МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕКОНОМІКИ І МЕНЕДЖМЕНТУ

Кафедра кібернетики та інформатики

КВАЛІФІКАЦІЙНА РОБОТА

освітній ступінь - «Бакалавр»

на тему: **ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПОШУКУ ТА СТРУКТУРУВАННЯ ПРЕЗЕНТАЦІЙ НА ОСНОВІ МЕТАДАНИХ**

Виконала: студентка спеціальності
F6(126) «Інформаційні системи та технології»

Мартинченко Софія Владиславівна

Керівник: ***Агаджанова Світлана Володимирівна***
к.т.н., доцент кафедри кібернетики та інформатики

Рецензент: ***Дегтярьова Неля Валентинівна***
к.п.н., доцент, завідувачка кафедри інформатики Сумського ДПУ ім.А.С.Макаренка

Суми – 2025

СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет	Економіки і менеджменту
Кафедра	Кібернетики та інформатики
Освітній ступінь	*«Бакалавр»*
Спеціальність	*(F6)126 «Інформаційні системи та технології»* *ОП «Інформаційні системи та технології»*

Затверджую:

Завідувач кафедри Світлана АГАДЖАНОВА

« 23 « червня 202 5 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ
Мартинченко Софії Владиславівні

1. Тема роботи: Інтелектуальна система пошуку та структурування презентацій на основі метаданих.

2. База практичного дослідження: ФОП Коваль м.Івано-Франківськ.

керівник роботи Агаджанова Світлана Володимирівна к.т.н., доцент

затверджена наказом по університету від: *«4» листопада 2024 № 3717/ос*

3. Строк подання студентом закінченого проекту (роботи): «23» червня 2025 року

4. Вихідні дані до проекту (роботи):

Результати аналітичних досліджень зарубіжних та вітчизняних вчених, інтернет-джерела, нормативні документи за чинним законодавством.

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

- *виконати постановку задачі;*
- *вибрати середовище для реалізації;*
- *розробити програмний продукт відповідно до поставлених задач;*
- *провести тестування системи.*

6. Дата видачі завдання: **« 04» листопада 2024 р.**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів проекту	Примітка
1	Визначення об'єкту, предмету дослідження, формулювання мети та задач кваліфікаційної роботи, складання плану	*січень 2025 р.*	*виконано*
2	Підбір та вивчення літературних джерел, технічної документації	*лютий 2025 р.*	*виконано*
3	Розробка системи	*лютий-квітень 2025 р.*	*виконано*
4	Оформлення пояснювальної записки	*до 30 травня 2025 р.*	*виконано*
5	Оформлення презентації	*до 1 червня 2025 р.*	*виконано*
6	Опрацювання зауважень керівника, перевірка на автентичність	*5 червня 2025 р.*	*виконано*
7	Підготовка доповіді та її попередній захист	*9 червня 2025 р.*	*виконано*
8	Перевірка з академічної доброчесності кваліфікаційної роботи	*9 червня 2025 р.*	*виконано*
9	Представлення кваліфікаційної роботи на кафедру, деканат	*13 червня 2025 р.*	*виконано*
10	Захист кваліфікаційної роботи	*27 червня 2025р.*	*виконано*

Студентка _____ ***Софія МАРТИНЧЕНКО***
(підпис)

Керівник роботи _____ ***Світлана АГАДЖАНОВА***
Світлана (підпис)

Перевірку на автентичність проведено _____ ***Надія БАРАННІК***
підпис)

Кваліфікаційна робота допущена до захисту _____ ***Світлана ЛУКАШ***
(підпис)

АНОТАЦІЯ

Мартинченко С.В. Інтелектуальна система пошуку та структурування презентацій на основі метаданих.

Кваліфікаційна робота зі спеціальності 126 «Інформаційні системи та технології» ОП Інформаційні системи та технології, СНАУ, Суми-2025р. – Рукопис.

Ця робота присвячена розробці інформаційної технології екстракції метаданих для аналізу презентацій. Основною метою є створення системи та моделей екстрактора метаданих, здатних аналізувати та виділяти метадані з текстових документів і зображень. Розроблена система забезпечує можливість користувачам завантажувати презентації для подальшої обробки, а також виконувати пошук серед вже оброблених файлів.

У дослідженні проведено огляд існуючих методів екстракції метаданих та визначені ключові виклики, пов’язані з аналізом тексту та зображень у презентаціях. Запропонована концептуальна модель системи екстракції, що включає опис функціональних можливостей, основних процесів та їх взаємозв’язків.

Для реалізації системи були використані сучасні технології програмування та баз даних, що забезпечують швидке та надійне функціонування. Розроблено модулі для завантаження, обробки презентацій та пошуку серед оброблених файлів. Проведено тестування системи на основі реальних даних.

Результати тестування підтвердили коректність роботи системи та її відповідність поставленим вимогам. Отримані результати свідчать про доцільність впровадження розробленої системи для автоматизації процесу аналізу презентацій та екстракції метаданих, що сприятиме покращенню ефективності обробки інформації та забезпечить зручний доступ до необхідних даних.

Ключові слова: інформаційна технологія, машинне навчання, моделі машинного навчання, PowerPoint, Node.js, Python, TypeScript, CockroachDB.

SUMMARY

Martynchenko S.V. Intelligent System for Searching and Structuring Presentations Based on Metadata.

Qualification thesis in specialty 126 “Information Systems and Technologies,” Educational Program “Information Systems and Technologies,” SNAU, Sumy-2024. – Manuscript.

This study is dedicated to the development of an intelligent system for searching and structuring presentations based on metadata. The main goal is to create a system and models for metadata extraction capable of analyzing and identifying metadata from text documents and images. The developed system enables users to upload presentations for further processing and perform searches among already processed files.

The research includes a review of existing metadata extraction methods and identifies key challenges related to text and image analysis in presentations. A conceptual model of the extraction system is proposed, detailing its functional capabilities, key processes, and their interconnections.

Modern programming technologies and database solutions were utilized to ensure fast and reliable system performance. Modules for uploading, processing presentations, and searching among processed files were developed. The system was tested using real data.

The testing results confirmed the system’s correct functionality and compliance with the specified requirements. The obtained results indicate the feasibility of implementing the developed system for automating the process of presentation analysis and metadata extraction, contributing to improved information processing efficiency and providing convenient access to essential data.

Keywords: information technology, machine learning, machine learning models, PowerPoint, Node.js, Python, TypeScript, CockroachDB.

ЗМІСТ

ВСТУП

Дані відіграють вирішальну роль у сучасному житті, вони є основою багатьох галузей і забезпечують функціонування повсякденних сервісів у нашому житті. Дані стають все більш важливими, оскільки вони дозволяють організаціям і приватним особам приймати кращі рішення, підвищувати ефективність і створювати нові продукти та послуги.

Проблема неструктурованих даних полягає у тому, що їх не можна аналізувати за допомогою стандартних методів обробки даних, таких як SQL. Ось кілька прикладів того, як дані використовуються в сучасному житті:

- компанії використовують дані, щоб отримати уявлення про поведінку клієнтів, оптимізувати свою діяльність та приймати кращі рішення;
- дані використовуються для покращення результатів лікування пацієнтів шляхом аналізу медичних записів, відстеження спалахів захворювань та надання персоналізованих планів лікування;
- дані використовуються для персоналізації навчального процесу, відстеження прогресу студентів та покращення загальної системи освіти;
- дані використовуються для виявлення шахрайства, управління ризиками та надання персоналізованих фінансових послуг;
- дані використовуються для покращення державних послуг, відстеження злочинності та прийняття кращих політичних рішень;
- дані використовуються для таргетування реклами та надання користувачам рекомендацій щодо контенту, а також для аналізу тенденцій у соціальних мережах та настроїв людей.

Неструктуровані дані можуть бути в вигляді тексту, презентацій, голосу, відео, зображень і т.д. Інформація з них може бути видобута за допомогою аналізу зображень, розпізнавання голосу та інших технологій. Однак, це може бути досить складно та дорого, особливо для великих об'ємів даних. Проблема обробки та

зберігання даних відноситься до викликів, пов'язаних з управлінням великими обсягами даних та їх використанням для різних цілей, таких як аналіз, прийняття рішень та звітність. Деякі з конкретних викликів у цій галузі включають в себе наступні:

- зі збільшенням обсягу даних, що генеруються з різних джерел, таких як соціальні мережі, пристрої Інтернету речей та онлайн-транзакції, організаціям складно зберігати та обробляти великі обсяги даних;
- дані генеруються все більш швидкими темпами, а це означає, що організації повинні мати можливість обробляти і аналізувати їх в режимі реального часу, щоб не відставати від них;
- дані бувають різних форм, наприклад, структуровані, напів структуровані та неструктуровані, що може ускладнювати їх обробку та аналіз;
- дані часто містять помилки, пропущені значення або невідповідності, що може ускладнити довіру до результатів аналізу даних;
- оскільки організації зберігають все більше конфіденційних даних, вони повинні забезпечити їх захист від несанкціонованого доступу, порушень та інших загроз безпеці;
- оскільки організації зберігають і обробляють все більше даних, їм необхідно забезпечити відповідність різним нормативним актом і законом щодо даних.

Для вирішення цих завдань організації використовують різні технології та підходи, такі як фреймворки для обробки великих даних, сховища даних, озера даних, хмарні сховища, рішення для управління даними та безпеки. Дані відіграють вирішальну роль у сучасному житті, вони є основою багатьох галузей і забезпечують функціонування повсякденних додатків та сервісів.

Дивлячись на присутність великого обсягу даних у майже всіх сферах у нашому житті. Постає необхідність у створенні такої моделі яка допоможе структурувати та аналізувати дані у презентаціях швидко та якісно.

Апробація результатів кваліфікаційної роботи. Результати кваліфікаційної роботи було представлено на конференціях:

1. Мартинченко С.В. Інтелектуальні методи обробки та категоризації мультимедійного контенту презентацій: матеріали Всеукраїнської наукової конференції студентів і аспірантів, присвяченої Міжнародному дню студента – (18-22 листопада 2024 р.). – Суми, 2024. – С.423

2. Мартинченко С.В. Виклики при обробці та управлінні неструктурованими даними: матеріали науково-практичної конференції викладачів, аспірантів та студентів Сумського НАУ – (14-18 квітня 2025 р.). – Суми, 2025. – С.306

Структура і обсяг роботи. Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел з 34 найменувань, додатків. Основний текст викладений на 63 сторінках комп'ютерного тексту, робота містить 1 таблицю, 20 рисунків, 3 додатки.

РОЗДІЛ 1

АНАЛІЗ ПРОБЛЕМИ СТРУКТУРУВАННЯ ТА ПОШУКУ ПРЕЗЕНТАЦІЙ

1.1 Огляд досліджень та публікацій

З появою нових каналів комунікації та технологій, таких як соціальні мережі, мобільні обчислення та динамічні презентації, дані, які створюються, більше не відповідають традиційним форматам чи структурам і не можуть бути оброблені за допомогою реляційних моделей. Вони надходять у вигляді тексту, XML, електронних листів, зображень, блогів, відео тощо, що призводить до появи нових типів даних.

Ці безформні дані є або напівструктурованими, або неструктурованими і ускладнюють пошук та аналіз. У цьому документі було зроблено огляд цих неструктурованих даних, які складають основу прогнозного аналізу.

У ньому окреслено джерела та елементи неструктурованих даних, а також те, як організація отримує вигоду від збору, аналізу та використання неструктурованих даних.

Обсяг неструктурованих даних, що генеруються такими компаніями, як нафтовидобувні, авіакомпанії, соціальні мережі, маркетингові та інші, сягає тисяч терабайт, і ці неструктуровані дані є настільки цінними, що компанії зараз розробляють методи вилучення з них сенсу. Обсяг неструктурованих даних, що генеруються у навчальних закладах, постійно зростає і стає важливою складовою їх інформаційного середовища. Неструктуровані дані - це дані, які не мають чіткої організації або формату, і не можуть бути легко впорядковані за допомогою традиційних методів обробки даних.

У навчальних закладах неструктуровані дані можуть бути представлені у різних формах, таких як текстові документи, електронні книги, лекційні записи,

наукові статті, електронні листи, веб-сторінки, блоги, фотографії, відео- та аудіозаписи, соціальні медіа-пости та багато іншого.

Обсяг неструктурованих даних в освітніх установах продовжує зростати з появою сучасних технологій та інструментів комунікації. Викладачі та студенти створюють значну кількість матеріалів, які можуть бути корисними в майбутньому, але потребують обробки та аналізу. Одним із прикладів неструктурованих даних в освітніх установах є текстові документи, включаючи студентські завдання, наукові роботи, звіти та нотатки. Аналіз таких документів може допомогти ідентифікувати ключові теми, тенденції, основні ідеї та розвиток знань у певних галузях.

Крім того, соціальні медіа, які широко використовуються освітніми установами, генерують величезну кількість неструктурованих даних. Через великий обсяг і різноманітність таких даних традиційні інструменти обробки стають недостатніми. Обсяги даних зростають експоненціально через машинно-генеровані дані та підвищену активність людей у соціальних мережах. На відміну від структурованих даних, неструктуровані дані складно впорядковувати, шукати, візуалізувати чи аналізувати. Тому потрібні нові інструменти та процеси для отримання цінної інформації, обміну даними та створення доданої вартості. Організації витрачають значний час і ресурси на створення неструктурованих даних, часто втрачаючи ефективність, коли ці дані не потрапляють до потрібних людей у потрібний час. Таблиця 1.1 ілюструє організації з високими темпами генерації даних [1].

Таблиця 1.1 - Організації з їх темпами генерування даних

Компанія	Згенеровані дані
Digital Universe Study	1 227 Ексабайт у 2010 році; Прогнозоване створення 1,8 Зеттабайт даних щорічно в 2011 році; 2,7 Петабайт у 2013 році.
YouTube	48 годин нових відео завантажуються щохвилини
Facebook	34 722 вподобання щохвилини; 100 терабайт завантажується щодня; 30+ Петабайт (зберігання, доступ, аналіз); 30 мільярдів одиниць контенту

	щомісяця;
Twitter	Приблизно 175 мільйонів твітів щодня; Понад 465 мільйонів рахунків

Джерело: запропоноване автором

Структуровані дані — це дані, що відповідають визначеному формату та довжині, завдяки чому їх легко зберігати та аналізувати. Дані організовані в зрозумілій структурі, що дозволяє виконувати запити для отримання необхідної інформації. Типовим прикладом структурованих даних є реляційні бази даних (наприклад, SQL або Access), які містять організовані числа, дати та текстові рядки (рядки/текст). Така структура дозволяє виконувати пошук за допомогою простих алгоритмів, адаптованих до типів даних. Традиційна аналітика переважно зосереджена на структурованих даних, часто ігноруючи інші типи.

Напівструктуровані дані — це нерегулярні дані, які можуть бути неповними, мати швидкозмінну або непередбачувану структуру, а також не мати чіткої схеми. Такі дані не організовані у вигляді таблиць, як у реляційних базах даних, чи графів, як у об’єктно-орієнтованих базах. За словами Ханіга, Шірле та Трабольда (2010) [2], модель напівструктурованих даних дозволяє інтегрувати інформацію з кількох джерел із пов’язаними, але різними атрибутами, такими як електронна пошта, XML-файли чи документи. На противагу цьому, неструктуровані дані не мають чіткої структури. Це часто включає растрові зображення, текст, електронну пошту та інші типи даних, які не є частиною бази даних. Наприклад, хоча електронна пошта організована в базоподібних форматах (наприклад, Lotus Notes, Microsoft Exchange), текст повідомлення є у неструктурованому вигляді. До неструктурованих даних також належать документи, такі як файли PowerPoint із описом стратегії компанії, електронні таблиці з потенційними клієнтами, листування між колегами та взаємодії клієнтів у соціальних мережах.

Цикл обробки даних — це послідовність кроків або операцій, які використовуються для обробки даних, перетворюючи сирі дані у придатний для використання формат. Обробка даних може включати декілька методів і етапів:

- Введення (Input) - сирі дані збираються і вводяться у цикл для обробки. Це перший і вкрай важливий крок.
- Обробка (Processing) - введені дані обробляються за допомогою вибраних методів, що дає бажані результати для подальшого використання.
- Виведення (Output) - результат, який є обробленими даними, надає цінну інформацію і більше не вважається сирими даними.

Кроки включають:

- Збір даних – це перший етап, який забезпечує інформацію для подальшої роботи. Сам процес збору даних є складним завданням, але він надзвичайно важливий, оскільки від його якості залежить кінцевий результат. Якість вхідних даних прямо впливає на якість вихідних. Дані можуть бути зібрані різними методами з первинних та вторинних джерел, включаючи статистику перепису населення, показники ВВП, фінансові показники компаній тощо. Під час збору потрібно врахувати, які джерела є найбільш релевантними для конкретного завдання.
- Підготовка/просіювання – хоча деякі вважають цей етап частиною обробки, насправді він не включає безпосередньої обробки. Підготовка полягає у сортуванні та фільтрації отриманих даних, щоб видалити зайву чи непотрібну інформацію. Це дозволяє значно підвищити ефективність подальшої обробки, оскільки зменшується обсяг даних і підвищується їх якість.
- Введення – етап, на якому підготовлені дані подаються для обробки. Якщо цей етап виконано неправильно, це негативно позначиться на результатах. Програмне забезпечення працює за принципом "сміття на вході – сміття на виході", тому важливо забезпечити точність та коректність вхідних даних.

- Обробка – основний етап, під час якого дані проходять через електронну, механічну чи автоматизовану обробку. Цей процес перетворює сирі дані на корисну інформацію, яка може бути використана користувачем. Тривалість обробки залежить від складності та обсягу даних. На цьому етапі результати стають зрозумілими та доступними для використання, а підготовчий етап допомагає зробити процес швидшим та ефективнішим.
- Вихід/результат – фінальний етап, де оброблені дані надаються у вигляді готової інформації. Отримані результати можуть бути використані для аналізу, прийняття рішень, створення презентацій або збереження для подальшої обробки. Якщо результат не використовується повторно, цикл обробки не завершується повністю і залишається одноразовою дією [3].

Нижче наведено перелік різних типів файлів, які є результатом обробки даних і можуть бути використані для подальшого аналізу або візуалізації.

- Простий текстовий файл – це базова форма оброблених даних, яка є найпростішою для розуміння. Більшість таких файлів зручно читати та аналізувати, оскільки вони мають легкий для сприйняття формат. Їх подальша обробка, як правило, мінімальна або зовсім не потрібна. Такі файли зазвичай експортуються у форматах, придатних для текстових редакторів, наприклад, блокнот або WordPad.
- Таблиці / електронні таблиці – це оптимальний формат для зберігання та обробки числових даних. Розташування інформації у вигляді рядків і стовпчиків дозволяє зручно працювати з числами та виконувати різноманітні операції. Наприклад, користувач може легко сортувати дані за зростанням або спаданням, а також застосовувати різні фільтри. Цей формат полегшує аналіз і дозволяє виконувати математичні обчислення, що робить його ідеальним для подальшого використання.
- Діаграми та графіки – надання оброблених даних у вигляді графіків або діаграм є надзвичайно зручним, і це вже стало стандартною функцією

багатьох програм. Цей формат особливо корисний для візуалізації числових даних, які демонструють певні тенденції, наприклад, зростання чи спад. У сучасному програмному забезпеченні доступна велика кількість стандартних типів діаграм і графіків, які відповідають різним потребам. Однак, якщо стандартні інструменти не підходять, користувач має можливість створювати власні, налаштовувані графіки для конкретних завдань.

- Карти / векторні та графічні файли – цей формат є важливим при роботі з просторовими даними. Зокрема, карти дуже корисні для містобудівників та інших фахівців, які займаються аналізом географічної інформації. Векторні та графічні файли забезпечують можливість точного відображення даних у вигляді зображень. Проте ці файли не завжди є зручними для безпосереднього читання, оскільки їх зазвичай використовують у спеціалізованому програмному забезпеченні для роботи з графікою.
- Інші формати / сирі файли – це специфічні програмні формати, які призначені для роботи в спеціалізованих середовищах. Вони не завжди є готовим продуктом і можуть вимагати подальшої обробки. У таких випадках обробка даних може включати кілька етапів, і деякі з них доведеться повторювати. Ці формати використовуються тоді, коли потрібно досягти високого рівня деталізації або адаптації до вузькопрофільних завдань [4] .

Для підготовки до публічних виступів сьогодні часто використовується програма для створення презентацій Microsoft PowerPoint. Серед найпопулярніших інструментів можна виділити: PowerPoint (MS), Actor (Macromedia), Animation Works Interactive (Gold Disc). Крім того, презентації можна створювати за допомогою Macromedia Flash або Google Slides. У цьому тексті зосередимо увагу на створенні презентацій у PowerPoint, програмі, розробленій спеціально для операційної системи Windows. Програма PowerPoint надає користувачам наступні можливості:

- створення презентацій із простими та компактними відеоматеріалами, що слугують для супроводу виступів і доповідей; д
- одавання мовного та музичного супроводу для покращення якості презентацій; розробка шаблонів, які включають елементи анімації для створення динамічних слайдів;
- створення великої кількості графіків, таблиць і діаграм для зручної візуалізації даних;
- можливість підготовки та передачі презентацій на інший комп’ютер, їх поширення в межах локальної мережі або через Інтернет [5].

Метадані – це стандартизований набір інформації, який містить основні відомості про файл, такі як ім'я автора, роздільна здатність, колірний простір, авторські права та ключові слова. Наприклад, більшість сучасних цифрових фотоапаратів додає до файлів із зображеннями базову інформацію, серед якої можна знайти висоту та ширину зображення, формат файлу або час, коли було зроблено знімок. Метадані часто використовуються для пришвидшення робочого процесу та організації файлів. Їх можна створювати як вручну, так і автоматично. Ручне створення метаданих зазвичай є більш точним, оскільки користувач має можливість вводити інформацію, яку вважає релевантною, або таку, що допоможе краще описати файл. Автоматизоване створення метаданих, своєю чергою, є простішим і обмежується базовими параметрами.

Воно зазвичай фіксує такі дані, як розмір файлу, розширення, дату створення файлу та інформацію про автора. Обидва підходи – ручний і автоматизований – мають свої переваги та використовуються залежно від цілей систематизації файлів.

Метадані створюються щоразу, коли документ, файл або будь-який інший інформаційний ресурс зазнає змін, включаючи його видалення. Точність і якість метаданих мають велике значення, оскільки вони можуть допомогти продовжити термін актуальності існуючих даних, відкриваючи нові способи їхнього використання. Метадані виконують функцію організації об'єкта даних,

застосовуючи терміни, що прямо пов'язані з цим конкретним об'єктом. Вони також дозволяють визначати об'єкти, які відрізняються один від одного, і водночас об'єднувати їх у пари із схожими об'єктами, що сприяє більш ефективному використанню інформаційних ресурсів. Як було зазначено, пошукові системи та браузери визначають, який саме вебконтент відображати користувачам, аналізуючи метадані, що містяться в HTML-документі. Мова метаданих створена таким чином, щоб бути зрозумілою як комп’ютерним системам, так і людям. Такий рівень стандартизації забезпечує кращу сумісність і дозволяє легко інтегрувати дані між різними додатками та інформаційними системами. Метадані активно використовуються компаніями з різних галузей, таких як цифрові публікації, інжиніринг, фінанси, охорона здоров'я та виробництво. Ці компанії застосовують метадані для аналізу, збору інформації та визначення шляхів вдосконалення своїх продуктів або модернізації процесів. Наприклад, постачальники потокового контенту використовують автоматизовані системи управління метаданими, щоб зберігати інформацію про права інтелектуальної власності у різних додатках. Це дозволяє одночасно захищати авторські права та надавати музику чи відео автентифікованим користувачам. Розвиток технологій штучного інтелекту значно полегшує управління метаданими, автоматизуючи багато процесів, які раніше виконувалися вручну. Завдяки цьому стає простішим каталогізувати та тегувати інформаційні активи, що сприяє більш ефективному управлінню великими обсягами даних.

Метадані поділяються на три основні типи: описові, адміністративні та структурні. Описові метадані служать для пошуку, ідентифікації та відбору ресурсів. Вони зазвичай містять такі елементи, як назва, ім'я автора та основна тематика. Адміністративні метадані забезпечують управління ресурсами й можуть включати дані про технічні характеристики, умови збереження, права доступу та використання. Структурні метадані переважно застосовуються для машинної обробки даних. Вони описують взаємозв’язки між різними частинами ресурсу,

наприклад, розділами книги або елементами мультимедійного контенту [7].

Дана проблема була також розглянута у тезах під назвою “Інтелектуальні методи обробки та категоризації мультимедійного контенту презентацій” та у роботі “Виклики при обробці та управлінні неструктурованими даними” [8],[9].

1.2 Актуальність вирішення проблеми структурування презентацій

Загальновідомо, що використання презентаційних матеріалів підвищує ефективність комунікації, сприяє легшому сприйняттю та розумінню інформації, а також підтримує взаємодію між людьми в різних сферах діяльності.

Щодня ми стикаємося з величезним обсягом даних — як на різних онлайн-платформах, так і під час написання та редагування статей, підготовки презентацій чи виконання завдань. Великі компанії та корпорації щодня працюють із даними, часто з великими масивами інформації. Саме тому питання організації та аналізу даних стає неминучим. Одним із головних викликів сучасності є великий обсяг інформації. Надмірна кількість даних може призвести до перенасичення слайдів. Перевантаження слайдів текстом і графічними елементами відволікає увагу аудиторії та ускладнює зосередження на головному повідомленні.

Екстракція метаданих із презентацій має великий потенціал для покращення управління та аналізу цих документів. Завдяки екстракції метаданих презентації можна швидко класифікувати, індексувати та організовувати, що полегшує їх пошук, навігацію та використання. Крім того, аналіз метаданих може виявляти тенденції, статистику та залежності у використанні презентацій у різних контекстах.

