

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

Кафедра кібернетики та інформатики

КВАЛІФІКАЦІЙНА РОБОТА

освітній ступінь - «Бакалавр»

на тему: **РОЗРОБКА ВЕБСЕРВІСУ ДЛЯ АВТОМАТИЧНОГО
АНАЛІЗУ ТЕКСТОВИХ ДОКУМЕНТІВ НА ОСНОВІ
ГЛИБОКИХ НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ**

Виконав:

студент спеціальності

126 «Інформаційні системи та технології»

Руденок Олександр Анатолійович

Керівник:

В'юненко Олександр Борисович

к.е.н., доцент кафедри кібернетики та інформатики

Рецензент:

Ободяк Віктор Корнелійович

к.т.н., доцент кафедри кібербезпеки СумДУ

Суми – 2024

СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет	<u>Економіки і менеджменту</u>
Кафедра	<u>Кібернетики та інформатики</u>
Освітній ступінь	<u>«Бакалавр»</u>
Спеціальність	<u>126 «Інформаційні системи та технології»</u> <u>ОП «Інформаційні системи та технології»</u>

Затверджую:
Завідувач кафедри Світлана АГАДЖАНОВА

«_____» «_____» 2024____ р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Руденку Олександрю Анатолійовичу

1. Тема роботи: Розробка вебсервісу для автоматичного аналізу текстових документів на основі глибоких нейромережових моделей.

2. База практичного дослідження: Сумський НАУ.

керівник роботи В'юненко Олександр Борисович к.е.н., доцент

затверджена наказом по університету від «_____» «_____» №_____

**3. Строк подання студентом закінченого проекту
(роботи)**

«28» червня 2024 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 технічне завдання;

- розробити програмний продукт відповідно до поставлених задач;

- розробити інструкцію для користувача.

6. Дата видачі завдання:

«_____» «_____»

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 2024 р.	
2	Підбір та вивчення літературних джерел, технічної документації	січень 2024 р.	
3	Розробка технічного завдання	січень 2024 р..	
4	Розробка вебсервісу	січень-квітень 2024 р.	
5	Створення інструкції користування вебсервісу для користувачів	до 14 травня 2024 р.	
6	Оформлення пояснювальної записки	до 30 травня 2024 р..	
7	Оформлення презентації	до 1 червня 2024 р.	
8	Опрацювання зауважень керівника, перевірка на автентичність	5 червня 2024 р.	
9	Підготовка доповіді та її попередній захист	10 червня 2024 р.	
10	Перевірка з академічної доброчесності кваліфікаційної роботи	10 червня 2024 р.	
11	Представлення кваліфікаційної роботи на кафедрі, деканат	14 червня 2024 р.	
12	Захист кваліфікаційної роботи	28 червня 2024р.	

Студент _____ **Олександр РУДЕНОК**
(підпис)

Керівник роботи _____ **Олександр В'ЮНЕНКО**
(підпис)

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

Кваліфікаційна робота допущена до захисту _____ **Маргарита ЛИШЕНКО**
(підпис)

АНОТАЦІЯ

Руденок О.А. Розробка вебсервісу для автоматичного аналізу текстових документів на основі глибоких нейромережевих моделей.

Кваліфікаційна робота зі спеціальності 126 «Інформаційні системи та технології» ОП Інформаційні системи та технології Сумського національного аграрного університету. Суми, 2024 р. – Рукопис.

Дана кваліфікаційна робота присвячена розробці вебсервісу для автоматичного аналізу текстових документів на основі глибоких нейромережевих моделей. У роботі розглядаються основні методи та засоби автоматизації процесу аналізу текстових даних, аналізуються сучасні інформаційні системи та технології, що застосовуються в цій сфері. Вебсервіс призначений для автоматизації збереження та обробки текстових документів, їх семантичного аналізу та надання інтерпретацій за допомогою нейромереж.

Об'єктом дослідження є система управління дистанційного навчання Moodle. Метою роботи є розробка вебсервісу, який буде здатний автоматично аналізувати текстові документи, надавати користувачеві інтерактивне пояснення результатів аналізу та покращувати залучення студентів до навчального процесу.

Проведено тестування підсистеми з використанням реальних текстових документів користувачів. Результати тестування підтвердили правильність функціонування підсистеми та її відповідність вимогам та очікуванням користувачів.

Результатом роботи є створення вебсервісу, який забезпечує зручний інтерфейс для користувачів та автоматизує процеси аналізу текстових документів. Застосунок дозволяє користувачам зручно отримувати пояснення та інтерпретації змісту документів, що є важливим для розуміння та використання інформації.

Ключові слова: вебсервіс, автоматизація, нейромережеві моделі, GPT-4, обробка природної мови, семантичний аналіз, інформаційні системи.

SUMMARY

Rudenok O.A. Development of a web service for automatic analysis of text documents based on deep neural network models.

Qualification work in the specialty 126 «Information Systems and Technologies» OP Information Systems and Technologies of Sumy National Agrarian University. Sumy, 2024 – Manuscript.

This qualification work is devoted to the development of a web service for automatic analysis of text documents based on deep neural network models. The paper discusses the main methods and tools for automating the process of analyzing text data, analyzes modern information systems and technologies used in this area. The web service is designed to automate the storage and processing of text documents, their semantic analysis and interpretation using neural networks.

The object of research is the Moodle distance learning management system. The purpose of the study is to develop a web service that will be able to automatically analyze text documents, provide the user with an interactive explanation of the analysis results, and improve student engagement in the learning process.

The subsystem was tested using real text documents of users. The test results confirmed the correct functioning of the subsystem and its compliance with the requirements and expectations of users.

The result of the work is the creation of a web service that provides a user-friendly interface and automates the process of analyzing text documents. The application allows users to conveniently receive explanations and interpretations of the content of documents, which is important for understanding and using information.

Keywords: web service, automation, neural network models, GPT-4, natural language processing, semantic analysis, information systems.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК	7
ВСТУП	8
РОЗДІЛ 1 Аналіз предметної області	11
1.1 Огляд останніх досліджень і публікацій	11
1.2 Аналіз програмних продуктів та аналогів	15
1.3 Постановка задачі	21
РОЗДІЛ 2 Моделювання та проектування вебсервісу автоматизованого аналізу текстових документів	30
2.1 Огляд глибоких нейромережевих моделей для аналізу текстових даних	30
2.2 Проектування вебсервісу	37
2.3 Проектування бази даних	46
РОЗДІЛ 3 Розробка та програмна реалізація вебсервісу для автоматичного аналізу текстових документів	55
3.1 Планування робіт	55
3.2 Розробка архітектури вебсервісу	57
3.3 Програмна реалізація	63
3.4 Розгортання вебсервісу із використанням граничних технологій	79
ВИСНОВКИ	86
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	88
ДОДАТКИ	91
Додаток А Технічне завдання	92
Додаток Б Лістинг коду компонента UploadFile	96
Додаток В Лістинг коду завантаження документа серверної частини	102
Додаток Г Копії публікацій	107

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАК

DNN – Deep Neural Networks;
LLM – Large Language Models;
GPT – Generative Pre-trained Transformer;
NLP – Natural Language Processing;
RNN – Recurrent Neural Networks;
LSTM – Long Short-Term Memory;
CNN – Convolutional Neural Networks;
API – Application Programming Interface;
AWS – Amazon Web Services;
BERT – Bidirectional Encoder Representations from Transformers;
SEO – Search Engine Optimization;
SSR – Server-Side Rendering;
SSG – Static Site Generation;
IDE – Integrated Development Environment;
SQL – Structured Query Language;
URL – Uniform Resource Locator;
REST – Representational State Transfer;
HTTP – Hypertext Transfer Protocol;
JSON – JavaScript Object Notation;
URI – Uniform Resource Identifier;
SOAP – Simple Object Access Protocol;
XML – Extensible Markup Language;
CRUD – Create Read Update Delete.

ВСТУП

Сфера аналізу текстових документів та дистанційного навчання постійно розвивається, підживлювана як академічними дослідженнями, так і реальними застосуваннями. Університети та дослідницькі лабораторії досліджують нові архітектури глибокого навчання і розробляють передові методи для вирішення складних завдань аналізу тексту, які використовуються в різних галузях промисловості для вилучення користі зі своїх зростаючих сховищ текстових даних. Водночас дистанційне навчання стало важливим компонентом сучасної освіти, дозволяючи студентам отримувати доступ до лекцій, навчальних матеріалів та оцінок дистанційно.

Кваліфікаційна робота присвячена розробці вебсервісу, спрямованого на полегшення автоматизованого аналізу текстових документів та покращення дистанційного навчання за допомогою сучасних глибоких нейронних мереж. Вебсервіс надає користувачам інтерактивну платформу для завантаження своїх документів, отримання всебічного аналізу та діалогу з мовною моделлю для поглибленого розуміння матеріалу. Це дозволить студентам взаємодіяти зі змістом лекцій за допомогою запитів на основі методів обробки природної мови.

