодальних мовних моделей" була обрана у зв'язку з революційними змінами на ринку нерухомості США та зростаючою потребою в автоматизації процесу аналізу технічної документації. З впровадженням нового законодавства, яке вимагає від покупців оплати комісійних агентам, виникла необхідність у розробці інструментів, які дозволять покупцям самостійно проводити професійний аналіз технічного стану нерухомості та приймати обґрунтовані рішення без залучення дорогих експертів.

Ступінь вивчення проблеми. Хоча існують окремі рішення для обробки документів та аналізу тексту, комплексних систем для автоматизованого аналізу звітів технічного огляду нерухомості практично немає. Поява потужних мультимодальних мовних моделей відкрила нові можливості для створення таких систем, але їх практичне застосування в цій сфері все ще залишається малодослідженим.

Специфіка джерельної бази. Інформаційною базою для дослідження послужили технічні документації провідних AI-компаній (OpenAI, Anthropic), документація хмарних сервісів (AWS, Vercel), наукові статті про мовні моделі, а також реальні звіти технічного огляду нерухомості.

Об'єктом дослідження є процес аналізу звітів технічного огляду нерухомості та формування структурованих висновків з оцінкою виявлених проблем.

Предметом дослідження є методи та засоби автоматизованого аналізу технічної документації з використанням мультимодальних мовних моделей.

Метою дослідження є створення та оцінка ефективності хмарної системи, яка використовує штучний інтелект для аналізу технічної документації, структурування виявлених проблем та надання рекомендацій щодо їх усунення з оцінкою приблизної вартості ремонтних робіт.

У процесі дослідження були вирішені такі **завдання**:

1. Аналіз існуючих підходів до автоматизованої обробки технічної документації
2. Дослідження можливостей мультимодальних мовних моделей для аналізу технічних текстів
3. Розробка архітектури хмарної системи для аналізу документів
4. Реалізація модулів обробки та валідації документів
5. Інтеграція з мовними моделями та оптимізація їх використання
6. Тестування системи та оцінка її ефективності

Методика дослідження базується на системному підході та включає:

- Аналіз сучасних технологій обробки документів
- Експериментальне дослідження мовних моделей
- Розробку та тестування програмного забезпечення
- Оцінку ефективності розробленої системи

Наукова новизна полягає у створенні комплексного підходу до автоматизованого аналізу технічної документації нерухомості з використанням мультимодальних мовних моделей, що включає методи обробки документів, валідації даних та генерації структурованих звітів.

Практична значущість результатів дослідження проявляється у можливості їх безпосереднього використання для:

- Автоматизації процесу аналізу технічної документації нерухомості

- Зниження залежності покупців від професійних консультантів
- Пришвидшення процесу прийняття рішень при купівлі нерухомості
- Адаптації розробленої системи для аналізу інших типів технічної документації

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ СИСТЕМИ АНАЛІЗУ ДОКУМЕНТІВ

1.1 Аналіз методів обробки PDF-документів

1.1.1 Формат PDF та його типи

PDF (Portable Document Format) - це формат файлів, розроблений компанією Adobe Inc. на початку 1990-х років. У 2008 році він став відкритим стандартом, який підтримується Міжнародною організацією стандартизації (ISO) [1].

Основні типи PDF-файлів:

1. **PDF (стандартний)**
   - Базовий формат для обміну та перегляду файлів онлайн
   - Найпоширеніший тип PDF-документів
2. **PDF/A**
   - Призначений для довгострокового архівного зберігання
   - Має обмежений набір функцій (без JavaScript, аудіо та відео)
   - Забезпечує незалежність від програмного забезпечення
3. **PDF/E**
   - Підтримує специфікації будівництва та інженерії
   - Використовується у виробничій документації
4. **PDF/X**

- Оптимізований для поліграфії та графічного дизайну
- Покращена підтримка графічних елементів

5. **PDF/VT**
   - Розширена версія PDF/X з додатковими можливостями налаштування
   - Призначений для професійного друку
6. **PDF/UA**
   - Сумісний з допоміжними технологіями
   - Покращена доступність для людей з обмеженими можливостями
7. **Searchable PDF**
   - Стандартний PDF з функцією пошуку
   - Дозволяє шукати текст у відсканованих документах

Основні можливості PDF-формату:

1. **Збереження форматування**
   - Точне відтворення оригінального макету
   - Збереження шрифтів, зображень та гіперпосилань
   - Підтримка коментарів та приміток
2. **Універсальна сумісність**
   - Незалежність від програмного забезпечення
   - Підтримка різними пристроями та операційними системами

- Легкість обміну документами через інтернет

3. **Захист інформації**
   - Можливість встановлення паролів
   - Обмеження доступу до функцій редагування
   - Захист конфіденційних даних
4. **Підтримка електронних підписів**
   - Можливість підписання документів
   - Управління процесом підписання
   - Відсутність необхідності друку

У контексті звітів технічного огляду нерухомості найчастіше використовуються стандартний PDF та PDF з можливістю пошуку.

1.1.2 Методи вилучення тексту з PDF

На сьогоднішній день існує декілька основних підходів до вилучення текстової інформації з PDF-документів. Кожен з них має свої особливості, переваги та обмеження.

Пряме вилучення тексту

Цей метод базується на доступі до вбудованого текстового шару PDF-документа. Текст у такому форматі зберігається у векторному вигляді, що забезпечує високу якість відтворення та можливість прямого копіювання.

Основні бібліотеки для роботи з PDF:

1. **PyPDF2**:

- Базові операції з PDF
- Швидка обробка документів
- Мінімальні вимоги до ресурсів

2. **pdfplumber**:
   - Покращене збереження форматування
   - Підтримка таблиць
   - Точне визначення позиції тексту
3. **PDFMiner**:
   - Глибокий аналіз структури
   - Підтримка різних кодувань
   - Розширені можливості обробки

Обмеження методу:

- Працює лише з цифровими PDF
- Може втрачати складне форматування
- Чутливий до структури документа

Оптичне розпізнавання символів (OCR)

OCR використовується для розпізнавання тексту з растрових зображень або сканованих документів. Цей метод перетворює графічне представлення тексту в редагований формат.

Основні OCR-рішення:

1. **Традиційні технології**:
    - Tesseract OCR (відкрите рішення від Google)
    - ABBYY FineReader (комерційне рішення)
    - Adobe Acrobat OCR (частина Adobe Acrobat)
    - OmniPage (спеціалізоване OCR-рішення)
2. **AI-based рішення**:
    - Amazon Textract
    - Google Cloud Vision OCR
    - Microsoft Azure Computer Vision
    - AI-powered OCR APIs

Специфіка роботи з технічною документацією

При роботі з технічними документами особливу увагу слід приділяти:

- Точності розпізнавання спеціальних термінів
- Збереженню структури документа
- Коректній обробці таблиць та діаграм
- Розпізнаванню технічних позначень

Кожен з методів має свої переваги та недоліки, і вибір конкретного підходу залежить від характеру документів та вимог до якості вилученого тексту.

Нестандартні підходи до OCR

В останній час з'явилися альтернативні підходи до розпізнавання тексту, зокрема використання мультимодальних мовних моделей (LLM). Ці моделі, розроблені такими компаніями як OpenAI, Anthropic та Google, здатні "розуміти" як текст, так і зображення.

1.1.3 Мультимодальні LLM для OCR

1. **Принцип роботи**:
    - Модель аналізує зображення цілісно
    - Розуміє контекст та структуру документа
    - Здатна інтерпретувати складні візуальні елементи
2. **Особливості підходу**:
    - Не потребує попередньої обробки зображення
    - Може працювати з різними форматами та стилями
    - Краще розуміє специфічну термінологію
    - Здатна обробляти таблиці та діаграми як єдине ціле
3. **Переваги порівняно з традиційними OCR**:
    - Вища точність на технічних документах
    - Краще розуміння контексту
    - Можливість одразу структурувати інформацію
    - Стійкість до різної якості сканування

4. **Обмеження**:
    - Залежність від API провайдерів
    - Вища вартість обробки
    - Можливі затримки при обробці
    - Обмеження на розмір вхідних даних

Цей підхід представляє особливий інтерес для обробки технічної документації, де важливе не лише точне розпізнавання тексту, але й розуміння його змісту та контексту.

1.2 Мультимодальні мовні моделі

1.2.1 Що таке мультимодальні мовні моделі (LLM)?

Мультимодальна мовна модель (Рисунок 1.1) - це вдосконалена версія моделі штучного інтелекту, яка здатна генерувати та обробляти дані різних модальностей або форм. На відміну від традиційних мовних моделей, які працюють переважно з текстовими даними, мультимодальні LLM здатні розуміти та генерувати контент з різних джерел, таких як текст, зображення, аудіо та навіть відео.

Простіше кажучи, мультимодальна LLM - це система штучного інтелекту, яка може сприймати та створювати контент, що включає не лише текст. Вона може розуміти та генерувати інформацію, що поєднує текст із візуальними елементами, такими як зображення чи діаграми, аудіозаписи або навіть відео.

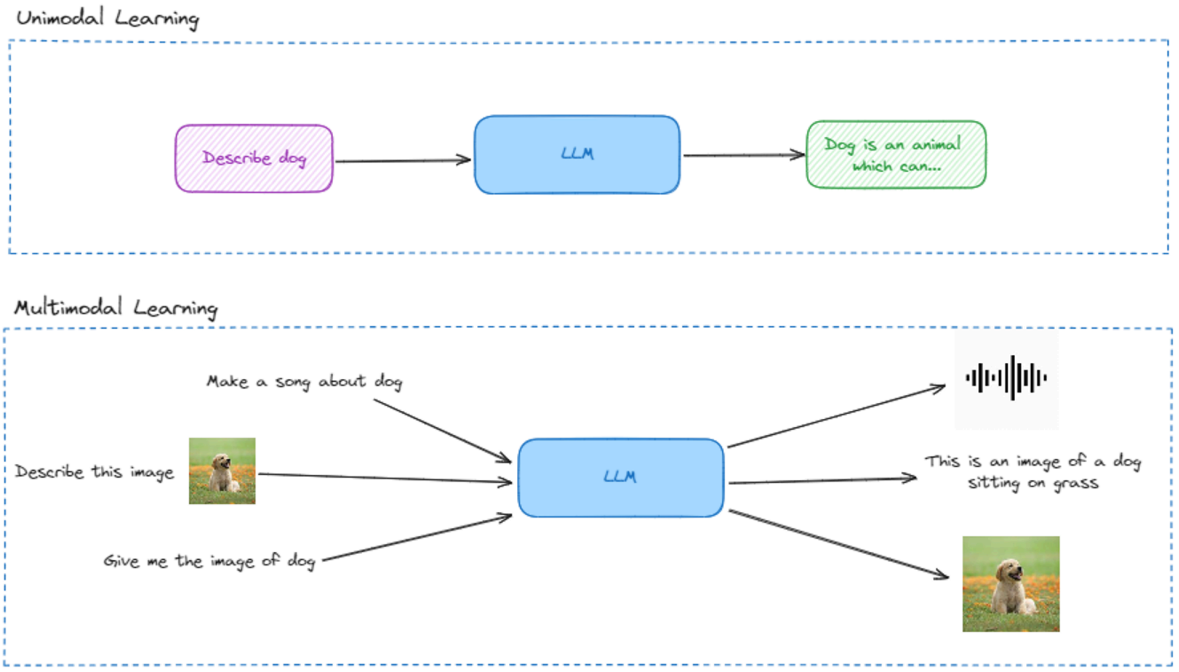


Рисунок 1.1 – Одномодальне та мультимодальне навчання

Розуміння мультимодального навчання

Мультимодальне навчання відноситься до процесу тренування моделей машинного навчання на даних, що поєднують кілька модальностей або джерел інформації. Замість того, щоб покладатися на одну модальність введення, ці моделі розроблені для одночасного навчання та інтеграції інформації з різних модальностей.

Важливість мультимодального навчання

Головною проблемою старих одномодальних LLM було те, що вони не могли розуміти більше однієї модальності одночасно, і іноді стає складно правильно пояснити вашу проблему чи результат як для LLM, так і для вас.

Ось чому мультимодальні LLM важливі:

- Заповнення прогалини між текстом і реальністю: Іноді текст сам по собі може бути складним для пояснення чи розуміння, і саме тут мультимодальні

LLM роблять це простішим. Наприклад, якщо ви хочете створити проект і отримати поради щодо архітектури цього проекту, ви можете легко надати зображення схеми архітектури, і модель може надати відгуки та також створити оновлену архітектуру.

- Більша креативність: Мультимодальні LLM розширили творчі можливості звичайних LLM, надаючи кілька модальностей одночасно. Вони можуть генерувати відео на основі тексту або створювати історії з зображень.

- Покращення взаємодії людина-комп'ютер: Мультимодальні LLM підвищили якість спілкування, роблячи його більш реалістичним з використанням кількох модальностей. Це схоже на спілкування з другом.

Порівняння з одномодальним навчанням

В одномодальному навчанні кожна модель тренується на наборі даних з однією модальністю. З іншого боку, мультимодальні моделі навчаються на даних різних модальностей або форм, що робить їх більш точними та креативними.

Одномодальне навчання дуже корисне в задачах класифікації, таких як розпізнавання зображень, де модель може класифікувати зображення, виявляти об'єкти всередині зображення. Однак у складних проблемах, таких як прогнозування захворювань, лише зображення недостатньо для точного прогнозування типу захворювання. У таких випадках мультимодальне навчання допомагає покращити розуміння захворювання.

1.2.2 Опис різних модальностей

Різні режими даних - це текст, зображення, аудіо, табличні дані тощо. Один режим даних може бути представлений або *приблизно відтворений* в іншому режимі даних. Наприклад:

- **Текст:** Текст є одним з найпоширеніших режимів або форм даних в LLM. Вони можуть обробляти та розуміти текст, що дозволяє їм аналізувати дані, генерувати текстові описи, перекладати мови та інформативно відповідати на ваші запитання

- **Зображення:** Зображення можуть бути перетворені у векторні вкладення або будь-який інший формат, щоб LLM могли їх розуміти та знаходити об'єкти на зображенні, генерувати зображення або знаходити подібні зображення для заданого зображення

- **Аудіо:** Аудіофайли можуть бути оброблені шляхом аналізу аудіоданих, розпізнавання мовленнєвих паттернів, розуміння емоційного забарвлення аудіо або навіть перетворення аудіо в текст чи навпаки

- **Відео:** Поєднуючи обробку зображень та аудіо, мультимодальні LLM можуть аналізувати відео, розбиваючи відео на кадри та аудіодоріжки, аналізувати вміст кожного кадру, знаходити об'єкти у відео або генерувати резюме відеовмісту

1.2.3 Компоненти мультимодальної LLM

Мультимодальні LLM - це складні системи з 5 основними компонентами, які працюють разом для розуміння та обробки інформації різних модальностей. Ось як виглядає архітектура мультимодальної LLM (Рисунок 1.2)

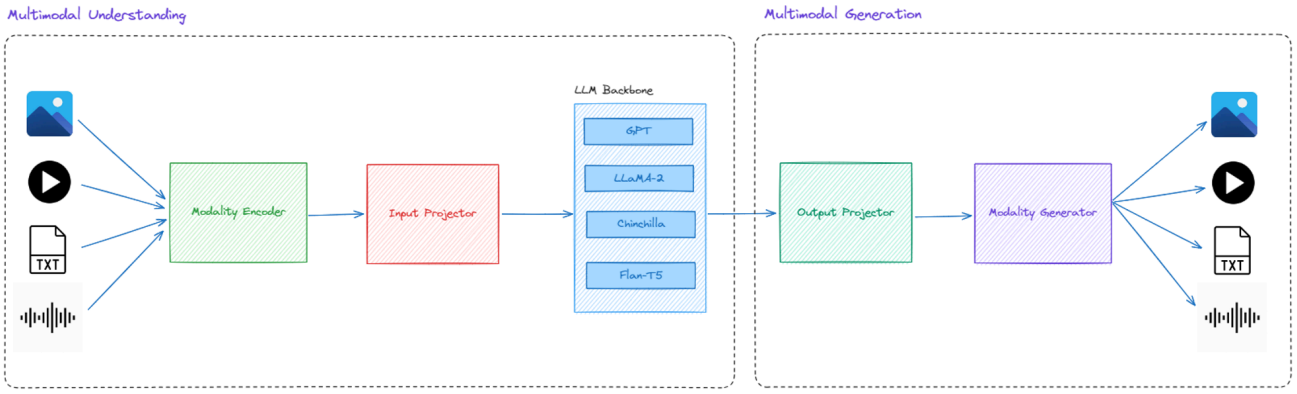


Рисунок 1.2 – Архітектура мультимодальної LLM

Архітектура мультимодальної LLM складається з 5 ключових компонентів:

1. Модальні енкодери Перетворюють різні типи даних (текст, зображення, аудіо, відео) у формат, зрозумілий для LLM. Кожен енкодер спеціалізується на своєму типі даних - текстовий витягує семантику, візуальний - характеристики зображень тощо.

2. Вхідний проектор Об'єднує закодовані дані з різних модальностей в єдиний формат шляхом конкатенації або проектування у спільний простір вкладень. Це дозволяє LLM одночасно працювати з різними типами даних.