У сучасному світі презентації широко використовуються у різних галузях, таких як бізнес, освіта, наука, медіа тощо. Існує кілька факторів, які сприяли збільшенню кількості презентацій:

- Презентації є ефективним засобом комунікації та обміну інформацією. У сучасному швидкому ритмі життя вони дозволяють коротко та зрозуміло представляти ідеї, проєкти, результати досліджень та іншу інформацію.
- Вони є потужним інструментом комунікації, який допомагає залучати та утримувати увагу аудиторії. Завдяки графіці, аудіо та відеоелементам презентації можуть бути більш привабливими та ефективними, ніж інші форми спілкування.
- У навчальних закладах презентації допомагають викладачам та студентам структурувати й візуалізувати матеріал, сприяють активному залученню студентів до навчального процесу.
- Цифровий формат презентацій дозволяє легко зберігати, редагувати та поширювати їх за допомогою електронних пристроїв, електронної пошти чи хмарних сервісів, роблячи їх доступними для ширшого кола людей.

Зростання використання презентацій пов’язане з розвитком технологій та збільшенням доступу до Інтернету. Сучасні інструменти для створення презентацій, такі як Microsoft PowerPoint, Google Slides, Keynote тощо, стають дедалі доступнішими та простішими у використанні. Крім того, розвиток комунікаційних мереж, соціальних медіа та онлайн-платформ сприяє швидкому поширенню та обміну презентаціями.

Збільшення кількості презентацій у різних сферах відображає сучасні тенденції комунікації та обміну інформацією. Презентації стали невід’ємною частиною бізнесу, освіти, науки та інших галузей. Вони забезпечують зручний та ефективний спосіб передачі інформації, сприяють залученню аудиторії та сприйняттю ідей.

Однак разом зі зростанням кількості презентацій виникають і певні труднощі. Складність створення ефективних презентацій, перенасичення інформацією, ускладнений пошук потрібних файлів і технічні проблеми можуть стати перешкодами у використанні презентаційних матеріалів.

Програма, яка здатна витягувати з презентації зображення, текст, шрифти та кількість слайдів для полегшення пошуку файлів, значно зменшить час на пошук і підготовку потрібної презентації. Така розробка сприятиме оптимізації роботи з презентаціями та полегшить їх використання.

1.3 Існуючі підходи до пошуку та структурування

Існують різноманітні рішення, які застосовують алгоритми машинного навчання для автоматичного аналізу та категоризації презентацій за такими критеріями, як тема, ключові слова, автор, дата тощо. Деякі системи використовують ручне маркування та сортування презентацій за допомогою міток або тегів. Інші підходи комбінують автоматичний аналіз із ручним для досягнення максимальної точності та ефективності при пошуку й організації презентацій.

Також існують спеціалізовані рішення, орієнтовані на конкретні галузі. Наприклад, у медичній сфері використовуються системи, здатні аналізувати текст та зображення у презентаціях, щоб автоматично визначати діагнози, методи лікування та інші важливі дані. Окремі інструменти дозволяють взаємодіяти з презентаціями в режимі реального часу, наприклад, швидко перемикатися між слайдами або залишати коментарі під час виступу. Доступні також хмарні рішення, які забезпечують зручне зберігання та обмін презентаціями, що підвищує їхню мобільність і доступність з будь-якого пристрою та місця. Крім того, деякі інструменти аналізують звук та відео, дозволяючи автоматично розпізнавати ключові моменти презентації, наприклад, початок нового розділу або зміну слайда. Це значно полегшує навігацію та акцентує увагу на важливих частинах.

Деякі системи інтерактивного навчання пропонують персоналізовані рекомендації, аналізуючи інтереси користувача й підбираючи релевантні матеріали. Інструменти візуального пошуку дозволяють швидко знаходити презентації за їхнім зовнішнім виглядом — кольорами, шрифтами, структурою

тощо. Більшість сучасних рішень можуть інтегруватися з іншими програмами та платформами для забезпечення сумісності й обміну даними. Розподілена обробка даних використовується для швидкої роботи з великими обсягами презентацій, розподіляючи завдання між серверами чи комп'ютерами. Усі ці підходи спрямовані на максимальну зручність і ефективність у роботі з презентаціями.

Існують сучасні технології, призначені для аналізу презентацій та документів, серед яких можна виділити Computer Vision, Object Recognition, Full-text Search та Image Analysis.

Для аналізу вмісту презентацій і фотографій активно застосовуються технології комп'ютерного зору (Computer Vision), такі як:

- Розпізнавання об'єктів - використовується для автоматичної ідентифікації різноманітних об'єктів, які присутні на фотографіях і в презентаціях.
- Розпізнавання тексту - технологія, що дозволяє видобувати текст із зображень або слайдів, який надалі може бути підданий аналізу за допомогою обробки природної мови (NLP).
- Розпізнавання облич - використовується для автоматичного визначення облич на фотографіях та у відео.
- Розпізнавання жестів - дозволяє ідентифікувати жести, що можуть слугувати для керування системою чи навігації по презентаціях.
- Аналіз шаблонів - ця технологія визначає взаємозв'язки між елементами та виявляє характерні шаблони у структурі презентацій або фотографій [10].

Окремо варто виділити технологію Object Recognition (розпізнавання об’єктів). Це процес автоматичного визначення та ідентифікації об’єктів на зображеннях чи відео. Вона ґрунтується на аналізі характеристик об’єктів, таких як форма, розміри, текстура, колір тощо. Одним із найбільш поширених методів у цій галузі є використання нейронних мереж, які після навчання на спеціалізованих наборах даних здатні розпізнавати навіть ті об'єкти, з якими не стикалися раніше [11]. Однією з ранніх архітектур для визначення об’єктів на зображеннях стала R-

CNN (Region Convolutional Neural Network). Її принцип роботи полягає у виконанні послідовності кроків, вона проілюстрована на зображенні 1:

1. Формування набору гіпотез для розташування об’єктів.
2. Витяг ознак із прогнозованих регіонів за допомогою згорткових нейронних мереж та кодування цих ознак у вектор.
3. Класифікація об’єкта всередині кожної гіпотези на основі отриманих векторів.
4. Уточнення координат гіпотез для підвищення точності.
5. Повторення процесу для всіх гіпотез із першого кроку, поки не будуть опрацьовані всі можливі варіанти.

Таким чином, використання технологій аналізу зображень і презентацій дозволяє ефективно опрацьовувати візуальний контент, автоматизуючи процеси і забезпечуючи високий рівень точності.

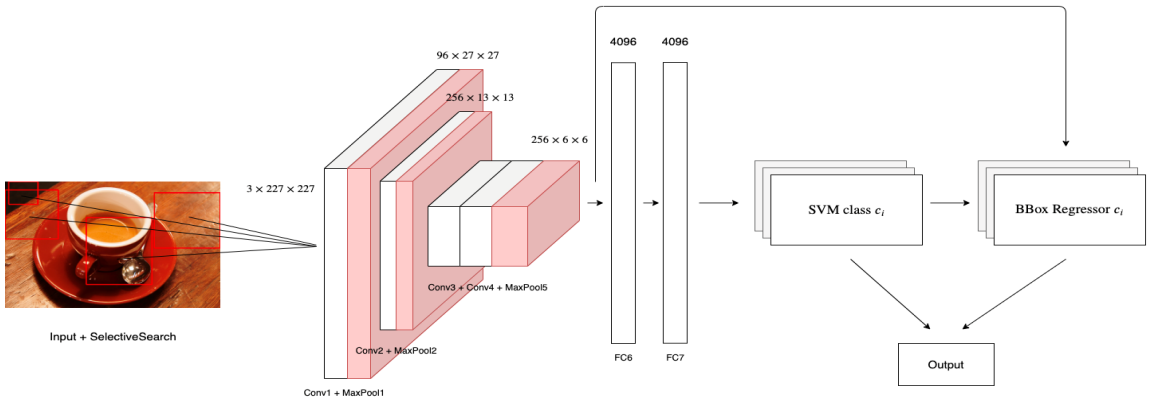


Рисунок 1.1 – Архітектура розпізнавання R-CNN

Джерело: запропоноване автором

Починаючи з заданого зображення, воно спочатку розбивається на невеликі гіпотези різних розмірів. У дослідженні використовується метод Selective Search, який створює набір гіпотез (незалежно від класу об’єкта) через сегментацію для визначення меж об'єктів. Це базується на інтенсивності пікселів, зміні кольорів, контрасті та текстурі. Хоча автори обрали Selective Search, можна застосовувати й

інші подібні алгоритми. У результаті виділяється приблизно 2000 регіонів, які можуть частково перекриватися. Для точнішої подальшої обробки кожну гіпотезу розширюють на 16 пікселів у всіх напрямках, щоб додати контекст.

Результат:

- Вхід: початкове зображення.
- Вихід: набір гіпотез із різними розмірами та співвідношеннями сторін.

Кожна гіпотеза обробляється окремо за допомогою згорткової нейронної мережі. Як базову архітектуру обрано AlexNet без softmax-шару. Основна мета мережі — перетворення вхідного зображення у векторне представлення, яке отримується з останнього повнозв'язаного шару FC7. Це векторне представлення має розмір 4096. Оскільки вхідні дані для AlexNet мають розмір 3×227×227, а гіпотези можуть мати будь-яке співвідношення сторін, вхідні дані стискаються або розтягуються до необхідного розміру.

Результат:

- Вхід: кожна гіпотеза з попереднього етапу.
- Вихід: векторне представлення для кожної гіпотези.

Отриманий вектор використовується для визначення об'єкта в регіоні. Для цього застосовується метод класифікації на основі опорних векторів (SVM) із Hinge loss. Необхідно мати Nc+1N_{c}+1 моделей, навчених за принципом OvR (один проти всіх), для багатокласової класифікації. Вихід моделі — Nc+1N_{c}+1-вимірний вектор, який показує впевненість щодо приналежності об'єкта до конкретного класу (нульовий клас відповідає фону).

Результат:

- Вхід: вектор для кожної гіпотези (з шару FC7).
- Вихід: матриця 2000×Nc2000×N_{c}, яка визначає класи об'єктів у гіпотезах.

Гіпотези можуть містити неточні координати (наприклад, об'єкт може бути частково «обрізаним»). Для покращення цього етапу використовується лінійна

регресія, яка коригує координати регіонів, що містять об'єкти. За словами авторів, це підвищує точність на 3–4%. Для кожного класу застосовується окремий регресор, а для покращення результатів використовується карта ознак з останнього шару Max Pooling (у AlexNet — це Max Pool 5, розмір 256×6×6). Це пояснюється тим, що карта ознак краще зберігає інформацію про місцезнаходження об’єкта, ніж вектор із FC7.

Результат:

- Вхід: карта ознак для кожної гіпотези з об’єктом (крім фону).
- Вихід: скориговані координати обмежувальної рамки [12].

Також існує технологія, відома як аналіз інформації з зображень (Image Analysis), яка використовується для вилучення різноманітної інформації з візуальних даних, включаючи текст, об'єкти, людей, транспортні засоби та інші елементи. До її основних можливостей належать розпізнавання текстів, визначення об'єктів, виявлення дефектів, отримання метаданих тощо. У рамках Image Analysis застосовуються різні методи та алгоритми, такі як обробка зображень, ідентифікація образів, розпізнавання облич, аналіз розмірів і форм об'єктів, вимірювання відстаней і кутів, дослідження кольорів і текстур, а також інші техніки.

Ця технологія знаходить широке застосування в багатьох галузях, зокрема медицині, науці, виробництві, рекламі, безпеці та інших сферах. Наприклад, у медицині Image Analysis активно використовується для аналізу зображень мозку, легенів, судин та інших органів, що допомагає діагностувати і виявляти різноманітні захворювання. У сфері виробництва ця технологія сприяє контролю якості продукції та виявленню дефектів на виробничих лініях. Використання Image Analysis значно полегшує і прискорює обробку великих обсягів даних, які містять зображення, забезпечуючи при цьому високу точність і надійність отриманих результатів [13].

Ще однією важливою технологією є Full-text search — інструмент, що дозволяє здійснювати пошук у текстових даних, включаючи об'ємні колекції документів. Full-text search (FTS) — це методика пошуку текстової інформації в базах даних, що надає змогу знаходити потрібні записи за ключовими словами або фразами у тексті документів, записів чи інших даних. Цей підхід дозволяє виявляти всі записи в базі, які містять задані ключові слова, незалежно від їхнього місця розташування в тексті. Використання FTS особливо корисне для роботи з великими масивами текстової інформації, як-от статті, документи, листування, записи чатів тощо. Для реалізації Full-text search у базах даних використовуються спеціальні індекси, які містять слова з текстів документів і вказівки на записи, де вони зустрічаються. У процесі пошуку система перевіряє ці індекси, щоб знайти записи, які відповідають запиту. Завдяки впровадженню технології FTS можна значно підвищити ефективність і точність пошуку текстової інформації, що робить її незамінним інструментом для багатьох застосувань, таких як пошукові системи, електронні каталоги, соціальні мережі, системи управління контентом тощо [14].

У сучасному світі технології, такі як Computer Vision, Object Recognition, Full-text search і Image Analysis, займають особливе місце завдяки своїй універсальності та широкому застосуванню. Вони є незамінними в таких сферах, як інформаційні системи, медицина, автоматизація процесів і забезпечення безпеки.

- Computer Vision та Object Recognition забезпечують здатність комп'ютерів розуміти та аналізувати візуальні дані. Вони дозволяють розпізнавати об'єкти, людей, класифікувати їх за групами, а також ідентифікувати емоції. Ці технології широко використовуються в системах безпеки для виявлення порушень і загроз на відеозаписах, а також у робототехніці та автономному транспорті.
- Full-text search є ефективним інструментом для організації та пошуку текстової інформації. Вона дозволяє швидко знаходити потрібні документи,

вебсторінки чи інші ресурси на основі ключових слів або фраз. Ця технологія активно використовується в базах даних, вебпошукових системах та інформаційних порталах, забезпечуючи високу швидкість і точність результатів.

- Image Analysis дає змогу комп'ютерам аналізувати та розуміти зображення, виконувати розпізнавання образів, класифікацію, вимірювання та інші операції. Ця технологія має велике значення в медичних дослідженнях, допомагаючи лікарям виявляти ознаки хвороб та патологій на медичних зображеннях, а також в автоматизованих системах якості та контролю, де вимагається точне розпізнавання деталей та вимірювання.

Запропонована система, яка може одночасно аналізувати текст та фотографії в презентаціях, має значну перевагу порівняно з іншими системами. Це дає можливість більш точного та повного розуміння контексту інформації, яка міститься в презентаціях. Завдяки поєднанню Full-text search з Image Analysis, система може ефективно виявляти та аналізувати як текстові дані, так і візуальну інформацію, що відкриває нові можливості для вдосконалення пошукових систем, автоматизації процесів та розуміння контексту в інформаційних системах. Переваги які роблять її більш ефективною та корисною в порівнянні з іншими системами.

По-перше, можливість одночасно аналізувати текст та фотографії дозволяє системі отримати більш повну та всебічну інформацію. Це дозволяє досягти більш точних та релевантних результатів в процесі пошуку, класифікації та інтерпретації даних. Наприклад, система може розпізнавати текст на зображеннях і використовувати його як додаткову інформацію для розуміння контексту та забезпечення кращих результатів пошуку.

По-друге, враховуючи швидкий темп зростання обсягу інформації, що вміщується в презентаціях, нова система здатна ефективно обробляти великі обсяги даних і забезпечувати швидкий доступ до результатів аналізу. Завдяки поєднанню

розуміння тексту та обробки зображень, система може прискорити процес пошуку та аналізу інформації, що є важливим в умовах швидко змінюючогося середовища.

Крім того, нова система дозволяє автоматизувати багато рутинних завдань, пов'язаних з аналізом презентацій. Вона може автоматично розпізнавати та класифікувати зображення, ідентифікувати ключові елементи тексту та витягувати важливу інформацію. Це зменшує навантаження на користувача та підвищує продуктивність роботи з презентаціями.

Отже, нова створена система, здатна одночасно аналізувати текст та фотографії в презентаціях, має значну перевагу, оскільки вона забезпечує більш повне розуміння та аналіз інформації, прискорює обробку великого обсягу даних та автоматизує рутинні завдання. Це робить її більш ефективною та корисною для користувачів, які працюють з презентаціями та потребують швидкого та точного пошуку та аналізу даних.

1.4 Постановка задачі

Задача програми екстракції метаданих для структурування та пошуку презентацій полягає у розробці програмного забезпечення, здатного автоматично аналізувати та категоризувати презентації за різними критеріями, такими як тема, ключові слова, автор, дата, тип файлу тощо. Головна мета такої системи — спростити пошук і доступ до необхідних презентацій, а також забезпечити зручний інтерфейс для користувачів.

Завдання системи включає наступні етапи: автоматичне розпізнавання та екстракція метаданих із презентацій, створення бази даних для їх збереження, розробка алгоритмів для категоризації та класифікації презентацій, а також створення інтерфейсу для їх пошуку та доступу.

Впровадження такої системи забезпечить ефективну та швидшу роботу з презентаціями, що є важливим для багатьох галузей, таких як освіта, наука, бізнес тощо.

Для реалізації цієї задачі необхідно виконати наступне:

- Розробити алгоритми для екстракції метаданих, включаючи розпізнавання тексту та зображень із презентацій.
- Обрати необхідні технології для розробки додатку.
- Створити базу даних для зберігання метаданих та презентацій.
- Розробити інтерфейс користувача, який дозволить швидко знаходити потрібну презентацію за ключовими словами чи іншими критеріями.
- Провести тестування програми перед запуском.
- Підготувати набір файлів для перевірки функціоналу додатку.
- Підтримувати систему, забезпечуючи її безперебійну роботу та оновлення даних.

РОЗДІЛ 2

ПРОЕКТУВАННЯ СИСТЕМИ ПОШУКУ ТА СТРУКТУРУВАННЯ ПРЕЗЕНТАЦІЙ

2.1 Модель інтелектуальної системи

Модель технології структурування та пошуку презентацій - це система, що дозволяє автоматично визначати та виділяти з презентацій метадані, такі як заголовки, автори, дата створення тощо, та дозволяє здійснювати пошук по цим метаданим. Ця модель базується на використанні попередньо навчених моделей машинного навчання для екстракції метаданих з тексту та зображень. Наприклад, для аналізу тексту можуть використовуватись моделі, навчені на задачах нерозміченого текстового аналізу, таких як виявлення іменованих сутностей, тематичне моделювання, сентимент-аналіз тощо.

Для аналізу зображень можуть використовуватись моделі, навчені на великих наборах даних, таких як ImageNet, щоб визначати об'єкти, сцени, та інші характеристики зображень. Після екстракції метаданих, система може структурувати ці дані та зберігати їх у відповідному форматі, такому як база даних. На рисунку 2 зображена структура нейронної мережі [15].

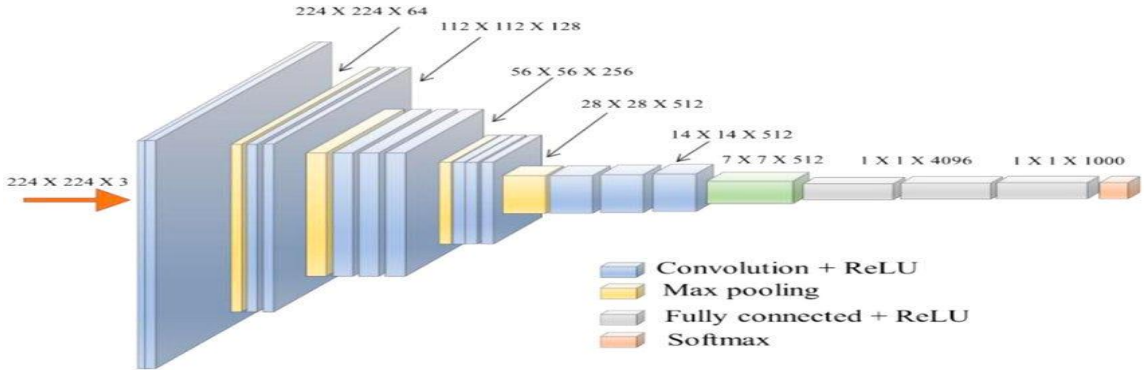


Рисунок 2.1 – Структура нейронної мережі

Джерело: запропоноване автором

Далі користувач може здійснювати пошук по цим метаданим, щоб швидко знайти необхідну презентацію. Модель технології структурування та пошуку презентацій може бути корисною для різних цілей, таких як пошук презентацій з певними характеристиками, або для створення збірок презентацій з певної тематики. Модель технології структурування та пошуку презентацій може бути використана для різних цілей, наприклад:

- за допомогою цієї моделі можна швидко і легко знайти необхідну презентацію в колекції, відкрити її та редагувати;
- модель може бути використана в освітніх закладах для структурування та пошуку презентацій, що дозволяє студентам та викладачам знайти потрібну інформацію для підготовки до занять або досліджень;
- модель може бути використана в бізнес-середовищі для структурування та пошуку презентацій з метою підготовки звітів, презентацій для клієнтів, партнерів тощо;
- модель може бути використана дослідниками для структурування та пошуку наукових презентацій з метою аналізу, порівняння, оцінки та подальшої роботи з ними.

2.2 Аналіз текстових даних

Аналіз зображень в презентаціях - це процес виявлення та розпізнавання об'єктів, розуміння їх змісту та використання отриманої інформації для досягнення певної мети. Цей процес може включати в себе розпізнавання облич, тексту, емоцій, об'єктів, що зображені на зображеннях, а також аналіз контексту, в якому вони знаходяться. Для аналізу зображень в презентаціях використовуються різноманітні методи та технології, такі як комп'ютерне зорове розпізнавання, нейронні мережі, машинне навчання, обробка природних мов, обробка сигналів та багато іншого.

Цей процес є корисним для різноманітних завдань, таких як автоматичне створення підписів до зображень, класифікація зображень за темами, виявлення подій та об'єктів на фотографіях та інше. Використання аналізу зображень в презентаціях дозволяє забезпечити більш ефективне та точне управління вмістом, поліпшити користувацький досвід та забезпечити більш ефективне використання часу.

Існує багато різних типів класифікації зображень, залежно від використаної методології та відповідної задачі класифікації. Нижче перераховані декілька основних типів класифікації зображень:

- бінарна класифікація: зображення класифікуються на два класи - "позитивний" та "негативний". Наприклад, класифікація зображень як "вміст для дорослих" та "не вміст для дорослих";
- класифікація з багатьма класами: зображення класифікуються на більше, ніж два класи. Наприклад, класифікація зображень на основі виду тварин або різних видів спорядження для різних видів спорту;
- класифікація з кількома мітками: зображення можуть бути класифіковані як належні до кількох класів. Наприклад, зображення вуличних сцен можуть бути класифіковані як "автомобільна дорога", "пішохідний перехід" та "велосипедист";
- класифікація зі змінною кількістю класів: кількість класів залежить від даних введення. Наприклад, класифікація зображень залежно від кольору одягу, де кількість класів залежить від кількості різних кольорів в зображенні;
- класифікація на основі семантичного змісту: зображення класифікуються на основі його семантичного змісту. Наприклад, класифікація зображень на основі сцени, яка зображена, такої як "парк", "місто", "офіс" [16].

ImageNet - це база даних з більш ніж 14 мільйонами зображень, розподілених на понад 20 000 категорій. Вона створена для розробки та навчання алгоритмів комп'ютерного зору, зокрема для класифікації зображень. ImageNet була створена

у 2009 році, і вона є однією з найбільших та найважливіших баз даних зображень у галузі комп'ютерного зору. Вона використовується для навчання та оцінки алгоритмів класифікації зображень, зокрема для глибокого навчання, такого як нейронні мережі. ImageNet містить зображення різних об'єктів, таких як тварини, люди, рослини, транспортні засоби, архітектура тощо. Кожне зображення в базі даних супроводжується міткою, що позначає категорію, до якої воно належить. Для роботи з ImageNet зазвичай використовують нейронні мережі, зокрема сверточні нейронні мережі (Convolutional Neural Networks, CNN), які можуть автоматично вивчати ознаки зображень та робити класифікацію. Під час навчання нейронної мережі використовуються зображення з ImageNet, що дозволяє мережі навчитися розпізнавати широкий спектр об'єктів та категорій [17].

Крім того, у ImageNet є панель експертів, яка перевіряє та оцінює якість та точність класифікації. Це дозволяє визначити переможців конкурсів та встановлювати нові рекорди точності в класифікації зображень. В цілому, ImageNet став важливим ресурсом для розвитку комп'ютерного зору та глибокого навчання. Багато з відкритих архітектур нейронних мереж, які здатні до класифікації зображень, були навчені на ImageNet. Крім того, ImageNet став основою для розробки інших даних для навчання та оцінювання нейронних мереж, які займаються розпізнаванням зображень. Для виконання аналізу зображень у документах та презентаціях використовують бібліотеку від Node.js під назвою node-efficient net.

Вона дозволяє використовувати архітектуру Efficient Net у Node.js проектах. Efficient Net - це тип згорткової нейронної мережі (CNN), яка була попередньо навчена на великому наборі даних і здатна досягати найсучаснішої точності в широкому спектрі завдань класифікації зображень. Використовуючи node-efficient net, можна легко імпортувати попередньо навчену модель EfficientNet і використовувати її для таких завдань, як класифікація зображень, виявлення об'єктів і семантична сегментація. Однією з переваг використання цієї бібліотеки є

те, що вона дозволяє використовувати потужну архітектуру Efficient Net без необхідності навчати модель з нуля, що може заощадити багато часу та обчислювальних ресурсів. Вона також надає простий API для завантаження та використання моделей і підтримує як синхронне, так і асинхронне використання.