У постановці задачі окреслюються проблеми, пов'язані з ручним аналізом документів та обмеженнями традиційних платформ дистанційного навчання, такі як часові обмеження, суб'єктивність і проблеми масштабованості, що підкреслює необхідність автоматизованих рішень. Робота має на меті розв'язати ці проблеми, пропонуючи комплексну систему для автоматизації аналізу текстових документів та створення динамічного вебсервісу для інтерактивного навчання, використовуючи передові методи NLP і хмарні сервіси.

Основна частина кваліфікаційної роботи зосереджена на моделюванні та розробці вебсервісу, з акцентом на виборі та адаптації моделей глибоких нейронних мереж для аналізу тексту та інтерактивного навчання. Це включає детальне вивчення різних архітектур і методологій для забезпечення оптимальної продуктивності та масштабованості. Дизайн самого вебсервісу ретельно

спланований, щоб забезпечити користувачам безперебійну та інтуїтивно зрозумілу роботу, від завантаження документа до аналізу та взаємодії з мовною моделлю.

У наступних розділах детально описано розробку та програмну реалізацію, включаючи архітектурний дизайн, процес розробки програмного забезпечення та практичне використання застосунку. Сюди входить інтеграція із зовнішніми сервісами для зберігання документів, автентифікації та для обробки природної мови. Особлива увага приділяється розробці функцій, що дозволяють студентам активно взаємодіяти з навчальним матеріалом у режимі реального часу.

Робота має на меті зробити внесок у розвиток аналізу текстових документів та покращення дистанційного навчання шляхом створення надійної та зручної платформи на основі глибоких нейронних мереж. Завдяки поєднанню передових технологій та інноваційного дизайну вебсервіс пропонує користувачам потужний інструмент для вилучення інформації, генерування пояснень і ведення змістовного діалогу з мовними моделями. Це, своєю чергою, покращить продуктивність, прийняття рішень і пошук знань у різних галузях, а також підвищить ефективність та залучення студентів до навчального процесу.

Актуальність теми. Зростання обсягу і складності текстових даних у різних галузях підкреслює нагальну потребу у впровадженні ефективних і точних методів аналізу документів. Традиційні підходи до аналізу тексту часто виявляються недостатніми для обробки сучасних масштабів і складнощів текстових документів. На противагу цьому, інтеграція глибоких нейронних мереж та хмарних сервісів демонструє перспективність у сфері автоматизації аналізу документів. Цей підхід дозволяє користувачам отримувати вибірки, генерувати пояснення та вести інтерактивний діалог із нейромережевими моделями. Розробка вебсервісу для автоматизованого аналізу текстів задовольняє цю потребу, забезпечуючи потужний інструмент для підвищення продуктивності, прийняття рішень та пошуку знань у різних галузях.

Мета і завдання виконання кваліфікаційної роботи. Метою роботи є розробка вебсервісу, який буде здатний автоматично аналізувати текстові документи, надавати користувачеві інтерактивне пояснення результатів аналізу та

покращувати залучення студентів до навчального процесу. Основними завданнями є аналіз останніх досліджень та публікацій в галузі аналізу текстових документів та дистанційного навчання, виявлення тенденцій, проблем та можливостей для інновацій. Розробити та впровадити вебсервіс, забезпечивши масштабованість, надійність та зручність для користувачів.

Об'єктом дослідження є система управління дистанційного навчання Moodle.

Предметом дослідження є система автоматичного аналізу текстових документів на основі глибоких нейромережевих моделей.

Апробація результатів кваліфікаційної роботи. Матеріали Всеукраїнської наукової конференції студентів і аспірантів, присвяченої Міжнародному дню студента – (13-17 листопада 2023 р.). – Суми, 2023. – 329 с. Матеріали науково-практичної конференції викладачів, аспірантів та студентів Сумського НАУ (14-16 травня 2024 р.). – Суми, 2024. – 432 с.

Публікації. За матеріалами кваліфікаційної роботи опубліковано 2 тези наукових конференцій.

Структура і обсяг роботи. Кваліфікаційна робота складається із вступу, трьох розділів, висновків та пропозицій, списку використаних джерел з 28 найменування, додатків. Основний текст викладений на 83 сторінках комп'ютерного тексту, робота містить 4 таблиць, 26 рисунків, 4 додатки.

РОЗДІЛ 1

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Зростаючий обсяг цифрових текстових даних створює як виклики, так і можливості в різних галузях. Автоматизований аналіз текстових документів став важливим інструментом для вилучення інформації, узагальнення даних і виявлення закономірностей у великих колекціях документів. У цьому підрозділі розглядаються останні досягнення в галузі аналізу текстових документів, з особливим акцентом на використанні глибоких нейронних мереж.

Поява глибоких нейронних мереж, особливо великих мовних моделей, таких як GPT, зробила революцію в задачах обробки природної мови. Ці моделі, навчені на великих текстових масивах, демонструють виняткові можливості в розумінні та маніпулюванні людською мовою. Використовуючи можливості DNN, аналіз текстових документів може досягти нових рівнів складності, уможливлуючи виконання таких завдань, як моделювання тем, аналіз настроїв, пошук інформації та автоматизоване узагальнення [14].

Аналіз текстових документів охоплює низку методів, спрямованих на вилучення значущої інформації з текстових даних. Ці методи можна умовно поділити на такі групи:

- інформаційний пошук, зосереджується на знаходженні потрібних документів у колекції на основі запиту користувача. Традиційні підходи передбачають зіставлення ключових слів, тоді як DNN можуть забезпечити більш складні можливості семантичного пошуку;

- реферування тексту, методи узагальнення спрямовані на скорочення документа до коротшої версії зі збереженням його ключових моментів. DNN можна використовувати для створення анотацій, які охоплюють основну інформацію і підтримують зв'язний наративний потік;

- тематичне моделювання, виявляє приховані тематичні структури в колекції документів. DNN можна використовувати для виявлення нових тем, відстеження еволюції тем у часі та виявлення взаємозв'язків між різними темами;

- аналіз настроїв, спрямований на визначення емоційного тону або думки, висловленої в текстовому документі. DNN чудово класифікують настрої як позитивні, негативні або нейтральні, і навіть можуть визначати нюанси емоційних станів;

- розпізнавання іменованих об'єктів, передбачає ідентифікацію та класифікацію іменованих об'єктів у документі, таких як люди, організації та географічні місця. DNN можуть досягти високої точності в завданнях розпізнаванні іменованих об'єктів, дозволяючи витягувати структуровану інформацію з текстових даних;

- класифікація тексту, класифікує документи на основі попередньо визначених міток. DNN можна використовувати для розробки надійних класифікаторів, здатних обробляти складні та багатогранні текстові дані.

Застосовуючи ці методи, аналіз текстових документів дає можливість дослідникам, компаніям і приватним особам розблокувати приховану цінність у величезних колекціях документів [19].

Глибокі нейронні мережі стали потужним інструментом для аналізу текстових документів завдяки своїй здатності вивчати складні закономірності з великих наборів даних. Існує декілька видів DNN, архітектура яких добре підходить для завдань аналізу тексту.

Рекурентні нейронні мережі являють собою тип глибоких нейронних мереж, спеціально розроблений для обробки послідовних даних, таких як текст. Завдяки своїй архітектурі, вони здатні зберігати і використовувати інформацію з попередніх етапів послідовності, що дозволяє ефективно захоплювати контекст і довготривалі залежності в текстових даних.

Мережі з довгою короткочасною пам'яттю особливий тип архітектури, оснащений механізмами для розв'язання проблеми зникаючого градієнта, що є поширеною проблемою, яка обмежує здатність DNN вивчати довгострокові

залежності. LSTM відмінно справляються із завданнями, що вимагають аналізу довгих послідовностей, що робить їх добре придатними для таких завдань, як узагальнення документів і тематичне моделювання. [3]

Трансформер відносно нова архітектура DNN, яка досягла значних успіхів у різних завданнях обробки природної мови. На відміну від RNN, трансформери обробляють цілі послідовності тексту одночасно, що дозволяє їм фіксувати зв'язки між словами в будь-якій позиції в послідовності. Можливість паралельної обробки робить трансформери високоефективними для таких завдань, як машинний переклад і узагальнення тексту.

Розглянуті види архітектури DNN можна використовувати для різних завдань аналізу текстових документів. До таких завдань можна віднести: підсумовування тексту, тематичне моделювання, аналіз настроїв, інформаційний пошук, розпізнавання іменованих об'єктів та класифікацію тексту [20].

Останні дослідження використовують LSTM і трансформери, для створення резюме, які не лише фіксують ключову фактичну інформацію, але й підтримують зв'язну наративну структуру. Ці моделі навчаються на великих масивах даних анотацій, написаних людиною, що дає їм змогу вивчати тонкощі узагальнення різних типів текстових документів.

LSTM і трансформери, досліджуються для задач тематичного моделювання. Ці моделі можуть ефективно виявляти приховані теми в колекціях документів, відстежувати їхню еволюцію в часі та виявляти взаємозв'язки між різними темами. Зокрема, дослідники вивчають можливість інтеграції знань з конкретних доменів у DNN для підвищення точності моделювання тем у спеціалізованих галузях.

DNN відмінно справляються з аналізом настроїв, перевершуючи традиційні підходи, засновані на лексиконі. Згорткові нейронні мережі та LSTM зазвичай використовуються для аналізу тексту та класифікації настроїв як позитивних, негативних або нейтральних. Крім того, дослідження просуваються в напрямку виявлення більш нюансованих емоційних станів у текстових даних [22].