3. Основа LLM Центральний компонент системи - велика мовна модель, навчена обробляти комбіновані дані та генерувати різноманітний контент (описи, резюме тощо).

4. Вихідний проектор Перетворює результати роботи LLM у потрібний формат. Наприклад, для завдання підпису зображення - генерує послідовність описових слів.

5. Модальний генератор Створює вихідні дані в різних модальностях (текст, аудіо, зображення) використовуючи різні техніки: GANs, VAEs або дифузійні моделі.

1.2.4 Сфери застосування мультимодальних мовних моделей

Розглянемо ключові напрямки практичного застосування мультимодальних мовних моделей у різних галузях:

- **Трансформація інформаційно-пошукових систем:** Сучасні пошукові системи переважно обмежені текстовим пошуком, з окремою функціональністю для пошуку зображень. Впровадження мультимодальних мовних моделей дозволить створити уніфіковані системи для одночасного пошуку та аналізу контенту різних типів - текстового, візуального та аудіального, що суттєво розширить можливості інформаційного пошуку.

- **Інновації в електронній комерції:** Мультимодальні мовні моделі відкривають можливості для створення систем віртуальної примірки, які здатні аналізувати антропометричні дані користувача та характеристики одягу для генерації реалістичних візуалізацій. Крім того, такі системи можуть формувати персоналізовані рекомендації щодо стилю на основі комплексного аналізу візуальних та текстових даних.

- **Розвиток медичної діагностики:** У сфері охорони здоров'я мультимодальні мовні моделі можуть здійснювати комплексний аналіз діагностичних зображень (рентгенограм, результатів МРТ), медичної документації та аудіозаписів консультацій для підтримки прийняття клінічних рішень. Системи здатні генерувати структуровані медичні висновки та розробляти рекомендації щодо терапевтичних стратегій на основі інтеграції різнорідних даних.

- **Модернізація освітнього процесу:** Застосування мультимодальних мовних моделей у навчанні забезпечує підвищення інтерактивності та ефективності освітнього процесу через інтеграцію різних форм подання інформації. У сфері технічної підтримки ці системи здатні аналізувати мультимедійні звернення користувачів для формування більш точних і детальних відповідей на основі комплексного розуміння проблеми.

1.3 Вилучення структурованих даних

1.3.1 Автореферування (summarization) тексту

Загальна характеристика

Автореферування тексту є методом обробки природної мови, що забезпечує конденсацію інформації з вхідних документів у коротший вихідний текст. Хоча існують різні погляди щодо оптимального коефіцієнта компресії - від 10% до 50% від початкового обсягу - основна мета залишається незмінною: ефективне вилучення ключової інформації.

Основні підходи до автореферування

У галузі автоматизованого автореферування тексту виділяють два принципових підходи: екстрактивний та абстрактивний. Екстрактивне автореферування функціонує шляхом селекції та комбінування існуючих речень з вихідного тексту без їх модифікації. Цей метод реалізується через три основні етапи: репрезентацію, де відбувається сегментація та підготовка тексту до аналізу; оцінювання речень, що визначає важливість кожного речення; та селекцію речень, яка обирає найбільш релевантні фрагменти при мінімізації redundантності. Екстрактивне автореферування зазвичай менш ресурсомістке та може бути реалізоване без використання нейронних мереж, хоча сучасні підходи часто застосовують їх для підвищення якості результатів.

Абстрактивний підхід та його особливості

Абстрактивне автореферування, натомість, генерує цілком нові речення, що відображають сутність вихідного матеріалу. Цей підхід потребує застосування складних нейронних мереж та великих мовних моделей для продукування семантично значущого тексту. Він використовує декілька ключових технік: компресію речень для конденсації довших фрагментів зі збереженням змісту, інформаційну фузію для об'єднання деталей з різних секцій та впорядкування інформації для тематичної організації контенту.

Методи оцінювання ефективності

Ефективність цих підходів вимірюється за допомогою спеціалізованих метрик. Показники BLEU оцінюють подібність між машинно-генерованими та написаними людьми рефератами шляхом аналізу послідовностей слів, що перекриваються (n-грам). ROUGE, похідна від BLEU, зосереджується на повноті відтворення - наскільки добре автоматизований реферат охоплює ключові елементи, присутні в еталонних текстах, створених людьми. Обидві метрики генерують оцінки від 0 до 1, де вищі значення свідчать про кращу відповідність рефератам, створеним людиною.

Практичне застосування

Дослідження, що порівнюють екстрактивні та абстрактивні методи, демонструють цікаві компроміси. Абстрактивні реферати зазвичай більш когерентні та природні, проте екстрактивні реферати часто зберігають більше фактичної інформації та релевантності. Вибір між методами часто залежить від конкретного випадку застосування та предметної області. Технологія знаходить застосування у різних галузях: від прискорення юридичних та академічних досліджень до обробки новинних статей та забезпечення міжмовної комунікації. Останні розробки демонструють перспективність у таких сферах, як виявлення

фейкових новин та модернізація історичних текстів, що підтверджує універсальність методів текстового автореферування.

1.3.2 Техніки вилучення структурованої інформації

Вилучення структурованої інформації є фундаментальним завданням у багатьох сучасних застосунках: від автоматизації робочих процесів до аналізу неструктурованого тексту. Великі мовні моделі (LLM), такі як GPT-4, довели свою універсальність для цієї мети завдяки здатності обробляти та інтерпретувати великі обсяги текстових даних. Розглянемо методи вилучення структурованих даних, використовуючи два основні підходи.

Простий промптинг

Цей метод передбачає формулювання запитів природною мовою для інструктування LLM щодо вилучення або перетворення тексту в структурований формат. Підхід є швидким, простим у реалізації та підходить для нескладних завдань.

Переваги:

- Простота реалізації без додаткових налаштувань
- Гнучкість застосування для різних задач

Обмеження:

- Можливі неточності при неоднозначних формулюваннях
- Обмежена масштабованість для складних завдань

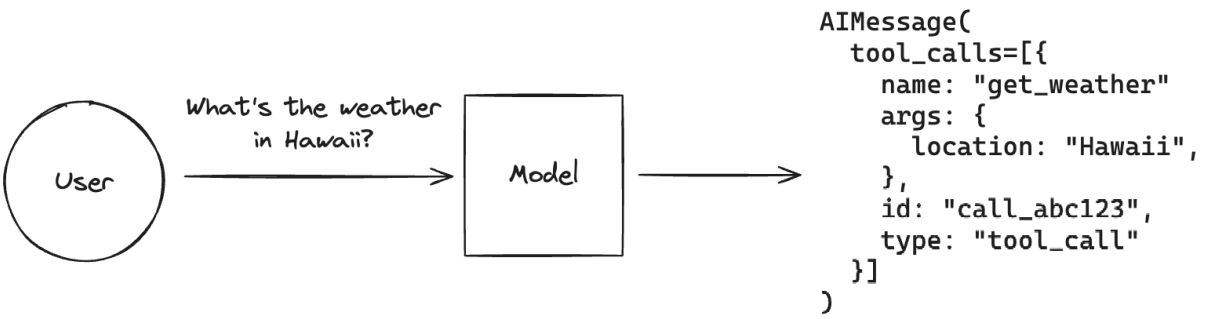


Рисунок 1.3 – Виклик функцій (function calling)

Використання інструментів (tools) або виклику функцій (function calling)

Цей метод (Рисунок 1.3) використовує структуровані інструменти або API, інтегровані з LLM, для керування процесом вилучення даних. Модель взаємодіє з попередньо визначеними функціями або схемами, забезпечуючи структуровані та послідовні результати. Приклад функції в синтаксисі OpenAI Tool показано на Рисунку 1.4.

```
listing_agent_email_extraction_tool = {
   "type": "function",
   "function": {
       "name": "listing_agent_email_extraction_tool",
        "description": "A tool to extract the email address of the listing agent from a
property listing.",
       "parameters": {
           "type": "object",
           "properties": {
               "listing_agent_email": {
                           "description": "The email of the Listing Agent. Example:
tcernohous@silverleafrealty.com",
                   "type": "string"
               }
           }
       }, "required": ["listing_agent_email"]   }}
```

Рисунок 1.4 – Приклад функції в синтаксисі OpenAI Tool

Переваги:

- Висока точність завдяки дотриманню попередньо визначених схем
- Можливість автоматизації та інтеграції з іншими системами
- Ефективна обробка помилок

Обмеження:

- Потребує попереднього налаштування схем або API
- Вимагає додаткових технічних знань

1.4 Архітектура сучасних хмарних застосунків

1.4.1 Фреймворк Next.js як основа сучасних веб-застосунків

Next.js є потужним React-фреймворком, що забезпечує оптимальну продуктивність та масштабованість веб-застосунків. Фреймворк надає вбудовану підтримку серверного рендерингу (SSR), статичної генерації (SSG) та інкрементальної статичної регенерації (ISR). Ці особливості дозволяють створювати високопродуктивні застосунки з оптимальним користувацьким досвідом та SEO. Next.js також пропонує вбудовану маршрутизацію на основі файлової системи, автоматичне розділення коду та оптимізацію зображень.

1.4.2 Безперервна інтеграція та розгортання

Сучасні хмарні застосунки потребують надійної системи CI/CD для автоматизації процесів розробки та розгортання. Поширеним рішенням є комбінація Github Actions з AWS Elastic Beanstalk або Vercel. Github Actions забезпечує автоматизацію процесів тестування, збірки та розгортання коду при кожному оновленні репозиторію. Інтеграція з AWS Elastic Beanstalk дозволяє автоматично розгортати застосунок у хмарній інфраструктурі AWS, тоді як Vercel пропонує спрощене розгортання з автоматичною оптимізацією продуктивності.

1.4.3 Стратегії масштабування

Для забезпечення високої доступності та продуктивності застосунків використовуються різні стратегії масштабування. AWS Elastic Beanstalk пропонує автоматичне масштабування на основі навантаження, керуючи створенням та видаленням екземплярів застосунку. Vercel забезпечує глобальне масштабування через мережу CDN та автоматичне балансування навантаження, оптимізуючи доставку контенту користувачам у різних географічних регіонах.

1.4.4 Управління даними в хмарі

Supabase представляє сучасну альтернативу традиційним базам даних, пропонуючи відкриту платформу, що поєднує реляційну базу даних PostgreSQL з реальночасовими можливостями. Платформа забезпечує автоматичне масштабування, резервне копіювання та високу доступність даних. Вбудована підтримка row level security (RLS) дозволяє реалізувати тонкий контроль доступу до даних на рівні записів.

1.4.5 Системи управління користувачами

Supabase надає комплексне рішення для автентифікації та авторизації користувачів. Платформа підтримує різні методи автентифікації (електронна пошта/пароль, OAuth, магічні посилання) та дозволяє легко інтегрувати сторонніх провайдерів автентифікації. Вбудована система ролей та дозволів забезпечує гнучке управління доступом користувачів до різних функцій застосунку.

Така архітектура забезпечує створення сучасних, масштабованих та безпечних хмарних застосунків з мінімальними витратами на інфраструктуру та обслуговування. Використання готових сервісів та платформ дозволяє розробникам зосередитися на бізнес-логіці застосунку, залишаючи питання інфраструктури та масштабування провайдерам хмарних послуг.

2 РОЗРОБКА СИСТЕМИ АНАЛІЗУ ДОКУМЕНТІВ

2.1 Аналіз та підготовка даних

2.1.1 Дослідження структури звітів технічного огляду та користувацькі дослідження

На початковому етапі розробки системи було проведено комплексне дослідження предметної області, що включало аналіз зразків звітів технічного огляду (див. Рисунки 2.1, 2.2, 2.3) та серію інтерв'ю з покупцями нерухомості. Метою дослідження було визначення ключових інформаційних потреб користувачів та розуміння того, як дані з технічних звітів використовуються при прийнятті рішень щодо купівлі нерухомості.

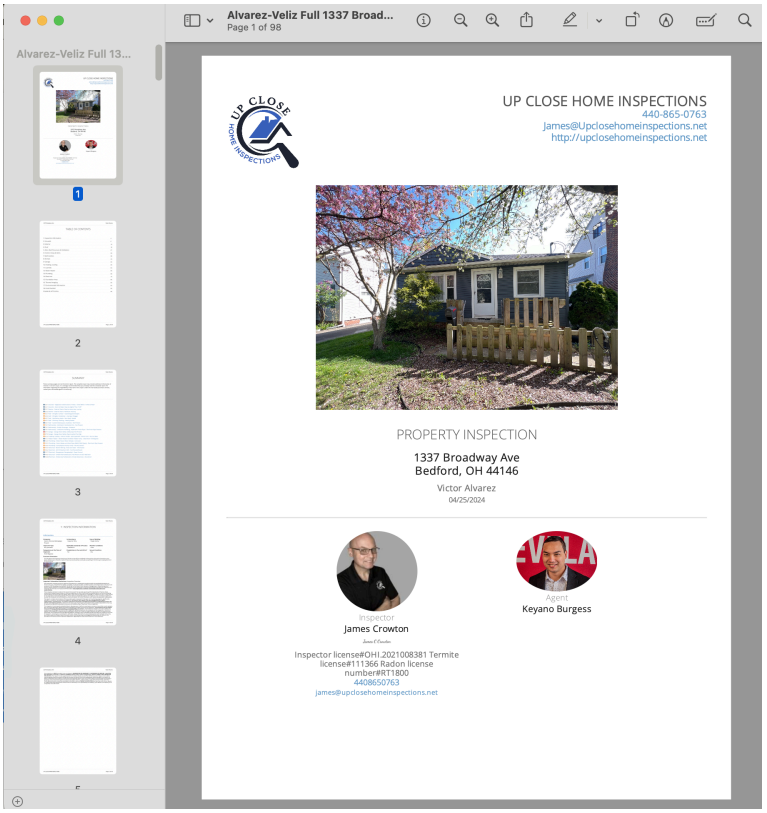
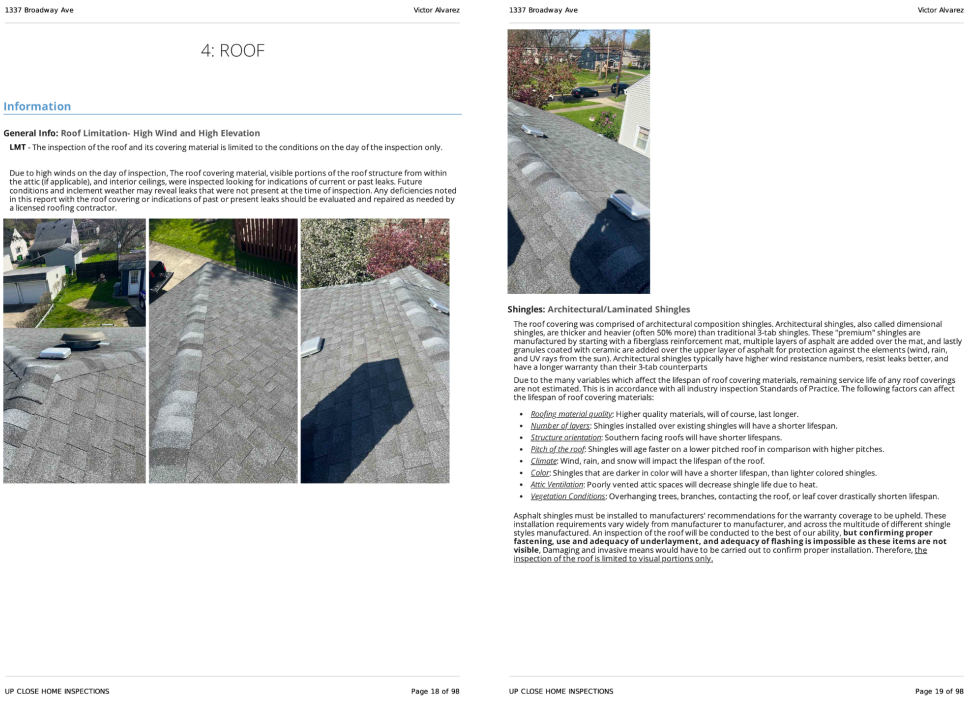


Рисунок 2.1 – Звіт про технічний огляд будинку



1337 Broadway Ave Victor Alvarez

4: ROOF

Information

General Info: Roof Limitation- High Wind and High Elevation

LMT - The inspection of the roof and its covering material is limited to the conditions on the day of the inspection only.

Due to high winds on the day of inspection, The roof covering material, visible portions of the roof structure from within the attic (if applicable), and interior ceilings, were inspected looking for indications of current or past leaks. Future conditions and inclement weather may reveal leaks that were not present at the time of inspection. Any deficiencies noted in this report with the roof covering or indications of past or present leaks should be evaluated and repaired as needed by a licensed roofing contractor.