Efficient Net - це серія глибоких нейронних мереж, розроблена компанією Google для класифікації зображень, яка має значну кількість параметрів та досягає високої точності при розпізнаванні об'єктів. Архітектура Efficient Net базується на концепції масштабування глибини, ширини та роздільної здатності (depth, width, resolution scaling) мережі, що дозволяє досягти максимальної точності при мінімальному використанні ресурсів. За допомогою коефіцієнта масштабування, можна відносно просто збільшувати кількість параметрів та розмір моделі, щоб підібрати оптимальну архітектуру для конкретного завдання. Наприклад, для класифікації зображень, які мають низьку роздільну здатність, можна використовувати меншу модель з меншою кількістю параметрів, тоді як для більш високої роздільної здатності можна використовувати більш складну модель з більшою кількістю параметрів. Efficient Net використовує стандартні прийоми в нейромережах, такі як свертки, групові нормалізації, функції активації та інші, а також додаткові методи, такі як об'єктивна функція для вибору коефіцієнта масштабування та механізм для автоматичного пошуку оптимальної архітектури мережі.

Efficient Net також використовує механізм розмірності для підвищення точності. Архітектура включає в себе компоненти, що забезпечують більш ефективне використання ресурсів обчислювальної мережі, такі як механізм збільшення (compound scaling) і фільтр змінних (swish activation function).

Механізм збільшення (compound scaling) дозволяє ефективно змінювати розмірність моделі, збільшуючи глибину, ширину та роздільну здатність зображення одночасно. Це допомагає забезпечити оптимальну точність при різних розмірах зображень та різній обчислювальній потужності.

Фільтр змінних (swish activation function) дозволяє збільшити точність класифікації шляхом використання не лінійної функції активації, що включає в себе параметризований параметр. Це дозволяє моделі вільно налаштовувати нелінійність залежно від типу даних, що аналізуються.

За допомогою цих технологій EfficientNet може досягати кращої точності за менший час та з меншими витратами на обчислення, що робить його привабливим вибором для завдань з обробки зображень.

Узагальнюючи, Efficient Net - це серія глибоких нейронних мереж, які застосовують концепцію масштабування глибини, ширини та роздільної здатності, щоб досягти максимальної точності при мінімальному використанні ресурсів [18].

2.3 Обробка візуальних компонентів презентацій

Один з алгоритмів, що використовують для аналізу та кластеризації тексту - це технологія keyword-extractor. Він зазвичай використовується для автоматичного виділення ключових слів або фраз з текстових документів. Вона базується на алгоритмах обробки природної мови (Natural Language Processing, NLP) та статистичних методах, що дозволяють аналізувати текст і визначати його найсуттєвіші елементи.

Алгоритм keyword-extractor спочатку обирає деяку кількість кластерів, що потрібно створити. Потім він обирає деякі початкові центри кластерів і призначає кожен зразок до ближчого до нього кластеру. Після цього він обчислює нові центри кластерів, перераховуючи середнє значення кожної групи і знову призначає кожен зразок до ближчого до нього кластеру. Цей процес повторюється, поки центри кластерів не стабілізуються і кластери не стануть більш однорідними.

У випадку аналізу презентацій, зразками можуть бути окремі слайди або фрагменти тексту з них, а кластери можуть відображати подібні теми або стилі. Після кластеризації можна використовувати отримані результати для створення

зручного інтерфейсу пошуку презентацій, що містять конкретну тему, стиль або ключові слова [19].

Основні етапи роботи технології keyword-extractor включають наступне:

- токенізація - спочатку текст розбивається на окремі слова або токени. Цей процес називається токенізацією і допомагає розділити текст на лексеми, з якими можна працювати;
- фільтрація стоп-слів - важливим кроком є фільтрація стоп-слів, які не несуть суттєвої інформації про контекст або тему тексту. Стоп-слова включають займенники, сполучники, частотні слова тощо. Вони виключаються з розгляду для більш точного виділення ключових термінів;
- обчислення ваги слів - кожному токену або слову надається вага, що вказує на його важливість у тексті. Ця вага може бути обчислена за допомогою різних алгоритмів, таких як TF-IDF (Term Frequency-Inverse Document Frequency), який враховує як частоту вживання слова в тексті, так і розповсюдження слова по інших текстах;
- ранжування ключових слів - на основі обчислених ваг слів проводиться їх ранжування в порядку спадання. Це дозволяє отримати найважливіші терміни або фрази, які найкраще відображають зміст і тему тексту.

Технологія "keyword-extractor" може мати різні варіації та алгоритми, які використовуються для виділення ключових слів. Вона може бути застосована у різних галузях, включаючи пошукову оптимізацію, аналіз тексту, автоматичну індексацію документів, автоматизацію обробки інформації та багато іншого.

РОЗДІЛ 3

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ

3.1 Структура програмного забезпечення

Для побудови системи пошуку та структурування презентацій було обрано клієнт серверну архітектуру. Клієнт-серверна архітектура - це розподілена архітектурна модель, в якій задачі і ролі між різними компонентами системи розподіляються між клієнтами та серверами. У такій архітектурі клієнти і сервери взаємодіють між собою через мережу, де клієнти запитують сервери для отримання ресурсів, послуг або інформації [20].

Основні компоненти клієнт-серверної архітектури включають:

1. клієнт - це користувацька частина системи, яка взаємодіє з сервером для отримання потрібної інформації або послуг. Клієнт може бути десктопною програмою, мобільним додатком, веб-браузером або будь-яким іншим інтерфейсом.
2. сервер - це централізований компонент, який надає послуги, ресурси або інформацію клієнтам. Сервер може бути фізичним комп'ютером або набором комп'ютерів, що виконують певні функції, оброблюють запити та надають відповіді клієнтам.

Переваги клієнт-серверної архітектури:

1. клієнт-серверна архітектура дозволяє легко масштабувати систему шляхом додавання або заміни серверів, що забезпечують певні послуги. Це дозволяє розподіляти навантаження і підвищує продуктивність системи.
2. сервер виконує централізовану обробку та керування ресурсами, що дозволяє забезпечити єдність даних та логіки обробки. Це полегшує управління системою і забезпечує консистентність даних.

3. клієнт-серверна архітектура дозволяє реалізувати різні рівні захисту даних та доступу до ресурсів. Сервер може контролювати доступ клієнтів і забезпечувати шифрування даних для забезпечення безпеки.
4. клієнт-серверна архітектура дозволяє реалізувати клієнтські додатки на різних платформах (десктоп, мобільні, веб), що робить систему більш доступною для різних користувачів.

Система має наступний вигляд (див рис 3)..

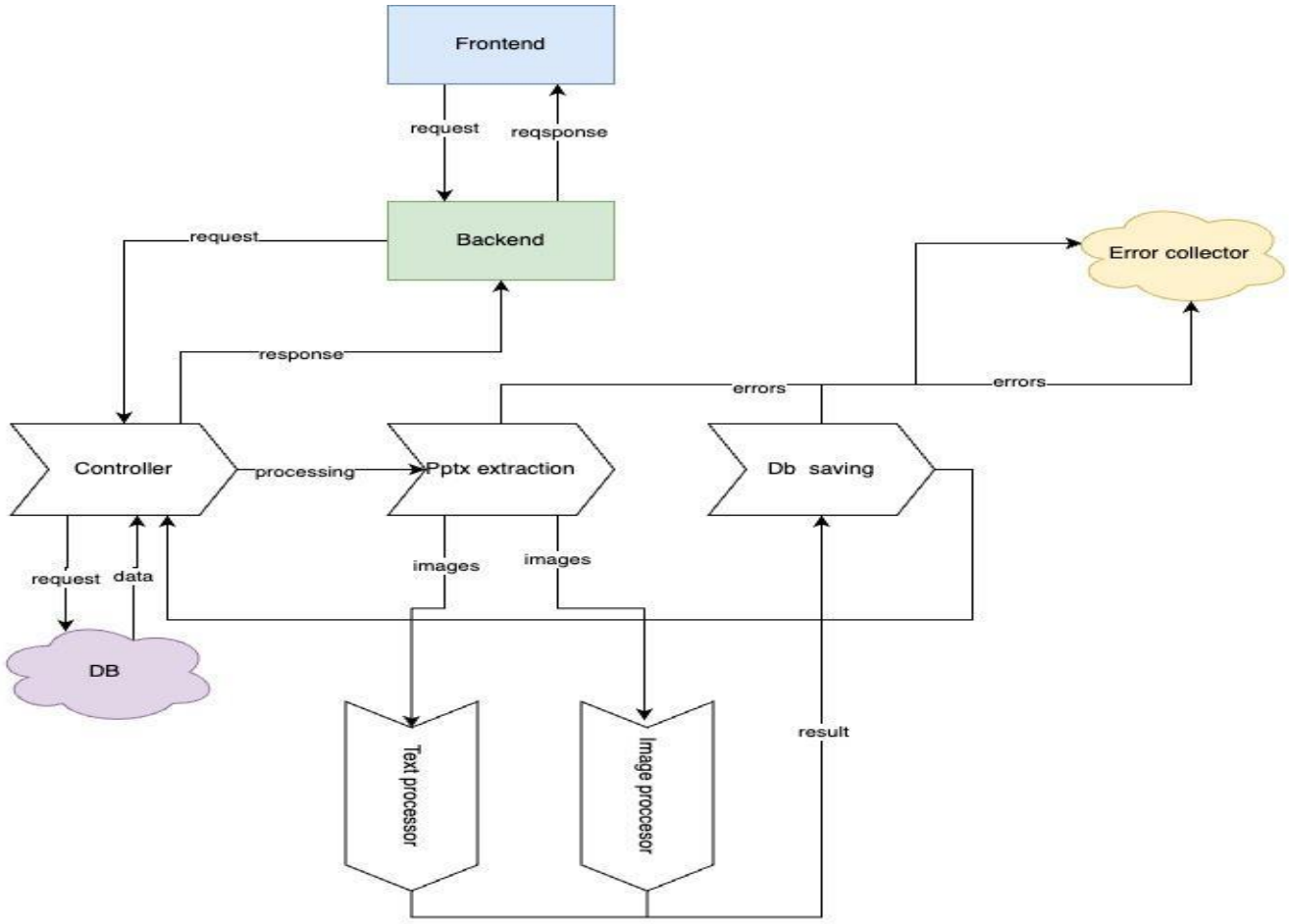


Рисунок 3.1 Архітектура системи пошуку та структурування презентацій

Джерело: запропоноване автором

У клієнт-серверній архітектурі сервер має контролери, які приймають запити від клієнтів та керують виконанням функціоналу аплікації. Фактично, контролери відіграють роль посередника між клієнтом і різними сервісами, які реалізують

функціональність серверної частини. Контролери отримують запити від клієнтів, перевіряють їх на валідність і передають відповідний запит до відповідного сервісу для обробки. Сервіси містять бізнес-логіку та виконують конкретні завдання, пов'язані з операціями над даними або іншими діями. Після обробки запиту сервіс повертає результат контролеру, який потім формує відповідь та передає її клієнту. Загалом, використання контролерів та сервісів в серверній архітектурі дозволяє забезпечити чітку роздільність функціональних компонентів, зробити розробку більш структурованою та покращити масштабованість системи. В даній системі можна виділити наступні сервіси - аналізу зображень та тексту. Їх основна мета полягає у виявленні та визначенні тегів, що описують зміст, характеристики або властивості контенту. Сервіс аналізу зображень використовує нейронні мережі для розпізнавання образів на зображеннях. Він проводить аналіз властивостей зображення, таких як форма, кольори, текстура, об'єкти та їхній контекст, щоб виявити релевантну інформацію та встановити зв'язки між об'єктами. На основі цього аналізу сервіс генерує теги, які описують зміст зображення. Сервіс аналізу тексту використовує алгоритми аналізу текстової інформації. Він виявляє ключові слова, фрази, теми, сутності та інші характеристики тексту, що допомагають класифікувати та описувати зміст. На виході обидва сервіси повертають набір тегів, які репрезентують зміст контенту.

Ці теги можуть включати описові характеристики, категорії, ключові слова або інші мітки, які допомагають ідентифікувати та організовувати контент. Вони можуть використовуватись для подальшого пошуку.

Також в було додано в систему сервіс моніторингу помилок. Сервіс моніторингу помилок в системі необхідний для збору, відстеження та аналізу помилок, що виникають під час роботи програмного забезпечення. Він допомагає виявляти, реєструвати та сповіщати про помилки, що виникають у реальному часі, забезпечуючи оперативну реакцію на проблеми. Цей сервіс допомагає розробникам

і адміністраторам системи швидко виявляти та виправляти помилки, поліпшувати стабільність та надійність системи, а також покращувати користувацький досвід.

3.2 Використані технології та бібліотеки

У процесі розробки було використано різноманітні технології, такі як Node.js, TypeScript, Nest.js, TypeORM, TensorFlow.js, Git, CockroachDB, Swagger та Sentry.

Розглянемо Node.js. Node.js - це середовище для виконання JavaScript коду. Раніше розробники запускали свій JavaScript-код лише за допомогою браузерів. Сьогодні Node.js працює незалежно від браузера і дозволяє запускати JavaScript-код і за межами браузера. Таким чином, він надає своїм додаткам API операційної системи, включаючи доступ до файлової системи, системних бібліотек або запущених процесів, включаючи HTTP-сервери. Node.js як середовище виконання не тільки дозволяє легко створювати веб-додатки. Це також інструмент для створення серверних додатків, скриптів і бібліотек різного роду. Він працює безпосередньо з операційною системою і може бути запущений на будь-якій з них, наприклад, Linux, macOS або Windows. Таким чином, за допомогою Node.js можна створювати веб-додатки з використанням таких фреймворків, як React, Angular, Vue або Svelte. Також можна створювати серверні додатки з використанням таких фреймворків, як Express, Nest, Koa. Можна створювати додатки як на JavaScript, так і на TypeScript.

Варто зазначити, що Node.js використовує движок V8. Це програмне забезпечення з відкритим вихідним кодом, на якому побудований браузер Chrome, а сьогодні і браузер Edge від Microsoft [21]. З точки зору розробки серверного програмного забезпечення, Node має ряд переваг. Найважливіші з них:

- висока продуктивність - Node призначений для оптимізації продуктивності і масштабованості веб-додатків і є хорошим рішенням багатьох поширених проблем при розробці такого типу додатків;

- код програми - це "старий добрий JavaScript", що означає, що ми витрачаємо менше часу на "перемикання контексту" при роботі з різними мовами на стороні клієнта та сервера;
- JavaScript є новою мовою програмування і має чимало переваг над іншими традиційно серверними мовами (наприклад, PHP, Python тощо);
- крім того, багато нових і популярних мов скомпільовані/перекладені на JavaScript, тому ви також можете використовувати TypeScript, CoffeScript, ClojureScript, Scala.js, LiveScript та інші;
- менеджер пакетів середовища Node (NPM - Node Package Manager) надає доступ до сотень тисяч пакетів для використання. Він має найкращий у своєму класі механізм вирішення залежностей і може бути використаний для автоматизації інструментів, які створюють додаток;
- Node.js є портативним. Він доступний для Microsoft Windows, macOs, Linux, Solaris, FreeBSD, OpenBSD, WebOS та NonStop OS. Він також підтримується постачальниками мережевих послуг, які надають відповідну інфраструктуру та документацію для розміщення додатків Node;
- має дуже активну сторонню екосистему та спільноту розробників, які готові допомогти.

Недоліки що має Node.js:

- середовище працює в однопотоковому режимі, заснованому на подіях;
- програма перестане реагувати або значно сповільниться при виконанні операцій, що вимагають великої кількості процесорних ресурсів, таких як обробка великих файлів, великих обсягів даних, редагування зображень;
- екосистема модулів все ще незріла;
- важко оцінити якість обраного модуля до його встановлення в додатку;
- простота публікації користувацьких пакетів у поєднанні з відсутністю надійного механізму затвердження модулів може означати, що нам слід

приділяти більше уваги при виборі модулів, перевіряючи їхню нещодавню активність, таку як виправлення помилок або оновлення [22].

Наступна технологія це TypeScript. TypeScript - це мова програмування з відкритим вихідним кодом, яка семантично є надбудовою над JavaScript. Основна команда, відповідальна за його підтримку, базується в Microsoft. Синтаксис мови майже ідентичний JavaScript. TypeScript додає деякі додаткові можливості, такі як статична типізація або інтерфейси. TypeScript передбачає надання змінним чітко визначеного типу даних та забезпечення його незмінності. Ще кілька років тому в PHP та JavaScript - найпопулярніших мовах програмування для веб-розробки - відсутність сильної типізації вважається перевагою, яка знижувала поріг входу для нових розробників. Однак зі зростанням складності проекту швидко стало очевидним, що це є недоліком, який збільшує ризик помилок. TypeScript компілюється в чистий JavaScript. TypeScript розширює JavaScript новими можливостями. У JavaScript відсутні багато механізмів, які полегшують створення складних фронтенд-проектів. І саме для цього був винайдений TypeScript (TS) [23]. Переваги TypeScript:

- завдяки типізації TypeScript дозволяє писати код, який легше зрозуміти, ніж код, створений на JavaScript. Набагато простіше здогадатися, за що відповідає та чи інша змінна, якщо крім самої назви відомо, яке саме значення вона зберігає. Програміст може не перейматися деталями, а зосередитися на більш важливих аспектах коду, таких як логічна правильність. Це може навіть сприяти зменшенню кількості помилок;
- на відміну від JavaScript, де розробник дізнається про помилки тільки під час запуску коду і тестування програми, TypeScript дозволяє виявити багато помилок вже на етапі трансполювання. Це процес, під час якого код перетворюється на зрозумілий код для програми, яка його виконує. Якщо частина коду не зрозуміла компілятору на цьому етапі, то програміст отримає попередження;

- TypeScript є основою не тільки надзвичайно популярного фреймворку Angular, але й безлічі інших сучасних веб-проектів. Зі знанням TypeScript розробнику легше і швидше освоїти багато нових інструментів;
- трансляція означає, що TypeScript автоматично заповнює прогалини між версіями JavaScript. Це означає, що розробник може використовувати найсучасніші функції, не турбуючись про те, чи буде його код працювати на старих браузерах або пристроях.

Недоліки використання TypeScript.

- TypeScript - це лише інструмент, і потрібно знати, коли він корисний, а коли ні. При створенні простих проектів ви можете виявити, що TypeScript буде просто зайвим шаром для освоєння;
- інша проблема полягає в тому, що використання TypeScript додає ще один набір інструментів для освоєння до вже значного набору, який ви повинні використовувати при створенні веб-додатків, і ще один блок для підключення в наш процес створення додатків, наприклад, доданий крок компіляції TS в Webpack;
- існує цілий ряд помилок, пов'язаних з недоречним використанням об'єктно-орієнтованого стилю програмування, але це вже більше питання до розробника коду, ніж до самої мови TypeScript [24].

Nest JS - це фреймворк для побудови серверних додатків на платформі Node.js. Він написаний на мові TypeScript. NestJS не тільки надає інструментарій, необхідний для створення проектів, а й вводить деякі правила для забезпечення дотримання найкращих практик. Nest JS дозволяє створювати серверні додатки з використанням різних протоколів. Найчастіше це REST-сервери на базі протоколу HTTP. Крім того, він забезпечує архітектуру проекту, впроваджує кращі практики, включає потужний механізм ін'єкції залежностей, використовує абстракції, що робить його дуже гнучким з точки зору вибору та підключення необхідних залежностей, наприклад, баз даних або бібліотек. Nest JS використовує механізм

декораторів, що робить його унікальним серед інших подібних фреймворків. Він запозичує найкраще з Angular і впорскує дані в окремі класи, так би мовити. Налаштувати його можна за допомогою ExpressJS, а також Fastify. Він був створений через досить поширену проблему в ExpressJS, а саме відсутність стандартизації. Код, написаний на ExpressJS, важко підтримувати і кожен писав його по-своєму. Найбільший акцент, який був зроблений на NestJS - це його філософія. Nest JS покликаний вирішити проблему відсутності архітектури та стандартизації додатків [25].

Наступна технологія це TypeORM. TypeORM - це простий у використанні ORM фреймворк з простим кодуванням. Він має наступні переваги:

- висока якість та слабозв'язані додатки;
- масштабовані додатки;
- легко інтегрується з іншими модулями;
- ідеально вписується в будь-яку архітектуру від малих до корпоративних додатків [26].

TypeORM має наступні можливості:

- автоматичне створення схем таблиць бази даних на основі ваших моделей;
- легко вставляти, оновлювати і видаляти об'єкти в базі даних;
- створення відображень (один-до-одного, один-до-багатьох і багато-до-багатьох) між таблицями;
- надає прості команди CLI.

TensorFlow.js - це бібліотека JavaScript з відкритим вихідним кодом, розроблена Google, яка дозволяє запускати моделі машинного навчання та глибокого навчання безпосередньо в браузері або на Node.js. Вона дозволяє розробникам створювати та розгортати додатки машинного навчання за допомогою JavaScript, що робить його доступним для широкого кола веб-розробників. Ось деякі ключові особливості та переваги TensorFlow.js:

- сумісність з браузерами: TensorFlow.js використовує можливості сучасних веб-браузерів для виконання моделей машинного навчання без необхідності обчислень на стороні сервера. Це дозволяє розробникам створювати додатки, які можуть виконувати завдання виведення безпосередньо на стороні клієнта, зменшуючи затримки та покращуючи взаємодію з користувачем;
- підтримка глибокого навчання: TensorFlow.js підтримує моделі глибокого навчання, що дозволяє розробникам навчати і розгортати складні нейронні мережі в браузері. Він надає високорівневий API, подібний до TensorFlow, що дозволяє легко визначати, навчати та запускати моделі глибокого навчання за допомогою JavaScript;
- перенесення навчання: TensorFlow.js включає в себе попередньо навчені моделі, які розробники можуть використовувати для трансферного навчання. Трансферне навчання дозволяє розробникам використовувати існуючі моделі, які були навчені на великих наборах даних, і застосовувати їх до нових, пов'язаних завдань з меншими наборами даних. Це полегшує побудову точних і ефективних моделей з обмеженим обсягом даних;
- висновок на пристрої: TensorFlow.js дозволяє робити висновки на пристрої, тобто моделі машинного навчання працюють безпосередньо на пристрої користувача, не вимагаючи мережевого зв'язку. Це забезпечує конфіденційність і безпеку даних, оскільки конфіденційні дані не потрібно надсилати на сервер для обробки;
- прискорення WebGL: TensorFlow.js використовує WebGL, графічну бібліотеку для Інтернету, щоб прискорити обчислення моделей машинного навчання в браузері. Це забезпечує ефективне паралельне виконання на графічному процесорі, що дозволяє швидше робити висновки та покращує продуктивність для завдань з інтенсивними обчисленнями;
- інтеграція з екосистемою TensorFlow: TensorFlow.js легко інтегрується з більшою екосистемою TensorFlow. Моделі, навчені в інших середовищах

TensorFlow, таких як TensorFlow Python, можуть бути перетворені і використані в TensorFlow.js. Ця сумісність полегшує повторне використання моделей на різних платформах.

TensorFlow.js - це комплексна бібліотека машинного навчання, яка надає розробникам JavaScript можливості TensorFlow, популярного фреймворку машинного навчання. Вона дозволяє створювати, навчати та розгортати моделі машинного навчання безпосередньо в браузері або на Node.js. TensorFlow.js поєднує в собі гнучкість і доступність JavaScript з обчислювальними можливостями машинного навчання, що робить його універсальним інструментом для широкого спектру застосувань [27].

Sentry - це сервіс з відкритим вихідним кодом для моніторингу помилок у проекті. Дозволяє швидко виявляти, аналізувати та усувати будь-які аномалії в коді, що значно полегшує роботу розробників. Його основна перевага полягає в тому, що він відстежує помилки в режимі реального часу і ті, які важко зловити на етапі тестування. Це виключає необхідність очікування звітів від користувачів програмного забезпечення, що має суттєвий вплив на ефективність та швидкість роботи у виробничих умовах. Sentry має приємний інтерфейс та інформує розробників про будь-які помилки, надсилаючи їм електронний лист. Більше того, вона достатньо розумна, щоб самостійно видаляти будь-які конфіденційні дані зі звітів, наприклад, паролі або номери рахунків та кредитних карток.

Сервіс дозволяє налаштовувати користувацькі функції та адаптувати їх до потреб розробника, а також відключати сповіщення про помилки для певних типів виключень. Крім того, Sentry дозволяє розробникам отримувати контекст навколишнього середовища, оскільки надає випадаючий список для фільтрації даних про повідомлення, проблеми та відгуки від інших користувачів. Саме тому цей сучасний хмарний сервіс використовують найбільші світові бренди, такі як Uber, Reddit, Instagram, Udemy та Dropbox [28].

Git - це система контролю версій, яка дозволяє розробникам відстежувати зміни в їхньому коді, співпрацювати з іншими розробниками та керувати кількома версіями своєї кодової бази. Git можна використовувати у сценаріях "що, якщо", використовуючи його функцію розгалуження. Git дозволяє розробникам створювати гілки своєї кодової бази, які є окремими копіями коду, над якими можна працювати незалежно. Це дозволяє розробникам експериментувати з різними змінами та версіями коду, не впливаючи на основну кодову базу. Вони можуть тестувати різні сценарії, оцінювати результати, а потім вирішувати, чи варто зливати зміни назад в основну кодову базу, чи відкинути їх. Наприклад, розробник може створити нову гілку для тестування нової функції, а потім, якщо вона працює добре, злити її в основну гілку або відкинути, якщо вона не відповідає вимогам. Крім того, Git також дозволяє розробникам відкочувати зміни до попередньої версії, що може бути корисним у випадку не очікуваного результату.