Інформаційний пошук зазнав революції, надаючи можливості семантичного пошуку. Сучасні DNN здатні виходити за рамки простого зіставлення ключових

слів і розуміти основне значення запитів користувачів, що призводить до більш релевантного і точного пошуку документів.

Штучні нейронні мережі досягають високої точності в задачах розпізнавані іменованих об'єктів. Ці моделі навчаються на великих наборах даних, анотованих іменованими сутностями, що дозволяє їм ідентифікувати та класифікувати різні типи сутностей у текстових документах з високою точністю.

DNN також використовуються для розробки надійних текстових класифікаторів, здатних обробляти складні та багатогранні текстові дані. Згорткові нейронні мережі і трансформери показують багатообіцяючі результати в класифікації документів на основі попередньо визначених категорій, таких як тема, жанр або настрій.

Хоча DNN зробили революцію в аналізі текстових документів, необхідна критична оцінка, щоб визначити сфери для вдосконалення і майбутні напрямки досліджень. До сильних сторін можна віднести:

- високу точність, оскільки DNN чудово вивчають складні закономірності з великих наборів даних, що призводить до кращої продуктивності в різних завданнях порівняно з традиційними методами;

- масштабованість, DNN можуть ефективно працювати з великими колекціями документів, що робить їх придатними для аналізу величезних масивів текстових даних;

- універсальність, різноманітні архітектури DNN дозволяють адаптувати їх до різних завдань аналізу тексту, пропонуючи гнучкий і потужний підхід.

Суттєвими обмеженнями, які варто зазначити є:

- інтерпретованість, внутрішня робота складних моделей DNN може бути непрозорою, що ускладнює розуміння того, як вони отримують конкретні результати. Відсутність інтерпретованості може перешкоджати довірі та обмежувати їх застосування в певних сферах;

- упередженість, DNN схильні до успадкування упереджень, присутніх у навчальних даних, на які вони впливають. Зменшення упередженості вимагає

ретельного відбору даних і розробки справедливих та неупереджених процедур навчання;

- обчислювальні витрати, навчання та запуск моделей DNN, особливо великих LLM, таких як GPT-4, може бути дуже дорогим процесом, що вимагає значних обчислювальних ресурсів [10].

Усуваючи ці обмеження DNN мають величезний потенціал для подальшої революції в аналізі текстових документів і отримання ще глибших знань з постійно зростаючого моря текстових даних [19].

У даному підрозділі зроблено огляд останніх досліджень і публікацій у галузі аналізу текстових документів на основі глибоких нейронних мереж. Розглянуто основні функціональні можливості завдань аналізу тексту, заглиблено в теоретичні основи відповідних архітектур DNN і проаналізовано, як ці моделі використовуються в сучасних дослідженнях.

Висвітлено сильні сторони DNN, зокрема їхню високу точність, масштабованість та універсальність. Однак, в ній також визнаються і обмеження, такі як інтерпретованість, упередженість та обчислювальні витрати.

Загалом, DNN представляють трансформаційну силу в аналізі текстових документів, дозволяючи дослідникам і практикам отримувати цінну інформацію з величезних колекцій текстових даних. Усуваючи поточні обмеження і розвиваючи перспективні напрямки, DNN мають потенціал для розкриття ще більших можливостей у цій критично важливій галузі.

1.2 Аналіз програмних продуктів та аналогів

Дослідженні наявних програмних продуктів, які використовують глибокі нейромережеві моделі для аналізу текстових документів допоможе зрозуміти, які рішення вже існують на ринку, які проблеми вони вирішують, а також їхні сильні та слабкі сторони.

Розглянуто різні аспекти цих продуктів, включаючи їхні функціональні можливості, використання, доступність, надійність, масштабованість та вартість.

Також проаналізовано, як вони використовують глибокі нейронні мережі для аналізу текстових документів, і які вони використовують моделі та алгоритми.

Цей аналіз допоможе визначити, які функції та характеристики мають бути додані в вебсервіс для аналізу текстових файлів, а також допоможе уникнути потенційних проблем, з якими стикалися інші розробники. Також допоможе визначити, як продукт може відрізнитися від наявних рішень та які унікальні переваги він може пропонувати користувачам. Це є важливим етапом у процесі розробки вебсервісу, оскільки він допоможе краще зрозуміти потреби користувачів та визначити, як можна задовольнити ці потреби.

Перший сервіс для аналізу текстових документів це Google Natural Language AI [2]. Потужний інструмент, який надає розробникам технології розуміння природної мови. Він дозволяє розробникам отримувати інформацію з неструктурованого тексту за допомогою машинного навчання Google. Цей сервіс є частиною Google Cloud і пропонує різноманітні функції для аналізу тексту.

Однією з ключових особливостей Google Natural Language AI є його здатність витягувати, аналізувати та зберігати текст. Він може навчати високоякісні користувацькі моделі машинного навчання без жодного рядка коду за допомогою AutoML. Це дозволяє розробникам завантажувати свої навчальні дані та тестувати кастомні моделі без жодного рядка коду.

Google Natural Language AI застосовує розуміння природної мови до додатків за допомогою API природної мови. API включає такі функції, як аналіз настроїв, аналіз сутностей, аналіз настроїв сутностей, класифікація змісту та аналіз синтаксису. Функція аналізу настроїв перевіряє заданий текст і визначає переважаючу емоційну думку в тексті, особливо для визначення ставлення автора як позитивного, негативного або нейтрального. З іншого боку, аналіз сутностей перевіряє текст на наявність відомих сутностей і повертає інформацію про ці сутності. Аналіз настроїв об'єктів поєднує ці дві функції, повертаючи інформацію про відомі об'єкти та визначаючи переважаючу емоційну оцінку об'єкта в тексті.

Синтаксичний аналіз є однією з функцій API для обробки природної мови, яка витягує лінгвістичну інформацію шляхом розбиття заданого тексту на окремі

речення та токени, забезпечуючи подальший аналіз на рівні цих токенів. Класифікація контенту здійснює аналіз змісту тексту та повертає відповідну категорію контенту.

Отже, Google Natural Language AI це комплексний інструмент для аналізу текстових документів, що пропонує широкий спектр функцій і можливостей, які можна використовувати для різних завдань у сфері розуміння природної мови.

Наступний сервіс обробки природної мови Amazon Comprehend. Використовує машинне навчання для пошуку висновків і взаємозв'язків у тексті. Є частиною набору Amazon Web Services і призначений для аналізу та вилучення висновків з тексту [5].

Amazon Comprehend використовує попередньо навчену модель для вивчення та аналізу документа або набору документів. Модель безперервно навчається на великому масиві тексту, що позбавляє користувачів необхідності надавати дані для навчання. Сервіс визначає мову тексту, виокремлює ключові фрази, місця, людей, бренди чи події, розуміє, наскільки позитивним чи негативним є текст, і аналізує текст за допомогою токенізації та частин мови. Він також може автоматично організувати колекцію текстових файлів за темами.

Amazon Comprehend здійснює аналіз документів, включаючи визначення таких елементів, як об'єкти, ключові фрази, мова, настрої та інші загальні компоненти. Об'єкти охоплюють імена людей, місця, предмети та локації, зазначені в документі. Ключові фрази представляють собою значущі фрази, які зустрічаються в тексті. Персональна інформація включає дані, що можуть ідентифікувати особу, наприклад, адресу, номер банківського рахунку або телефонний номер. Додатково визначається домінуюча мова документа. Настрій документа класифікується як позитивний, нейтральний, негативний або змішаний, причому цільовий настрій пов'язується з конкретними об'єктами у тексті. Синтаксис охоплює визначення частин мови для кожного слова в документі.

Amazon Comprehend може обробляти будь-які текстові файли у форматі UTF-83. Також виконує аналіз у режимі реального часу для невеликих робочих навантажень або запускає асинхронні завдання аналізу для великих наборів

документів. Користувачі можуть використовувати попередньо навчені моделі, які надає вебсервіс, або створювати власні моделі для класифікації та розпізнавання об'єктів. Amazon Comprehend може зберігати дані користувача, щоб постійно покращувати якість своїх попередньо навчених моделей.

На додаток до цих функцій, Amazon Comprehend підтримує різні мови залежно від конкретної функціональності. Функція визначення панівної мови може аналізувати документи та ідентифікувати основну мову з великого набору мов. Таким чином, Amazon Comprehend є надійним інструментом для аналізу текстових документів, пропонуючи широкий спектр функцій і можливостей для виконання різних завдань у сфері обробки природної мови.

Кожен існуючий сервіс має свої переваги та недоліки, що робить їх придатними для різних сценаріїв використання та уподобань. AWS Intelligent Document Processing пропонує найповніший набір функцій та інтеграцій, що робить його ідеальним для користувачів, глибоко інтегрованих в екосистему AWS. Google Cloud Natural Language забезпечує надійні можливості NLP та безперешкодну інтеграцію з іншими хмарними сервісами Google. Microsoft Azure Text Analytics пропонує широку мовну підтримку та конкурентоспроможну продуктивність, що робить його відмінним вибором для організацій, які вже використовують служби Azure.