UP CLOSE HOME INSPECTIONS Page 18 of 98

1337 Broadway Ave Victor Alvarez

Shingles: Architectural/Laminated Shingles

The roof covering was comprised of architectural composition shingles. Architectural shingles, also called dimensional shingles, are thicker and heavier (often 50% more) than traditional 3-tab shingles. These "premium" shingles are manufactured by starting with a fiberglass reinforcement mat, multiple layers of asphalt are added over the mat, and lastly granules coated with ceramic are added over the upper layer of asphalt for protection against the elements (wind, rain, and UV rays from the sun). Architectural shingles typically have higher wind resistance numbers, resist leaks better, and have a longer warranty than their 3-tab counterparts

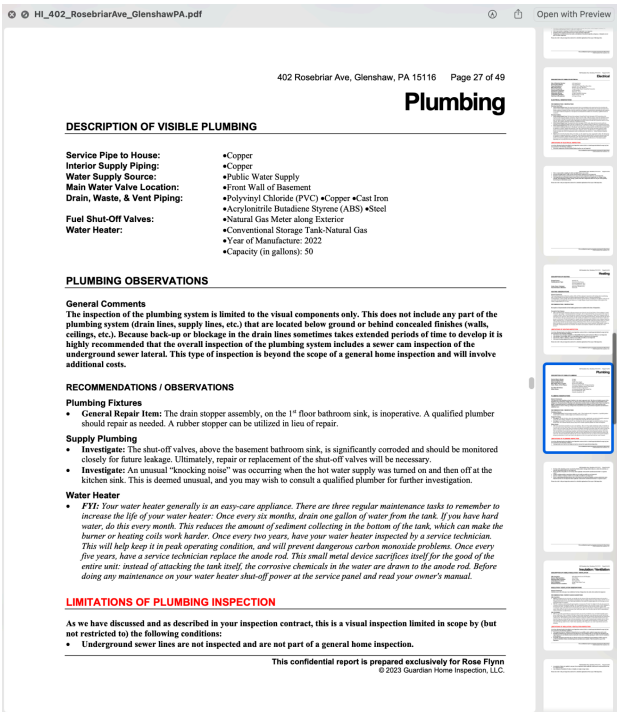
Due to the many variables which affect the lifespan of roof covering materials, remaining service life of any roof coverings are not estimated. This is in accordance with all industry inspection Standards of Practice. The following factors can affect the lifespan of roof covering materials:

- *Roofing material quality*: Higher quality materials, will of course, last longer.
- *Number of layers*: Shingles installed over existing shingles will have a shorter lifespan.
- *Structure orientation*: Southern facing roofs will have shorter lifespans.
- *Pitch of the roof*: Shingles will age faster on a lower pitched roof in comparison with higher pitches.
- *Climate*: Wind, rain, and snow will impact the lifespan of the roof.
- *Color*: Shingles that are darker in color will have a shorter lifespan, than lighter colored shingles.
- *Attic Ventilation*: Poorly vented attic spaces will decrease shingle life due to heat.
- *Vegetation Conditions*: Overhanging trees, branches, contacting the roof, or leaf cover drastically shorten lifespan.

Asphalt shingles must be installed to manufacturers' recommendations for the warranty coverage to be upheld. These installation requirements vary widely from manufacturer to manufacturer, and across the multitude of different shingle styles manufactured. An inspection of the roof will be conducted to the best of our ability, **but confirming proper fastening, use and adequacy of underlayment, and adequacy of flashing is impossible as these items are not visible**. Damaging and invasive means would have to be carried out to confirm proper installation. Therefore, the inspection of the roof is limited to visual portions only.

UP CLOSE HOME INSPECTIONS Page 19 of 98

Рисунок 2.2 – Звіт про технічний огляд будинку, сторінки з фото



HI_402_RosebriarAve_GlenshawPA.pdf Open with Preview

402 Rosebriar Ave, Glenshaw, PA 15116 Page 27 of 49

Plumbing

DESCRIPTION OF VISIBLE PLUMBING

Service Pipe to House:	•Copper
Interior Supply Piping:	•Copper
Water Supply Source:	•Public Water Supply
Main Water Valve Location:	•Front Wall of Basement
Drain, Waste, & Vent Piping:	•Polyvinyl Chloride (PVC) •Copper •Cast Iron •Acrylonitrile Butadiene Styrene (ABS) •Steel
Fuel Shut-Off Valves:	•Natural Gas Meter along Exterior
Water Heater:	•Conventional Storage Tank-Natural Gas •Year of Manufacture: 2022 •Capacity (in gallons): 50

PLUMBING OBSERVATIONS

General Comments

The inspection of the plumbing system is limited to the visual components only. This does not include any part of the plumbing system (drain lines, supply lines, etc.) that are located below ground or behind concealed finishes (walls, ceilings, etc.). Because back-up or blockage in the drain lines sometimes takes extended periods of time to develop it is highly recommended that the overall inspection of the plumbing system includes a sewer cam inspection of the underground sewer lateral. This type of inspection is beyond the scope of a general home inspection and will involve additional costs.

RECOMMENDATIONS / OBSERVATIONS

Plumbing Fixtures

- **General Repair Item:** The drain stopper assembly, on the 1st floor bathroom sink, is inoperative. A qualified plumber should repair as needed. A rubber stopper can be utilized in lieu of repair.

Supply Plumbing

- **Investigate:** The shut-off valves, above the basement bathroom sink, is significantly corroded and should be monitored closely for future leakage. Ultimately, repair or replacement of the shut-off valves will be necessary.
- **Investigate:** An unusual "knocking noise" was occurring when the hot water supply was turned on and then off at the kitchen sink. This is deemed unusual, and you may wish to consult a qualified plumber for further investigation.

Water Heater

- ***FYI:*** *Your water heater generally is an easy-care appliance. There are three regular maintenance tasks to remember to increase the life of your water heater: Once every six months, drain one gallon of water from the tank. If you have hard water, do this every month. This reduces the amount of sediment collecting in the bottom of the tank, which can make the burner or heating coils work harder. Once every two years, have your water heater inspected by a service technician. This will help keep it in peak operating condition, and will prevent dangerous carbon monoxide problems. Once every five years, have a service technician replace the anode rod. This small metal device sacrifices itself for the good of the entire unit: instead of attacking the tank itself, the corrosive chemicals in the water are drawn to the anode rod. Before doing any maintenance on your water heater shut-off power at the service panel and read your owner's manual.*

LIMITATIONS OF PLUMBING INSPECTION

As we have discussed and as described in your inspection contract, this is a visual inspection limited in scope by (but not restricted to) the following conditions:

- **Underground sewer lines are not inspected and are not part of a general home inspection.**

Рисунок 2.3 – Звіт про технічний огляд будинку, детальні текстові висновки

В ході інтерв'ю з 8 потенційними покупцями було виявлено кілька важливих аспектів:

1. Більшість покупців (85%) відчувають труднощі з інтерпретацією технічної інформації у звітах
2. Практично всі респонденти (92%) хотіли б бачити чітку категоризацію проблем за рівнем критичності
3. Значна частина опитаних (78%) відзначила потребу в конкретних рекомендаціях щодо подальших дій
4. Майже всі учасники (95%) висловили зацікавленість в отриманні приблизної оцінки вартості необхідних ремонтних робіт

Аналіз структури наданих звітів технічного огляду показав, що хоча вони містять всю необхідну технічну інформацію, формат її подання не оптимальний для швидкого прийняття рішень непрофесійними користувачами. Було виявлено, що критичні проблеми часто "загублені" серед менш важливих зауважень, а рекомендації щодо подальших дій розпорошені по тексту документа.

На основі отриманих даних було розроблено систему класифікації проблем за п'ятьма рівнями критичності та створено шаблон для структурованого представлення результатів аналізу. Цей шаблон ліг в основу промпту для мовної моделі, що дозволяє автоматизувати процес аналізу та представлення даних у форматі, оптимізованому для прийняття рішень.

Розроблений шаблон включає:

- Загальне враження про стан об'єкта
- Категоризовані списки виявлених проблем з поясненнями
- Оцінку сильних та слабких сторін об'єкта
- Чіткі рекомендації щодо доцільності купівлі

- Конкретні кроки для подальших дій

Така структура дозволяє трансформувати технічну інформацію зі звітів у формат, що відповідає реальним потребам користувачів при прийнятті рішень щодо купівлі нерухомості.

2.1.2 Розробка системи конвертації PDF в текстовий формат

На етапі дослідження структури звітів технічного огляду було виявлено, що хоча звіти містять значну кількість фотографій, вся важлива інформація дублюється в текстовому описі. Фотографії слугують лише для візуального підтвердження описаних проблем, але не містять унікальної інформації, необхідної для аналізу. Це спостереження значно спростило процес розробки системи конвертації, оскільки дозволило зосередитись виключно на вилученні текстового контенту.

Процес розробки конвертації PDF у текстовий формат включав наступні етапи:

1. **Початкова розробка в Jupyter Notebook**
    - Створено експериментальне середовище для тестування різних підходів до конвертації
    - Випробувано різні бібліотеки для роботи з PDF (PyPDF2, pdfplumber, PDFMiner)
    - Проведено експерименти з вилучення тексту на тестових звітах
2. **Реалізація базової функціональності**
    - Розроблено функції для читання PDF файлу та отримання plain text
    - Додано попередню обробку тексту для видалення зайвих пробілів та форматування
    - Реалізовано збереження результатів у текстовому форматі

Розроблений модуль конвертації став основою для подальшої розробки системи автоматизованого аналізу звітів. Відмова від обробки зображень дозволила значно спростити архітектуру системи та підвищити її надійність, зосередившись на аналізі текстового контенту, який містить всю необхідну інформацію для формування висновків про стан нерухомості.

2.1.3 Реалізація OCR-потоку для сканованих документів

В ході розробки системи виявилось, що частина звітів технічного огляду не містить текстового шару і складається лише зі сканованих зображень. Для обробки таких документів було необхідно реалізувати додатковий OCR-потік. Початковий аналіз показав наявність кількох потенційних рішень - від традиційних OCR-систем, таких як Amazon Textract та PyTesseract, до сучасних підходів з використанням штучного інтелекту та мультимодальних мовних моделей.

Для вибору оптимального рішення було проведено порівняльне тестування різних підходів з оцінкою швидкості обробки, точності розпізнавання та вартості використання. Особлива увага приділялась здатності систем коректно розпізнавати специфічну технічну термінологію, характерну для звітів технічного огляду нерухомості.

Несподіваним результатом дослідження стало те, що використання мультимодальних мовних моделей виявилось не лише ефективнішим, але й економічно вигіднішим рішенням. Вартість обробки однієї сторінки склала $0.01-0.02 порівняно з $0.05-0.06 при використанні Amazon Textract. Крім того, мовні моделі забезпечували швидшу обробку - 2-3 секунди на сторінку проти 5-7 секунд у традиційних OCR-рішень. Додатковою перевагою стала можливість використання єдиного API для всіх етапів обробки документів. Промпт для OCR за допомогою мовних моделей показано на Рисунку 2.4.

```
markdown_parser_system_prompt = '''
You're a helpful assistant for blind property buyers helping to read the texts of the
home inspection reports.

You read the text of the report, and convert it to markdown using the proper syntax.

User will send you the scanned pages one by one, and you would respond with the markdown
code, without any additional text.

1. Don't include the header with the title and footnotes in the content.
2. Use markdown for titles, lists, tables, etc.
3. Don't add anything, just respond with the parsed text.
5. For the photos, explain them with text, see <image> example below. Don't insert the
markdown image. Be as precise and detailed as possible.

<image>
The photo shows a door with visible missing hardware components. The following details
can be observed:
- **Strike Plate**: The strike plate is missing, leaving an empty hole where it should
be installed.
- **Handle Set**: There is no handle or knob present on the door.
- **Fasteners and Hinges**: Potential absence or improper installation of fasteners and
hinges, which may impact the door's alignment and functionality.
The condition suggests that the door may not function properly until the missing parts
are installed. The lack of a strike plate, handle, and potentially other components can
prevent the door from latching or locking securely, posing a security and operational
concern.
</image>
'''
```

Рисунок 2.4 – Промпт для OCR

З урахуванням специфіки взаємодії з користувачами, було розроблено систему сповіщень про необхідність та тривалість OCR-обробки. Користувачі отримують повідомлення про додатковий час обробки відразу після завантаження документа, а після завершення розпізнавання надсилається email-нотифікація. Для оптимізації роботи системи також реалізовано кешування результатів розпізнавання, що дозволяє уникнути повторної обробки однакових документів.

Тестування розробленого OCR-потоку (Рисунок 2.5) на реальних звітах технічного огляду підтвердило його ефективність незалежно від формату вхідних документів. Система успішно обробляє як документи з текстовим шаром, так і

скановані звіти, забезпечуючи високу якість розпізнавання тексту для подальшого аналізу.

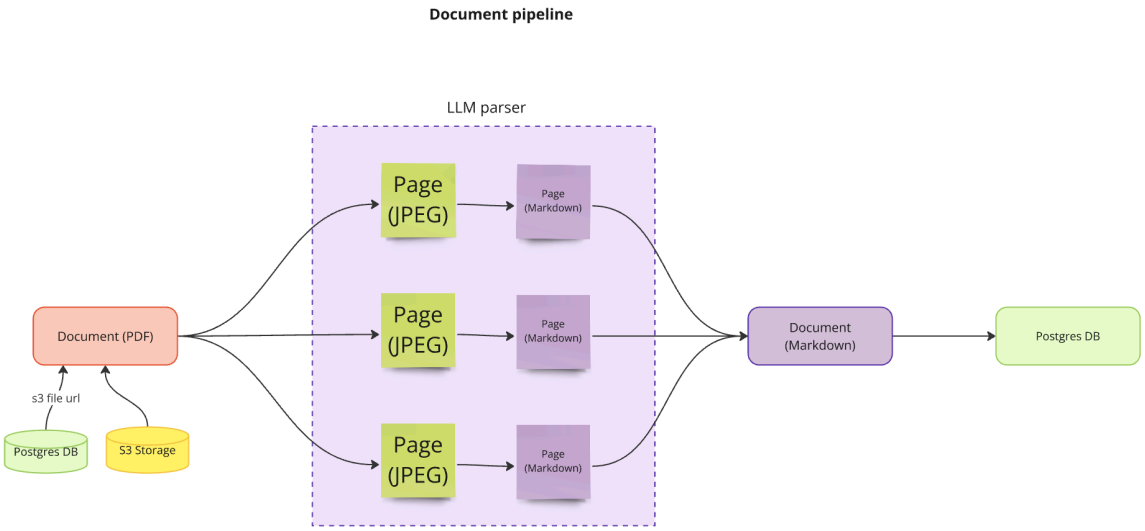


Рисунок 2.5 – Діаграма роботи алгоритму OCR

2.2 Розробка модуля аналізу документів

2.2.1 Розробка системи вилучення структурованих даних

На основі результатів дослідження структури звітів та потреб користувачів було розроблено систему вилучення структурованих даних, що складається з двох основних етапів. На першому етапі відбувається загальне автореферування документа з виділенням ключових моментів. На другому етапі здійснюється детальне структурування даних з використанням спеціально розроблених інструментів для роботи з LLM.

Для підвищення якості вилучення даних процес було розділено на два кроки замість прямої конвертації в JSON. На першому кроці система генерує markdown-резюме звіту, що містить основні знахідки у форматі, зручному для

подальшої обробки. Цей проміжний крок дозволяє отримати більш якісні та послідовні результати порівняно з прямою екстракцією даних.

На другому кроці система використовує спеціалізовані tool calls для конвертації markdown-резюме у структурований JSON-формат. Розроблена схема містить масив знахідок, де кожна знахідка представлена об'єктом з полями для назви та опису проблеми. Додатково для кожної виявленої проблеми система генерує детальний опис, оцінку вартості необхідних ремонтних робіт та рекомендації щодо усунення.

Початкова розробка та тестування системи проводились у середовищі Jupyter Notebook, що дозволило швидко експериментувати з різними підходами та оптимізувати процес обробки. Після підтвердження ефективності обраного підходу код було перенесено в Flask API для інтеграції з іншими компонентами системи.

Важливим аспектом розробки стало забезпечення надійної валідації отриманих структурованих даних. Для кожного етапу обробки було впроваджено механізми перевірки коректності та повноти даних, що дозволяє гарантувати якість результатів аналізу для кінцевих користувачів.

2.2.2 Вилучення даних у markdown

Ключовим етапом у розробці системи аналізу документів стала реалізація модуля генерації markdown. Основою для цього модуля став спеціально розроблений промпт, що визначає правила та структуру генерації резюме звіту.

В розробленому промпті (Рисунок 2.6) система представляється як "Home Inspection Report Reviewer Pro" - професійний аналітик звітів технічного огляду нерухомості. Промпт детально описує послідовність дій: спочатку читання звіту, потім його аналіз, і нарешті - генерація структурованого резюме за визначеним шаблоном.

Шаблон резюме включає:

- Адресу об'єкта
- Початкові враження про стан нерухомості
- Детальний перелік знайдених проблем, класифікованих за п'ятьма рівнями критичності
- Підсумки та рекомендації, включаючи сильні та слабкі сторони об'єкта
- Загальну оцінку за десятибальною шкалою
- Чіткі рекомендації щодо купівлі
- Конкретні наступні кроки

Для оптимізації роботи LLM з цим шаблоном було використано розмітку markdown, що забезпечує чітку структуру вихідного тексту та полегшує його подальшу обробку. Кожен розділ має свій рівень заголовків, а ключові елементи виділяються спеціальним форматуванням.