Таким чином, Git дозволяє розробникам легко керувати різними версіями своєї кодової бази та тестувати різні сценарії "що, якщо", що робить його важливим інструментом для розробки та супроводу програмного забезпечення. Git був створений для розробників, оскільки він вирішує кілька ключових проблем, з якими стикаються розробники під час роботи над програмними проектами:

- Git дозволяє декільком розробникам одночасно працювати над однією і тією ж кодовою базою, відстежуючи зміни та запобігаючи конфліктам;
- Git зберігає історію всіх змін, внесених до кодової бази, що дозволяє розробникам легко повертатися до попередніх версій або порівнювати різні версії коду;
- Git дозволяє розробникам створювати гілки своєї кодової бази, які можна використовувати для експериментів з новими функціями, тестування або виправлення помилок;
- Git є швидким і легким, що робить його чудовим інструментом для використання розробниками у повсякденному робочому процесі;

- Git - це програмне забезпечення з відкритим вихідним кодом, що означає, що його може вільно використовувати, змінювати та поширювати будь-хто;
- Git є найпоширенішою системою контролю версій, а це означає, що багато розробників вже знайомі з нею, що полегшує їм спільну роботу над проектами.

  [29].

CockroachDB - це розподілена база даних SQL. Її розподілений дизайн забезпечує надмірність даних. CockroachDB зберігає ваші дані в безпеці, навіть якщо один сервер бази даних виходить з ладу. CockroachDB може пережити відмову диска, машини, стійки та центрів обробки даних. Ця здатність до виживання є причиною того, чому розробники цієї бази даних вибрали назву "CockroachDB". CockroachDB фокусується на підтримці цілісності даних, навіть якщо операція з базою даних зазнає невдачі.

Для цього CockroachDB фокусується на узгодженості. Він використовує сильно узгоджене сховище ключ-значення. Вона створена для підтримки систем обробки транзакцій, SoR (Systems of Record) тощо. Ця СУБД з відкритим вихідним кодом (реляційна система управління базами даних) масштабується горизонтально.

CockroachDB пропонує наступні можливості:

- довговічність ваших даних: Розподілена СУБД CockroachDB підтримує реплікацію та автоматизований ремонт. Це забезпечує живучість. Ваші дані залишаються, навіть якщо один сервер бази даних виходить з ладу;
- ваші дані залишаються правдивими та послідовними: Як ми вже обговорювали, CockroachDB суворо відповідає керівним принципам ACID. Це підтримує цілісність даних. CockroachDB не дозволяє вашим даним погоджуватися;
- CockroachDB використовує алгоритм консенсусу Raft для операцій "запису". Для операцій "читання" використовується власний алгоритм

синхронізації за часом. Ці фактори дозволяють CockroachDB пропонувати сильну узгодженість.

- простота використання: Ви можете легко розгорнути і використовувати CockroachDB;
- підтримка SQL: CockroachDB підтримує SQL, тому розробники можуть використовувати схеми, таблиці, рядки, стовпці та індекси. CockroachDB також є трансакційним сховищем ключів-значень, і він розподілений;
- підтримує розподілені транзакції: CockroachDB підтримує розподілені транзакції в кластері. Незалежно від того, чи є у вас кілька серверів в одному місці або багато серверів в декількох місцях, CockroachDB може підтримувати розподілені транзакції;
- масштабованість: Ви можете масштабувати Cockroach DB горизонтально, і це не вимагає операційних накладних витрат.

CockroachDB пропонує наступні переваги:

- команда Cockroach Labs забезпечує чуйну, оперативну та ефективну підтримку клієнтів. Це справедливо для безкоштовної версії;
- є можливість використовувати "CockroachDB Core", версію CockroachDB з відкритим вихідним кодом для більшості цілей. Вона безкоштовна. Вартість ліцензії на процесор повністю підтримуваної версії CockroachDB Enterprise менше, ніж у Oracle RAC;
- можливість розгорнути Cockroach DB локально. Розгортка в публічній та приватній хмарі. CockroachDB підтримує розгортання контейнерів, віртуальних машин і голих серверів. Cockroach Labs пропонує Cockroach Cloud, базу даних як послугу в хмарі AWS і Google;
- CockroachDB відповідає керівним принципам ACID ("Атомарність", "Узгодженість", "Ізоляція" і "Довговічність"). Вона підтримує цілісність даних. Це перевага для SoR і OLTP систем;
- CockroachDB пропонує високий ступінь доступності;

- CockroachDB пропонує високу продуктивність в порівнянні з системою бенчмаркінгу TPC-C;
- Cockroach DB пропонує хмарну та горизонтальну масштабованість. Це високамасштабовона СУБД;
- можливість легкого розгорнення CockroachDB;
- підтримка PostgreSQL: PostgreSQL - популярна СУБД з відкритим вихідним кодом. CockroachDB підтримує SQL діалект PostgreSQL. Можливість використання клієнтських бібліотек PostgreSQL для підключення до CockroachDB;
- CockroachDB підтримує активні динамічні зміни схем. Підтримує додатки, що враховують особливості центру обробки даних, завдяки підтримці гео розподілу.

CockroachDB та PostgreSQL мають наступні спільні риси:

- обидві є СУБД. Обидві підтримують схеми, SQL-запити, таблиці, рядки, стовпці, індекси тощо;
- PostgreSQL і Cockroach DB є ACID-сумісними;
- обидві СУБД пропонують високу продуктивність і доступність;
- можливість отримати відмінну підтримку як для PostgreSQL, так і для Cockroach DB;
- CockroachDB і PostgreSQL підтримують як хмарне, так і локальне розгортання [30].

Для швидкого конфігурування кожного компонента системи було використано Docker. Docker - це платформа контейнеризації з відкритим вихідним кодом. Вона дозволяє розробникам упаковувати додатки в контейнери - стандартні виконувані компоненти, які об'єднують вихідний код програми з залежностями та бібліотеками операційної системи, необхідними для запуску цього коду в будь-якому середовищі. Контейнери спрощують доставку розподілених додатків і стають все більш популярними, оскільки організації все частіше створюють хмаро

орієнтовані рішення і використовують гібридні мульти хмарні середовища [31]. Додатково до Docker використовується Docker Compose який автоматизує створення і запуск декількох контейнерів. Це значно спрощує роботу і виключає людський фактор, особливо якщо нам потрібно запустити кілька контейнерів, які залежать один від одного. Він дозволяє здійснювати дії в умовах, порядку та строки. Ось чому Docker Orchestration, або, як її ще називають, Compose, часто використовується для опису правил спільної роботи контейнерів.

Є невелика різниця між Docker і Docker Compose. Docker застосовується для управління окремими контейнерами (сервісами), з яких складається додаток. Docker Compose використовується для одночасного керування кількома контейнерами, що входять до складу програми. Цей інструмент пропонує ті самі можливості, що й Docker, але дає змогу працювати зі складними застосунками [32].

Swagger - це програмний фреймворк з відкритим вихідним кодом, який дозволяє розробникам проектувати, створювати, документувати та використовувати RESTful веб-сервіси. Він надає набір інструментів і специфікацій, які полегшують процес розробки та сприяють співпраці між різними командами, що працюють над проектом.

В основі Swagger лежить специфікація OpenAPI (раніше відома як Swagger Specification) для визначення API. Специфікація OpenAPI - це стандартний формат для опису REST ful API, включаючи кінцеві точки, параметри, формати запитів і відповідей, методи автентифікації та багато іншого. Він слугує контрактом між сервером, що надає API, та клієнтами, які його споживають, забезпечуючи безперебійну комунікацію та інтеграцію.

Swagger пропонує кілька ключових особливостей, які роблять його популярним серед розробників:

- за допомогою Swagger розробники можуть проектувати API, використовуючи зручний для читання і зрозумілий для машин формат. Специфікація OpenAPI дозволяє їм визначати кінцеві точки, структури

запитів/відповідей, типи даних тощо. Крім того, Swagger надає зручний інтерфейс для документування API, що полегшує розробникам їх розуміння та використання;

- Swagger може автоматично генерувати серверні заглушки та клієнтські SDK для різних мов програмування на основі специфікації API. Ця функція економить час і зусилля, зменшуючи ручну роботу, необхідну для впровадження та використання API;
- інтерактивне тестування API: Swagger надає інтерактивний інтерфейс під назвою Swagger UI, який дозволяє розробникам тестувати і вивчати API безпосередньо з документації. Він надає середовище пісочниці, де розробники можуть вводити параметри, надсилати запити та переглядати відповіді в режимі реального часу. Ця функція спрощує процес тестування та налагодження під час розробки API;
- управління версіями та життєвим циклом API: Swagger підтримує керування версіями API, що дозволяє розробникам ефективно керувати різними версіями своїх API. Він також надає можливості для управління життєвим циклом API, включаючи застарівання старих версій, випуск нових версій і забезпечення зворотної сумісності;
- інтеграція з інструментами розробника: Swagger легко інтегрується з популярними інструментами та фреймворками для розробників. Він пропонує плагіни та інтеграції для редакторів коду, систем збірки, конвеєрів безперервної інтеграції/безперервного розгортання (CI/CD) тощо. Ця сумісність дозволяє розробникам інтегрувати Swagger в існуючі робочі процеси без жодних перешкод;
- підтримка екосистеми та спільноти: Swagger має енергійну та активну спільноту, яка робить свій внесок у його розвиток та надає підтримку користувачам. Існують численні ресурси, навчальні посібники та форуми, доступні для вивчення та усунення несправностей, пов'язаних зі Swagger.

Широке розповсюдження Swagger також означає, що існує багато сторонніх інструментів і бібліотек, створених на його основі, що розширює його функціональність і можливості розширення [33].

Використовуючи Swagger під час розробки, розробники можуть оптимізувати життєвий цикл розробки API, покращити співпрацю та комунікацію, автоматизувати документування та генерацію коду, а також полегшити тестування та налагодження. Це допомагає створювати добре спроектовані, добре задокументовані та легко використовувані API, що в кінцевому підсумку призводить до більш ефективних та успішних проектів розробки [34].

3.3 Тестування системи та аналіз результатів

Веб додаток складається з двох сторінок які доступні для юзера. Веб додаток має підключений Swagger який було описано раніше. Додаток має 4 запити до сервера, такі як:

- POST/pptx/upload - процес завантаження презентацій до системи, для подальшого аналізу файлу. Під час цього запиту успішний код це 201. Що означає що запис був доданий до бази. У разі відсутності файлу у відповідь буде інформація що необхідно додати файл;
- GET/pptx/list - за допомогою даного запиту відображається список всіх завантажених презентацій. Дана дія зображена на фотографіях з додатку нижче;
- GET/pptx/search - для пошуку необхідної презентації за кодовим словом є дана дія. Необхідно ввести слово та система знайде необхідну презентацію;
- GET/pptx/download - користувач має можливість завантажити знайдену презентацію для подальшого використання.

На рисунку 3.2-3.5 зображенні всі 4 запити. На наступних рисунках зображені інші запити.