Проте, існує суттєвий недолік у цих сервісів, який робить їх менш підходящими для масового користування - це складний інтерфейс та відсутність інтерактивної форми взаємодії, такої як чат інтерфейс. Основною частиною користувачів наведених вебсервісів є розробники, які використовуючи API-інтерфейс інтегрують частину функціоналу в свої додатки.

Створення власного вебсервісу для аналізу текстових документів вирішує цю проблему. Цей сервіс може бути спеціально розроблений для вирішення конкретних потреб загальної аудиторії. Він забезпечує можливість користувачам не лише отримувати аналіз тексту, а й взаємодіяти з даними у найбільш зручний та доступний спосіб. Інтеграція глибоких нейронних мереж дозволить отримувати

більш точні та високоякісні результати аналізу тексту, що перевершують можливості існуючих сервісів.

Останнім з розглянутих вебсервісів є Lexalytics. Лідер у галузі текстової аналітики та обробки природної мови. Пропонує набір програмних бібліотек, які перетворюють неструктурований текст на придатні для використання дані та висновки. Ці бібліотеки підходять для аналітиків даних та архітекторів, які потребують повного доступу до базової технології або потребують локального розгортання з міркувань безпеки чи конфіденційності [24].

Ядром пропозиції Lexalytics є технологія текстової аналітики. Текстова аналітика - це процес перетворення неструктурованих текстових документів на придатні для використання структуровані дані. Текстова аналітика Lexalytics працює шляхом розбиття речень і фраз на складові частини, а потім оцінює роль і значення кожної частини за допомогою складних програмних правил і алгоритмів машинного навчання. Це формує основу численних функцій обробки природної мови, включаючи розпізнавання іменованих об'єктів, категоризацію та аналіз настроїв.

Технологія текстової аналітики Lexalytics призначена для аналізу чотирьох основних аспектів: визначення суб'єкта висловлювання, теми обговорення, тверджень щодо цих тем та виражених емоцій. Для цього Lexalytics використовує інструменти інтелектуального аналізу тексту, які дозволяють витягувати корисну інформацію та робити контекстні висновки з великих обсягів необробленого тексту, таких як коментарі в соціальних мережах, онлайн-огляди та новинні статті.

На додаток до своєї основної технології текстової аналітики, Lexalytics пропонує ряд послуг, щоб допомогти клієнтам отримати максимальну віддачу від своїх рішень для текстової аналітики. Ці послуги включають конфігурацію об'єктів, власну категоризацію, налаштування настроїв і розробку власних моделей машинного навчання. Наприклад, Lexalytics може створювати і налаштовувати списки об'єктів, адаптовані до конкретної галузі, включаючи продукти, бренди і навіть адреси. Також може підвищити точність категоризації та вилучення тем, створюючи власні таксономії.

Коли певні слова та фрази мають особливе значення в конкретному бізнесі або галузі, Lexalytics може налаштувати свої системи оцінки настроїв для відображення цієї специфіки. Для розв'язання складних завдань використовуються моделі машинного навчання, що забезпечують більш економічно ефективні рішення. Ці моделі навчаються виконувати дуже специфічні завдання, такі як розпізнавання об'єктів за неоднозначною назвою компанії або категоризація харчових продуктів і соусів.

Lexalytics пропонує комплексний набір інструментів і сервісів для аналізу тексту та обробки природної мови. Його можливості варіюються від базової текстової аналітики до більш просунутих функцій, таких як аналіз настроїв, розпізнавання іменованих об'єктів і кастомні моделі машинного навчання. Це робить його надійним інструментом для аналізу текстових документів.

Хоча Google Natural Language AI, Amazon Comprehend і Lexalytics пропонують потужні можливості аналізу тексту, вони також мають певні обмеження, які можуть перешкоджати їхньому використанню не розробниками та звичайними користувачами.

По-перше, ці сервіси вимагають певного рівня технічних знань для ефективного використання. Вони розроблені з думкою про розробників, а тому їхні інтерфейси та документація часто складні та наповнені технічними термінами. Це може зробити їх недоступними для звичайних користувачів, які не мають досвіду в програмуванні або науці про дані.

По-друге, ці сервіси часто вимагають ручної конфігурації та налаштування для досягнення оптимальних результатів. Наприклад, аналіз настроїв може потребувати коригування для точного відображення настроїв у певній галузі чи контексті. Це вимагає розуміння технології, що лежить в основі, і може зайняти багато часу.

По-третє, ці сервіси зазвичай не мають зручного інтерактивного інтерфейсу. Користувачам часто доводиться працювати з кодом або складними інформаційними панелями, щоб проаналізувати свої текстові документи, що може бути складно для нетехнічних користувачів.

Ці обмеження підкреслюють необхідність створення нового вебсервісу для аналізу текстових документів. Цей сервіс має бути розроблений з урахуванням простоти використання, що зробить його доступним для звичайних користувачів. Він би обробляв усі складнощі аналізу тексту за лаштунками, дозволяючи користувачам зосередитися на своїх документах і висновках, отриманих з них.

Однією з ключових особливостей цього нового сервісу був би чат інтерфейс. Це дозволить користувачам взаємодіяти з системою в природній, розмовній манері, ставлячи запитання та отримуючи відповіді в режимі реального часу. Це зробить процес аналізу тексту більш цікавим та інтуїтивно зрозумілим, усунувши бар'єри, які часто асоціюються з більш технічними системами.

Отже, існуючі сервіси, такі як Google Natural Language AI, Amazon Comprehend і Lexalytics, пропонують потужні можливості, їхня складність і відсутність зручного інтерфейсу створюють значні проблеми для звичайних користувачів. Розробка нового вебсервісу, розробленого з урахуванням простоти використання та інтерактивності, могла б вирішити ці проблеми і зробити можливості аналізу тексту доступними для всіх.

1.3 Постановка задачі

Поширення цифрових текстових документів у різних галузях зумовило необхідність розробки передових інструментів для ефективного аналізу та розуміння. З появою моделей глибоких нейронних мереж, зокрема GPT-4 від компанії OpenAI, потенціал автоматизованого аналізу тексту досяг нових висот. У цьому контексті розробка вебсервісу, який використовує такі моделі для надання інтерактивних пояснень текстових документів, відкриває широкі можливості як для досліджень, так і для практичного застосування.

Основна проблема полягає у складності та недоступності існуючих сервісів аналізу текстових документів, таких як Google Natural Language AI, Amazon Comprehend та Lexalytics. Ці сервіси, хоча й потужні, розроблені для розробників і потребують певного рівня технічних знань для ефективного використання. Для

досягнення оптимальних результатів вони часто потребують ручної конфігурації та налаштування, що може забирати багато часу і бути складним завданням для нетехнічних користувачів. Крім того, наведені сервіси не мають зручного інтерактивного інтерфейсу, що ускладнює ефективний аналіз текстових документів для звичайних користувачів.

Тому метою є розробка нового вебсервісу для аналізу текстових документів, який буде простим у використанні і доступним для звичайних користувачів, а не лише для розробників. Цей сервіс повинен обробляти всі складнощі аналізу тексту за лаштунками, дозволяючи користувачам зосередитися на своїх документах і висновках, отриманих з них.

Для вирішення цієї проблеми необхідно розробити зручний вебсервіс, який використовує можливості глибоких нейронних мереж для аналізу та пояснення текстових документів в інтерактивному режимі. Цей сервіс повинен пропонувати чат-інтерфейс, який дозволить користувачам взаємодіяти з системою в природній, розмовній манері, ставлячи запитання та отримуючи відповіді в режимі реального часу. Це зробить процес аналізу тексту більш цікавим та інтуїтивно зрозумілим, усунувши бар'єри, які часто асоціюються з більш технічними системами.

Проблема, що розглядається, полягає в розробці та реалізації вебсервісу, здатного безперешкодно аналізувати текстові документи і надавати користувачам глибокі пояснення. Основні завдання містять:

- інтеграцію найсучасніших моделей глибоких нейронних мереж, у вебсервіс для полегшення аналізу та інтерпретації документів;
- створення інтуїтивно зрозумілого користувацького інтерфейсу, який дозволяє користувачам завантажувати документи, взаємодіяти з системою та отримувати значущі дані без зайвих зусиль;
- впровадження надійної системи управління даними для ефективної обробки користувацьких документів, запитів та відповідей, забезпечення безперешкодної інтеграції із зовнішніми сервісами;

- розробку методів перетворення текстових документів і запитів користувачів у семантично значущі представлення для ефективного пошуку інформації та генерації контексту;

- забезпечення можливості масштабування системи для обслуговування великої кількості користувачів і документів, зберігаючи при цьому високу продуктивність і швидкість реагування.

Вирішення цих завдань призведе до створення комплексного рішення, яке дозволить користувачам швидко та ефективно отримувати цінну інформацію з текстових документів.

Інтеграції сучасних моделей глибоких нейронних мереж, зокрема моделі GPT-4 від компанії OpenAI, у вебсервіс є важливим аспектом для забезпечення автоматизованого аналізу та інтерпретації текстових документів. Цей процес включає кілька ключових кроків для забезпечення ефективної взаємодії між вебсервісом і нейромережевою моделлю, а також генерування точних і контекстно-релевантних відповідей.