Тестування показало, що такий підхід до структурування даних значно підвищує якість та послідовність аналізу, забезпечуючи формат виводу, який легко читається як людьми, так і машинами для подальшої обробки.

```
system_prompt_home_inspection = """
You are Home Inspection Report Reviewer Pro ⭐
Your mission is to conduct precise reviews of the Home Inspection Reports.
You will strictly follow the steps.

# Steps
Step 1. Read Home Inspection Report.
Step 2. Review the Home Inspection Report, and give a summary based on the Home
Inspection Report Review Template.
Step 3. If the user raises a question, please use the markdown format to give a specific
and lengthy reply.

<Template>
## Home Inspection Report Review

### Property Address:
[Address of the property. String or 'N/A'. Example: '123 Main St, Anytown, CA 12345']
```

```
- **Initial Thoughts**:
[General impression of the property's condition.]

### Findings

#### 1. Critical Issues
[List of Critical issues. Title + One sentence to explain why this is an issue. 'None'
if none.]

#### 2. Major Issues
[List of Major issues. Title + One sentence to explain why this is an issue.'None' if
none.]

#### 3. Moderate Issues
[List of Major issues. Title + One sentence to explain why this is an issue.'None' if
none.]

#### 4. Minor Issues
[List of Major issues. Title + One sentence to explain why this is an issue. 'None' if
none.]

#### 5. Information only Issues
[List of Information only issues. Title + One sentence to explain why this is an issue.
'None' if none.]

### Summary and Final Recommendations
- **Strengths**: [Highlight the key strengths of the property.]
- **Weaknesses**: [Summarize the main weaknesses or concerns.]
- **Overall Score**: x/10
- **Final Recommendation**: [Recommend to proceed with buying the property, appraisal,
or contingency exit, with justifications.]


### Actionable Next Steps
[More specific suggestions on the next steps.]

</Template>

"""
```

Рисунок 2.6 – Промпт для автореферування за заданим шаблоном

2.2.3 Конвертація в JSON структуру

Другим етапом вилучення структурованих даних стала розробка системи конвертації markdown-резюме в JSON формат. Розробка базувалась на спостереженні, що структура даних в markdown-форматі ідентична необхідній

JSON структурі, що дозволило оптимізувати процес через використання різних моделей на різних етапах обробки.

Було розроблено tool call (Рисунок 2.7) для вилучення структурованих даних, який точно відображає структуру markdown-шаблону. Наприклад, якщо в markdown ми маємо розділи для критичних, основних та помірних проблем, то в JSON схемі ці проблеми представлені як масив об'єктів з полями title, description та severity, де severity може приймати відповідні значення: "critical", "major", "moderate", "minor" або "informational".

Використання Langchain дозволило застосовувати різні моделі для різних етапів обробки, зберігаючи єдиний інтерфейс через tool calls. Так, для початкового аналізу та генерації markdown використовується потужніша модель, а для конвертації в JSON - легша версія, оскільки завдання вже спрощене наявністю чіткої структури. Це оптимізує як швидкодію, так і вартість обробки.

Тестування на великій кількості звітів показало, що такий двоетапний підхід з використанням проміжного markdown-формату забезпечує вищу якість та надійність вилучення даних порівняно з прямою конвертацією в JSON. Ідентичність структур markdown та JSON значно спрощує процес перетворення та зменшує ймовірність помилок.

```
home_inspection_report_tool = {
   "type": "function",
   "function": {
       "name": "home_inspection_report_tool",
       "description": "Parses a home inspection report and returns the data in a structured
format. The report includes initial thoughts, findings (each with a severity), strengths,
weaknesses, overall score, final recommendations, and actionable next steps.",
       "parameters": {
           "type": "object",
           "properties": {
               "property_address": {
                   "description": "Specify the property address. Expected output: String or
'N/A'. Example: '123 Main St, Anytown, CA 12345'",
                   "type": "string"
               },
               "initial_thoughts": {
                   "description": "General impression of the property's condition.",
```

```
                    "type": "string"
                },
                "findings": {
                    "description": "List of findings, each with a title, description, and severity
level.",
                    "type": "array",
                    "items": {
                        "type": "object",
                        "properties": {
                            "title": {
                                "description": "Title of the issue. Choose and prepend a relevant
emoji to the title, e.g. 'Foundation Water Seepage' -> '💧 Foundation Water Seepage'.",
                                "type": "string"
                            },
                            "description": {
                                "description": "Detailed description of the issue.",
                                "type": "string"
                            },
                            "severity": {
                                "description": "Severity level of the issue.",
                                "type": "string",
                                "enum": [
                                    "critical",
                                    "major",
                                    "moderate",
                                    "minor",
                                    "informational"
                                ]
                            }
                        },
                        "required": ["title", "description", "severity"]
                    }
                },
                "summary": {
                    "type": "object",
                    "properties": {
                        "strengths": {
                            "description": "Highlight the key strengths of the property.",
                            "type": "string"
                        },
                        "weaknesses": {
                            "description": "Summarize the main weaknesses or concerns.",
                            "type": "string"
                        },
                        "overall_score": {
                            "description": "A score from 1 to 10 representing the overall
condition of the property.",
                            "type": "number"
                        },
                        "final_recommendation": {
                            "description": "A final recommendation on whether to proceed with
buying the property, appraisal, or contingency exit, along with justifications.",
                            "type": "string"
                        }
                    }
                },
```

```
                "actionable_next_steps": {
                    "description": "Specific suggestions on the next steps that should be taken.",
                    "type": "array",
                    "items": {
                        "type": "string"
                    }
                }
            },
            "required": ["initial_thoughts", "findings", "summary", "actionable_next_steps"]
        }
    }
}
```

Рисунок 2.7 – Функція екстрагування даних в синтаксисі OpenAI Tool

2.2.4 Деталізація виявлених проблем

На основі результатів загального аналізу звіту, система виконує детальний аналіз кожної виявленої проблеми. Для забезпечення найбільш точних результатів, кожен запит супроводжується повним текстом звіту технічного огляду, що надає моделі максимально повний контекст для аналізу.

Процес деталізації також реалізований через двоетапний підхід, аналогічний до загального аналізу звіту. На першому етапі використовується промпт, що інструктує модель надати детальний опис проблеми за визначеним шаблоном, який включає:

- Детальний опис проблеми
- Потенційні причини виникнення
- Рекомендовані дії для усунення
- Орієнтовний діапазон вартості ремонту

На другому етапі (Рисунок 2.8), отриманий markdown-текст конвертується у структурований JSON формат за допомогою спеціалізованого tool call. Схема JSON включає всі необхідні поля та забезпечує валідацію даних, зокрема перевірку наявності числових значень для діапазону вартості ремонту.

Особливістю цього етапу є те, що деталізація кожної проблеми відбувається у контексті повного звіту, що дозволяє моделі враховувати взаємозв'язки між різними проблемами та надавати більш точні оцінки їх серйозності та вартості усунення. Використання Langchain дозволяє ефективно керувати взаємодією з різними моделями, зберігаючи єдиний інтерфейс для обробки даних.

```
system_prompt_home_inspection_issue_analysis = """
You are a helpful assistant that helps with the home inspection report analysis.

User will provide you with a specific issue, and you will respond with the details using
the <Issue Details Template>. Don't add any tags, just respond with the text content.

<Issue Details Template>
## Issue details
...

## Potential causes
...

## Recommended actions
...

## Estimated fix price range
... price range for the state/city where the property is located
</Issue Details Template>

The home inspection report is provided below:
{raw_content}
"""
```

Рисунок 2.8 – Промпт для автореферування однієї проблеми за заданим шаблоном

2.2.5 Оптимізація моделей через LangSmith

Після створення основних компонентів системи було проведено тестування та оптимізацію запитів через платформу LangSmith. Аналіз статистики, представленої на Рисунку 2.9, показав оптимальний баланс вартості та якості при використанні різних моделей на різних етапах обробки.

Рисунок 2.9 демонструє обробку одного звіту, що включає:

- Початковий аналіз звіту (~7,000 токенів, $0.022)
- Вилучення структурованих даних (~800 токенів, $0.0002 за запит)
- Детальний аналіз кожної проблеми (~6,500 токенів, $0.018 за проблему)

На основі цих даних було визначено оптимальну комбінацію моделей:

- Для первинного аналізу та генерації markdown – Claude 3.5 Sonnet, враховуючи необхідність глибокого розуміння контексту
- Для конвертації в JSON та деталізації проблем – GPT-4o Mini або Claude 3 Haiku, оскільки завдання вже структуроване через tool calls

Така оптимізація дозволила знизити середню вартість обробки одного звіту до приблизно $0.20 при збереженні високої якості аналізу, що робить сервіс економічно ефективним для користувачів.

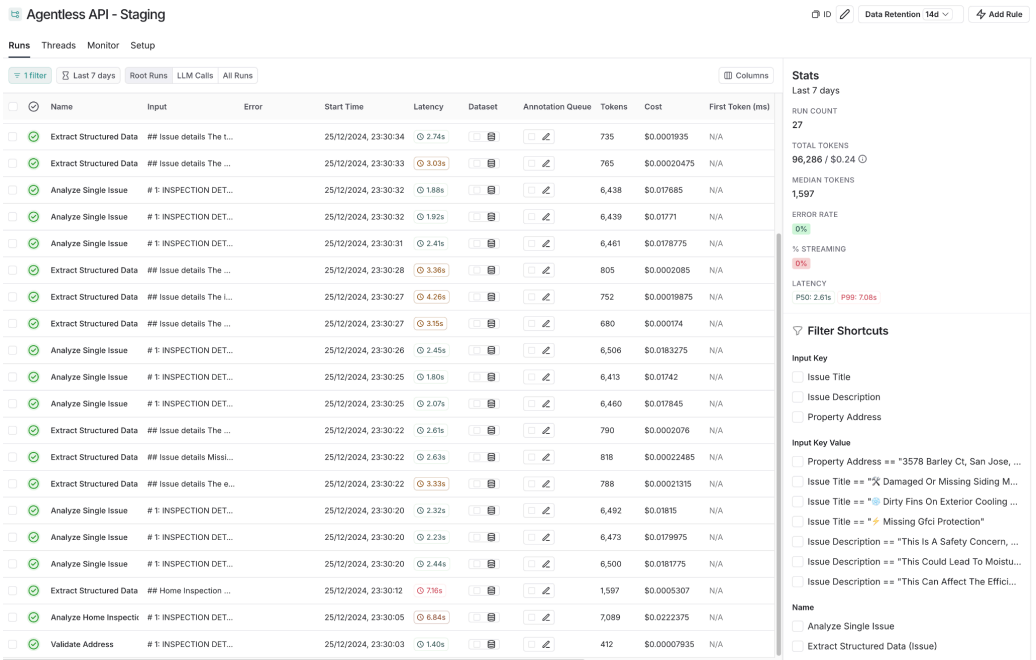


Рисунок 2.9 – Обробка звіту технічного огляду в LangSmith

2.3 Розробка серверної частини

2.3.1 Архітектура Flask API

Серверна частина системи реалізована як RESTful API на базі Flask з використанням розширення Flask-RESTx для автоматичної генерації документації Swagger. API застосовує модульну архітектуру з чітким розділенням відповідальності між компонентами.

Основним компонентом системи є група ендпоінтів для обробки звітів технічного огляду. Базовий ендпоінт /home-inspection/validate_address забезпечує валідацію відповідності адреси в звіті та властивості, що є важливим першим кроком аналізу. Наступний ендпоінт /home-inspection/parse відповідає за початкову обробку PDF-документа, включаючи конвертацію в текстовий формат та за необхідності OCR-розпізнавання.

Центральним елементом API є ендпоінт /home-inspection/analyze (Рисунок 2.10), який виконує загальний аналіз звіту з генерацією markdown-резюме. Важливою особливістю цього ендпоінту є підтримка асинхронної обробки - після завершення основного аналізу, детальний розбір кожної виявленої проблеми виконується у фоновому режимі з використанням багатопотоковості. Це дозволяє суттєво прискорити відгук системи для користувача.

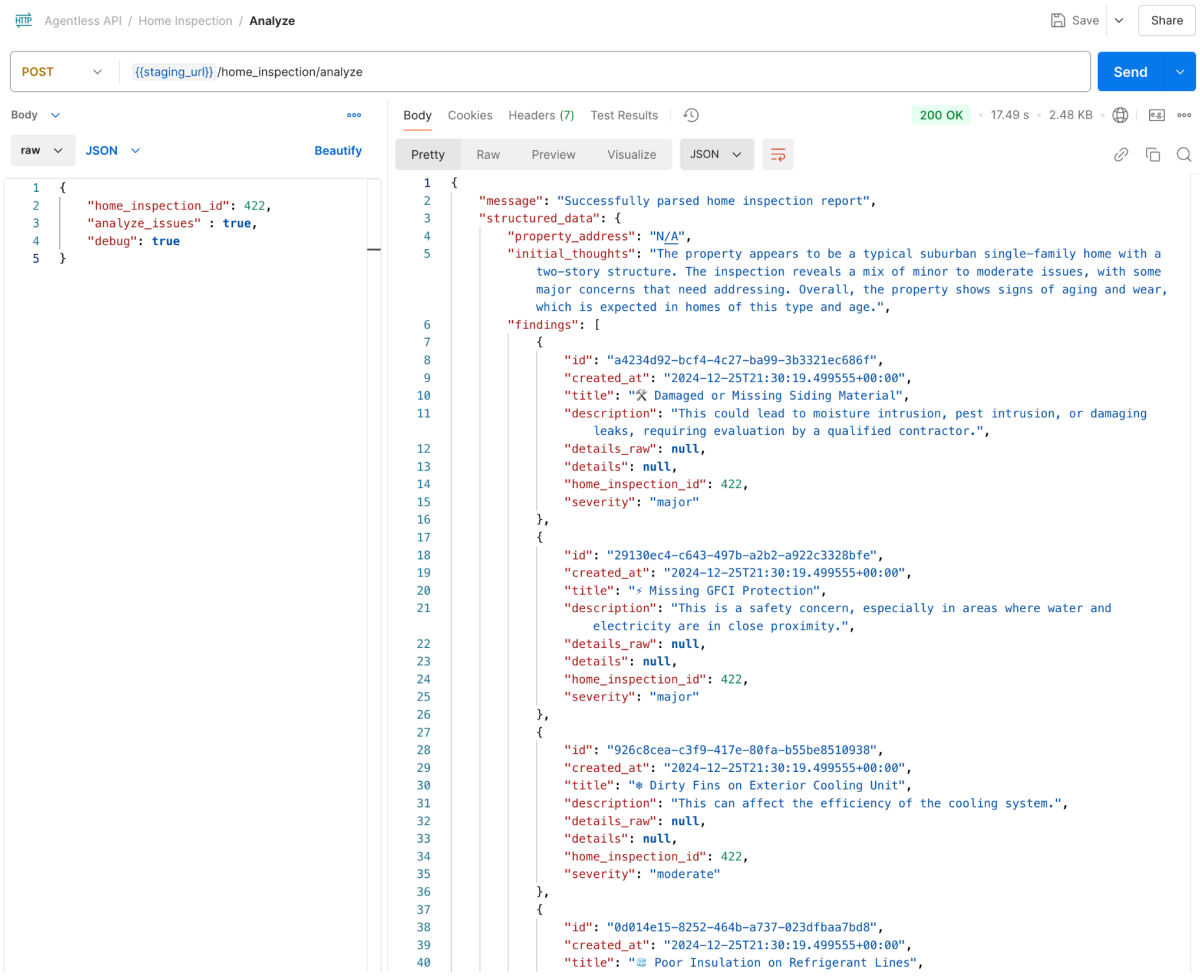


Рисунок 2.10 – Приклад роботи API

Для обробки окремих елементів звіту реалізовано два спеціалізовані ендпоінти. /home-inspection/single_issue забезпечує детальний аналіз окремої проблеми з генерацією рекомендацій та оцінки вартості ремонту. /home-inspection/analyze_issues_bulk призначений для масової паралельної обробки всіх виявлених проблем, що оптимізує загальну швидкодію системи.

Архітектура API (Рисунок 2.11) передбачає гнучке налаштування процесу обробки через систему параметрів. Користувачі можуть обирати різні моделі для кожного етапу аналізу, включати режим налагодження та запускати повторний аналіз існуючих даних. Безпека системи забезпечується декоратором auth_required, який перевіряє автентифікацію кожного запиту.

Допоміжна функціональність включає генерацію пресайн URL для завантаження файлів в S3 та систему логування, яка відстежує процес обробки та вимірює час виконання кожної операції. Валідація вхідних даних реалізована через моделі API, що забезпечує надійність та передбачуваність роботи системи.

```
api/
├── endpoints/
│   ├── customer/
│   ├── home_inspection/
│   │   ├── __init__.py
│   │   ├── validate_address.py        # Перевірка відповідності адрес
│   │   ├── parse.py                   # Обробка PDF та OCR
│   │   ├── analyze.py                 # Основний аналіз звіту
│   │   ├── single_issue.py            # Аналіз окремих проблем
│   │   └── analyze_issues_bulk.py     # Масова обробка проблем
│   └── property/
├── utils/
│   ├── langchain/
│   │   ├── models.py                  # Налаштування моделей
│   │   ├── prompts.py                 # Системні промпти
│   │   └── tools.py                   # Визначення інструментів
│   ├── pdf/
│   │   ├── converter.py               # Вилучення тексту з PDF
│   │   └── ocr.py                     # Обробка сканованих документів
│   └── s3.py                          # Операції з S3 сховищем
├── models/
│   └── home_inspection/
│       ├── request.py                 # Моделі запитів
│       └── response.py                # Моделі відповідей
├── main.py                           # Головний файл застосунку
├── requirements.txt                  # Залежності проекту
└── Dockerfile                        # Конфігурація контейнера
```

Рисунок 2.11 – Архітектура Flask API

2.3.2 Розробка системи зберігання документів в Amazon S3

Для надійного зберігання та управління документами було розроблено систему на базі Amazon S3.