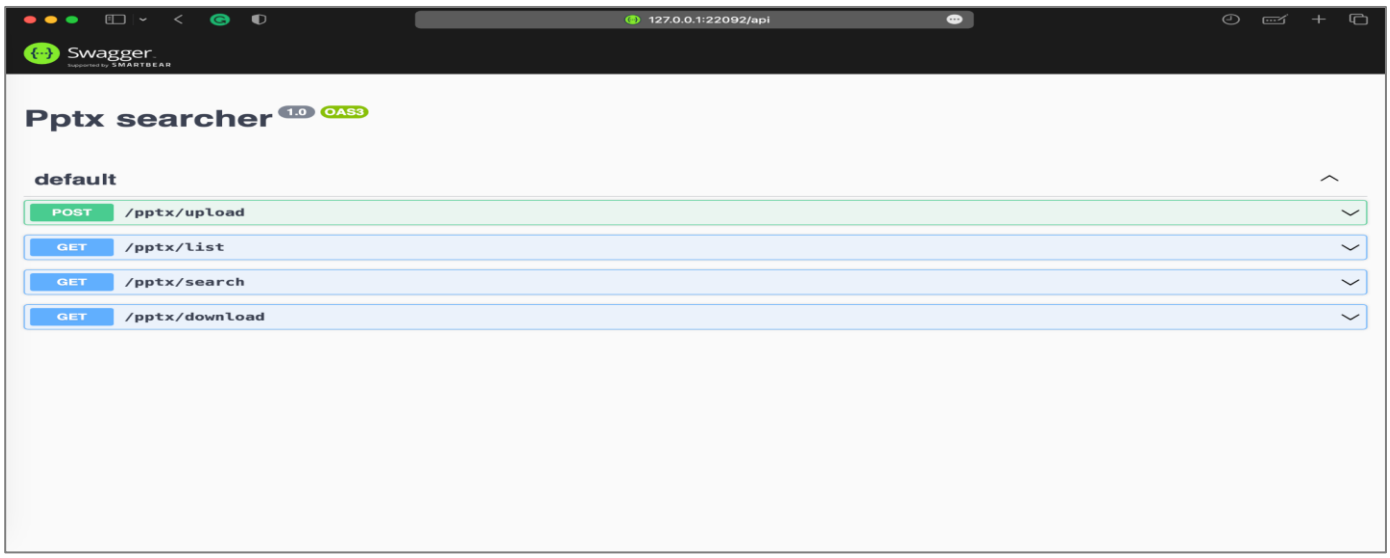


Рисунок 3.2 – Всі запити

Джерело: запропоноване автором

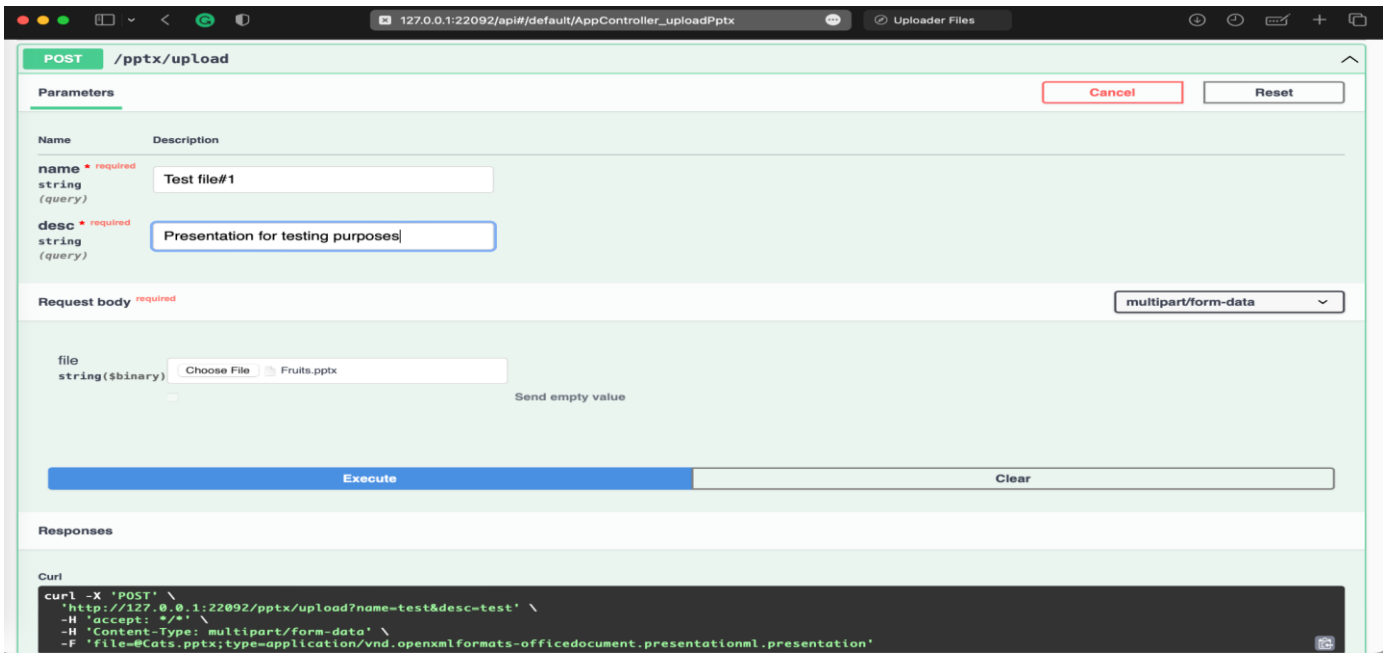


Рисунок 3.3 – Завантаження файлів

Джерело: запропоноване автором

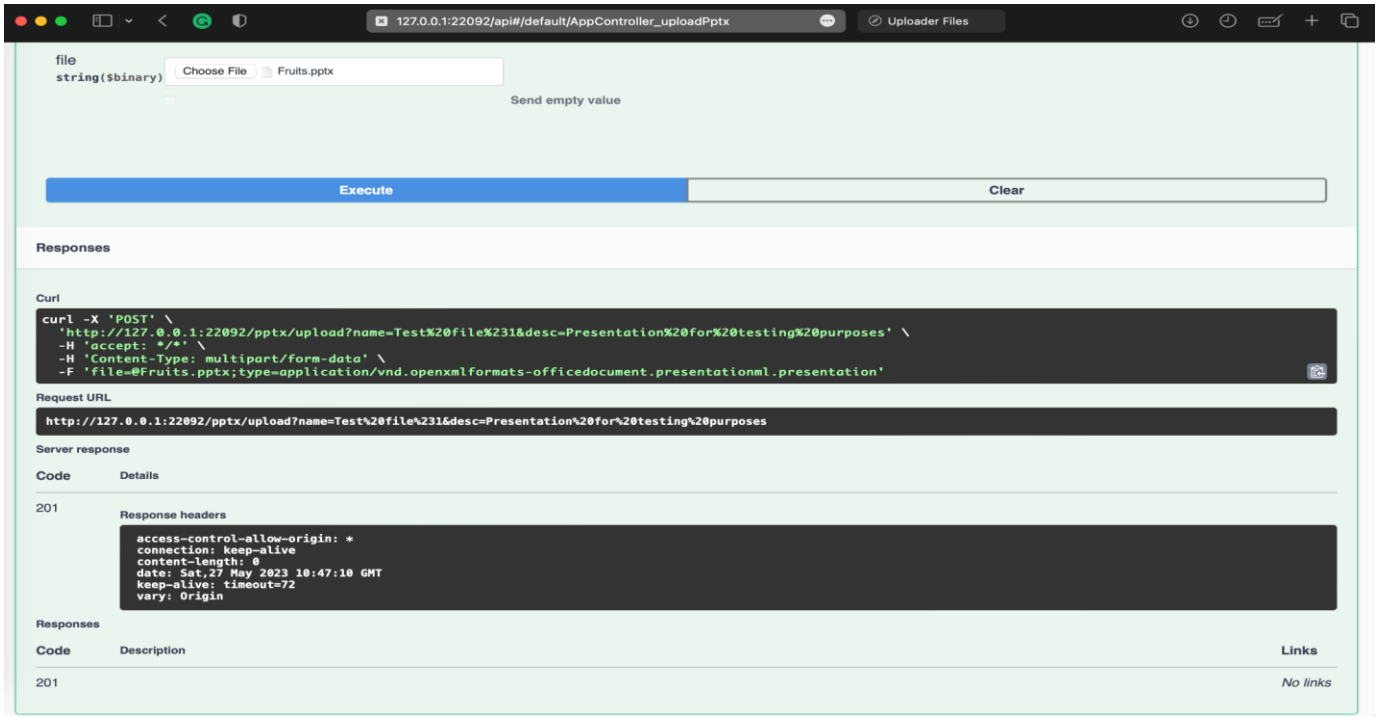


Рисунок 3.4 – Завантаження файлів

Джерело: запропоноване автором

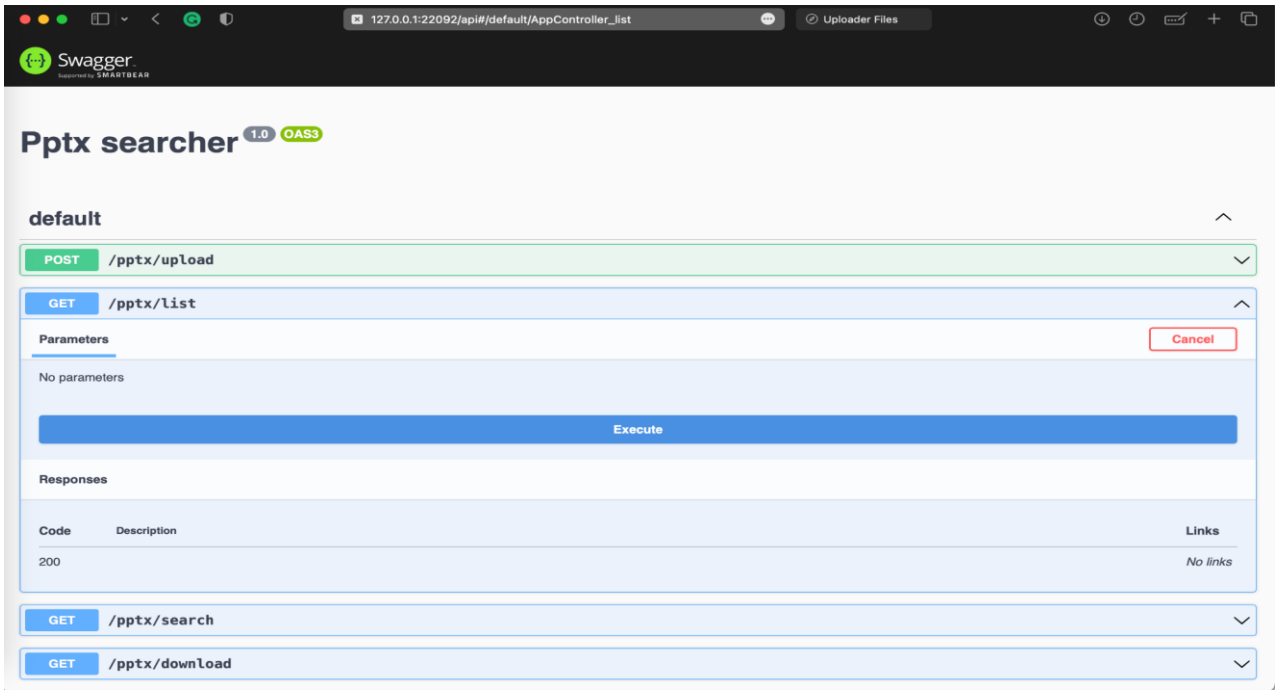


Рисунок 3.5 – Перегляд завантажених файлів

Джерело: запропоноване автором

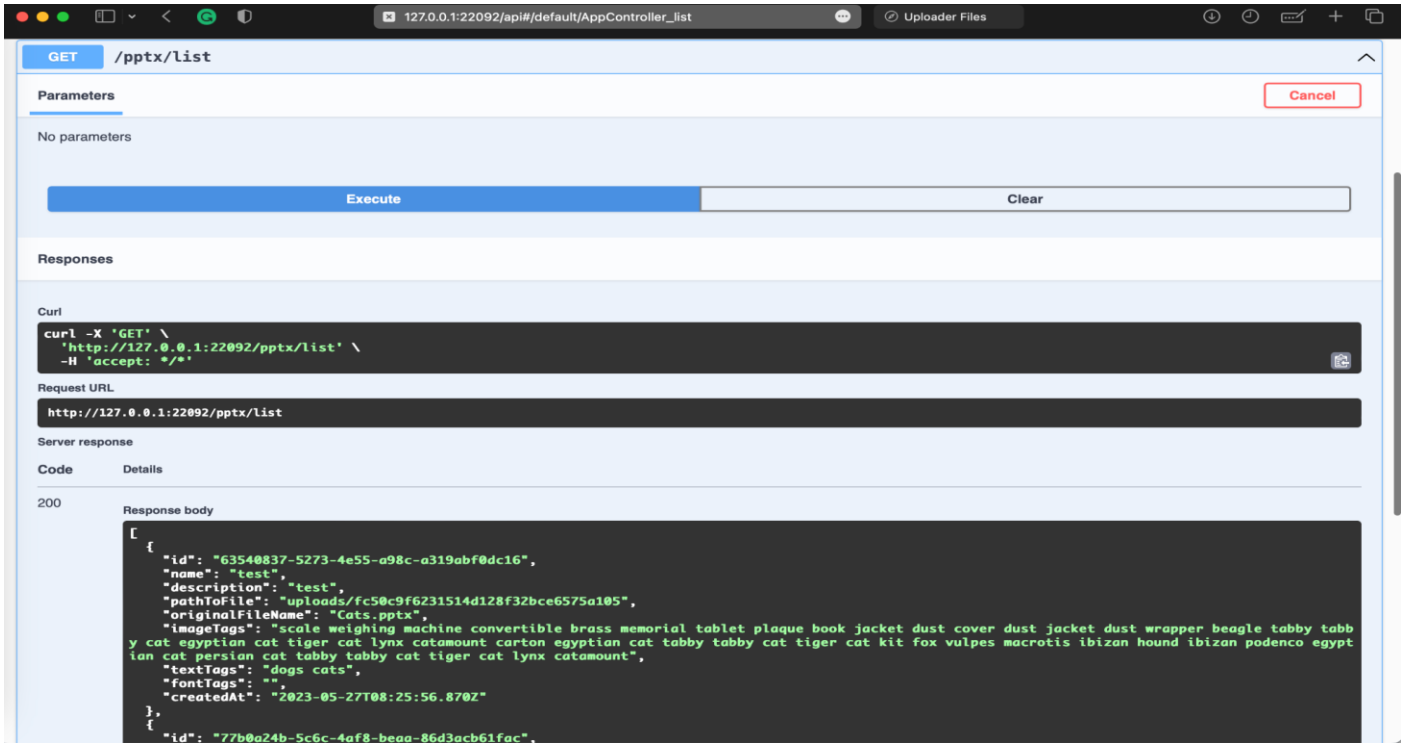


Рисунок 3.6. – Перегляд завантажених файлів

Джерело: запропоноване автором

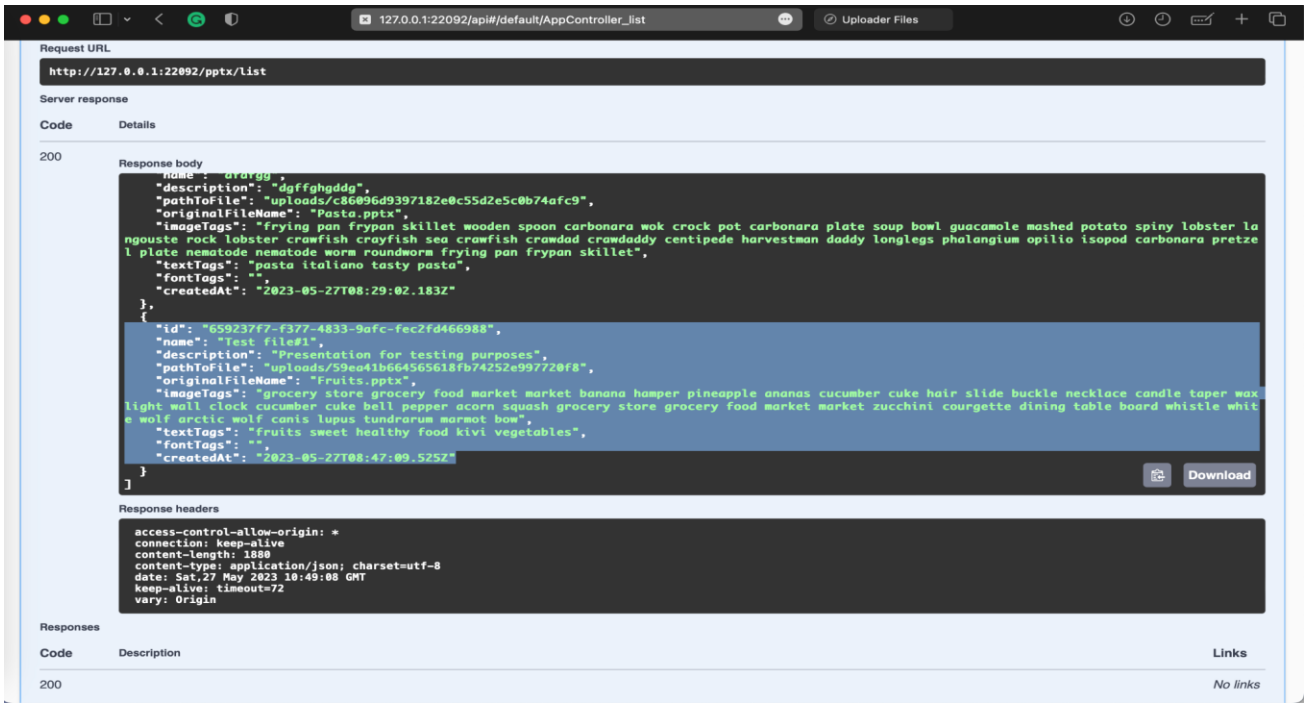


Рисунок 3.7. – Перегляд завантажених файлів

Джерело: запропоноване автором

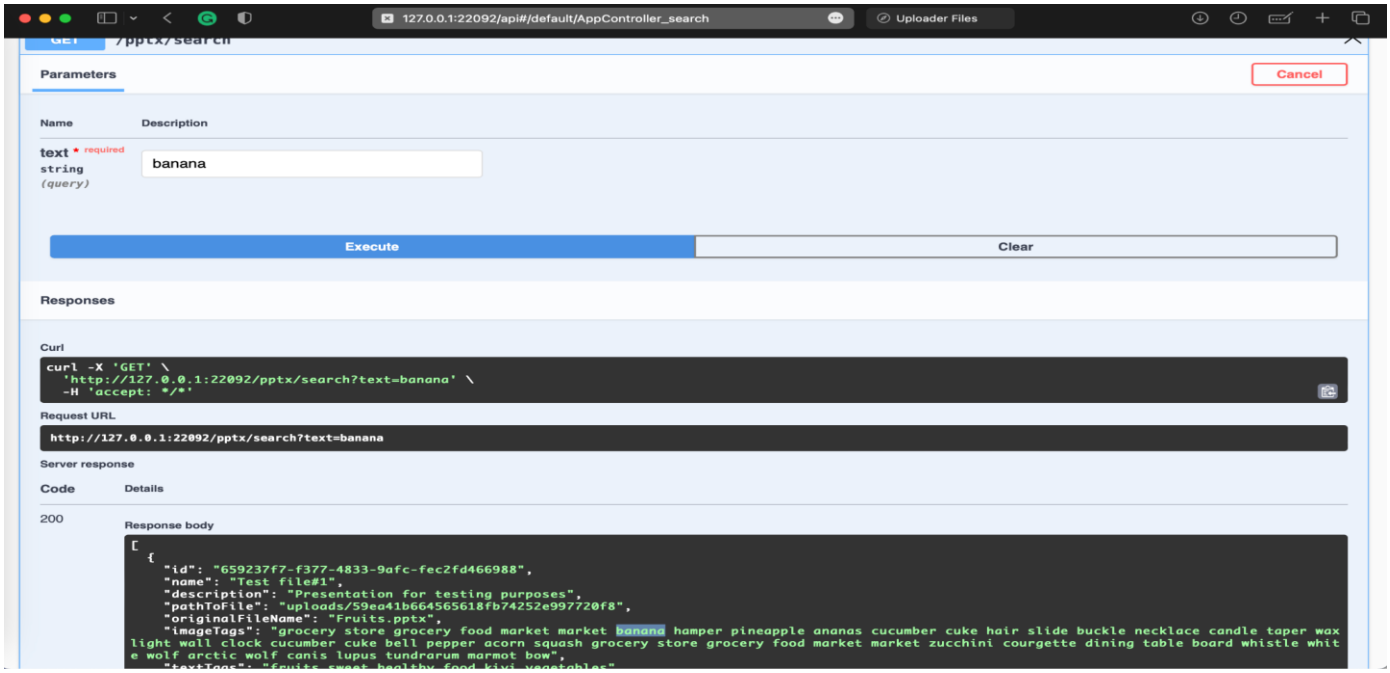


Рисунок 3.8. – Пошук серед завантажених файлів

Джерело: запропоноване автором

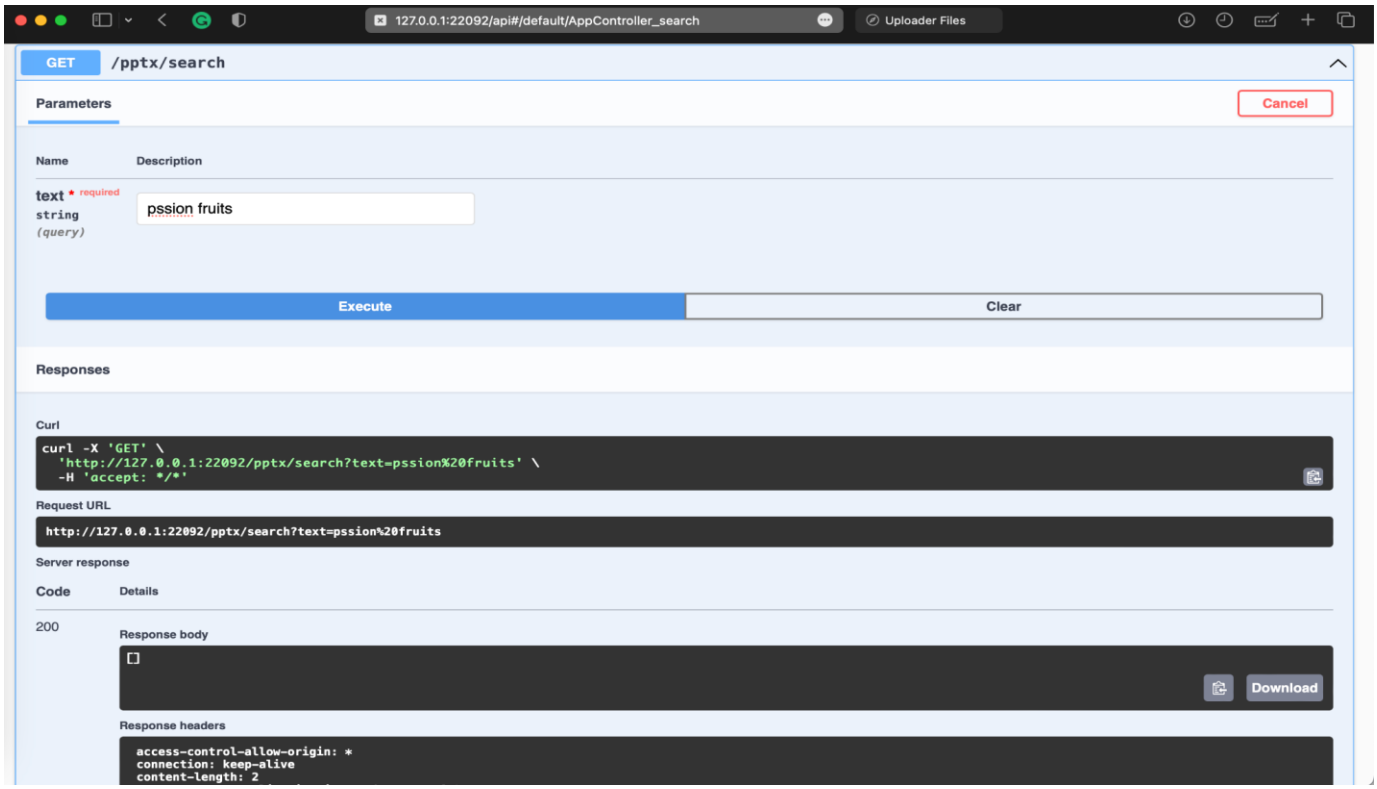


Рисунок 3.9 – Пошук серед завантажених файлів

Джерело: запропоноване автором

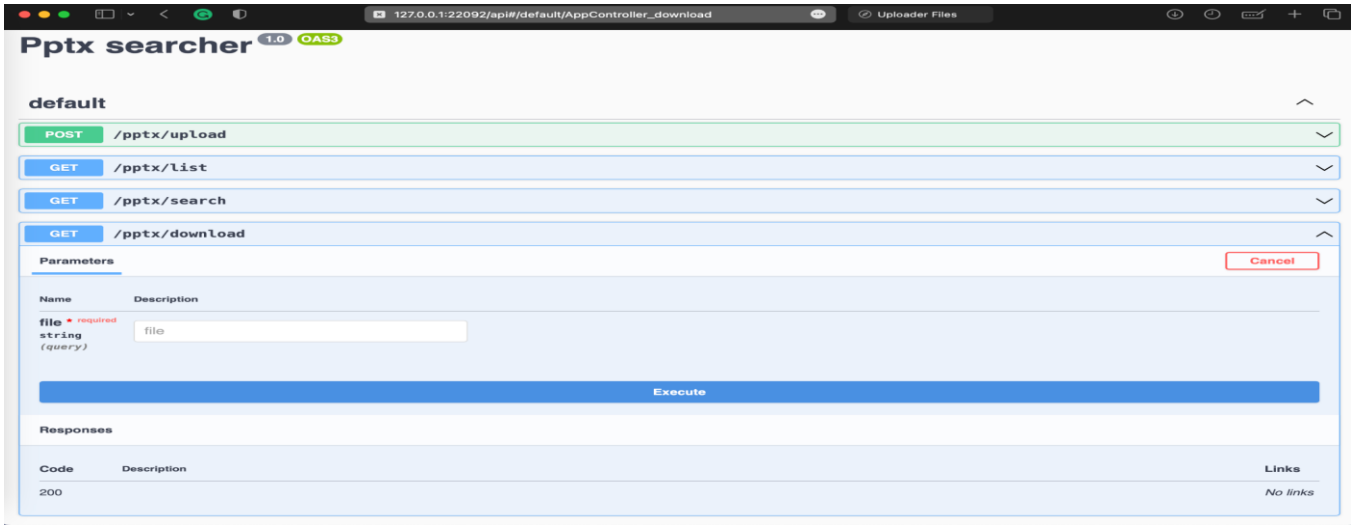


Рисунок 3.10. – Можливість завантаження обраного файлу

Джерело: запропоноване автором

Проводилось тестування веб - додатку за допомогою підготовлених файлів для тестування. Файли з різними наборами фотографій та тексту, щоб перевірити позитивні та негативні сценарії. Нижче на фотографії зображені файли для тестування системи. Для тестування достатньо було 8 презентацій. Презентації мають різні назви, часом не пов'язані зі змістом всередині, щоб перевірити, як програма аналізує картинки.

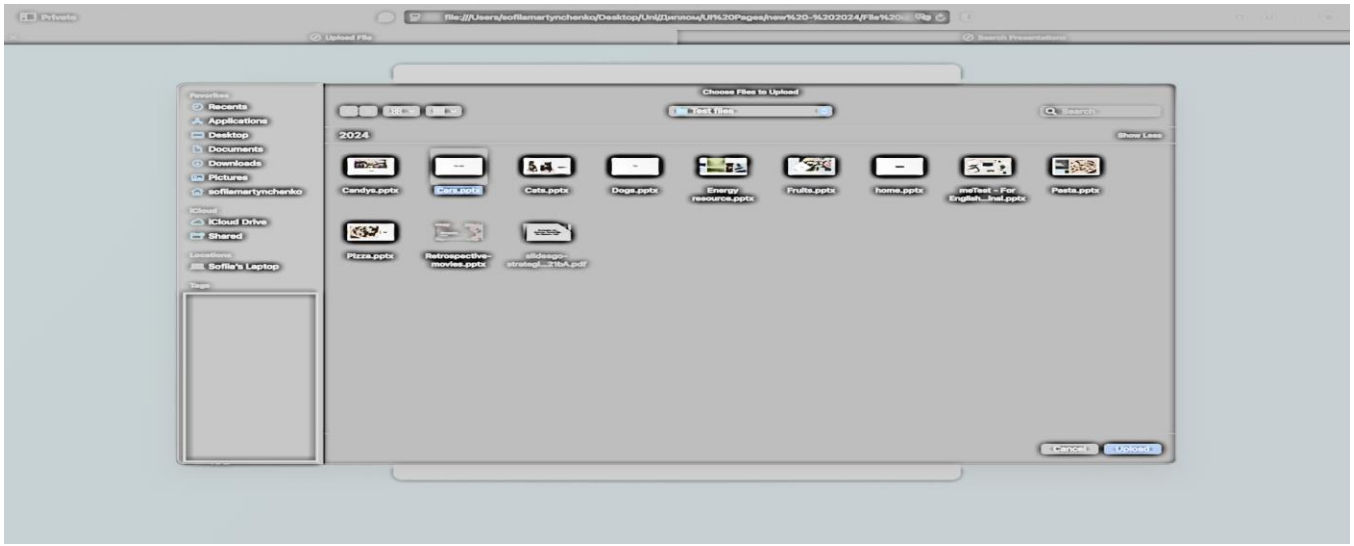


Рисунок 3.11. – Файли для тестування

Джерело: запропоноване автором

Приклад презентації що була використана для тестування програми наведена нижче на рисунку 14.

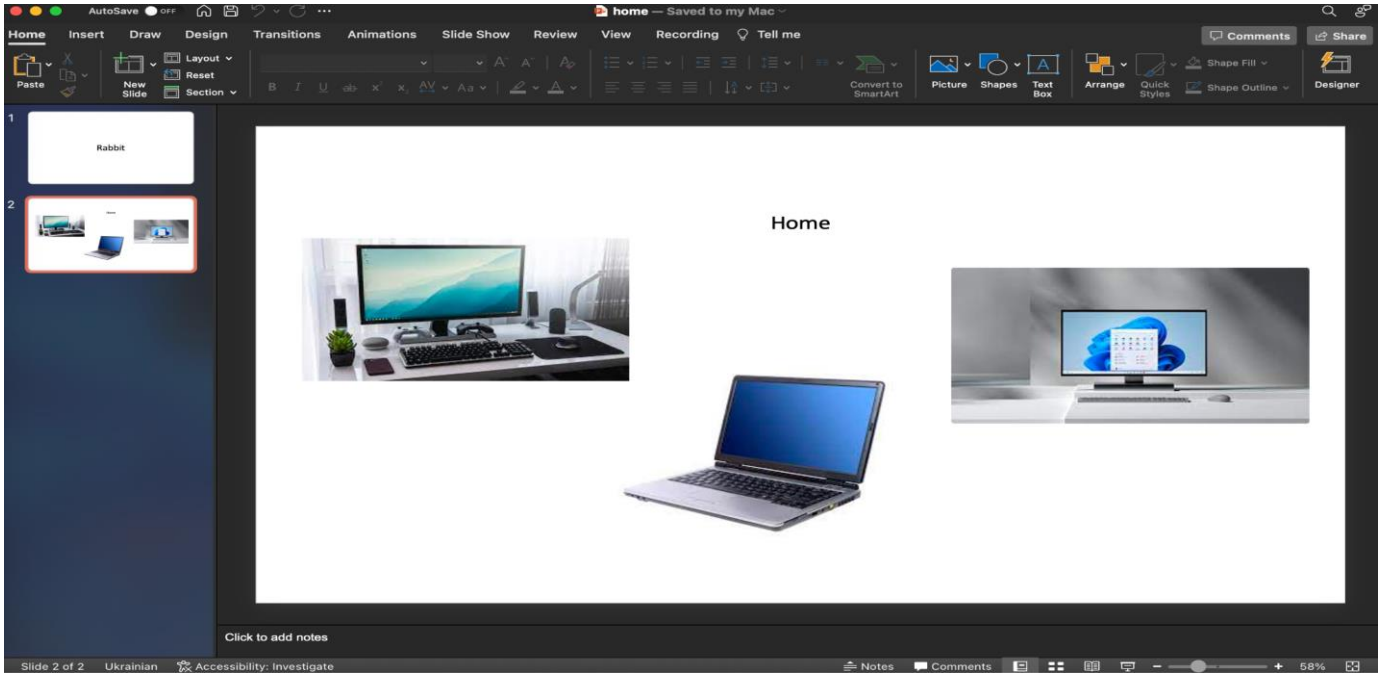


Рисунок 3.12. – Прикладова презентація для тестування

Джерело: запропоноване автором

Користувачі мають доступ до інтерфейсу. Нижче, на скріншоті, зображено сторінку "Upload file", де користувач має змогу завантажити файл до додатку. Якщо користувач натисне на кнопку "Click here to start searching", то юзер буде перенаправлений на сторінку з пошуку серед завантажених презентацій. Далі будуть додані скріншоти.

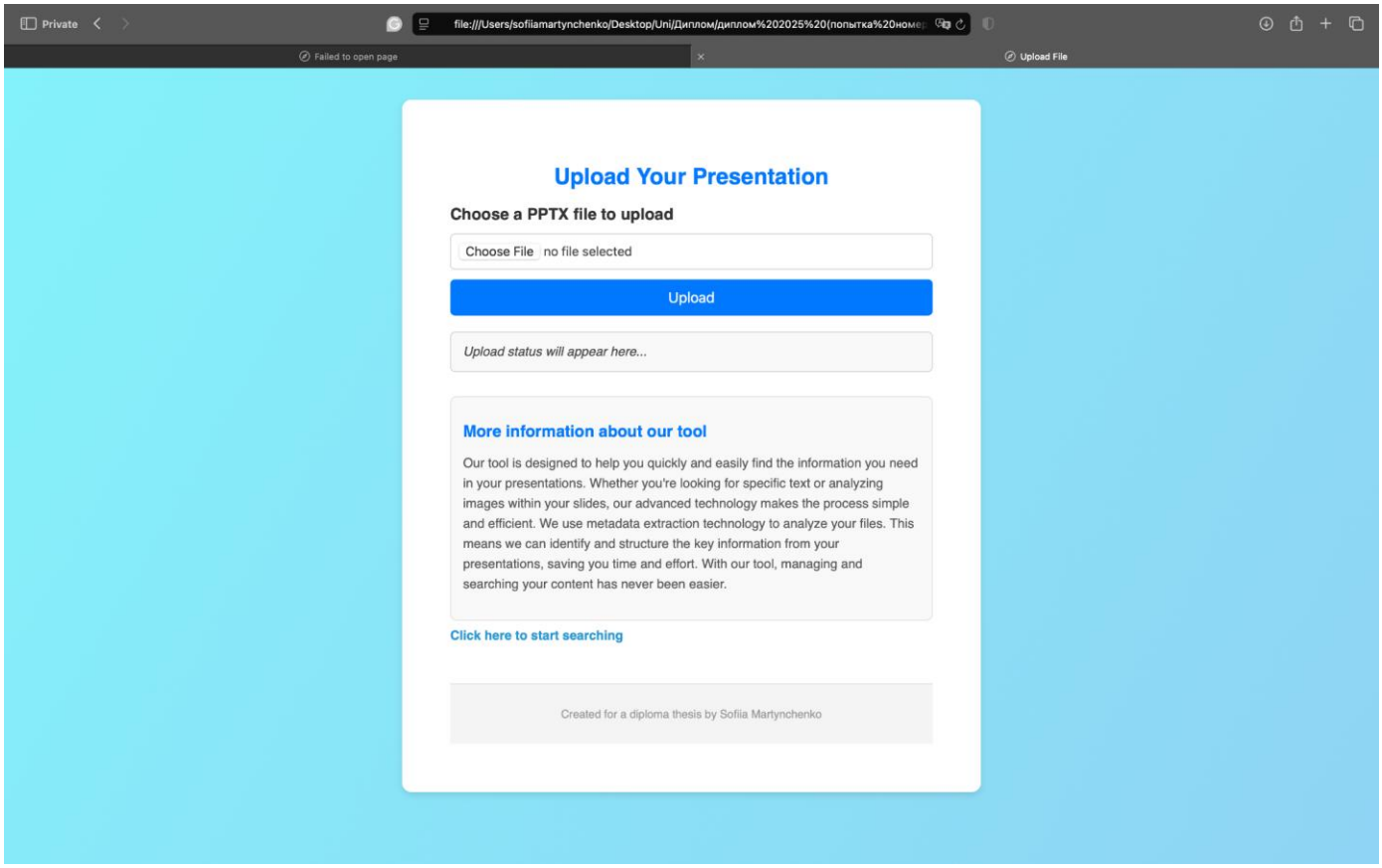


Рисунок 3.13. – Сторінка Uploader

Джерело: запропоноване автором

Сторінка під назвою "Searcher in Presentations" дає змогу користувачеві шукати контекст презентації за словом, а саме за наявними у презентації фотографіями. У випадку, коли немає завантажених презентацій, під полем пошуку не буде презентацій. Коли є завантажені презентації, вони є видимі на цій сторінці.

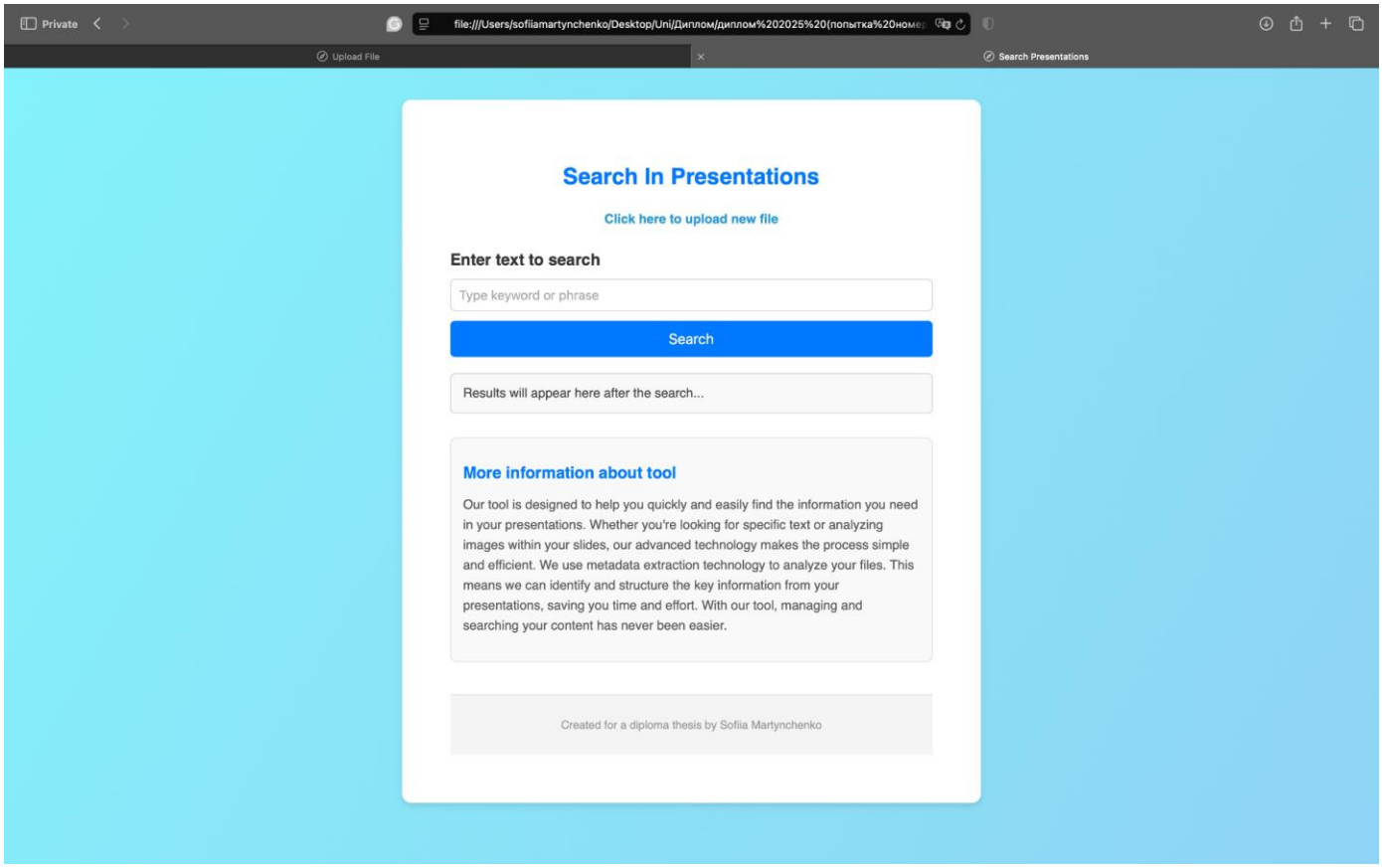


Рисунок 3.14. – Сторінка Searcher in Presentations

Джерело: запропоноване автором

Коли користувач ввів слово, за яким система повинна шукати презентацію, працює алгоритм, і програма видає результати, тобто презентації, які мають картинки з введеним словом. У даному випадку це картинка кота. Тому користувач може побачити цю презентацію та завантажити її.