Першим кроком в інтеграції моделей є ретельний відбір моделей, які добре підходять для завдань аналізу тексту. Модель GPT-4 виділяється як потужна мовна модель, здатна розуміти та генерувати текст, подібний до людського. Інші моделі, такі як BERT, також можуть бути розглянуті залежно від конкретних вимог [11].

Після вибору моделей необхідна інтеграція з OpenAI API, щоб отримати доступ до їхніх можливостей. Це передбачає налаштування кінцевих точок API для зв'язку з моделлю, обробку автентифікації та налаштування параметрів запиту, таких як версія моделі та ліміти токенів. Перед передачею текстових даних до моделі може знадобитися попередня обробка для забезпечення сумісності та оптимальної продуктивності. Це може включати токенізацію, видалення спеціальних символів та обробку будь-яких мовних нюансів або проблем з форматуванням. Під час аналізу текстових документів важливо надати моделі контекст, щоб вона могла генерувати змістовні відповіді. Для цього необхідно зробити сегментацію документа на менші фрагменти або надання відповідної довідкової інформації, щоб керувати процесом аналізу. Після того, як модель

опрацює вхідний текст, згенеровані відповіді потрібно проаналізувати, відформатувати і представити користувачеві в чіткій і зрозумілій формі. Це може включати кроки постобробки, такі як узагальнення, виділення ключових моментів або створення пояснень.

Як і для будь-якої зовнішньої інтеграції з API, необхідно мати механізми обробки помилок для розв'язання таких проблем, як збої в мережі, обмеження швидкості API або неочікувані помилки в моделі. Впровадження стратегій повторних спроб і резервних механізмів може допомогти забезпечити надійність сервісу.

Для підтримки швидкості реагування і масштабованості можуть застосовуватися методи оптимізації продуктивності, такі як кешування часто використовуваних відповідей, пакетна обробка запитів або використання асинхронної обробки для довготривалих завдань. Завдяки ефективній інтеграції моделей глибоких нейронних мереж у вебсервіс, користувачі можуть скористатися розширеними можливостями аналізу тексту, включаючи узагальнення, аналіз настроїв, розпізнавання об'єктів і відповіді на запитання. Інтеграція формує основу вебсервісу, дозволяючи йому надавати точні та глибокі пояснення текстових документів в інтерактивному режимі.

Створення інтуїтивно зрозумілого та цікавого користувацького інтерфейсу має важливе значення для того, щоб користувачі могли легко завантажувати документи, взаємодіяти з системою та отримувати змістовну інформацію зі своїх текстових документів. Взаємодія з користувачем охоплює різні аспекти вебсервісу, включаючи завантаження документів, подання запитів і представлення результатів.

Взаємодія користувача з системою складається з наступних елементів:

- інтерфейс завантаження документів має бути візуально зрозумілим і привабливим, щоб допомогти користувачам легко завантажувати свої PDF-документи. Він повинен надавати зворотний зв'язок про хід та успішність завантаження. Впровадження перевірки валідації, щоб переконатися, що для завантаження приймаються лише документи у форматі PDF. Надавання інформативних повідомлень про помилки, якщо завантаження не вдалося через

неправильний формат файлу, обмеження розміру або інші проблеми, підказуючи користувачам, як розв'язувати проблему;

- інтерфейс чату повинен бути інтуїтивно зрозумілим, нагадуючи розмову з системою. Користувачі повинні мати можливість вводити свої запити в поле для введення тексту і легко надсилати їх. Надавання підказок або пропозиції, які допоможуть користувачам зрозуміти, які типи запитів вони можуть задавати або як ефективно взаємодіяти з системою. Зворотний зв'язок у режимі реального часу під час подання запиту, наприклад, повідомляючи, коли система обробляє запит і коли відповідь буде готова;

- відображення результатів аналізу в інтерактивній формі, наприклад, за допомогою чат інтерфейсу або візуалізацій, що динамічно оновлюються. Це дозволяє користувачам взаємодіяти з результатами і шукати подальші роз'яснення, якщо це необхідно. Виділення ключових ідей або важливих розділів документа, які мають відношення до запиту користувача. Це допоможе користувачам швидко зрозуміти найбільш релевантну інформацію;

- представити потік взаємодії з користувачем у логічній послідовності, безперешкодно проводячи користувачів через різні етапи завантаження, запитів та вивчення результатів. Розробити елементи навігації, такі як верхня панель навігації та зворотні посилання, щоб допомогти користувачам легко переходити між різними розділами вебсервісу. Підтримання дизайну і шаблонів взаємодії в інтерфейсі, щоб забезпечити цілісний користувацький досвід;

- інтерфейс повинен бути доступним для користувачів з обмеженими можливостями, реалізувавши такі функції, як сумісність з пристроями зчитування з екрана, клавіатурна навігація та альтернативні варіанти тексту для візуальних елементів. Оптимізація інтерфейсу для різних пристроїв і розмірів екранів, щоб забезпечити однакову роботу на настільних, планшетних і мобільних пристроях.

Зосередившись на цих аспектах взаємодії з користувачами, вебсервіс може забезпечити безперебійний і цікавий досвід для користувачів, дозволяючи їм легко завантажувати документи, ставити запитання та отримувати змістовну інформацію.

Ефективне управління даними має вирішальне значення для забезпечення безперебійної роботи вебсервісу, особливо при обробці документів користувачів, запитів і відповідей. Управління даними охоплює кілька ключових компонентів, включаючи зберігання, пошук та інтеграцію із зовнішніми сервісами. Використання сторонніх сервісів для зберігання завантажених користувачами PDF-документів. Після завантаження документа сервіс генерує унікальну URL-адресу, яка потім зберігається разом з додатковими метаданими в реляційній базі даних. Ці метадані включають таку інформацію, як ідентифікатор користувача, назву документа, мітку часу завантаження та будь-які відповідні теги або категорії, пов'язані з документом.

Реляційна база даних слугує центральним сховищем для керування даними користувача, що складається з декількох трьох основних таблиць. Таблиця для зберігання інформації про користувачів, включаючи імена користувачів, паролі представлені у вигляді хешу, токени автентифікації. Таблиця для зберігання завантажених документів, маючи зв'язок «один-до-багатьох» з попередньою таблицею, щоб пов'язати документи з їхніми власниками. Та таблиця для збереження запитів та відповідей, також має зв'язок «один-до-багатьох», щоб пов'язати запити з документами до яких вони були зроблені.

Забезпечення узгодженості, безпеки та масштабованості даних є пріоритетним завданням у процесі управління даними. Для досягнення цих цілей вебсервіс впроваджує різноманітні заходи:

- дані користувача, включаючи паролі та вміст документів, шифруються як під час передачі, так і в стані спокою, щоб запобігти несанкціонованому доступу;
- суворий контроль доступу застосовується для обмеження доступу до даних користувачів на основі їхньої автентифікації та рівнів авторизації. Тільки автентифіковані користувачі можуть отримати доступ до власних документів;
- система керування даними розроблена з можливістю горизонтального масштабування, щоб пристосуватися до зростаючих баз користувачів і колекцій документів. Це передбачає розподіл даних між кількома серверами та

використання методів балансування навантаження для ефективного розподілу вхідних запитів;

- регулярне резервне копіювання даних користувачів здійснюється для запобігання втрати даних у разі збоїв обладнання або інших непередбачуваних обставин. Крім того, існують надійні процедури відновлення для швидкого відновлення даних у разі збою;

Загалом, ефективне управління даними гарантує, що вебсервіс може безпечно зберігати, отримувати та обробляти дані користувачів, забезпечуючи безперешкодну взаємодію та аналіз текстових документів. Інтегруючись із зовнішніми сервісами вебсервіс може використовувати їхні можливості, щоб надати користувачам всебічний і надійний досвід аналізу документів.

Семантичне розуміння є ключовим аспектом вебсервісу, що забезпечує точну інтерпретацію значень текстових документів і запитів користувачів. Вебсервіс використовує OpenAI Embeddings API для перетворення текстових документів і запитів користувачів у семантичні вектори [11]. Семантичний вектор представляє собою щільні, високорозмірні репрезентації тексту, що відображають семантичні зв'язки між словами і фразами. Перетворюючи текст у векторний простір, вебсервіс може виконувати семантичні операції, такі як порівняння подібності та контекстний аналіз.

Коли користувач завантажує PDF-документ, його вміст обробляється для вилучення текстової інформації. Потім цей текст пропускається через модель Embeddings для створення семантичного векторного представлення. Вектор інкапсулює семантичне значення документа, відображаючи його ключові поняття, теми та зв'язки між словами. Аналогічно, коли користувач надсилає запит, він також перетворюється на семантичний вектор, який відображає семантичний зміст запиту та слугує основою для вилучення відповідної інформації з бази даних документів.

Методи семантичного пошуку застосовуються для зіставлення семантичних векторів запитів користувачів зі змістом документів, передбачаючи обчислення балів подібності між векторами за допомогою таких методів, як косинусоїдна

подібність. Документи з векторами, найбільш схожими на вектор запиту, знаходяться і розглядаються як потенційний контекст для генерації відповідей. Після визначення релевантних документів їхні семантичні вектори використовуються для надання контексту моделям глибоких нейронних мереж. Включаючи цей контекст у процес аналізу, моделі можуть генерувати точніші та контекстуально релевантні відповіді на запити користувачів [27].