Структура зберігання

Документи зберігаються в єдиному бакеті agentless-app з наступною структурою папок:

- {s3_folder}/home_inspection/ - для звітів технічного огляду
- {s3_folder}/other/ - для інших документів пов'язаних з нерухомістю

Система завантаження файлів

Для безпечного завантаження файлів було реалізовано систему генерації presigned URL, що дозволяє клієнтам завантажувати файли напряму в S3. Ключова логіка реалізована в функції generate_presigned_url (Рисунок 2.12)

```
def generate_presigned_url(data):
    property_id = data.get('property_id')
    filename = data.get('filename')
    file_type = data.get('file_type')
    content_type = data.get('content_type', 'application/octet-stream')

    if not property_id or not filename:
        return jsonify({'error': 'Missing required fields'}), 400

    properties_data = supabase.table('properties').select('s3_folder').eq('id', property_id).execute().data
    if not properties_data:
        return jsonify({'error': 'Property not found'}), 404
        s3_folder = properties_data[0]['s3_folder']
    s3_key = f"{s3_folder}/home_inspection/{filename}" if file_type == 'home_inspection_report' else
f"{s3_folder}/other/{filename}"

    try:
        presigned_url = s3_client.generate_presigned_url(
            'put_object',
            Params={
                'Bucket': S3_BUCKET,
                'Key': s3_key,
                'ContentType': content_type
            },
            ExpiresIn=3600
        )
        return jsonify({'url': presigned_url})
    except NoCredentialsError:
        return jsonify({'error': 'AWS credentials not found'}), 500
```

Рисунок 2.12 – Генерація Presigned URL

Цей підхід забезпечує безпечне та ефективне завантаження файлів безпосередньо з клієнта в S3, при цьому зберігаючи контроль над доступом та структурою зберігання документів.

2.4 Розробка клієнтської частини

2.4.1 Імплементація завантаження документів

Для забезпечення зручного та надійного завантаження документів було розроблено React-компонент, що поєднує drag-and-drop інтерфейс з можливістю вибору файлів через стандартний діалог.

Розроблений компонент надає користувачам два способи завантаження документів: через перетягування файлів у спеціальну зону або через стандартний діалог вибору файлів, який відкривається при кліку на цю зону. Інтерфейс реагує на дії користувача, змінюючи свій вигляд при перетягуванні файлів та під час завантаження.

Процес завантаження файлів реалізований через двоетапний механізм. Спочатку компонент робить запит до серверного API для отримання presigned URL. Після успішного отримання URL, файл завантажується безпосередньо в S3 бакет, минаючи сервер застосунку. Такий підхід забезпечує ефективне використання ресурсів та покращує швидкість завантаження.

Для забезпечення надійності завантаження реалізовано валідацію файлів: приймаються тільки PDF документи розміром до 10MB. Під час завантаження компонент блокується для запобігання повторним завантаженням, а користувач отримує візуальну індикацію процесу. У випадку помилок система відображає відповідні повідомлення, що допомагає користувачам зрозуміти та виправити проблему.

Компонент (Рисунок 2.13) можна легко інтегрувати в будь-яку частину застосунку, передавши йому ідентифікатор об'єкта нерухомості та callback-функцію, яка буде викликана після успішного завантаження. Це дозволяє гнучко реагувати на завершення завантаження та оновлювати стан застосунку відповідним чином.

```
<FileUpload
  propertyId="123"
  onSuccess={() => {
    console.log('File uploaded successfully');
  }}
/>
```

Рисунок 2.13 – Приклад використання компонента FileUpload

2.4.2 Розробка інтерфейсу перегляду результатів аналізу

На основі структури даних, що генерується AI-моделлю, було розроблено інтерактивний інтерфейс (Рисунок 2.14, 2.15) для перегляду результатів аналізу звіту. Важливою особливістю реалізації стала можливість миттєвого відображення загальних результатів аналізу, в той час як детальний аналіз кожної проблеми продовжується у фоновому режимі. Таке рішення значно покращує користувацький досвід, дозволяючи приступити до роботи з результатами, не чекаючи повного завершення аналізу.

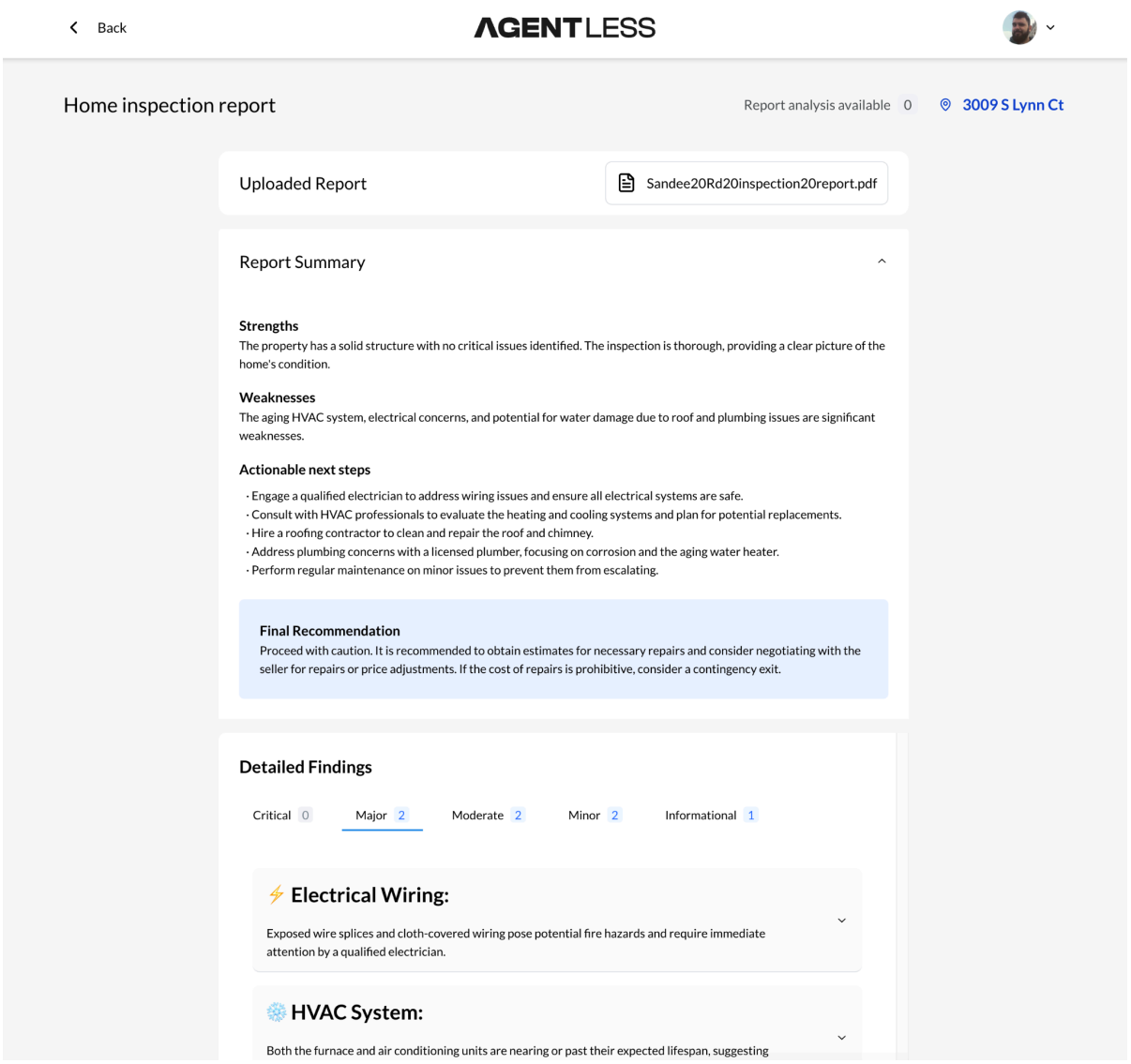


Рисунок 2.14 – Інтерфейс користувача Agentless

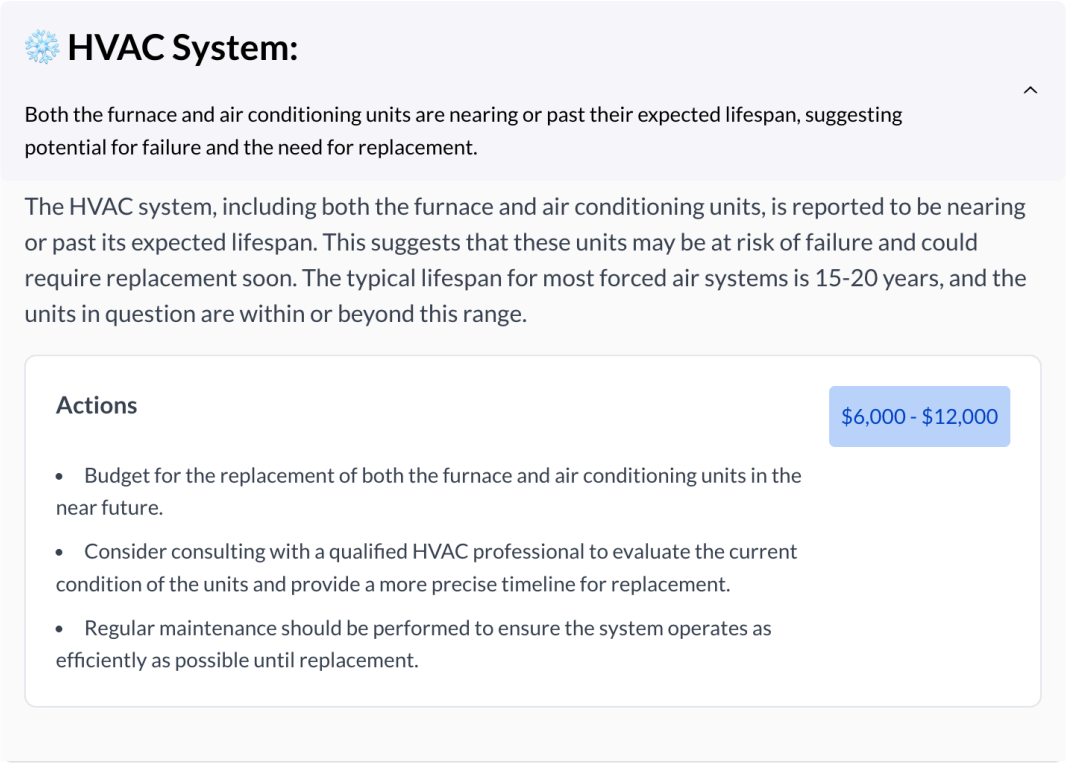


Рисунок 2.15 – Інтерфейс детального опису виявленої проблеми

Розроблений інтерфейс структурує результати аналізу в декілька основних блоків. Головний екран відображає загальну оцінку стану нерухомості та ключову рекомендацію щодо подальших дій. Під ним розміщені секції сильних та слабких сторін об'єкта, що дозволяє швидко оцінити його переваги та недоліки.

Детальні знахідки організовані в інтерактивний акордеон, де кожна проблема візуально маркована відповідно до її критичності - від червоного кольору для критичних проблем до сірого для інформаційних повідомлень. Такий підхід дозволяє користувачам швидко ідентифікувати найважливіші проблеми, не втрачаючи доступу до повної інформації.

Особливістю реалізації є те, що користувач отримує доступ до загального аналізу звіту відразу після його обробки, в той час як система продовжує детальний аналіз кожної виявленої проблеми у фоновому режимі. Результати цього аналізу автоматично зберігаються в базі даних та стають доступними

користувачу в міру їх готовності, що забезпечує оптимальний баланс між швидкістю відгуку системи та повнотою наданої інформації.

2.4.3 Інтеграція системи сповіщень користувача

Оскільки процес аналізу документів вимагає певного часу, було розроблено комплексну систему сповіщень для інформування користувачів про статус обробки їхніх документів.

Для своєчасного інформування користувачів було реалізовано багаторівневу систему сповіщень:

1. **Миттєві результати**: Стандартний аналіз звіту займає 17-40 секунд, що дозволяє користувачам отримати основні результати майже миттєво. Вони бачать загальну оцінку, рекомендації та список виявлених проблем одразу після завершення первинної обробки.
2. **Email сповіщення** (Рисунок 2.16, 2.17): Інтеграція з сервісами Make та SendGrid забезпечує автоматичне надсилання email-повідомлень користувачам. Коли аналіз повністю завершено (включаючи детальний аналіз кожної проблеми), користувач отримує email з посиланням на повний звіт.

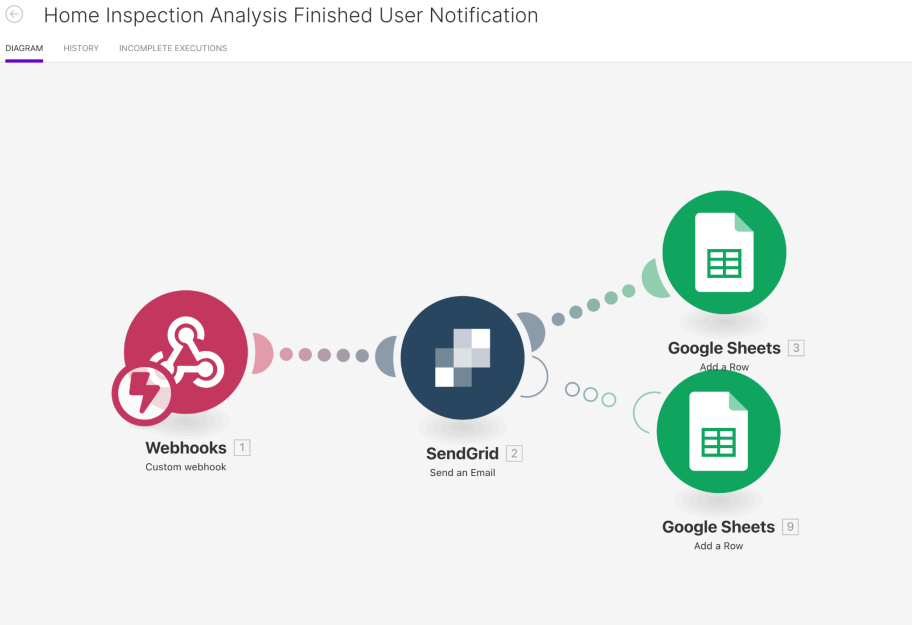


Рисунок 2.16 – Імейл сповіщення, інтеграція Make+Sendgrid

3. **Обробка помилок** (Рисунок 2.17, 2.18): У випадку виникнення помилок система не тільки показує відповідне повідомлення користувачу, але й автоматично сповіщає адміністраторів. Адміністратори мають доступ до детальної інформації про помилку та можуть:
    - Переглянути логи обробки
    - Запустити процес аналізу вручну
    - Внести корективи в налаштування системи

Така багаторівнева система сповіщень забезпечує прозорість процесу обробки та дозволяє оперативно реагувати на можливі проблеми, що значно покращує користувацький досвід.