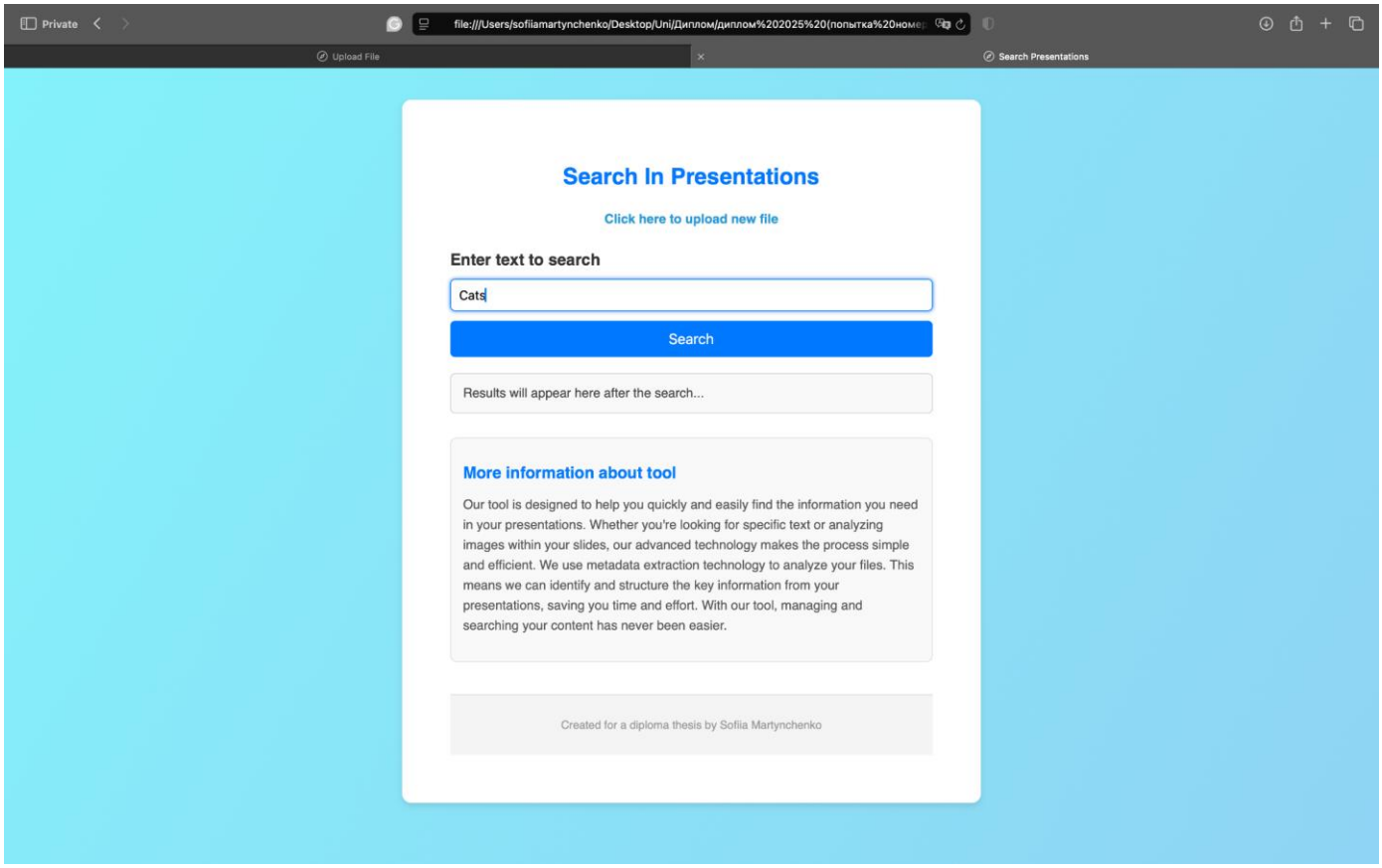


Рисунок 3.15. – Пошук презентації

Джерело: запропоноване автором

У випадку коли немає презентації з введеним словом, користувач не побачить результатів пошуку.

На рисунку 3.16 зображена завантажена знайдена презентація. Презентація відкривається без помилок.

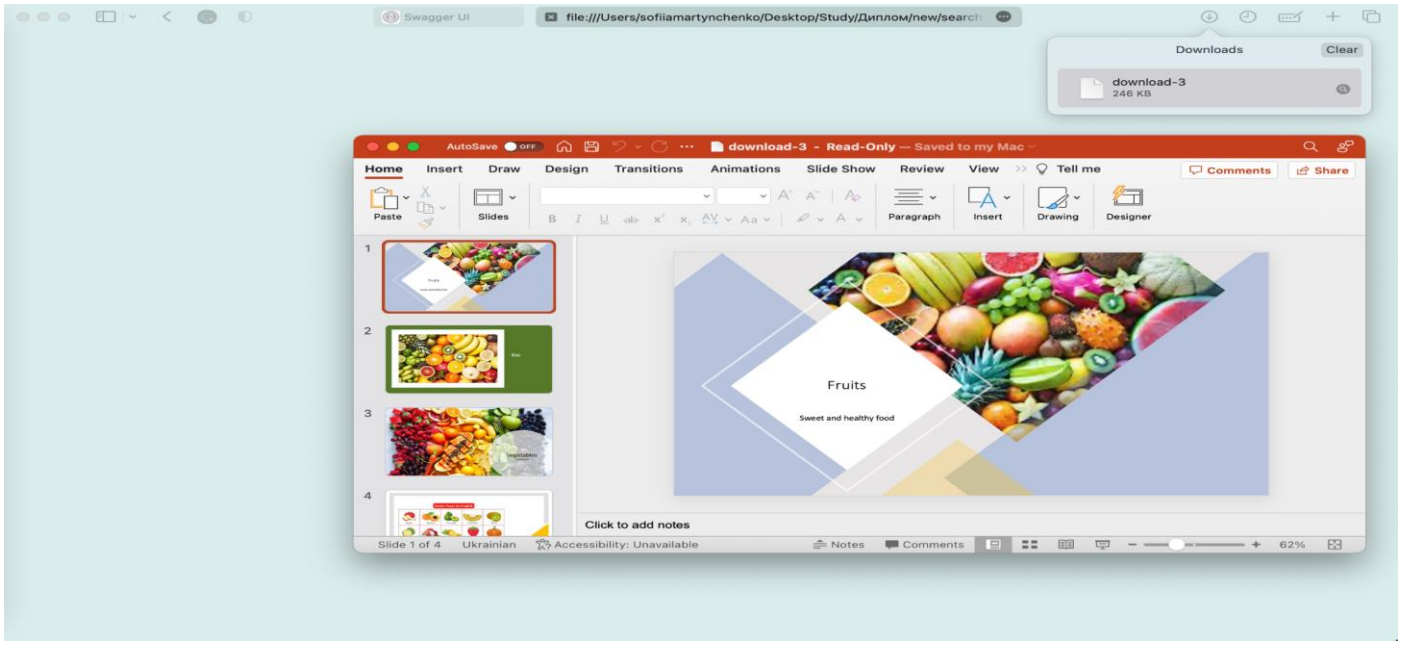


Рисунок 3.16. – Завантажена презентація

Джерело: запропоноване автором

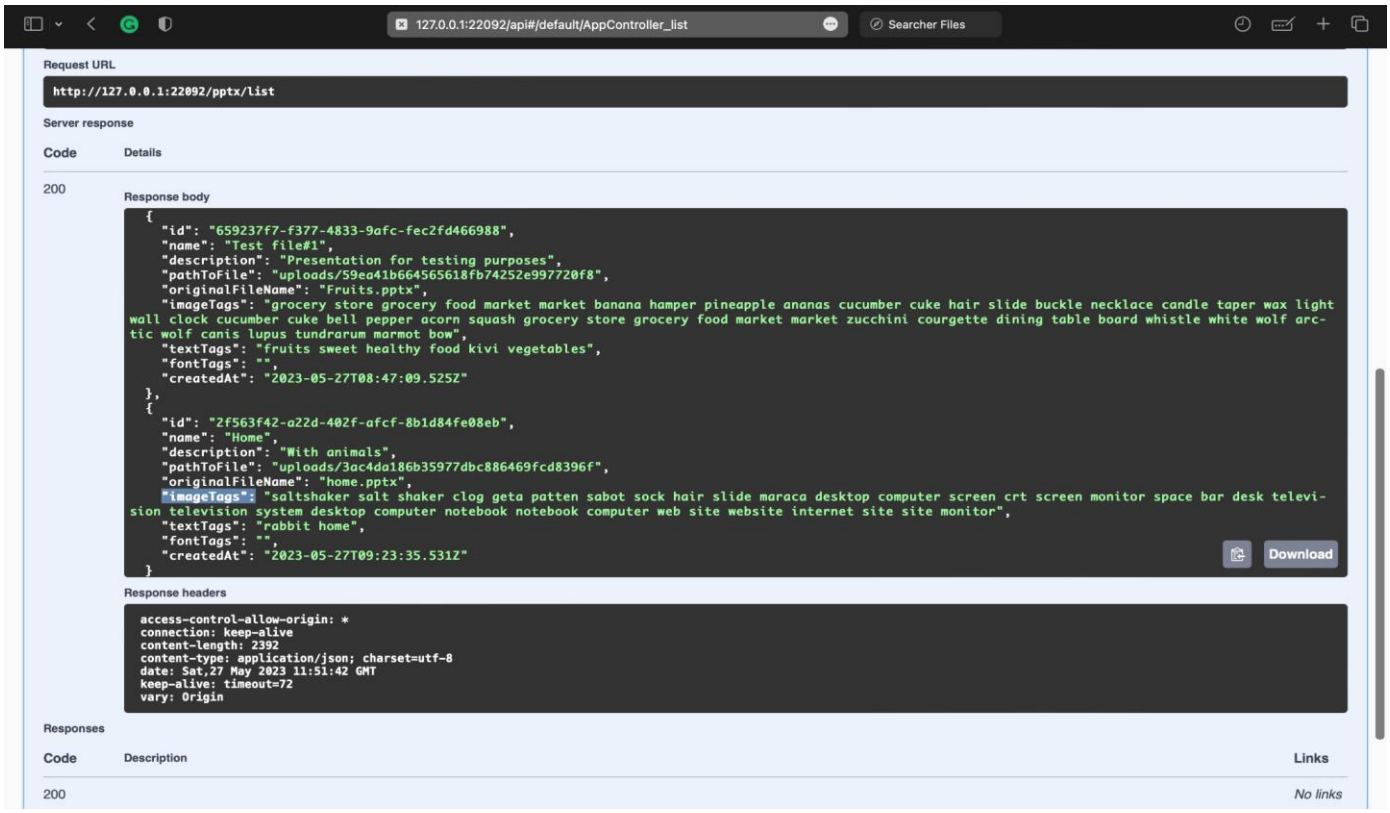


Рисунок 3.17. – Згенеровані Image Tags

Джерело: запропоноване автором

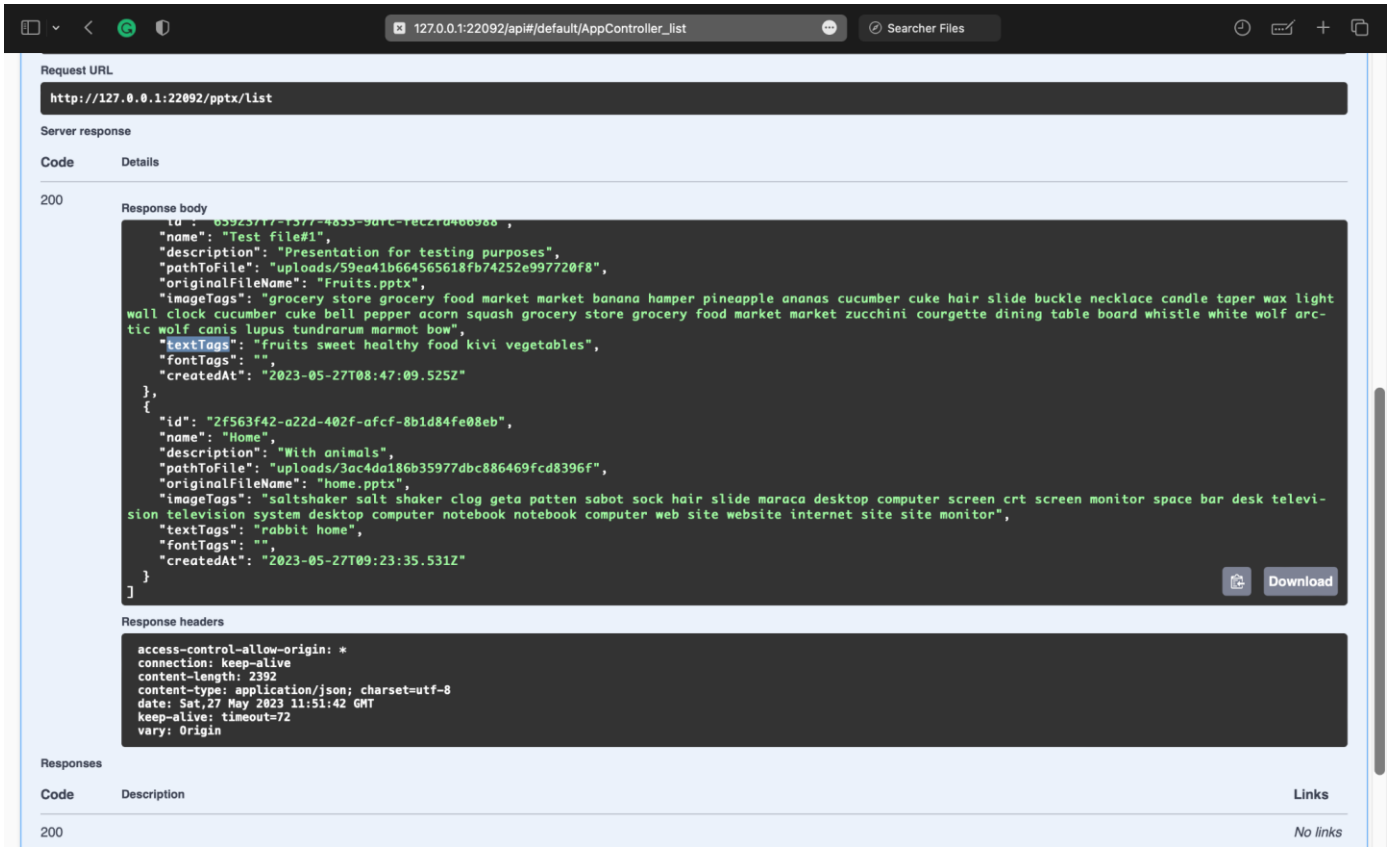


Рисунок 3.18. – Згенеровані Text Tags

Джерело: запропоноване автором

ВИСНОВКИ

Було проведено аналіз та розроблено модель для структурування даних у презентаціях. Моделі, які пройшли попереднє навчання з аналізу тексту та зображень у презентаціях, були використані для отримання інформації про метадані. Була створена програмна реалізація, що дозволяє користувачам завантажувати документи для обробки та здійснювати пошук серед оброблених документів. Результати демонструють практичну застосовність інформаційної системи.

В процесі роботи було вирішено кілька завдань, зокрема:

- Зібрано набір презентацій для аналізу та визначено формати файлів, що підтримуються системою;
- Розроблено алгоритми для вилучення метаданих з презентацій, включаючи розпізнавання тексту та зображень;
- Створено базу даних для зберігання вилучених метаданих та презентацій;
- Розроблено користувацький інтерфейс, що дозволяє користувачам швидко й легко знаходити потрібну презентацію за ключовими словами або іншими критеріями;
- Забезпечено безперебійну роботу та оновлення даних шляхом підтримки системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Unstructured Data: an overview of the data of Big Data [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/309393428_Unstructured_Data_an_overview_of_the_data_of_Big_Data.
2. Structured vs. Unstructured Data: A Comprehensive Guide. Medium. 12.11.2024. [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@redswitches/structured-vs-unstructured-data-a-comprehensive-guide-6117960b4a26.
3. Stages of data processing [Електронний ресурс] – Режим доступу до ресурсу: https://planningtank.com/computer-applications/data-processing-cycle.
4. Different types of output files [Електронний ресурс] – Режим доступу до ресурсу: https://planningtank.com/computer-applications/data-processing.
5. Мультимедийные технологии обработки информации Microsoft Power Point [Електронний ресурс] – Режим доступу до ресурсу: https://slonimsmc.grodno.by/infotex/p4aa1.html.
6. Metadat [Електронний ресурс] – Режим доступу до ресурсу: https://www.techtarget.com/whatis/definition/metadata.
7. Metadata Types [Електронний ресурс] – Режим доступу до ресурсу: https://guides.lib.utexas.edu/metadata-basics/key-concepts.
8. Мартинченко С.В. Інтелектуальні методи обробки та категоризації мультимедійного контенту презентацій: матеріали Всеукраїнської наукової конференції студентів і аспірантів, присвяченої Міжнародному дню студента – (18-22 листопада 2024 р.). – Суми, 2024. – С.423.
9. Мартинченко С.В. Виклики при обробці та управлінні неструктурованими даними: матеріали науково-практичної конференції викладачів, аспірантів та студентів Сумського НАУ – (14-18 квітня 2025 р.). – Суми, 2025. – С.306.

10. Szeliski R. Computer Vision: Algorithms and Applications / Richard Szeliski., 2022.
11. How does R-CNN work for object detection [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/how-does-r-cnn-work-for-object-detection.
12. Object Detection [Електронний ресурс] – Режим доступу до ресурсу: https://viso.ai/deep-learning/object-detection/.
13. Класифікація зображень на основі застосування автоасоціативних нейронних мереж [Електронний ресурс] – Режим доступу до ресурсу: https://journals.indexcopernicus.com/api/file/viewByFileId/590423.pdf.
14. What is Full-Text Search and How Does it Work? [Електронний ресурс] – Режим доступу до ресурсу: https://www.mongodb.com/basics/full-text-search#:~:text=Full%2Dtext%20search%20refers%20to,search%20would%20return%20exact%20matches.
15. ImageNet [Електронний ресурс] – Режим доступу до ресурсу: https://www.image-net.org/.
16. How to train CNNs on ImageNet [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/analytics-vidhya/how-to-train-a-neural-network-classifier-on-imagenet-using-tensorflow-2-ede0ea3a35ff
17. About ImageNet [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/ImageNet.
18. EfficientNet [Електронний ресурс] – Режим доступу до ресурсу: https://huggingface.co/docs/timm/models/efficientnet.
19. Keyword Extraction Methods — The Overview [Електронний ресурс] – Режим доступу до ресурсу:

https://towardsdatascience.com/keyword-extraction-methods-the-overview-35557350f8bb.

20. КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА [Електронний ресурс] – Режим доступу до ресурсу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/.
21. Node.js [Електронний ресурс] – Режим доступу до ресурсу: https://geek.justjoin.it/pierwsze-kroki-w-node-js/#Co_to_jest_i_za_co_odpowiada_Nodejs.
22. Node.js – jakie ma zalety i wady [Електронний ресурс] – Режим доступу до ресурсу: https://geek.justjoin.it/pierwsze-kroki-w-node-js/#Nodejs_%E2%80%93_jakie_ma_zalety_i_wady
23. Що такє TypeScript [Електронний ресурс] – Режим доступу до ресурсу: https://clockworkjava.pl/2019/07/co-to-typescript/.
24. TypeScript [Електронний ресурс] – Режим доступу до ресурсу: https://teamquest.pl/blog/704_typescript-czym-jest-i-dlaczego-warto-sie-go-uczyc.
25. NestJS framework do budowania wydajnych aplikacji [Електронний ресурс] – Режим доступу до ресурсу: https://boringowl.io/tag/nestjs.
26. TypeORM [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialspoint.com/typeorm/typeorm_introduction.htm.
27. Tensorflow [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/js.
28. Sentry [Електронний ресурс] – Режим доступу до ресурсу: https://boringowl.io/tag/sentry.
29. What is Git? [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlassian.com/git/tutorials/what-is-git

30. CockroachDB vs (MySQL, PostgreSQL, MongoDB & Cassandra) [Електронний ресурс] – Режим доступу до ресурсу: https://devathon.com/blog/cockroachdb-vs-mysql-vs-postgresql-vs-mongodb-vs-cassandra/.
31. Docker [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/pl-pl/cloud/learn/docker#toc-dlaczego-w-BeaJn--P.
32. Docker composer [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/ruvds/blog/450312/.
33. API Development [Електронний ресурс] – Режим доступу до ресурсу: https://swagger.io/solutions/api-development/.
34. API Developer Experience: Why it Matters, and How Documenting Your API with Swagger Can Help [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://swagger.io/blog/api-documentation/api-documentation-and-developer-experience/.

ДОДАТКИ

ДОДАТОК А.

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи

«ІНТЕЛЕКТУАЛЬНА СИСТЕМА ПОШУКУ ТА СТРУКТУРУВАННЯ ПРЕЗЕНТАЦІЙ НА ОСНОВІ МЕТАДАНИХ»

ПОГОДЖЕНО:

доцент кафедри кібернетики та інформатики

Світлана АГАДЖАНОВА

_____

Студент групи ІСТ 2101

Софія МАРТИНЧЕНКО

_____

ПЛАНУВАННЯ РОБІТ

Продуктом дипломного проекту є веб-додаток, який реалізує функціональність по витягуванню метаданих з презентацій і надає можливість користувачам здійснювати пошук презентацій за вказаним ключовим словом. Цей веб-додаток буде забезпечувати автоматичне аналізування фотографій у презентаціях і екстракцію релевантної інформації, що дозволить здійснювати ефективний пошук за ключовими словами.

Користувачі можуть завантажувати свої презентації в додаток, після чого він автоматично обробляти кожну презентацію, витягуючи метадані і зберігаючи їх у базі даних. Крім того, веб-додаток наддасть інтерфейс для пошуку презентацій за ключовими словами, дозволяючи користувачам знайти потрібну презентацію швидко та ефективно.

Результати деталізації методом SMART розміщені у табл. Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Specific (конкретна):	Розробити веб-додаток, який витягуватиме метадані з презентацій та дозволить здійснювати пошук презентацій за вказаним ключовим словом.
Measurable (вимірювана):	Визначити успішність проекту на основі оцінки замовника.
Achievable (досяжна):	Реалізація системи буде здійснюватися з використанням відповідного середовища розробки.

Relevant (релевантна):	Наявні необхідні технічні та програмні ресурси. Розробники мають достатню кваліфікацію для виконання поставлених завдань.
Time-framed (обмежена у часі):	Ціль має чітко визначений часовий рамок. Робота повинна бути завершена у визначені строки, які були погоджені замовником проекту. Проект має бути виконаний відповідно до календарного плану.

Джерело: запропоноване автором

Нижче наведена таблиця з результатами деталізації планування методом SMART для дипломного проекту, який полягає у розробці додатка для оптимізації та пошуку презентацій за словом, яке знаходиться у самій презентації.

Таблиця Б.2 − Результатами деталізації планування методом SMART

Мета	Специфічна	Міркування	Досяжна	Релевантна	Обмежена в часі
Розробити додаток	ТАК	ТАК	ТАК	ТАК	ТАК
Покращити оптимізацію пошуку	ТАК	ТАК	ТАК	ТАК	ТАК
Додаток повинен шукати за словом у презентації	ТАК	ТАК	ТАК	ТАК	ТАК
Результативність	ТАК	ТАК	ТАК	ТАК	ТАК

Джерело: запропоноване автором

У таблиці використані наступні позначення:

Специфічна: Чи чітко визначена мета проекту?

Міркування: Чи можливо виміряти прогрес або досягнення мети?

Досяжна: Чи є мета досяжною з використанням наявних ресурсів?

Релевантна: Чи пов'язана мета проекту з загальними цілями організації або проекту?

Обмежена в часі: Чи встановлені конкретні терміни досягнення мети?

Згідно з результатами деталізації планування методом SMART, мети проекту є специфічними, вимірюваними, досяжними, релевантними та обмеженими в часі. Це допоможе забезпечити чіткість, вимірюваність та реалізованість проекту, спрямованого на розробку додатка для оптимізації та пошуку презентацій за словом, яке міститься у презентаціях.

Для планування змісту структури робіт використовується WBS (Work Breakdown Structure) діаграма, яка графічно відображає згруповані елементи проекту у вигляді пакетів робіт. Ці пакети робіт ієрархічно пов'язані з продуктом проекту. Ми побудуємо структуру WBS, детально описавши роботи, що потрібно виконати на кожному етапі створення проекту. Ми також здійснимо декомпозицію робіт для цього проекту. На рисунку Б.1 зображена WBS діаграма.