Семантичне розуміння лежить в основі здатності вебсервісу аналізувати текстові документи і надавати змістовні пояснення. Використовуючи модель Embeddings і методи семантичного пошуку, сервіс може ефективно подолати семантичний розрив між запитами користувачів і вмістом документів, що дозволяє проводити більш точний і глибокий аналіз.

Отже, постановка проблеми окреслила кілька ключових викликів, які необхідно вирішити при розробці. Ці проблеми включають інтеграцію моделей, взаємодію з користувачами, управління даними та семантичне розуміння.

Ефективне вирішення цих проблем призведе до розробки надійного вебсервісу, здатного забезпечити автоматизований аналіз і пояснення текстових документів в інтерактивний і глибокий спосіб.

Аналіз текстових документів за допомогою глибоких нейронних мереж став потужним інструментом для вилучення значущої інформації з великих обсягів текстових даних. Методи аналізу текстових документів включають інформаційний пошук, реферування тексту, тематичне моделювання, аналіз настроїв, розпізнавання іменованих об'єктів та класифікацію тексту. DNN використовуються для виконання цих завдань з високою точністю, масштабованістю та універсальністю.

Дослідження наявних програмних продуктів, які використовують глибокі нейромережеві моделі для аналізу текстових документів, дозволило зрозуміти, які рішення вже існують на ринку, які проблеми вони вирішують, а також їхні сильні та слабкі сторони. Розглянуто різні аспекти цих продуктів, включаючи їхні функціональні можливості, використання, доступність, надійність,

масштабованість та вартість. Також було проаналізовано, як вони DNN для аналізу текстових документів, і які вони використовують моделі та алгоритми.

Цей аналіз сприяв ідентифікації потрібних функцій та характеристик для включення в вебсервіс, а також допоміг уникнути потенційних проблем, які зустрічали інші розробники. Додатково було визначено, як продукт може відрізнятися від наявних рішень і які унікальні переваги він може пропонувати користувачам.

Кожен існуючий сервіс має свої переваги та недоліки, що робить їх придатними для різних сценаріїв використання та уподобань. Проте, існує суттєвий недолік у цих сервісів, який робить їх менш підходящими для масового користування - це складний інтерфейс та відсутність інтерактивної форми взаємодії, такої як чат інтерфейс. Створення вебсервісу для аналізу текстових документів вирішує цю проблему. Цей сервіс може бути спеціально розроблений для вирішення конкретних потреб загальної аудиторії. Він забезпечує можливість користувачам не лише отримувати аналіз тексту, а й взаємодіяти з даними у найбільш зручний та доступний спосіб. Технічне завдання наведено у додатку А.

РОЗДІЛ 2

МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ВЕБСЕРВІСУ АВТОМАТИЗОВАНОГО АНАЛІЗУ ТЕКСТОВИХ ДАНИХ

2.1 Огляд глибоких нейромережових моделей для аналізу текстових документів

У сфері обробки природної мови та аналізу тексту глибокі нейронні мережі стали потужним інструментом, який революціонував підхід дослідників і практиків до цих складних завдань. Моделі глибокого навчання, завдяки своїй здатності вивчати складні закономірності і уявлення з величезних обсягів даних, продемонстрували чудову продуктивність у різних додатках NLP, включаючи класифікацію текстів, аналіз настроїв, машинний переклад і відповіді на запитання.

Однією з найвідоміших архітектур глибокого навчання для аналізу тексту є модель Трансформер, яку представили Vaswani та ін. у 2017 році. Модель використовує механізми уваги для фіксації довгострокових залежностей всередині послідовностей, що дозволяє їй ефективно обробляти і розуміти текстові дані. Архітектура трансформер отримала широке розповсюдження і послужила основою для багатьох успішних мовних моделей, таких як Generative Pre-trained Transformer і Bidirectional Encoder Representations from Transformers [8, 11].

Модель GPT-4, розроблена OpenAI, є великомасштабною мовною моделлю, яка навчалася на великому корпусі текстових даних з Інтернету. Маючи 170 трильйонів параметрів, GPT-4 продемонструвала чудові можливості в розумінні та створенні тексту, схожого на людську мову, в широкому діапазоні областей і завдань. Здатність цієї моделі фіксувати контекстну інформацію і генерувати зв'язні та релевантні відповіді зробила її цінним активом для різних застосувань NLP, включаючи узагальнення тексту, відповіді на питання і генерацію текстів [13].

У контексті аналізу текстових документів глибокі нейронні мережі, такі як GPT, можуть бути використані для таких завдань, як екстрактивне та абстрактне

узагальнення, вилучення інформації та генерація контенту. Ці моделі можуть аналізувати і розуміти зміст певного текстового документа, визначаючи ключові моменти, витягуючи релевантну інформацію і генеруючи стислі резюме або інтерпретації. Крім того, їх можна використовувати для створення пояснень, уточнень або додаткового контенту на основі вхідного документа, що сприяє інтерактивному та динамічному дослідженню тексту.

В основі моделі GPT лежить Трансформер – архітектура глибокого навчання, розроблена для вирішення задач послідовного навчання. На відміну від традиційних рекурентних нейронних мереж, трансформер використовує механізм уваги, що дозволяє динамічно ідентифікувати релевантні ділянки вхідної послідовності. Завдяки цьому механізму модель здатна ефективно захоплювати довгострокові залежності в текстових даних, що є критично важливим для розуміння складних взаємозв'язків між словами [1].

Трансформер використовує два ключові підмодулі: стек кодерів-декодерів та механізм само уваги. Кодер обробляє вхідну послідовність, генеруючи контекстне представлення для кожного слова. Механізм уваги в кодері дозволяє кожному елементу звертати увагу на інші елементи послідовності, сприяючи всебічному розумінню структури та значення вхідних даних. Декодер, відповідальний за створення вихідної послідовності, використовує закодоване представлення для передбачення наступного слова. Знову ж таки, механізм уваги відіграє важливу роль, дозволяючи декодеру зосередитися на відповідних частинах закодованого контексту, коли він генерує вихідні дані. Механізм самоуваги дозволяє кожному елементу послідовності приділяти увагу іншим елементам, ефективно оцінюючи їх релевантність для поточного кроку обробки. Це дозволяє моделі розуміти складні взаємозв'язки між словами, навіть якщо вони не є безпосередніми сусідами в послідовності [7]. Модель GPT проходить двоетапний процес навчання:

1) некероване навчання за допомогою маскованого мовного моделювання. На модель подається великий масив даних з текстом і текстовими фрагментами. Слова в корпусі випадковим чином маскуються, і модель отримує завдання передбачити замасковані слова на основі навколишнього контексту. Завдяки

такому підходу самонавчання модель GPT розвиває глибоке розуміння мовних моделей, синтаксису та семантики;

2) точне налаштування для конкретних завдань. Попередньо навчена модель може бути додатково налаштована для конкретних завдань NLP, таких як узагальнення тексту, машинний переклад або відповіді на запитання. Під час тонкого налаштування параметри моделі коригуються на основі маркованих наборів даних, релевантних цільовому завданню, що дозволяє їй спеціалізуватися в конкретній галузі.

Процес генерації в моделі GPT є ітеративним. Модель починає роботу з вхідних даних, які можуть бути як окремим словом, так і короткою фразою. Декодер використовує механізм самоконтролю для аналізу поточного контексту. На основі проаналізованого контексту декодер генерує розподіл ймовірностей для наступного слова в послідовності. Слово вибирається з розподілу ймовірностей або шляхом вибору найбільш ймовірного, або за допомогою техніки вибірки, яка включає певний ступінь випадковості. Вибране слово додається до поточної послідовності, і процес починається з кроку 2 – прогнозування слів, продовжуючись доти, доки не буде виконано заздалегідь визначений критерій зупинки.

Здатність моделі GPT обробляти та генерувати текст, подібний до людського, є свідченням потужності архітектур глибокого навчання. Однак важливо визнати, що вона має певні обмеження. Результати роботи моделі можуть бути упередженими через дані, на яких вона навчається, і вона може не справлятися із завданнями, що вимагають фактичної точності або реальних міркувань. Незважаючи на ці обмеження, потенціал моделі GPT незаперечний. З розвитком досліджень і вдосконаленням навчальних наборів даних можна очікувати появи ще більш досконалих можливостей генерації тексту, що прокладе шлях до прогресу у взаємодії людини з комп'ютером, творчому письмі та безлічі застосувань NLP.

Успіх GPT базується на вмілому використанні архітектури Transformer та передньому навчанні на величезних обсягах текстових даних. Однак залишається ключовим питанням: як модель GPT, навчена на ізольованих словах, здатна

вловлювати глибоке значення та взаємозв'язки в мові? В цьому контексті велике значення має концепція вкладання слів (англ. word embedding) [18]. Вкладання слів виступають як засіб перетворення дискретних даних, таких як окремі слова, у безперервний векторний простір, що сприяє глибшому розумінню мови моделями машинного навчання.