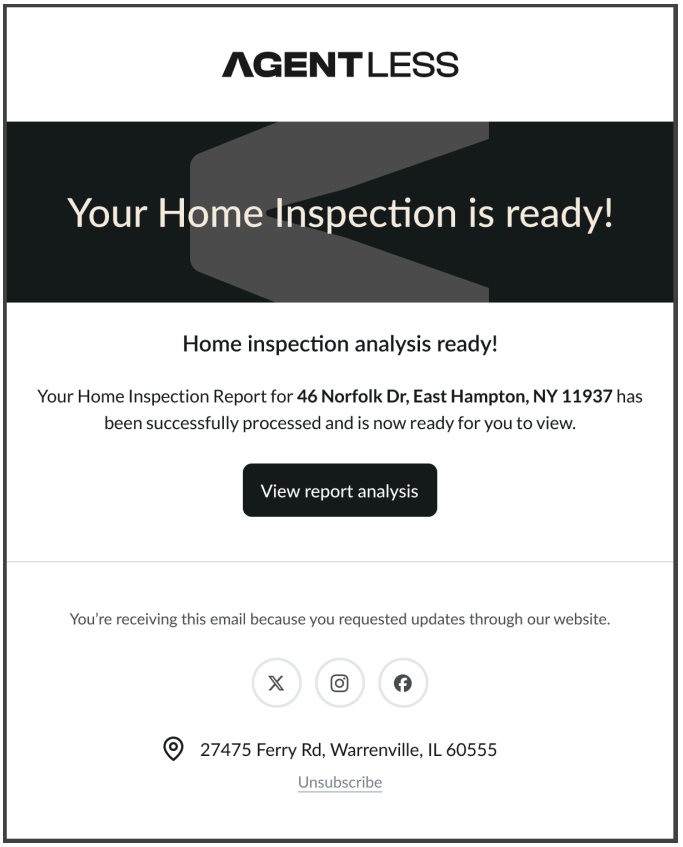


Рисунок 2.17 – Імейл сповіщення, шаблон листа у Sendgrid

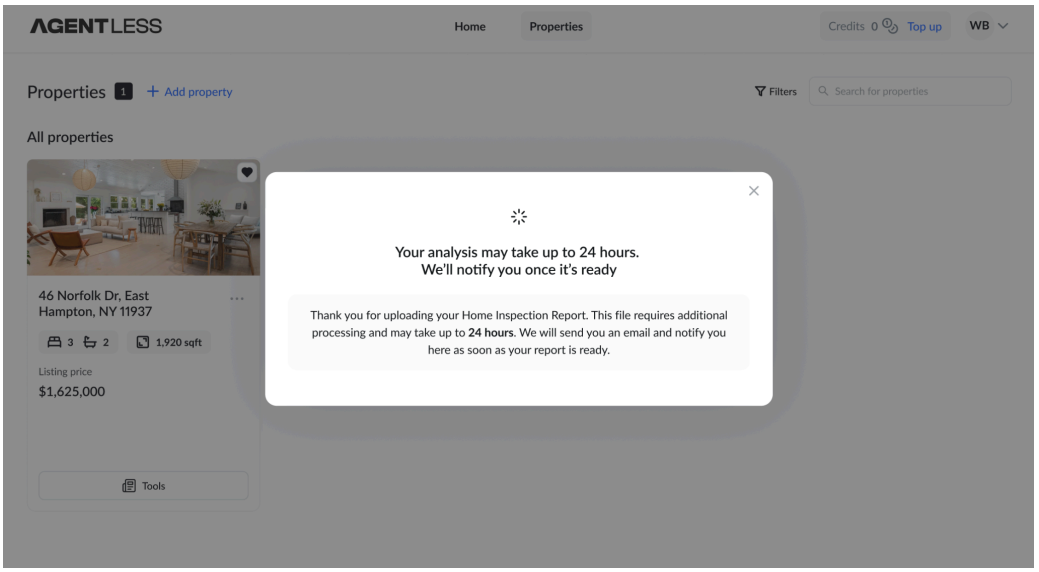


Рисунок 2.18 – Повідомлення користувача про помилку

3 РОЗГОРТАННЯ, ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ

3.1 Розгортання та впровадження

3.1.1 Налаштування CI/CD pipeline

Для забезпечення надійного процесу розробки та розгортання було налаштовано CI/CD pipeline на базі GitHub Actions з розгортанням в AWS Elastic Beanstalk. Особливістю реалізації стала підтримка двох середовищ - staging та production, що дозволяє безпечно тестувати зміни перед виходом в продакшн.

Pipeline автоматично запускається при коміті в гілки main або staging та включає наступні ключові етапи:

1. Управління конфігурацією: На початку процесу система автоматично генерує відповідний .env файл в залежності від цільового середовища. Це забезпечує правильну конфігурацію для кожного середовища розгортання.

2. Підготовка Docker-образу: Застосунок пакується у Docker-контейнер та завантажується до Amazon ECR. Для кожного середовища використовується окремий репозиторій (agentless-api-prod та agentless-api-staging), що забезпечує ізоляцію артефактів.

3. Розгортання: Фінальний етап використовує AWS Elastic Beanstalk для розгортання застосунку. В залежності від цільової гілки, розгортання відбувається в різні оточення (AgentlessApiEnv для production та AgentlessApiStagingEnv для staging).

   Особливості реалізації:

- Автоматичне створення ECR репозиторіїв при їх відсутності
- Використання тегів latest та SHA коміту для версіонування образів

- Налаштування логування через Nginx
- Конфігурація портів для правильної маршрутизації трафіку

Така архітектура забезпечує надійний процес доставки коду від розробника до продакшену з можливістю тестування на staging середовищі перед випуском в production.

3.1.2 Налаштування AWS Elastic Beanstalk та масштабування системи

В рамках розгортання системи було налаштовано два середовища AWS Elastic Beanstalk (Рисунок 3.1): виробниче (AgentlessApiEnv) та тестове (AgentlessApiStagingEnv). Обидва середовища використовують Docker як платформу для розгортання, що забезпечує ідентичність конфігурації та мінімізує проблеми при оновленнях.

```
option_settings:
  aws:autoscaling:asg:
    MinSize: '1'
    MaxSize: '5'

  aws:autoscaling:trigger:
    MeasureName: NetworkOut
    Statistic: Average
    Unit: Bytes
    Period: '5'
    BreachDuration: '5'
    UpperThreshold: '6000000'
    LowerThreshold: '2000000'
    UpperBreachScaleIncrement: '1'
    LowerBreachScaleIncrement: '-1'

  aws:elasticbeanstalk:environment:
    EnvironmentType: LoadBalanced
    LoadBalancerType: application
    ServiceRole: aws-elasticbeanstalk-ec2-role
```

```
aws:ec2:instances:
  InstanceTypes: t3.xlarge,t3.large

aws:elasticbeanstalk:application:environment:
  PYTHONDONTWRITEBYTECODE: 1
  FLASK_ENV: production
```

Рисунок 3.1 – Налаштування Elastic Beanstalk

Для забезпечення стабільної роботи системи реалізовано автоматичне масштабування з наступними параметрами:

- Мінімальна кількість екземплярів: 1
- Максимальна кількість екземплярів: 5
- Тип інстансів: t3.xlarge та t3.large для оптимального співвідношення ціна/продуктивність
- Метрика масштабування: NetworkOut (мережевий трафік) з 5-хвилинним періодом оцінки

Система моніторингу налаштована на автоматичне масштабування при досягненні порогових значень мережевого навантаження. При перевищенні верхнього порогу (6MB/s) додається новий екземпляр, при падінні нижче нижнього порогу (2MB/s) - видаляється. Такий підхід забезпечує оптимальне використання ресурсів та стабільну роботу під навантаженням.

Application Load Balancer рівномірно розподіляє навантаження між усіма активними екземплярами, а також виконує health-checks для автоматичного виключення проблемних серверів з пулу. Налаштовані security groups обмежують вхідний трафік тільки необхідними портами, забезпечуючи базовий рівень мережевої безпеки.

Для моніторингу та діагностики системи використовуються вбудовані можливості AWS Elastic Beanstalk, що дозволяють переглядати логи та метрики в реальному часі через веб-інтерфейс або AWS CLI.

3.1.3 Конфігурація хмарних сервісів

Для забезпечення надійної роботи системи було налаштовано інтеграцію з кількома ключовими хмарними сервісами:

Supabase для бази даних та автентифікації

База даних (Рисунок 3.2) побудована на PostgreSQL з чіткою структурою для зберігання звітів, результатів аналізу та користувацьких даних. Важливим аспектом стала реалізація зв'язків між таблицями через зовнішні ключі, що забезпечує цілісність даних. Автентифікація налаштована через провайдери Email та Google для максимальної зручності користувачів.

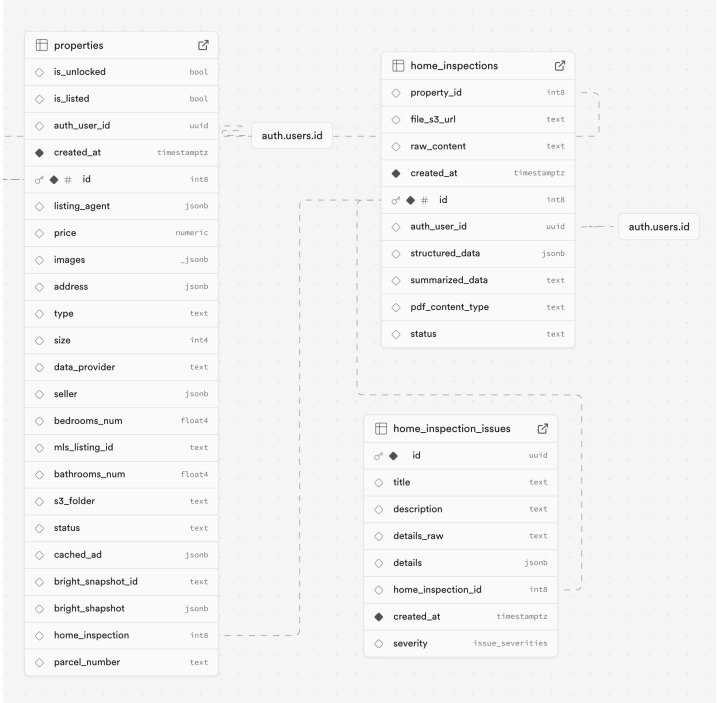


Рисунок 3.2 – Схема бази даних

Vercel для фронтенду

Frontend застосунок розгорнуто на Vercel з налаштованим автоматичним розгортанням при push в основні гілки (main та staging). Конфігурація включає автоматичне генерування preview середовищ для кожної feature гілки, що спрощує процес review коду.

Amazon S3 для зберігання файлів

Налаштовано бакет для зберігання PDF файлів звітів з відповідними IAM політиками. Для безпечного завантаження файлів реалізовано систему з presigned URL, що дозволяє клієнтам завантажувати файли напряму в S3.

SendGrid для email комунікацій

Налаштовано шаблони email повідомлень у SendGrid для сповіщення користувачів про завершення аналізу. Інтеграція реалізована через Make (раніше Integromat) для гнучкого управління тригерами та форматом повідомлень.

Всі сервіси налаштовані з урахуванням потреб як production, так і staging середовищ, з відповідними обмеженнями та моніторингом для кожного середовища. Це забезпечує безпечне тестування нового функціоналу без ризику для продакшн даних.

3.1.4 Моніторинг роботи системи

Для комплексного відстеження роботи системи та швидкого виявлення проблем було впроваджено Sentry (Рисунки 3.3, 3.4) як основний інструмент моніторингу помилок та продуктивності.

```
import * as Sentry from "@sentry/nextjs";

Sentry.init({
  dsn: process.env.NEXT_PUBLIC_SENTRY_DSN,
  enabled: process.env.NODE_ENV === 'production',
```

```
  environment: process.env.NEXT_PUBLIC_ENVIRONMENT,
  tracesSampleRate: 0.1,
  integrations: [
    new Sentry.BrowserTracing({
      tracePropagationTargets: ['app.agentless.us', /^\//],
    }),
  ],
});
```

Рисунок 3.3 – Налаштування Sentry для Next.js фронтенду

```
# config.py
import sentry_sdk
from sentry_sdk.integrations.flask import FlaskIntegration

sentry_sdk.init(
    dsn=os.environ.get("SENTRY_DSN"),
    environment=os.environ.get("ENVIRONMENT"),
    traces_sample_rate=0.1,
    integrations=[
        FlaskIntegration(),
    ],
    before_send=lambda event, hint: event if event.get('level') >= 'error' else None
)
```

Рисунок 3.4 – Налаштування Sentry для Flask бекенду

Інтеграція Sentry забезпечує:

1. Відстеження помилок в реальному часі - всі помилки автоматично фіксуються з повним стеком викликів та контекстом, включаючи інформацію про браузер користувача, версію застосунку та середовище виконання.

2. Моніторинг продуктивності - Sentry відслідковує час завантаження сторінок, тривалість API запитів та інші метрики продуктивності, що допомагає виявляти проблемні місця в системі.

3. Групування схожих помилок - система автоматично групує однотипні помилки, що спрощує їх аналіз та дозволяє визначати пріоритети при виправленні.

4. Сповіщення команди - налаштовано інтеграцію з Slack для миттєвого сповіщення розробників про критичні помилки. Сповіщення містять всю необхідну інформацію для швидкого реагування.

Особлива увага приділена налаштуванню фільтрації подій - в production середовищі фіксуються тільки помилки рівня Error та вище, тоді як в staging середовищі відслідковуються також попередження (warnings) для більш детального аналізу.

Для оптимізації витрат налаштовано вибіркову фіксацію метрик продуктивності (sample rate 10%), що забезпечує репрезентативні дані при збереженні розумного обсягу трафіку.

3.2 Тестування та оптимізація

3.2.1 Порівняльний аналіз мовних моделей

Для оптимізації роботи системи було проведено порівняльний аналіз різних мовних моделей та їх комбінацій для кожного етапу обробки документів. Тестування проводилось через платформу LangSmith, що дозволило зібрати детальні метрики по кожній моделі.

Аналіз статистики показав оптимальний баланс вартості та якості при використанні різних моделей на різних етапах обробки.

Рисунок 3.5 демонструє обробку одного звіту, що включає:

- Початковий аналіз звіту (~7,000 токенів, $0.022)

- Вилучення структурованих даних (~800 токенів, $0.0002 за запит)
- Детальний аналіз кожної проблеми (~6,500 токенів, $0.018 за проблему)

На основі цих даних було визначено оптимальну комбінацію моделей:

- Для первинного аналізу та генерації markdown – Claude 3.5 Sonnet, враховуючи необхідність глибокого розуміння контексту
- Для конвертації в JSON та деталізації проблем – GPT-4o Mini або Claude 3 Haiku, оскільки завдання вже структуроване через tool calls

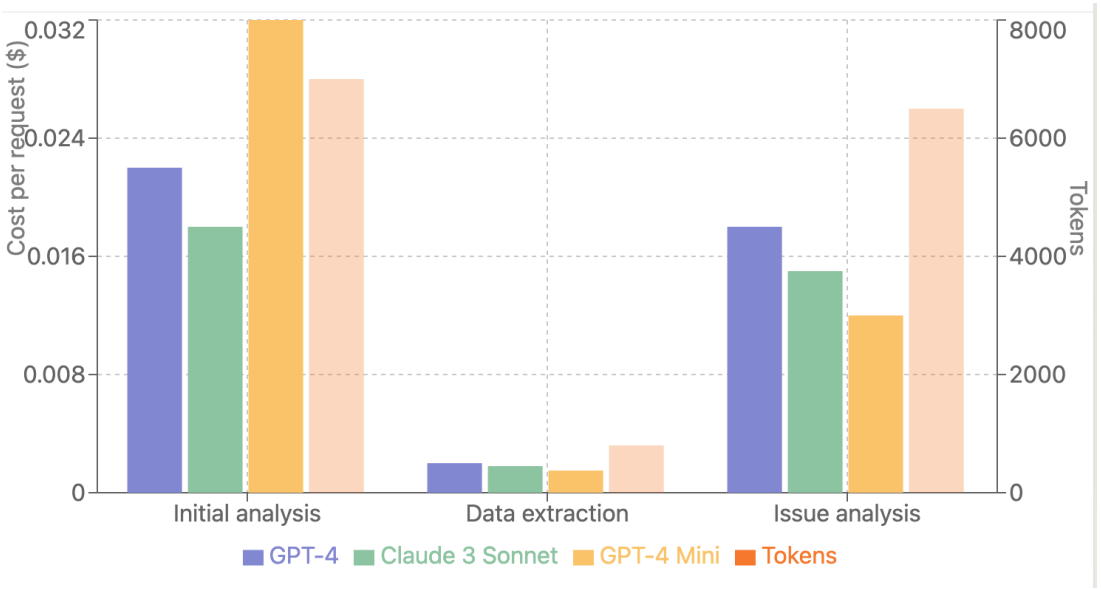


Рисунок 3.5 – Порівняльний аналіз витрат на LLM

Така оптимізація дозволила знизити середню вартість обробки одного звіту до приблизно $0.20 при збереженні високої якості аналізу, що робить сервіс економічно ефективним.

3.2.2 Інтеграційне тестування системи

Для забезпечення надійності системи було розроблено набір інтеграційних тестів, що перевіряють взаємодію всіх компонентів. Основна увага приділяється перевірці повного циклу обробки звіту - від завантаження файлу до отримання

результатів аналізу. Тести використовують pytest як основний фреймворк та охоплюють всі ключові аспекти роботи системи.

Перший блок тестів зосереджений на процесі завантаження файлів. Він включає перевірку генерації presigned URL для S3, валідацію допустимих типів файлів та контроль правильності створення структури папок у сховищі. Це забезпечує надійність процесу завантаження документів користувачами.

Другий важливий компонент тестування стосується самого процесу аналізу звітів. Тести перевіряють коректність структури результатів, відповідність рівнів критичності (severity levels) заданим критеріям, а також правильність форматування markdown та JSON даних. Це гарантує якість та консистентність результатів аналізу.

Особлива увага приділяється тестуванню фонової обробки даних. Перевіряється коректність асинхронного аналізу findings, процес збереження результатів у базу даних та надсилання email нотифікацій користувачам. Це забезпечує надійність довготривалих процесів обробки.

Для роботи з хмарними сервісами в тестах використовуються mock-об'єкти, зокрема moto для AWS. Це дозволяє тестувати взаємодію з S3 без реальних API-викликів, забезпечуючи стабільне виконання тестів у CI/CD pipeline. Такий підхід також значно прискорює процес тестування та робить його більш надійним.

Важливим аспектом тестування є перевірка обробки помилок та граничних випадків. Тести охоплюють сценарії з некоректними форматами файлів, відсутніми властивостями, помилками при аналізі та потенційними таймаутами. Всі ці тести запускаються автоматично при кожному push в репозиторій, що забезпечує постійний контроль якості системи.

3.2.3 Налаштування моніторингу продуктивності

Для забезпечення стабільної роботи системи було реалізовано комплексний моніторинг продуктивності, що включає відстеження ключових метрик через AWS CloudWatch та візуалізацію даних у режимі реального часу.