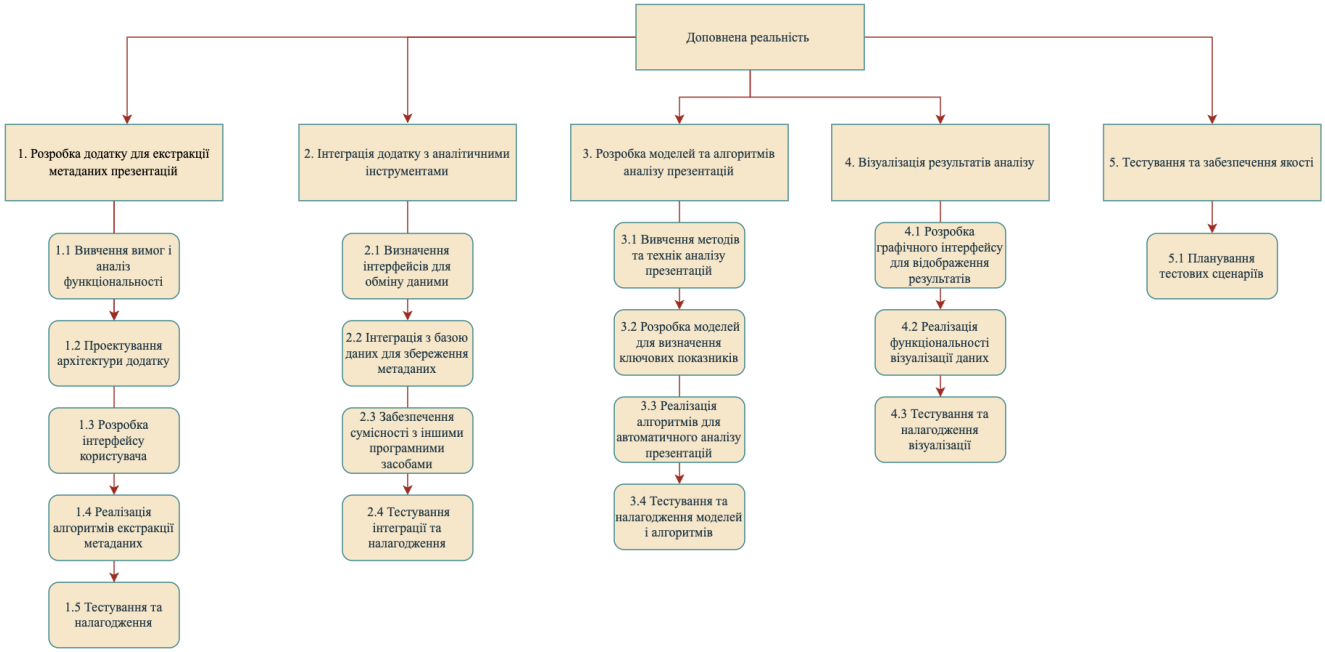


Рисунок Б.1 − WBS. Структура робіт проекту

Джерело: запропоноване автором

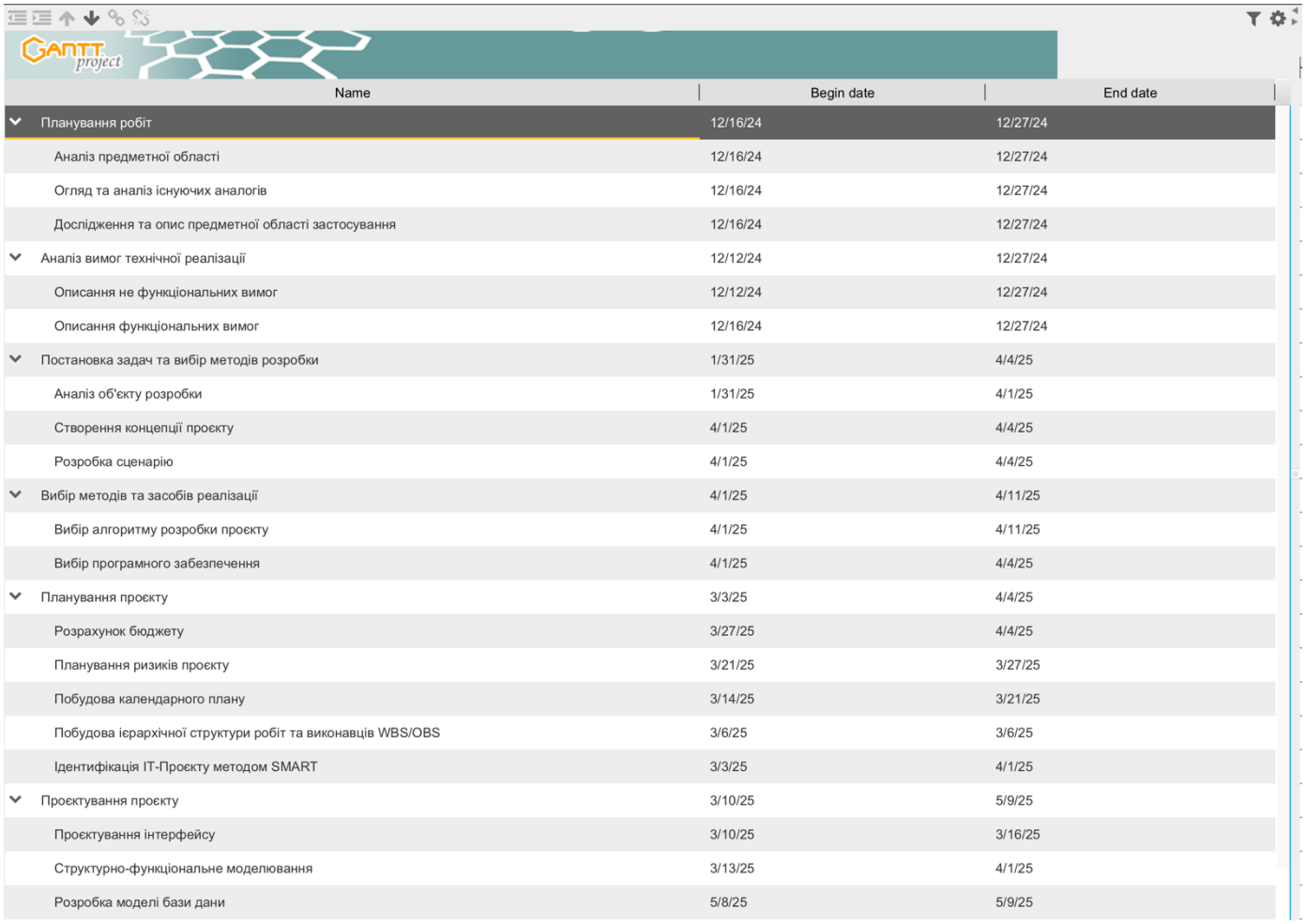
Планування структури організації, для впровадження готового проекту (OBS). Діаграма OBS зображена на рис. Б.2.

Рисунок Б.2 − Організаційна структура проекту (OBS)

Джерело: запропоноване автором

Після побудови діаграми Ганта, ми перейдемо до створення календарного плану для виконання дипломного проекту. Діаграма Ганта є найпоширенішим форматом графіка в будь-якій галузі. Вона надає менеджерам проекту і всій команді розробників можливість візуалізувати графіки часу і зв'язок між окремими завданнями та етапами проекту. Тривалість робіт вказана у днях, але фактична тривалість виконання завдань приблизно складає 3-4 години на день. Щоб отримати реалістичне уявлення про тривалість виконання робіт з урахуванням обмежень у використанні ресурсів, вихідних та святкових днів, ми побудували календарний графік.

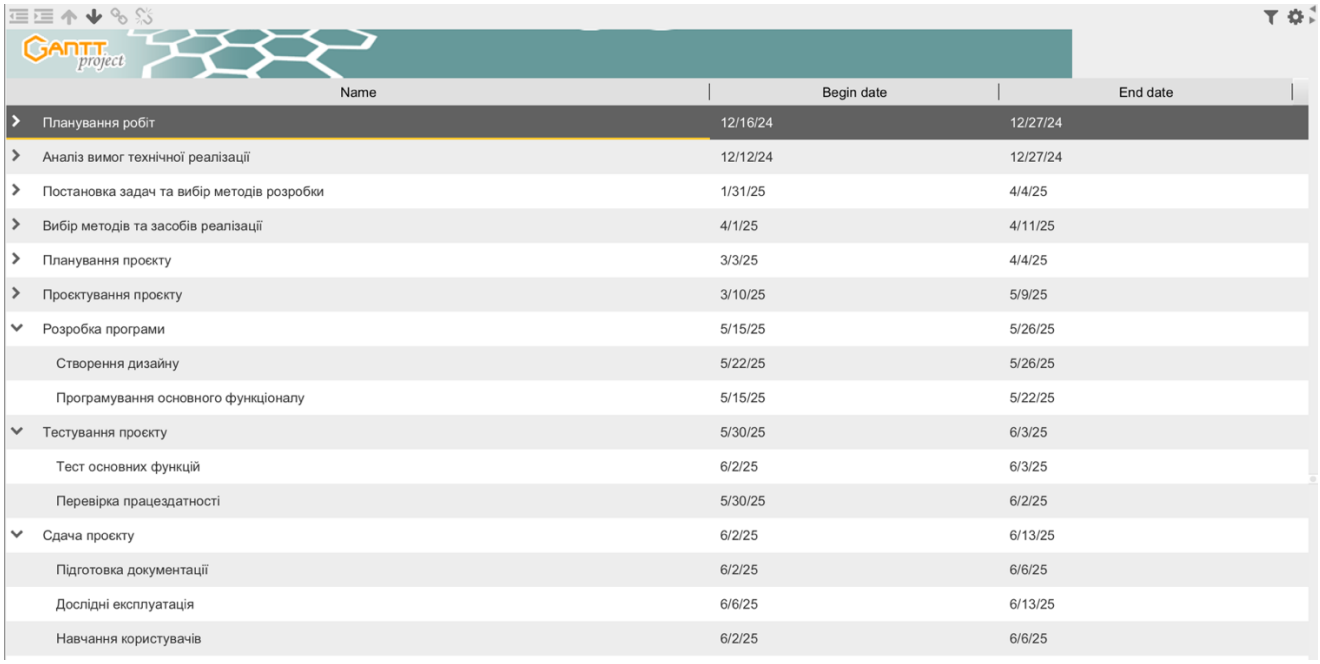
Діаграма Ганта та список робіт діаграми Ганта зображені на рис. Б.3 - Б.9.



Name	Begin date	End date
Планування робіт	12/16/24	12/27/24
Аналіз предметної області	12/16/24	12/27/24
Огляд та аналіз існуючих аналогів	12/16/24	12/27/24
Дослідження та опис предметної області застосування	12/16/24	12/27/24
Аналіз вимог технічної реалізації	12/12/24	12/27/24
Описання не функціональних вимог	12/12/24	12/27/24
Описання функціональних вимог	12/16/24	12/27/24
Постановка задач та вибір методів розробки	1/31/25	4/4/25
Аналіз об'єкту розробки	1/31/25	4/1/25
Створення концепції проєкту	4/1/25	4/4/25
Розробка сценарію	4/1/25	4/4/25
Вибір методів та засобів реалізації	4/1/25	4/11/25
Вибір алгоритму розробки проєкту	4/1/25	4/11/25
Вибір програмного забезпечення	4/1/25	4/4/25
Планування проєкту	3/3/25	4/4/25
Розрахунок бюджету	3/27/25	4/4/25
Планування ризиків проєкту	3/21/25	3/27/25
Побудова календарного плану	3/14/25	3/21/25
Побудова ієрархічної структури робіт та виконавців WBS/OBS	3/6/25	3/6/25
Ідентифікація ІТ-Проєкту методом SMART	3/3/25	4/1/25
Проєктування проєкту	3/10/25	5/9/25
Проєктування інтерфейсу	3/10/25	3/16/25
Структурно-функціональне моделювання	3/13/25	4/1/25
Розробка моделі бази дани	5/8/25	5/9/25

Рисунок Б.3 – Список робіт для побудови діаграми Ганта

Джерело: запропоноване автором



Name	Begin date	End date
Планування робіт	12/16/24	12/27/24
Аналіз вимог технічної реалізації	12/12/24	12/27/24
Постановка задач та вибір методів розробки	1/31/25	4/4/25
Вибір методів та засобів реалізації	4/1/25	4/11/25
Планування проєкту	3/3/25	4/4/25
Проєктування проєкту	3/10/25	5/9/25
Розробка програми	5/15/25	5/26/25
Створення дизайну	5/22/25	5/26/25
Програмування основного функціоналу	5/15/25	5/22/25
Тестування проєкту	5/30/25	6/3/25
Тест основних функцій	6/2/25	6/3/25
Перевірка працездатності	5/30/25	6/2/25
Сдача проєкту	6/2/25	6/13/25
Підготовка документації	6/2/25	6/6/25
Дослідні експлуатація	6/6/25	6/13/25
Навчання користувачів	6/2/25	6/6/25