Традиційно категоріальні дані, такі як слова або ідентифікатори користувачів, представляють за допомогою однократного кодування. Цей метод присвоює кожній категорії вектор високої розмірності, в якому лише один елемент дорівнює 1, а всі інші – 0. Незважаючи на простоту, цей підхід має два ключові недоліки. Перший недолік це вектори високої розмірності можуть призвести до обчислювальної неефективності та перешкоджати роботі моделей машинного навчання. А другий при однократному кодуванні кожна категорія розглядається як абсолютно окрема, не враховуючи притаманних їй взаємозв'язків між ними.

Вкладання усувають ці обмеження шляхом перетворення категоріальних даних у векторний простір нижчої розмірності. Кожен елемент у цьому просторі, відомий як вектор вкладання, інкапсулює семантичну інформацію про відповідну категорію. Подібні категорії розташовуються ближче одна до одної у векторному просторі, що відображає властиві їм взаємозв'язки.

Існують два основні методи генерації векторів. Перший метод, відомий як вкладання слів, особливо популярний у обробці природної мови і зосереджується на представленні слів у вигляді високорозмірних семантичних векторів. Серед поширених підходів виділяється Word2Vec, який використовує дві моделі: модель безперервного мішка слів, що передбачає слово на основі контексту, та модель Skip-gram, що передбачає навколишні слова на основі заданого слова. Завдяки цим процесам слова зі схожими значеннями розташовуються близько один до одного у векторному просторі. Інший підхід, GloVe, використовує статистичну інформацію з великого текстового корпусу для навчання вкладання слів, аналізуючи шаблони повторюваності слів для виявлення семантичних зв'язків між ними [2].

Другий метод, матрична факторизація, використовується переважно для задач спільної фільтрації. У цьому підході дані про взаємодію користувачів з

елементами представлені у вигляді розрідженої матриці. Матрична факторизація спрямована на розкладання цієї матриці на дві матриці меншої розмірності, одна з яких представляє вкладання користувачів, а інша – вкладання елементів. Отримані вектори відображають вподобання користувачів та характеристики товарів, що дозволяє моделі надавати персоналізовані рекомендації.

На рис. 2.1 наведено візуалізацію перетворення текстового виразу на семантичний вектор використовуючи модель вкладання слів [12].

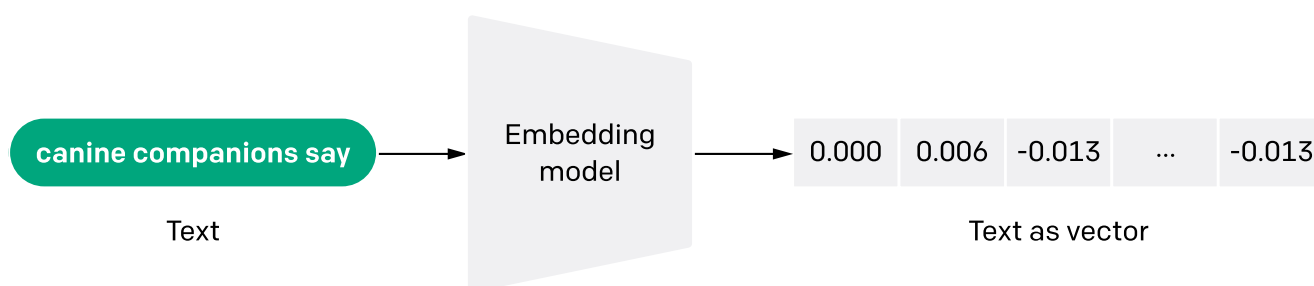


Рисунок 2.1 – Візуалізація перетворення текстового виразу на семантичний вектор використовуючи вкладання слів

Джерело: [12]

Одним з найбільш популярних застосувань вкладання слів є його використання у сфері семантичного пошуку. Семантичний пошук є технологією, яка використовується в інформаційно-пошукових системах для знаходження релевантної інформації, враховуючи контекстуальне значення та семантичну подібність між запитом користувача та наявними даними. Цей підхід виходить за рамки класичних методів пошуку за ключовими словами, оскільки він розуміє основні концепції та взаємозв'язки в тексті. Ключовим елементом семантичного пошуку є використання моделей Embeddings і семантичних векторів. Детальний огляд того, як працює семантичний пошук з використанням цих технологій, демонструє їхню ефективність у визначенні схожих за змістом текстових даних і забезпеченні контекстуально релевантних результатів для користувачів:

1) перший крок це попередня обробка текстових даних, які потрібно проіндексувати для пошуку. Зазвичай це включає в себе токенізацію, видалення стоп-слів, стемінг або лематизацію та інші методи очищення тексту;

2) попередньо оброблений текст пропускається через попередньо навчену мовну модель, таку як Word2Vec, GloVe, BERT або GPT, яка генерує вставки для кожного слова, фрази або документа. Ці вставки є щільними числовими векторами, які фіксують семантичну та контекстну інформацію тексту [1];

3) вкладення окремих слів або фраз об'єднуються для створення єдиного семантичного вектора, що представляє весь документ або сегмент тексту. Цей семантичний вектор відображає загальний зміст і контекст тексту;

4) семантичні вектори всіх документів у корпусі зберігаються в ефективній структурі даних для індексування, такій як приблизний індекс найближчого сусіда або векторна база даних. Цей індекс дозволяє здійснювати швидкий пошук схожості між семантичними векторами;

5) коли користувач надсилає пошуковий запит, та сама мовна модель використовується для створення вкладання слів в текст запиту. Це вкладання слів запиту відображає семантичне значення інформаційної потреби користувача;

6) вкладення запиту порівнюється з семантичними векторами проіндексованих документів за допомогою метрики схожості, наприклад, косинусної схожості або евклідової відстані. Документи, семантичні вектори яких найближчі до вложеного запиту, вважаються найбільш релевантними результатами;

7) система фільтрує знайдені документи на основі їхньої схожості і повертає документи з найвищим рейтингом як результати пошуку. В алгоритм фільтрування також можуть бути включені додаткові фактори ранжування, такі як популярність або свіжість документа.

Ключовою перевагою семантичного пошуку над традиційним пошуком за ключовими словами є його здатність розуміти контекстне значення і взаємозв'язки між словами, фразами і поняттями. Це дозволяє отримувати більш точні та

релевантні результати пошуку, навіть якщо запит і відповідні документи не мають багато точних збігів за ключовими словами.

Семантичний пошук особливо корисний у сферах, де мова є багатозначною, неоднозначною або залежною від контексту, наприклад, у юридичних документах, медичній літературі або наукових роботах. Він також може покращити користувацький досвід у загальному вебпошуку, пошуку продуктів електронної комерції та системах управління знаннями підприємства, надаючи більш інтуїтивно зрозумілі та змістовні результати пошуку.

Вбудовування універсальний інструмент для різних завдань машинного навчання. До найвідоміших застосувань можна віднести обробку природної мови, системи рекомендацій та комп'ютерний зір. В обробці природної мови вбудовування слів мають фундаментальне значення для таких завдань, як машинний переклад, аналіз настроїв та узагальнення тексту. Захоплюючи семантичні зв'язки, вставки дозволяють моделям розуміти нюанси мови. Вбудовування користувачів і предметів, отримані з даних про взаємодію користувача з предметом, дають змогу рекомендаційним системам пропонувати релевантні предмети користувачам на основі їхніх минулих вподобань і притаманних їм характеристик. Вбудовування можна використовувати для представлення зображень або характеристик зображень, що дозволяє вирішувати такі завдання, як пошук зображень і розпізнавання об'єктів. Подібні зображення матимуть близькі вектори вбудовування у векторному просторі.

Підсумовуючи, вбудовування слугують потужним мостом між символічним світом даних та числовим світом моделей машинного навчання. Перетворюючи дискретні дані в безперервний векторний простір, вбудовування дозволяють моделям відображати семантичні зв'язки і відкривають ширший спектр застосувань. З розвитком досліджень у цій галузі можна очікувати появи ще більш досконалих методів вкладання слів, які ще більше збагатять сферу машинного навчання.

Модель GPT і модель Embeddings представляють дві ключові компоненти в області просунутих методів обробки природної мови. Використовуючи свою

потужну архітектуру та переднє навчання, GPT створює текст, що максимально наближений до людського. У той же час, вкладання слів відіграють важливу роль у розумінні семантичних взаємозв'язків у мові. Вони перетворюють дискретні дані на числові представлення, що дозволяє моделям машинного навчання обробляти та генерувати мову з вищою ступеню складності.

Оскільки дослідження в області глибоких нейронних мереж продовжують активно розвиватися, можна очікувати значних досягнень у сфері обробки природної мови. Це відкриває нові можливості для розробки нових додатків і глибшого розуміння людської комунікації.

2.2 Проєктування вебсервісу

Розробка вебсервісу передбачає врахування різних чинників, включаючи досвід користувача, масштабованість, безпеку та інтеграцію із зовнішніми сервісами. Мета - створити зручну платформу, яка дозволить користувачам завантажувати свої текстові документи, отримувати глибокий аналіз та інтерактивно взаємодіяти з системою через чат інтерфейс.