Основні метрики, що відстежуються, включають час відгуку API, використання ресурсів (CPU, пам'ять), кількість оброблених звітів та успішність їх обробки. Особлива увага приділяється моніторингу часу, необхідного для різних етапів аналізу - від початкового парсингу PDF до генерації фінального звіту.

Окремо налаштовано моніторинг взаємодії з зовнішніми сервісами, такими як API мовних моделей та SendGrid. Це дозволяє швидко виявляти та реагувати на можливі проблеми з інтеграціями. Встановлено автоматичні сповіщення при перевищенні порогових значень метрик або виникненні помилок.

Важливим компонентом системи став дашборд для візуалізації метрик (Рисунок 3.6), що дозволяє команді розробки та підтримки швидко оцінювати стан системи та виявляти потенційні проблеми. Графіки показують тренди у використанні ресурсів та продуктивності системи, що допомагає в плануванні масштабування та оптимізації.

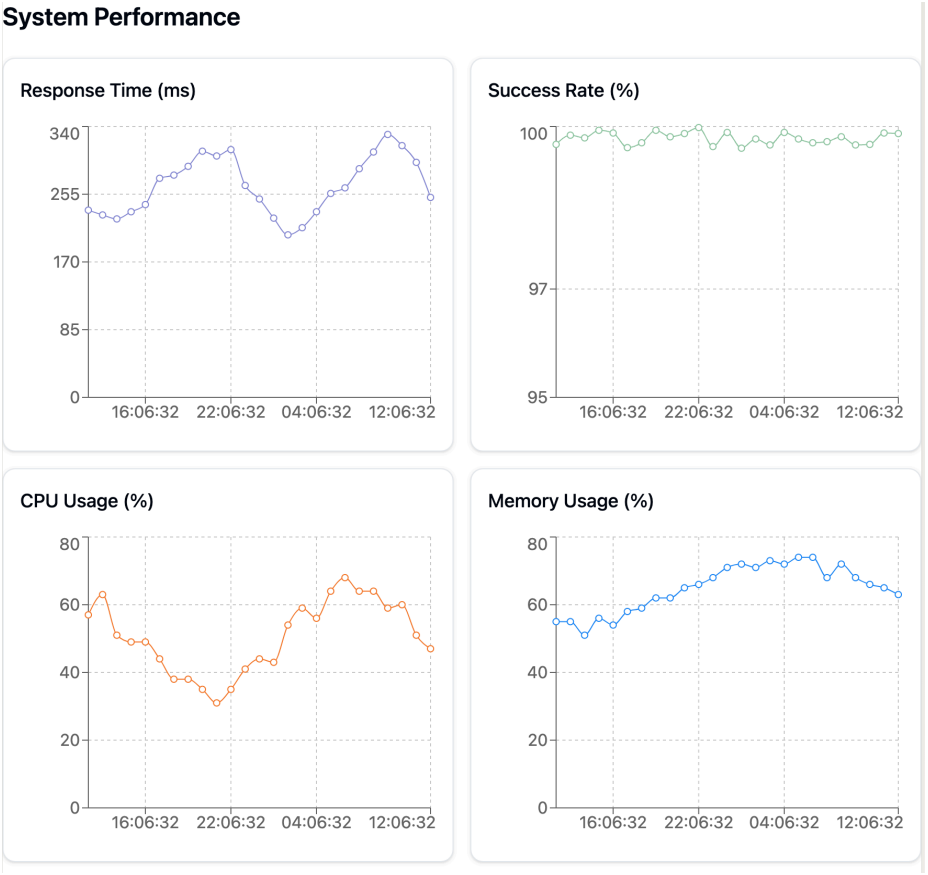


Рисунок 3.6 – Дашборд для візуалізації метрик

Також впроваджено логування деталізованої інформації про кожен запит та його обробку, що значно спрощує діагностику проблем та дозволяє проводити глибокий аналіз продуктивності системи при необхідності.

ВИСНОВКИ

1. Розроблена хмарна система аналізу документів на базі мультимодальних мовних моделей успішно вирішує задачу автоматизації процесу аналізу звітів технічного огляду нерухомості. Порівняльний аналіз різних мовних моделей (GPT-4, Claude 3.5, GPT-4 Mini) дозволив визначити оптимальні комбінації для різних етапів обробки, що забезпечує баланс між якістю аналізу та економічною ефективністю.

2. Розроблена архітектура системи включає ефективну обробку документів через S3 з використанням presigned URL для безпечного завантаження, Flask API для обробки та аналізу даних, та Next.js для інтерактивного користувацького інтерфейсу. Впровадження двоетапної обробки (загальний аналіз з подальшою деталізацією проблем) дозволяє користувачам отримувати перші результати протягом 17-40 секунд, що значно покращує користувацький досвід.

3. Реалізована повноцінна CI/CD система з використанням GitHub Actions та AWS Elastic Beanstalk забезпечує надійне розгортання та масштабування системи. Налаштоване автоматичне масштабування на основі мережевого навантаження (від 1 до 5 instances) та комплексний моніторинг через Sentry гарантують стабільну роботу сервісу під навантаженням. Інтеграційні тести охоплюють всі критичні компоненти системи, забезпечуючи надійність оновлень.

4. Розроблена система значно спрощує процес аналізу технічної документації нерухомості, надаючи структуровані висновки та рекомендації в зручному для сприйняття форматі. Середня вартість аналізу одного звіту становить близько $0.20, що робить сервіс економічно привабливим для користувачів та відкриває можливості для його широкого впровадження на ринку нерухомості.

ПЕРЕЛІК ПОСИЛАНЬ

1. Different PDF file types explained. Adobe: веб-сайт. URL: https://www.adobe.com/uk/acrobat/resources/document-files/pdf-types.html (дата звернення: 16.12.2024).
2. ISO 32000-2:2020 "Portable document format". International Organization for Standardization: веб-сайт. URL: https://www.iso.org/standard/75839.html (дата звернення: 16.12.2024).
3. Raschka S. Understanding Multimodal LLMs: A Deep Dive into Vision-Language Models. Sebastian Raschka's ML Magazine: веб-сайт. URL: https://magazine.sebastianraschka.com/p/understanding-multimodal-llms (дата звернення: 16.12.2024).
4. A Comprehensive Guide to Multimodal LLMs and How They Work. IONIO: веб-сайт. URL: https://www.ionio.ai/blog/a-comprehensive-guide-to-multimodal-llms-and-how-they-work (дата звернення: 16.12.2024).
5. Murel J., Kavlakoglu E. Text Summarization: Understanding the Basics. IBM Think: веб-сайт. URL: https://www.ibm.com/think/topics/text-summarization (дата звернення: 16.12.2024).
6. DevOps: Understanding CI/CD. GitHub Resources: веб-сайт. URL: https://github.com/resources/articles/devops/ci-cd (дата звернення: 16.12.2024).
7. Introduction to AWS Elastic Beanstalk. GeeksForGeeks: веб-сайт. URL: https://www.geeksforgeeks.org/introduction-to-aws-elastic-beanstalk/ (дата звернення: 16.12.2024).
8. Supabase User Management Examples. Restack: веб-сайт. URL: https://www.restack.io/docs/supabase-knowledge-supabase-user-management-examples (дата звернення: 17.12.2024).
9. Brown, T., et al. “Language Models are Few-Shot Learners.” Advances in Neural Information Processing Systems, 2020. URL: https://arxiv.org/abs/2005.14165 (дата звернення: 16.12.2024).

10. Understanding the Home Inspection Report. HomeGauge: веб-сайт. URL: https://www.homegauge.com/learning/what-is-a-home-inspection-report/ (дата звернення: 17.12.2024).
11. Konovalchuk N. OCR Algorithms: Types, Use Cases and Best Solutions. Itransition: веб-сайт. URL: https://www.itransition.com/computer-vision/ocr-algorithm (дата звернення: 27.12.2024).
12. LangSmith Documentation. LangChain: веб-сайт. URL: https://docs.smith.langchain.com/ (дата звернення: 17.12.2024)
13. How to Build an API With Python Flask. Moesif: веб-сайт. URL: https://www.moesif.com/blog/technical/api-development/Building-RESTful-API-with-Flask/ (дата звернення: 17.12.2024).
14. What is Next.js? Overview of the Powerful React Framework. Sanity: веб-сайт. URL: https://www.sanity.io/glossary/next-js (дата звернення: 17.12.2024).
15. Next.js Documentation. Vercel: веб-сайт. URL: https://nextjs.org/docs (дата звернення: 17.12.2024).
16. Flask Documentation. Pallets Projects: веб-сайт. URL: https://flask.palletsprojects.com/ (дата звернення: 17.12.2024).
17. Amazon S3 Documentation. AWS: веб-сайт. URL: https://docs.aws.amazon.com/s3/ (дата звернення: 17.12.2024).
18. Supabase Documentation. Supabase: веб-сайт. URL: https://supabase.com/docs (дата звернення: 17.12.2024).
19. API Documentation. OpenAI: веб-сайт. URL: https://platform.openai.com/docs/ (дата звернення: 17.12.2024).
20. Sentry Documentation. Sentry: веб-сайт. URL: https://docs.sentry.io/ (дата звернення: 17.12.2024).
21. GitHub Actions Documentation. GitHub: веб-сайт. URL: https://docs.github.com/en/actions (дата звернення: 17.12.2024).

22. A Complete Overview of LangChain Framework. Towards Data Science: веб-сайт. URL: https://towardsdatascience.com/langchain-framework-a-complete-overview-a295436152bf (дата звернення: 17.12.2024).
23. AWS Elastic Beanstalk Documentation. AWS: веб-сайт. URL: https://docs.aws.amazon.com/elastic-beanstalk/ (дата звернення: 17.12.2024).
24. Dosovitskiy A., et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. International Conference on Learning Representations (ICLR), 2021. URL: https://arxiv.org/abs/2010.11929 (дата звернення: 27.12.2024).
25. Radford A., Kim J.W., Hallacy C., et al. “Learning Transferable Visual Models From Natural Language Supervision”. Proceedings of the 38th International Conference on Machine Learning, 2021. URL: https://arxiv.org/abs/2103.00020 (дата звернення: 27.12.2024).
26. Brown T., Mann B., Ryder N., et al. “Language Models are Few-Shot Learners”. Advances in Neural Information Processing Systems (NeurIPS), 2020. URL: https://arxiv.org/abs/2005.14165 (дата звернення: 27.12.2024).
27. Zhang H., Goodfellow I., Metaxas D., Odena A. “Self-Attention Generative Adversarial Networks”. International Conference on Machine Learning (ICML), 2019. URL: https://arxiv.org/abs/1805.08318 (дата звернення: 27.12.2024).
28. He K., Zhang X., Ren S., Sun J. “Deep Residual Learning for Image Recognition”. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. URL: https://arxiv.org/abs/1512.03385 (дата звернення: 27.12.2024).
29. Vaswani A., Shazeer N., Parmar N., et al. “Attention is All You Need”. Advances in Neural Information Processing Systems (NeurIPS), 2017. URL: https://arxiv.org/abs/1706.03762 (дата звернення: 27.12.2024).

30. Sutskever I., Vinyals O., Le Q.V. “Sequence to Sequence Learning with Neural Networks”. Advances in Neural Information Processing Systems (NeurIPS), 2014. URL: https://arxiv.org/abs/1409.3215 (дата звернення: 27.12.2024).
31. Liu Y., Ott M., Goyal N., et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. arXiv preprint, 2019. URL: https://arxiv.org/abs/1907.11692 (дата звернення: 27.12.2024).
32. Ramesh A., Pavlov M., Goh G., et al. “Zero-Shot Text-to-Image Generation”. Advances in Neural Information Processing Systems (NeurIPS), 2021. URL: https://arxiv.org/abs/2102.12092 (дата звернення: 27.12.2024).
33. Kenton J.D., Toutanova L. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2019. URL: https://arxiv.org/abs/1810.04805 (дата звернення: 27.12.2024).

ПРЕЗЕНТАЦІЯ ДО МАГІСТЕРСЬКОЇ РОБОТИ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

ДИПЛОМНА РОБОТА
на ступінь вищої освіти магістр
із спеціальності 122 Комп'ютерні науки

ХМАРНА СИСТЕМА АНАЛІЗУ ДОКУМЕНТІВ НА ОСНОВІ МУЛЬТИМОДАЛЬНИХ МОВНИХ МОДЕЛЕЙ

Виконав:
студент 6 курсу групи КНДМ-61
Пархоменко Антон Володимирович

Керівник:
зк.т.н., доцент Гніденко Микола Петрович

Київ • 2024

Слайд 1

2

ЗАГАЛЬНА ХАРАКТЕРИСТИКА ДИПЛОМНОЇ РОБОТИ

Тема

Хмарна система аналізу документів на основі мультимодальних мовних моделей

Мета дослідження

Підвищення ефективності процесу аналізу технічної документації шляхом створення системи, що автоматично виявляє проблеми та надає рекомендації щодо їх усунення

Наукове завдання

Розробка хмарної системи для аналізу технічної документації з використанням мультимодальних мовних моделей

Об'єкт дослідження

Процес аналізу технічної документації нерухомості та формування звітів про виявлені проблеми

Предмет дослідження

Методи та засоби автоматизованого аналізу технічної документації з використанням мультимодальних мовних моделей

Слайд 2

3

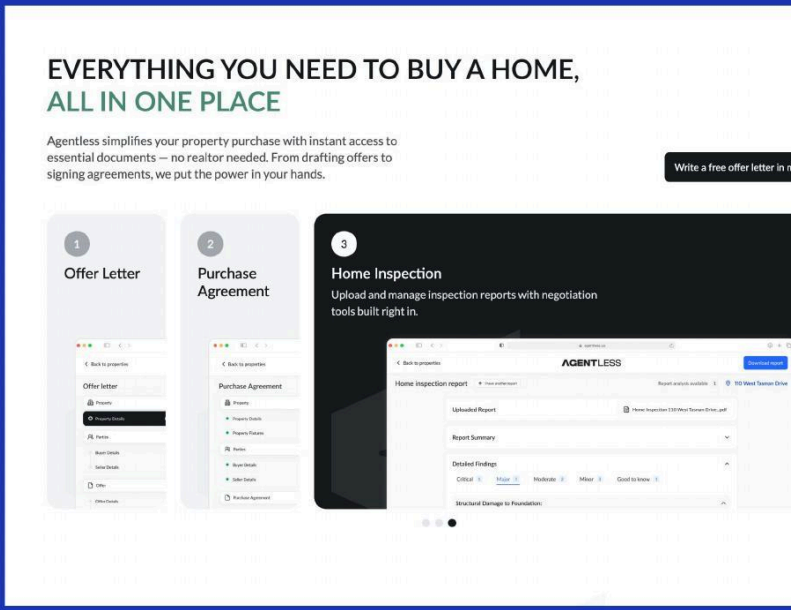
Проект Agentless

Призначення

Спрощення процесу купівлі нерухомості

Цільова аудиторія

покупці нерухомості в США



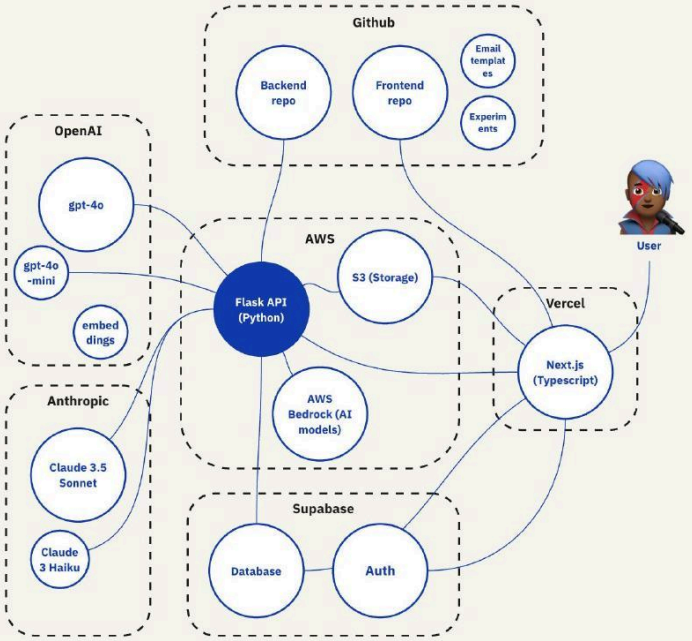
Слайд 3

4

Архітектура проекту

Frontend: Next.js (TypeScript)
Backend: Flask (Python)
Хмарні сервіси:

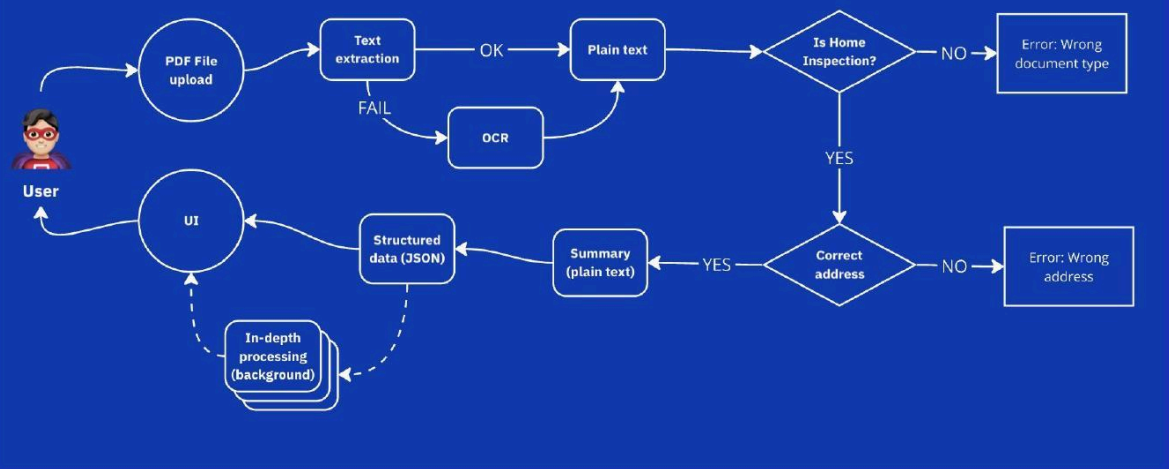
- Vercel (Frontend hosting)
- AWS Elastic Beanstalk (Backend)
- Amazon S3 (Storage)
- Supabase (Database)



Слайд 4

5

Процес аналізу документів



Слайд 5

6

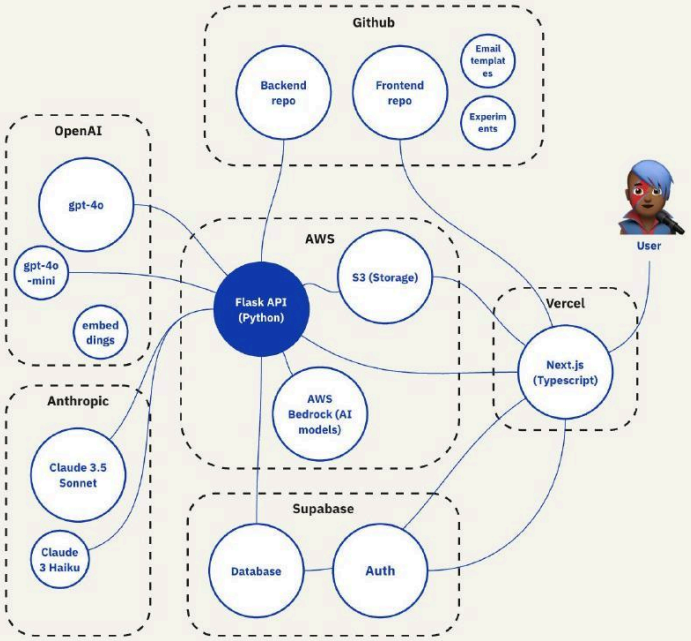
Мовні моделі та API

Використані API:

- OpenAI GPT-4
- Anthropic Claude
- LangChain Framework

Особливості:

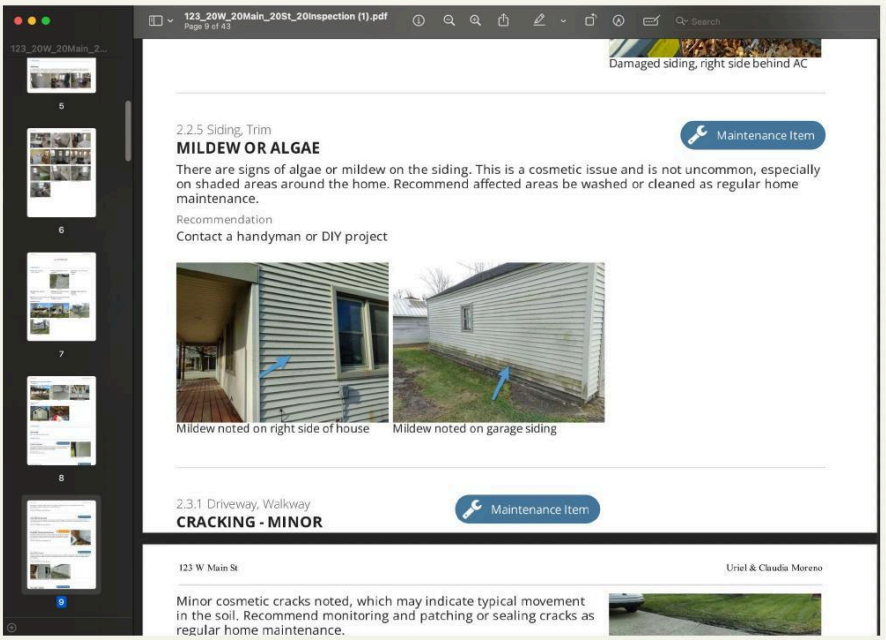
- менші моделі для простіших задач
- OCR за допмогою мультимодальних моделей
- багатопоточність та резервні хмарні провайдери (AWS Bedrock, Together)



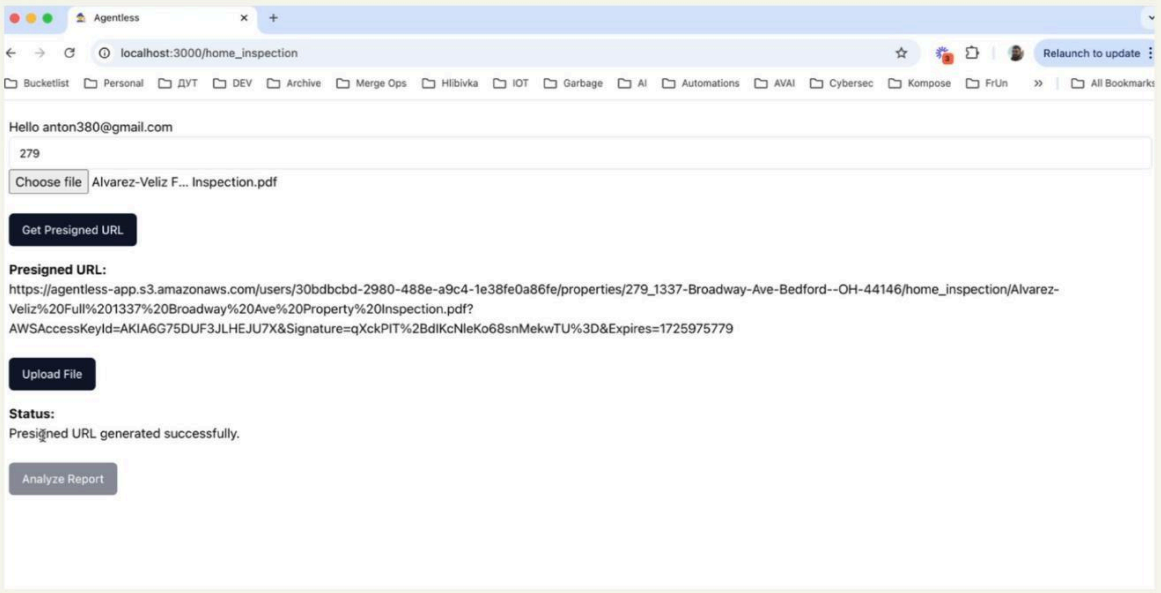
Слайд 6

Приклад документа

PDF, 40-100 сторінок, іноді – заблоковані або зібрані з зображень

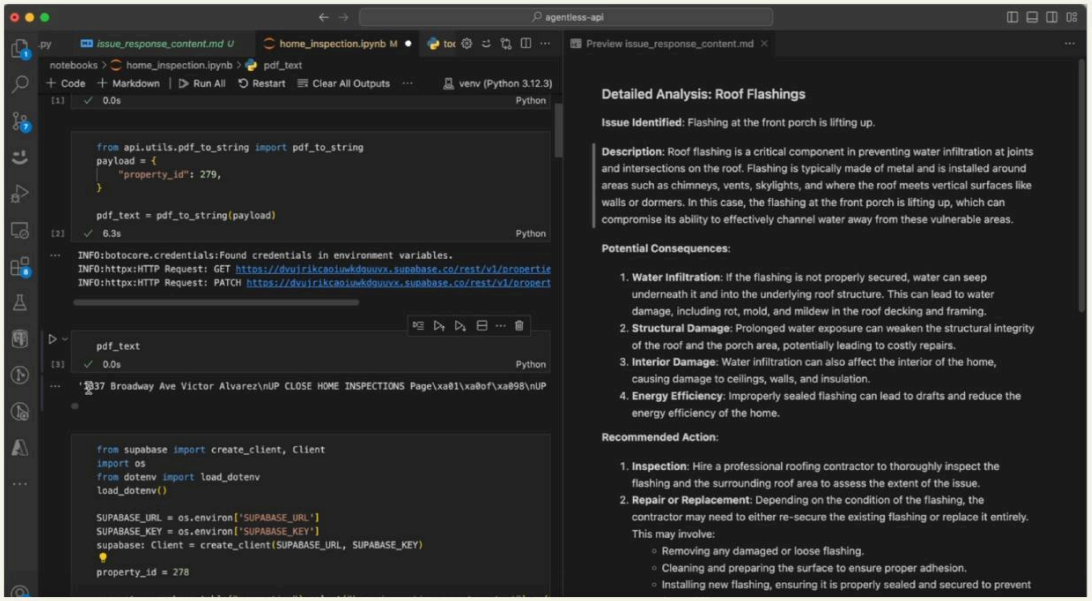


Слайд 7



Завантаження файлів на S3 за допомогою підписаних (Presigned) URL

Слайд 8



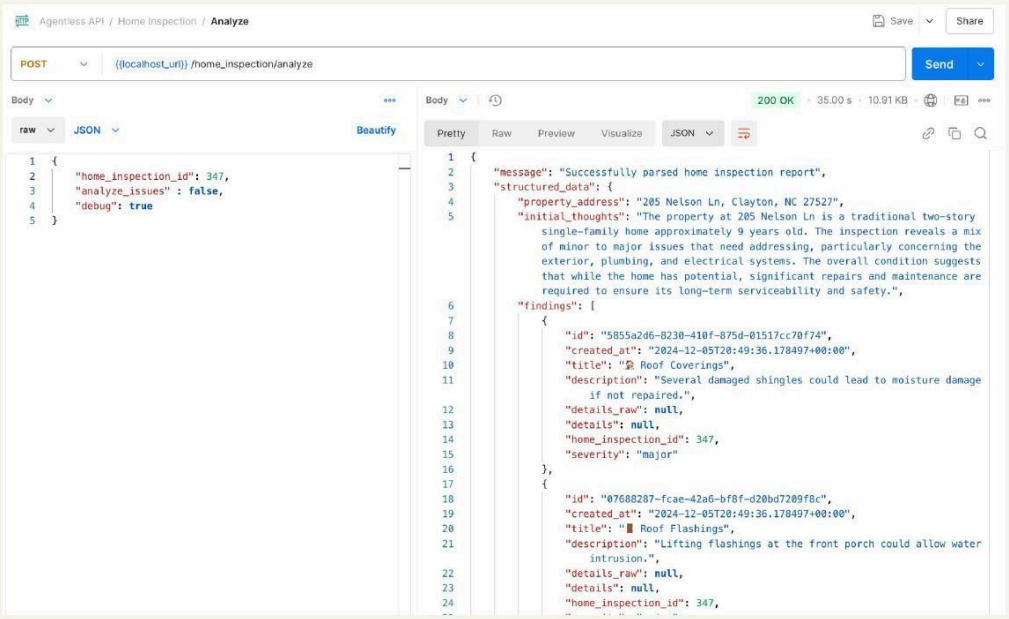
Тестування гіпотез: Jupyter Notebook

Слайд 9

```python
def analyze_home_inspection(data) -> tuple:
    """
    Analyze a home inspection report based on the provided home inspection ID.

    Expected payload schema:
    {
        "home_inspection_id": str,     # Required: The ID of the home inspection report to parse
        "analysis_model": str,         # Optional: Model to use for analysis
        "extraction_model": str,       # Optional: Model to use for data extraction
        "validate_address_model": str, # Optional: Model to use for address validation
        "debug": bool                  # Optional: Enable debug mode
    }

    Example response:
    {
        "message": "Successfully parsed home inspection report",
        "structured_data": {}
    }

    """
    start_time = time.time()

    if 'home_inspection_id' not in data:
        return {"error": "home_inspection_id is required"}, 400

    home_inspection_id = data.get('home_inspection_id')
    analysis_model = data.get('analysis_model', default_models.home_inspection_analysis)
    extraction_model = data.get('extraction_model', default_models.small)
    validate_address_model = data.get('validate_address_model', default_models.small)

    debug = 'true' if data.get('debug') is True else os.environ['LANGCHAIN_TRACING_V2']
    os.environ['LANGCHAIN_TRACING_V2'] = debug
```

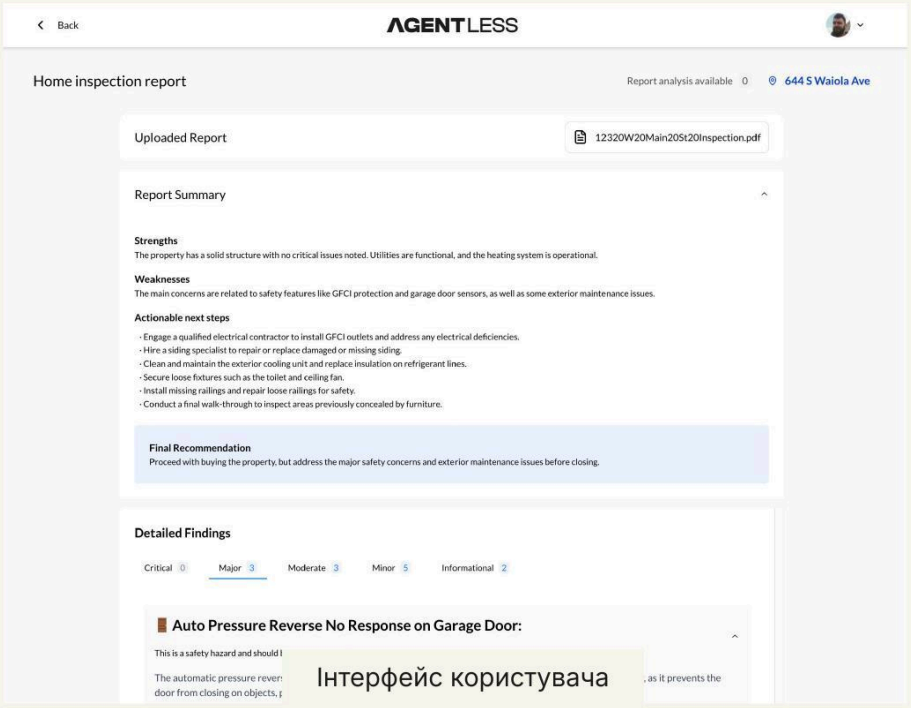
Код, перенесений у Flask API

Слайд 10



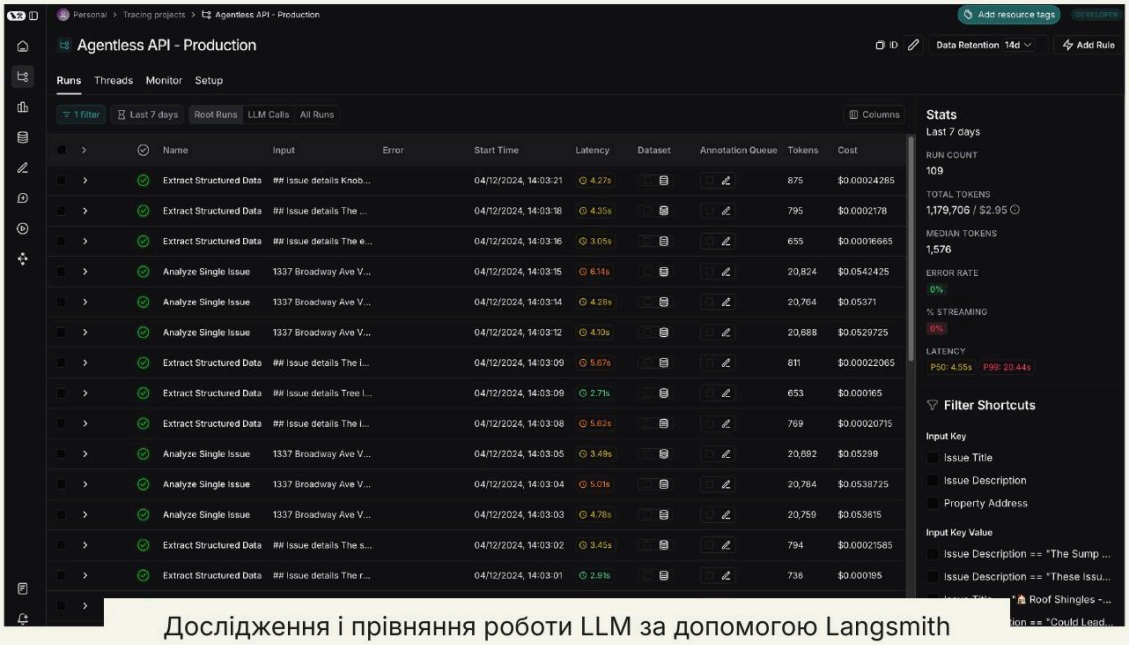
Тестування API за допомогою Postman

Слайд 11



Інтерфейс користувача

Слайд 12



Дослідження і прівняння роботи LLM за допомогою Langsmith

Слайд 13