Рисунок Б.4 – Список робіт для побудови діаграми Ганта

Джерело: запропоноване автором

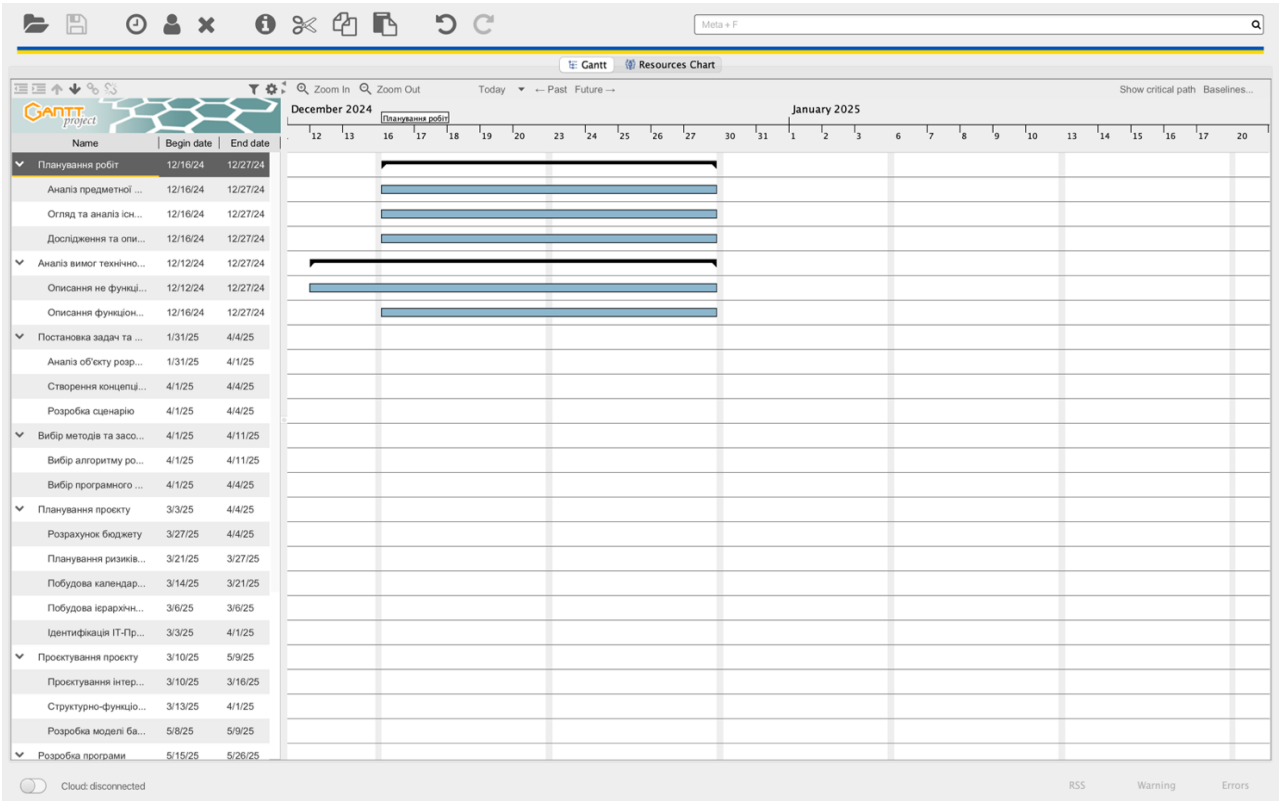


Рисунок Б.5 – Продовження діаграми Ганта

Джерело: запропоноване автором

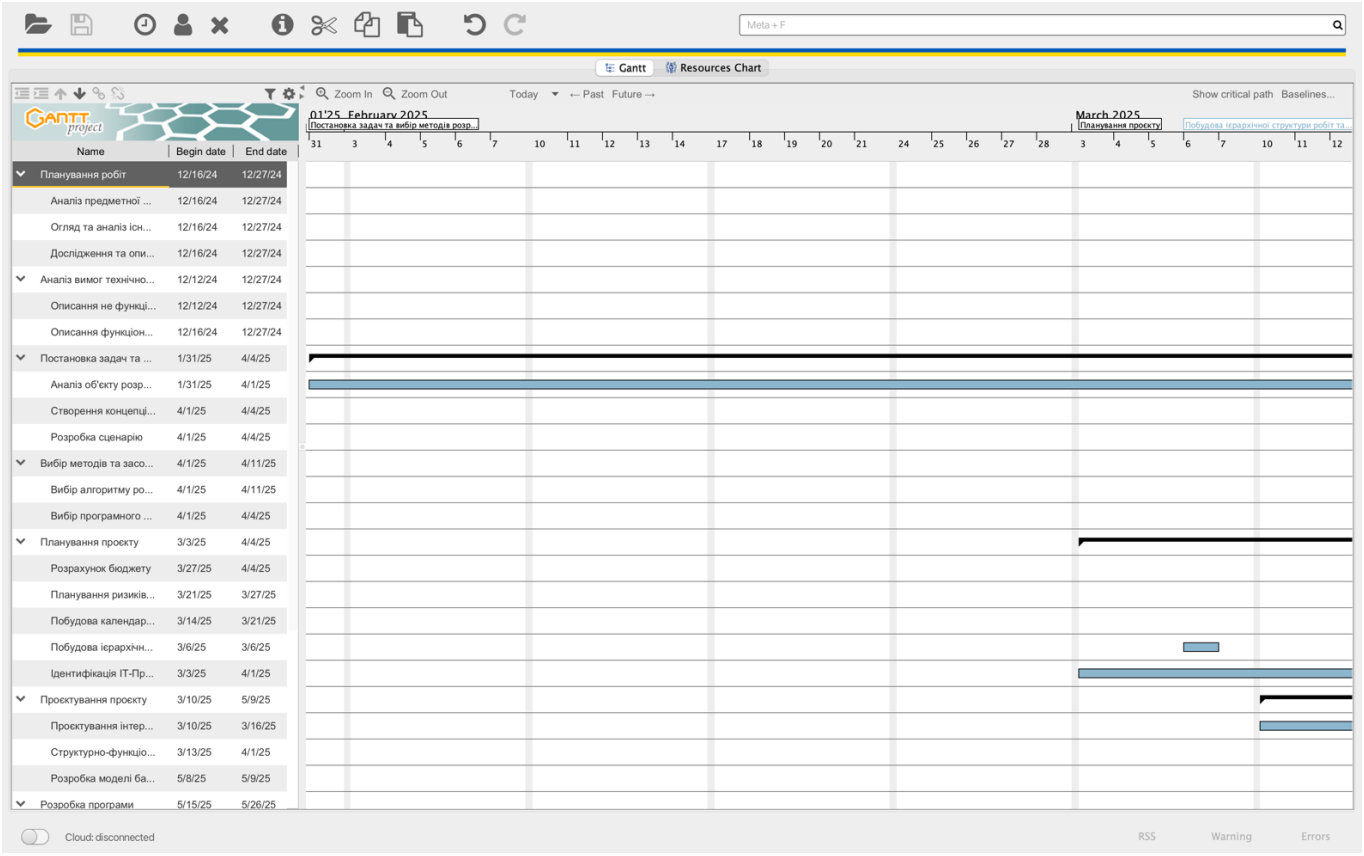


Рисунок Б.6 – Продовження діаграми Ганта

Джерело: запропоноване автором

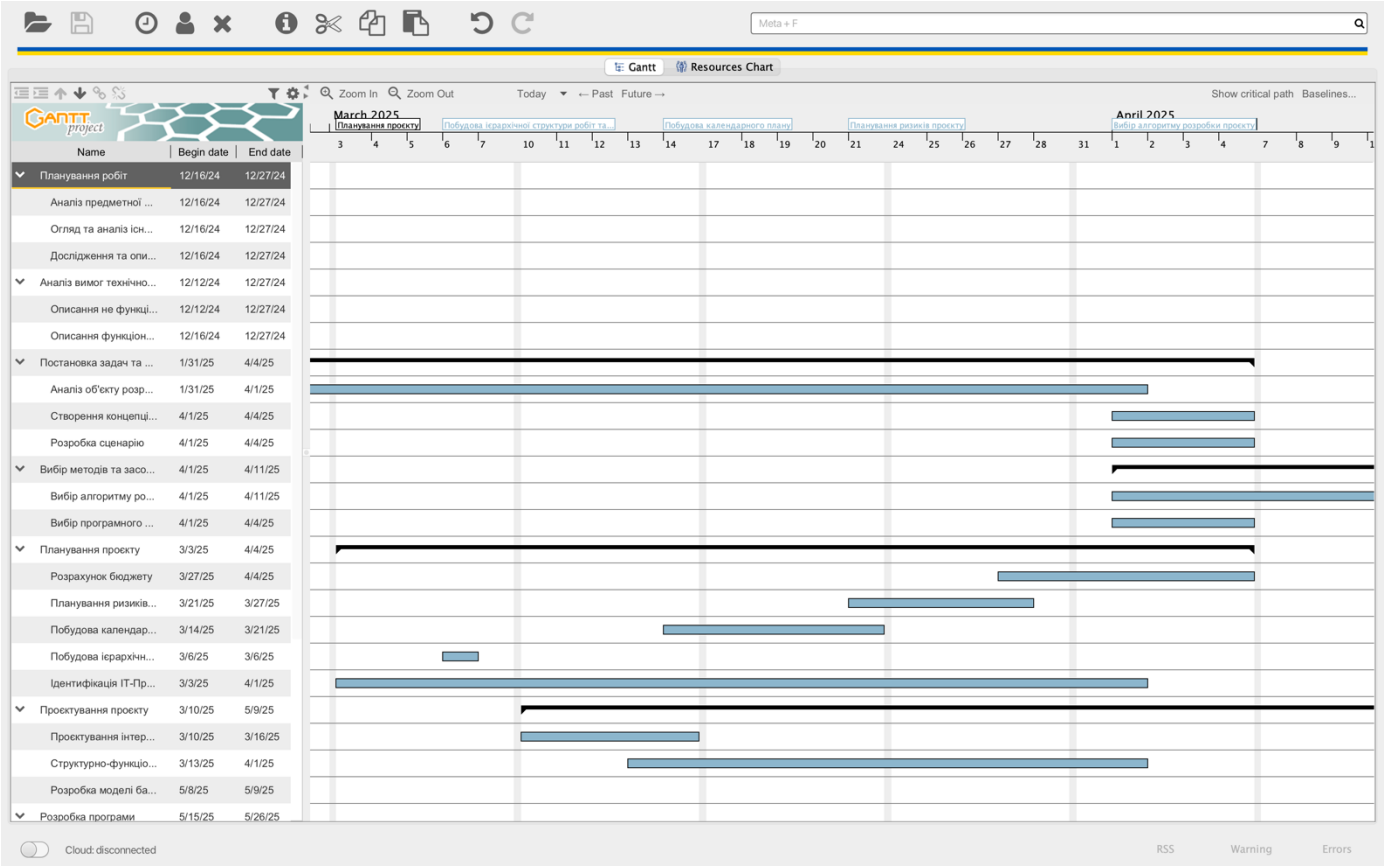


Рисунок Б.7 – Продовження діаграми Ганта

Джерело: запропоноване автором

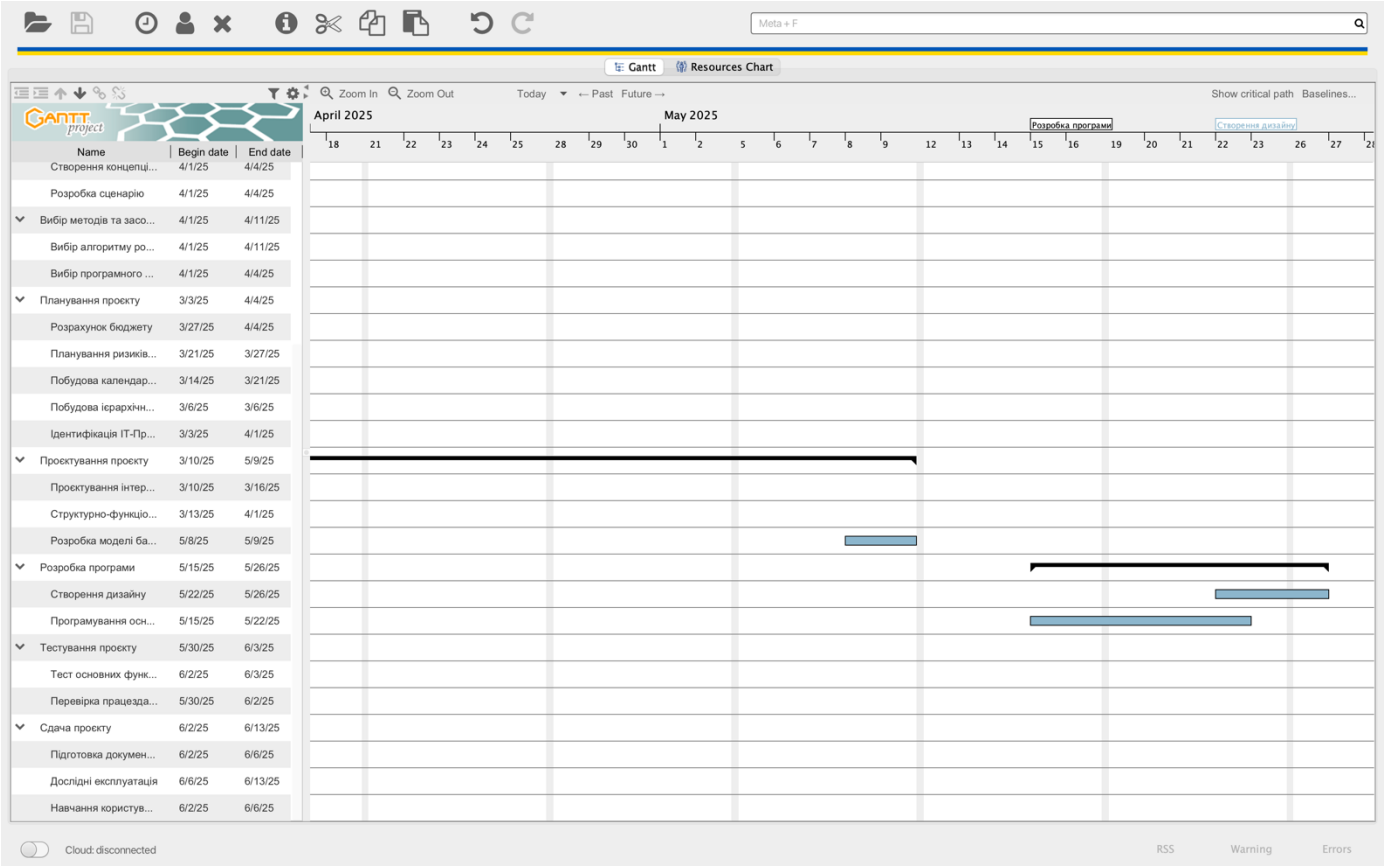


Рисунок Б.8 – Продовження діаграми Ганта

Джерело: запропоноване автором

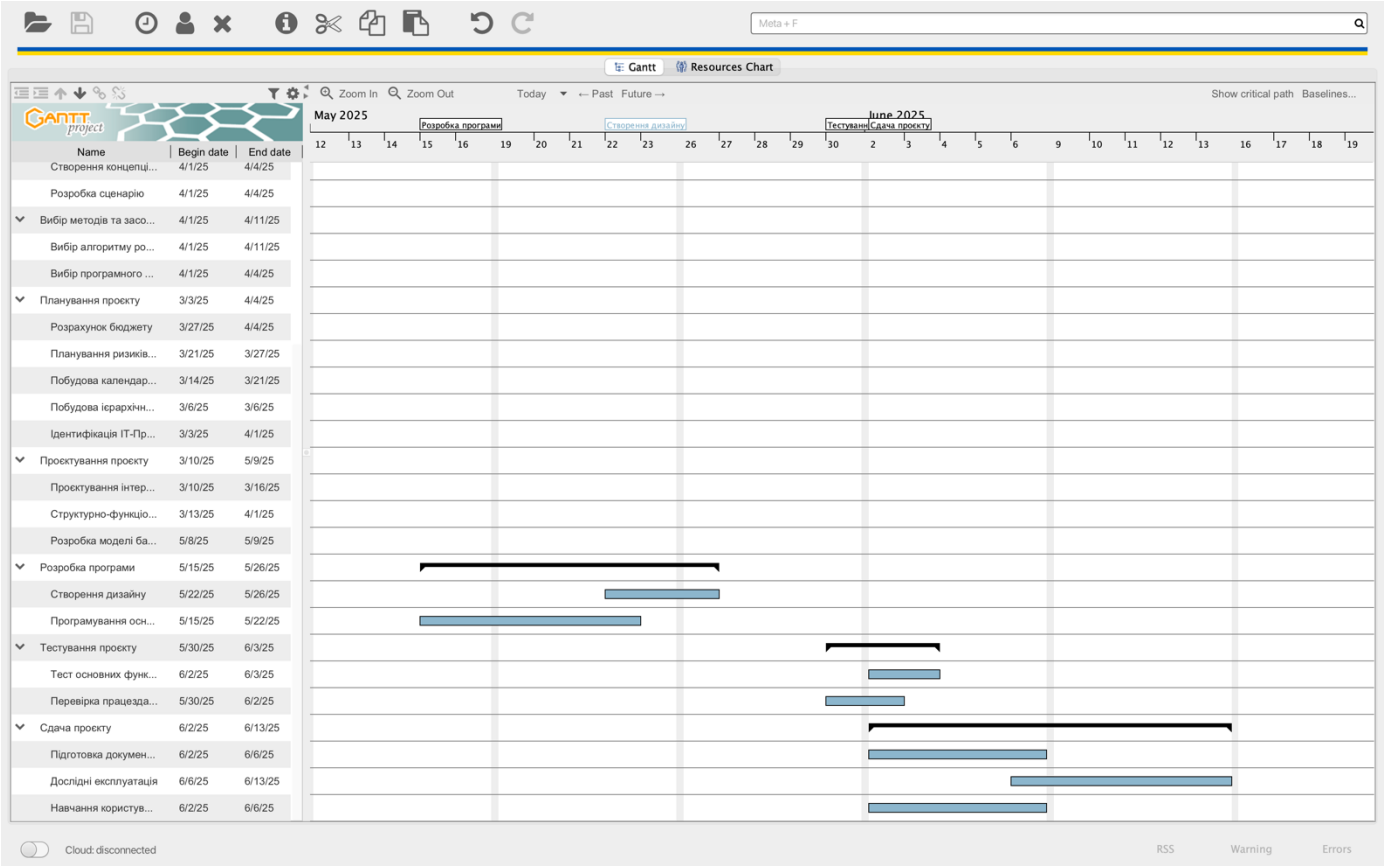


Рисунок Б.9 – Продовження діаграми Ганта

Джерело: запропоноване автором

ДОДАТОК Б.

ЛІСТНИГИ ПРОГРАМ

Лістинг основних файлів керування роботою додатку:

Файл: image.context.extractor.service.ts

```
import { Injectable, OnModuleInit } from '@nestjs/common';
import {
 EfficientNetCheckPoint,
 EfficientNetCheckPointFactory,
 EfficientNetModel,
} from 'node-efficientnet';

@Injectable()
export class ImageContextExtractorService implements OnModuleInit {
 private readonly models: EfficientNetModel[] = [];

 async extractTags(pathToImage: string) {
  const resultSet = new Set();
  for (const model of this.models) {
   const resultOfAnalyze = (
    await model.inference(pathToImage, {
     topK: 5,
    })
   ).result.map((i) => i.label);
   resultOfAnalyze.forEach((res) => {
    res.split(',').forEach((i) => resultSet.add(i.toLowerCase().trim()));
```

```
   });
  }
  const result = [];
  resultSet.forEach((i) => result.push(i));
  return result;
 }
 async onModuleInit() {
  const list = [
   EfficientNetCheckPoint.B1,
   EfficientNetCheckPoint.B2,
   EfficientNetCheckPoint.B3,
   EfficientNetCheckPoint.B4,
   EfficientNetCheckPoint.B5,
   EfficientNetCheckPoint.B6,
   EfficientNetCheckPoint.B7,
  ];
  for (const modelType of list) {
   const model = await EfficientNetCheckPointFactory.create(modelType);
   this.models.push(model);
  }
 }
}
```

Файл: list.service.ts

```
import { Injectable } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { PptxEntity } from '../entities/pptx.entity';
import { Like, Repository } from 'typeorm';
```

```
@Injectable()
export class ListService {
 constructor(
  @InjectRepository(PptxEntity)
  private readonly pptxEntityRepository: Repository<PptxEntity>,
 ) {}

 list() {
  return this.pptxEntityRepository.find();
 }

 search(key: string) {
  const filter = `%${key}%`;
  return this.pptxEntityRepository.find({
   where: [
    {
     name: Like(filter),
    },
    {
     description: Like(filter),
    },
    {
     imageTags: Like(filter),
    },
    {
     textTags: Like(filter),
    },
```

```
    {
     fontTags: Like(filter),
    },
    {
     originalFileName: Like(filter),
    },
   ],
  });
 }
}
```

Файл: upload.service.ts

```
import { Injectable, Logger } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { PptxEntity } from '../entities/pptx.entity';
import { ImageContextExtractorService } from './image.context.extractor.service';
import * as decompress from 'decompress';
import { copyFileSync, existsSync, readdirSync } from 'fs';
import * as keyword_extractor from 'keyword-extractor';
import * as officeParser from 'officeparser';

@Injectable()
export class UploadService {
 private readonly logger = new Logger(UploadService.name);

 constructor(
  @InjectRepository(PptxEntity)
  private readonly pptxEntityRepository: Repository<PptxEntity>,
  private readonly imageContextExtractorService: ImageContextExtractorService,
 ) {}

 async upload(
  name: string,
  pathToFile: string,
  originalFileName: string,
  description: string,
 ) {
  const distFolder = pathToFile + '_dist';
```

```
const imgDistFolder = distFolder + '/ppt/media';
const fontsDistFolder = distFolder + '/ppt/fonts';
await decompress(pathToFile, distFolder);

this.logger.log(`Text analyzing is started`);
copyFileSync(pathToFile, pathToFile + '.pptx');
const text = await officeParser.parseOfficeAsync(pathToFile + '.pptx');
// @ts-ignore
const textTags = keyword_extractor.extract(text, {
 remove_digits: true,
 return_changed_case: true,
 remove_duplicates: false,
});
this.logger.log(`Text analyzing is finished`);

this.logger.log(`Image analyzing is started`);
const imageTags = [];
if (existsSync(imgDistFolder)) {
 for (const fileName of readdirSync(imgDistFolder)) {
  imageTags.push(
   ...(await this.imageContextExtractorService.extractTags(
    imgDistFolder + '/' + fileName,
   )),
  );
 }
}
this.logger.log(`Image analyzing is finished`);
```

```
  this.logger.log(`Font analyzing is started`);
  const fontTags = [];
  if (existsSync(fontsDistFolder)) {
   for (const fileName of readdirSync(fontsDistFolder)) {
    fontTags.push(fileName.toLowerCase().replace('.fntdata', ''));
   }
  }
  this.logger.log(`Image analyzing is finished`);

  await this.pptxEntityRepository.save({
   name,
   originalFileName,
   description,
   pathToFile,
   imageTags: imageTags.join(' '),
   fontTags: fontTags.join(' '),
   textTags: textTags.join(' '),
  } as PptxEntity);
 }
}
```

Файл: search.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Search Presentations</title>
  <link rel="stylesheet" href="style.css">
</head>

<body>
  <div class="container">
    <header>
      <h1>Search In Presentations</h1>
      <nav>
        <a href="File uploader.html">Click here to upload new file</a>
      </nav>
    </header>
    <main>
      <form id="searchForm">
        <label for="searchQuery" class="search-label">Enter text to search</label>
        <input type="text" id="searchQuery" class="search-input" placeholder="Type
keyword or phrase">
        <button type="button" class="search-btn">Search</button>
      </form>
      <div id="results" class="results">
        Results will appear here after the search...
```

```
    </div>
    <!-- Information about technology -->
    <section class="info-section">
      <h2>More information about tool</h2>
      <p>
        Our tool is designed to help you quickly and easily find the information you need in your
        presentations. Whether you're looking for specific text or analyzing images within your slides, our
        advanced
        technology makes the process simple and efficient. We use metadata extraction technology to analyze
        your files.
        This means we can identify and structure the key information from your presentations, saving you
        time and effort. With our tool, managing and searching your content has never been easier.
      </p>
    </section>
  </main>
  <footer>
    <p>Created for a diploma thesis by Sofiia Martynchenko</p>
  </footer>
 </div>
</body>

</html>
```

Файл: upload.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Upload File</title>
  <link rel="stylesheet" href="style.css">
</head>

<body>
  <div class="container">
    <header>
      <h1>Upload Your Presentation</h1>

    </header>
    <main>
      <form id="uploadForm">
        <label for="fileUpload" class="file-label">Choose a PPTX file to upload</label>
        <input type="file" id="fileUpload" class="file-input" accept=".pptx">
        <button type="button" class="upload-btn">Upload</button>
      </form>
      <div id="status" class="status">
        Upload status will appear here...
      </div>
      <!--Information about technology -->
      <section class="info-section">
```

```
        <h2>More information about our tool</h2>
        <p>
          Our tool is designed to help you quickly and easily find the information you need in your
          presentations. Whether you're looking for specific text or analyzing images within your slides, our
          advanced
          technology makes the process simple and efficient. We use metadata extraction technology to analyze
          your files.
          This means we can identify and structure the key information from your presentations, saving you
          time and effort. With our tool, managing and searching your content has never been easier.
        </p>
      </section>
      <nav>
        <a href="Search Presentations.html">Click here to start searching</a>
      </nav>
    </main>
    <footer>
      <p>Created for a diploma thesis by Sofiia Martynchenko</p>
    </footer>
  </div>
  <script src="script.js"></script>
</body>

</html>
```

Файл: script.js

```
// Upload file validation
document.querySelector('.upload-btn').addEventListener('click', function () {
  const fileInput = document.getElementById('fileUpload');
  const statusDiv = document.getElementById('status');

  if (!fileInput.files.length) {
    statusDiv.textContent = 'Please select a file to upload.';
    statusDiv.style.color = 'red';
    return;
  }

  const file = fileInput.files[0];
  const fileExtension = file.name.split('.').pop().toLowerCase();

  if (fileExtension !== 'pptx') {
    statusDiv.textContent = 'Only .pptx files are allowed.';
    statusDiv.style.color = 'red';
    return;
  }

  statusDiv.textContent = `File "${file.name}" is valid and ready to upload.`;
  statusDiv.style.color = 'green';
  // uploadFile(file);
});

// Search input validation
```

```
document.querySelector('.search-btn').addEventListener('click', function () {
  const searchInput = document.getElementById('searchQuery');
  const resultsDiv = document.getElementById('results');

  if (searchInput.value.trim() === '') {
    resultsDiv.textContent = 'Please enter a keyword or phrase to start the search.';
    resultsDiv.style.color = 'red';
    return;
  }

  resultsDiv.textContent = `Searching for: "${searchInput.value}"...`;
  resultsDiv.style.color = 'green';
  // performSearch(searchInput.value);
});
```

Файл: script.js

```
/* style.css */

body {
  font-family: 'Roboto', sans-serif;
  margin: 0;
  padding: 0;
  background: linear-gradient(120deg, #84f2fa, #8fd3f4);
  color: #444;
  min-height: 100vh;
}
```

```
.container {
  width: 90%;
  max-width: 600px;
  margin: 40px auto;
  padding: 60px;
  background-color: #ffffff;
  border-radius: 12px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

header {
  text-align: center;
  margin-bottom: 20px;
}

header h1 {
  font-size: 28px;
  color: #0078ff;
}

nav a {
  text-decoration: none;
  font-weight: bold;
  color: #2a98d4;
  margin: 10px 0;
  display: inline-block;
}
```

```
nav a:hover {
  text-decoration: underline;
}


nav a:hover {
  text-decoration: underline;
}

form {
  display: flex;
  flex-direction: column;
  gap: 12px;
  margin-bottom: 20px;
}

.file-label, .search-label {
  font-size: 20px;
  font-weight: bold;
  color: #333;
}

.file-input, .search-input {
  padding: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 6px;
  outline: none;
```

```
  transition: box-shadow 0.3s ease;
}

.file-input:focus, .search-input:focus {
  box-shadow: 0 0 5px #0078ff;
  border-color: #0078ff;
}

.upload-btn, .search-btn {
  padding: 12px;
  font-size: 18px;
  background-color: #0078ff;
  color: #fff;
  border: none;
  border-radius: 6px;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.upload-btn:hover, .search-btn:hover {
  background-color: #005bbb;
}

.status, .results {
  padding: 15px;
  border: 1px solid #ccc;
  border-radius: 6px;
  background-color: #f9f9f9;
```

```
  margin-top: 20px;
}

.status {
  color: #444;
  font-style: italic;
}

.results ul {
  padding-left: 20px;
}

.results li {
  margin: 5px 0;
  list-style-type: disc;
  font-size: 16px;
}

/* Module with information about technology */
.info-section {
  margin-top: 30px;
  padding: 15px;
  border: 1px solid #ddd;
  border-radius: 8px;
  background-color: #f9f9f9;
}

.info-section h2 {
```

```
  font-size: 20px;
  margin-bottom: 10px;
  color: #007bff;
}

.info-section p {
  font-size: 16px;
  line-height: 1.6;
  color: #555;
}

/* Footer */
footer {
  margin-top: 40px;
  padding: 15px;
  text-align: center;
  font-size: 14px;
  color: #999;
  border-top: 1px solid #ddd;
  background-color: #f4f4f4;
}
```

ДОДАТОК С

КОПІЇ ПУБЛІКАЦІЙ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

МАТЕРІАЛИ
ВСЕУКРАЇНСЬКОЇ НАУКОВОЇ КОНФЕРЕНЦІЇ СТУДЕНТІВ ТА АСПІРАНТІВ, ПРИСВЯЧЕНОЇ МІЖНАРОДНОМУ ДНЮ СТУДЕНТА

(18-22 листопада 2024 р., м. Суми)

ІНТЕЛЕКТУАЛЬНІ МЕТОДИ ОБРОБКИ ТА КАТЕГОРИЗАЦІЇ МУЛЬТИМЕДІЙНОГО КОНТЕНТУ ПРЕЗЕНТАЦІЙ

Мартинченко С.В., студ., 4 курс ФЕІМ, спец. «Інформаційні системи та технології»
Науковий керівник: к.т.н., доц. С.В. Агаджанова С.В.
Сумський НАУ

В епоху інформаційних технологій обсяг цифрового контенту постійно збільшується, що створює нові перешкоди для його ефективного опрацювання та використання. Проблема управління знаннями, особливо тими, що містяться в презентаційних матеріалах, є важливим джерелом інформації для бізнесу та освіти. Презентації є складними документами, які містять текстові, графічні та мультимедійні елементи, і їх автоматизація вимагає спеціальних методів.

Сучасні методи штучного інтелекту відкривають нові можливості для створення систем, здатних автоматично аналізувати та категоризувати контент презентацій. Ключовим викликом залишається розробка ефективних алгоритмів для автоматизованого вилучення та структурування метаданих з презентаційних матеріалів, що дозволить оптимізувати процеси пошуку та використання накопичених знань [1].

Запропонована система базується на комплексному підході, який включає:

1. Розробку багаторівневої архітектури, яка розділяє презентаційні матеріали на основні компоненти.
2. Аналіз зображень за допомогою нейронної мережі.
3. Створити унікальну базу метаданих, яка використовує оптимізовану структуру зберігання.
4. Використання бібліотек для машинного навчання, таких як Keras та TensorFlow, для побудови моделей обробки даних.

Python — це основний інструмент для виконання завдань, який надає доступ до бібліотек і датасетів, таких як ImageNet, для навчання згорткових нейронних мереж. Такі технології полегшують оптимізацію пошуку та аналізу неструктурованих даних, що особливо важливо для бізнесу та навчальних закладів.

Впровадження інтелектуальних методів екстракції та аналізу метаданих презентаційних матеріалів суттєво підвищує точність та ефективність процесів пошуку й обробки інформації. Розроблена система спирається на комплексний підхід до аналізу різнорідних компонентів презентацій, використовуючи передові алгоритми машинного навчання для розпізнавання взаємозв'язків між текстовими та візуальними елементами. В основі системи лежить згорткова нейронна мережа архітектури VGG16, яка забезпечує високу точність класифікації та розпізнавання візуальних елементів презентацій.

Особливу увагу в розробленій системі приділено інтеграції технологій обробки природної мови (NLP) з алгоритмами глибинного навчання. Такий синергетичний підхід дозволяє здійснювати комплексний аналіз текстового контенту на різних рівнях абстракції, включаючи морфологічний, синтаксичний та семантичний аналіз. Система автоматично виділяє ключові терміни, визначає тематичну спрямованість контенту та будує семантичні мережі, що відображають концептуальні зв'язки між елементами презентацій.

Система екстракції та аналізу метаданих з презентацій забезпечує ефективну обробку та структурування неструктурованих даних. Створення масштабованої архітектури, здатної ефективно обробляти великі обсяги презентаційних матеріалів, стало результатом успішного поєднання технологій глибинного навчання з методами обробки природної мови.

Список використаних джерел

1. An Artificial Intelligence Driven Multi-Feature Extraction Scheme for Big Data [Електронний ресурс] // IEEE Access. – 2019. – Режим доступу до ресурсу: https://www.researchgate.net/publication/334005616_An_Artificial_Intelligence_Driven_Multi-Feature_Extraction_Scheme_for_Big_Data_Detection

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ НАЦІОНАЛЬНИЙ
АГРАРНИЙ УНІВЕРСИТЕТ

МАТЕРІАЛИ

науково-практичної конференції
викладачів, аспірантів та студентів
Сумського НАУ

(14-18 квітня 2025 р.)

ВИКЛИКИ ПРИ ОБРОБЦІ ТА УПРАВЛІННІ НЕСТРУКТУРОВАНИМИ ДАНИМИ

Мартинченко С.В, студ. 4 курсу ФЕІМ, спец. «Інформаційні системи та технології»
Науковий керівник: к.т.н., доц. С.В. Агаджанова
Сумський НАУ

У сучасному цифровому світі робота з масштабними даними, також відомими як Big Data, є важливою для розвитку різноманітних організацій. Збільшення кількості даних, які надходять від сенсорних систем, пристроїв Інтернету речей і соціальних мереж, створює нові можливості для отримання важливих бізнес-інсайдів і покращення процесів прийняття рішень.

Однак адаптація та розширення систем обробки даних викликає низку технічних проблем:

1. Інфраструктурні обмеження – розробка систем обробки даних вимагає постійного збільшення обчислювальної потужності. Зростання обсягів даних викликає зростання вимог до продуктивності системи, що призводить до значного збільшення операційних витрат для компаній, які купують нове обладнання та розширюють інфраструктуру.
2. Керування ресурсами пам'яті – ефективне управління пам'яттю особливо важливе для стабільної роботи систем, що працюють з великими обсягами даних. Неправильний розподіл ресурсів може призвести до втрати продуктивності, зависань систем і порушень у роботі програмного забезпечення, оскільки обробка великих даних вимагає значного використання як постійної, так і оперативної пам'яті. Щоб уникнути цих проблем, необхідно впроваджувати рішення для використання пам'яті, які забезпечують баланс між продуктивністю та стабільністю систем.
3. Складність побудови архітектури – одним із найважливіших завдань у сфері Big Data є проєктування архітектури для зберігання, агрегації та аналізу великих обсягів даних. Побудова ефективних систем зберігання може бути складною через відсутність універсальних рішень та специфіку кожного проєкту. Це може спричинити проблеми з об'єднанням даних та швидкістю доступу до них.
4. Безпека інформації – збільшення кількості чутливих даних підвищує ймовірність несанкціонованого доступу до них. Захист даних є життєво важливим, особливо для компаній, що обробляють конфіденційні дані, оскільки витік таких даних може призвести до втрати репутації та серйозних наслідків.
5. Пропускна здатність мережі – у розподілених системах вона може стати серйозним обмеженням через низьку пропускну здатність мережі. Такі системи потребують високошвидкісної передачі даних між вузлами, щоб мінімізувати затримки в процесах обробки та аналізу. Проте, значні затримки можуть виникнути через обмеження пропускної здатності, особливо коли йдеться про обробку даних у режимі реального часу.
6. Обробка неструктурованих даних – робота з неструктурованими даними, які не піддаються традиційним схемам зберігання та аналізу, є однією з основних проблем. Це можуть бути текстові повідомлення, зображення, відео, звукові записи та інші неструктуровані дані. Хоча використання методів глибокого навчання та нейронних мереж дозволяє автоматизувати процеси обробки таких даних, перетворення неструктурованих даних у корисну інформацію вимагає значних обчислювальних ресурсів і складних алгоритмів. Тому, розробка ефективних алгоритмів для роботи з різними типами неструктурованих даних є важливим викликом. Це дозволяє значно покращити процеси аналізу та інтеграції таких даних з іншими даними.
7. Сумісність та стандартизація – ще одна проблема, з якою стикаються різні системи обробки даних. Багато різних інструментів, технологій і платформ, які використовуються для роботи з великими даними, можуть зробити інтеграцію та обмін даними між різними системами складними. Для вирішення цієї проблеми важливо створити відкриті стандарти та інтерфейси, щоб забезпечити безшовну взаємодію різних технологій і зменшити технічні перешкоди під час обробки великих даних.
8. Аналіз даних і прийняття рішень – необхідність швидкого прийняття рішень на основі результатів аналізу великих обсягів даних часто робить його складним. Виявлення важливих закономірностей у великих обсягах інформації та їх використання для стратегічних рішень є складним процесом, що вимагає ефективних алгоритмів і здатності системи визначати відповідні патерни. Для підвищення точності таких рішень важливо застосовувати рекомендації та методи прогнозування, які спрощують роботу з великими кількостями даних [1].

Використання новітніх технологій машинного навчання та штучного інтелекту дозволяє значно підвищити ефективність аналізу даних і отримувати глибші аналітичні висновки. Здатність обробляти дані в режимі реального часу особливо важлива в галузях, де швидкість реагування на зміни є критичною для успішної діяльності. Впровадження сучасних технологій штучного інтелекту та машинного навчання для швидкого та ефективного аналізу даних є необхідним для розв'язання цих проблем.

Список використаних джерел

1. Challenges to Managing Unstructured Data When Standard Approaches No Longer Work [Електронний ресурс] // Atempo. – 2024. – Режим доступу до ресурсу: https://blog.atempo.com/en/blog/6-challenges-to-managing-unstructured-data.