У процесі проєктування вебсервісу одним з основоположних рішень є вибір вебфреймворку. Цей вибір суттєво вплине на процес розробки, можливості сервісу та його продуктивність. Три популярні вебфреймворки, які виділяються на сучасному ландшафті - Next.js, Nuxt та SvelteKit. Кожен з цих фреймворків має свої унікальні переваги і може бути більш придатним для певних типів проєктів.

Next.js це надійний фреймворк на основі бібліотеки React, що пропонує серверний рендеринг і статичну генерацію сайтів. Відомий своєю гнучкістю та сумісністю з широким спектром браузерів [16]. Nuxt є більш незалежним фреймворком, заснованим на бібліотеці Vue.js. Забезпечує більш визначені конвенції та стандарти, що може спростити та пришвидшити процес розробки [17]. SvelteKit - це мета-фреймворк, побудований на основі легкої бібліотеки Svelte. Virізняється своєю продуктивністю та простотою коду завдяки реактивній моделі даних [23].

Для здійснення об'єктивного порівняння між Next.js, Nuxt і SvelteKit, було враховано кілька важливих параметрів. Результати цього порівняння представлені у таблиці 2.1.

Таблиця 2.1 – Порівняння фреймворків Next.js, Nuxt та SvelteKit

Функція	Next.js	Nuxt.js	SvelteKit
Базова бібліотека	React	Vue	Svelte
Маршрутизація	Директорія-заснована, API-заснована	Директорія-заснована, Nuxt-специфічні маршрути	Директорія-заснована
Рендеринг	Серверний, Клієнтський, Статичний	Серверний, Клієнтський, Універсальний	Серверний, Клієнтський, Статичний
Розширення	Custom Next.js API, сторонні бібліотеки React	Nuxt модулі, сторонні бібліотеки Vue	SvelteKit adapters, сторонні бібліотеки Svelte
Розгортання	Vercel, Netlify, AWS Amplify	Netlify, AWS Amplify, Vercel	Vercel, Netlify, AWS Amplify
Переваги	Велика спільнота, багатий набір функцій, добре документований	Гнучкість, зручність використання, інтеграція з Nuxt ecosystem	Мала вага, висока продуктивність, компоненти Svelte
Недоліки	Складніша крива навчання, більший розмір пакета	Менша спільнота, менша кількість сторонніх бібліотек	Менша екосистема, менша кількість документації
Використання	Електронна комерція, SaaS, маркетингові сайти	Блоги, форуми, односторінкові вебдодатки	Мобільні вебдодатки, високопродуктивні вебсайти

Джерело: запропоновано автором

З усіх вказаних параметрів, Next.js виявляється провідним у більшості категорій. Цей фреймворк ґрунтується на бібліотеці React, що є однією з найпопулярніших для розробки вебдодатків, та має значну спільноту розробників. Власний API Next.js і підтримка великої кількості сторонніх бібліотек спрощують процес інтеграції нових функцій. Крім того, фреймворк відрізняється великою спільнотою користувачів, розширеним набором функціональностей і добре структурованою документацією, що робить його перевагою для розробки веб-сервісів.

Оскільки Next.js підтримує дві мови програмування JavaScript та TypeScript необхідно обрати яку саме мову використовувати. Для цього в табл. 2.2 наведено порівняння за головними параметрами, які важливі при розробці вебсервісу.

Таблиця 2.2 – Порівняння мов програмування JavaScript та TypeScript

Параметр	JavaScript	TypeScript
Типізація	Динамічно типізована	Статично типізована
Помилки під час компіляції	Помилки виявляються під час виконання	Помилки виявляються під час компіляції
Функціональна сумісність	Безперешкодно працює з існуючими бібліотеками JavaScript	Повністю сумісний з кодом JS
Підтримка інструментів	Велика екосистема інструментів, але може вимагати ручного налаштування	Відмінна інтеграція з IDE та інструменти перевірки типів
Продуктивність	Вважається трохи швидшою за TypeScript через відсутність етапу компіляції	Продуктивність виконання майже ідентична до JavaScript

Джерело: запропоновано автором

JavaScript є найпопулярнішою мовою для веброзробки і пропонує знайомий синтаксис для тих, хто має досвід програмування. Її динамічна природа забезпечує швидку розробку, але може призвести до помилок під час виконання, які буває важко відстежити.

TypeScript, підмножина JavaScript, додає до мови необов'язкову статичну типізацію. Це покращує зручність супроводу коду та зменшує кількість помилок під час виконання, відловлюючи їх на етапі компіляції. Однак TypeScript вимагає компіляції перед запуском коду, що додає додатковий крок до процесу розробки.

Для цього проєкту обрано TypeScript як основну мову розробки. Хоча JavaScript на початковому етапі пропонує швидший цикл розробки, переваги TypeScript переважають недоліки в проєкті такого масштабу. Статична типізація значно покращить зручність обслуговування коду та зменшить ймовірність помилок під час виконання складного вебсервісу. Крім того, чудова інструментальна підтримка TypeScript покращить досвід розробки.

Наступним важливим кроком після вибору фреймворку та мови програмування є визначення способу завантаження та зберігання документів, зокрема PDF-файлів. Для цього є кілька сервісів, кожен з яких має свій набір функцій та міркувань.

Одним із найвідоміших варіантів є Amazon S3, широко використовуваний хмарний сервіс зберігання даних, який пропонує масштабоване, надійне та безпечне зберігання різних типів даних, включаючи документи. За допомогою Amazon S3 користувачі можуть завантажувати свої PDF-документи безпосередньо в хмару, а сервіс надає унікальну URL-адресу для кожного збереженого файлу. Цю URL-адресу можна зберігати в базі даних для легкого пошуку та управління [4].

Ще один варіант, який варто розглянути, - це Google Cloud Storage, який надає функції, подібні до Amazon S3. Пропонує масштабоване сховище для великих об'єктів, включаючи PDF-файли, з можливістю доступу до них і управління ними програмно за допомогою API-інтерфейсів. Google Cloud Storage також забезпечує високу доступність і довговічність збережених даних [7].

Альтернативно, Microsoft Azure Blob Storage є ще одним претендентом у цьому просторі. Azure Blob Storage дозволяє користувачам зберігати великі обсяги неструктурованих даних, включаючи PDF-документи, і надає такі функції, як шифрування, контроль доступу та інтеграція з іншими службами Azure [6].

Кожен сервіс зберігання даних має свої переваги та особливості. Amazon S3, наприклад, широко використовується і пропонує широкий набір функцій, включаючи управління версіями, життєвим циклом і політиками контролю доступу. Його модель ціноутворення за принципом «плати, як користуєшся» робить його придатним для проєктів різного масштабу, хоча витрати можуть зростати зі збільшенням обсягу використання.

З іншого боку, Google Cloud Storage забезпечує тісну інтеграцію з іншими сервісами Google Cloud Platform, що може бути корисним, якщо проєкт використовує інші хмарні сервіси Google. Воно також пропонує такі функції, як керування версіями об'єктів та управління життєвим циклом об'єктів. З точки зору вартості, Google Cloud Storage, як правило, є конкурентоспроможним, і залежно від конкретних вимог, він може запропонувати варіанти економії коштів, такі як класи зберігання Nearline і Coldline.

Microsoft Azure Blob Storage, як частина екосистеми Azure, пропонує безперешкодну інтеграцію з іншими службами Azure. Вона надає такі функції, як

Azure Storage Explorer для зручного керування, керування версіями Blob та управління життєвим циклом Blob. Ціна може змінюватися залежно від варіантів резервування сховища та вибору рівня доступу.

Рішення про те, який сервіс використовувати, має враховувати не лише технічні аспекти, але й такі фактори, як підтримка, документація та екосистема інструментів і сервісів, доступних на кожній платформі. Крім того, наявність клієнтських бібліотек та SDK для інтеграції з обраним сервісом може спростити процес розробки.

Таблиця 2.3 – Порівняння вебсервісів Amazon S3, Google Cloud Storage та Microsoft Azure Blob Storage

Параметр	Amazon S3	Google Cloud Storage	Microsoft Azure Blob Storage
Масштабованість	Висока масштабованість з практично необмеженою ємністю	Висока масштабованість з можливістю вибору регіона	Висока масштабованість з можливістю вибору температурних режимів
Безпека	Пропонує шифрування, контроль доступу та цілісність даних	Забезпечує шифрування, контроль доступу та політики IAM	Пропонує шифрування, контроль доступу та інтеграцію з Azure AD
Доступність	Забезпечує високу доступність з SLA для часу роботи	Пропонує високу доступність з міцною мережевою інфраструктурою	Забезпечує високу доступність з договорами рівня обслуговування
Інтеграція з іншими сервісами	Добре інтегрується з екосистемою AWS та інструментами сторонніх розробників	Має сильну інтеграцію з послугами Google Cloud Platform	Забезпечує безшовну інтеграцію з іншими службами Azure
Вартість ефективності	Пропонує конкурентну цінову політику з платою за фактом використання	Конкурентоспроможні ціни з можливістю вибору економічних класів	Вартість варіюється в залежності від видів збереження та рівня доступу
Управління життєвим циклом об'єктів	Надає політики життєвого циклу для автоматизованого керування даними	Пропонує керування життєвим циклом об'єктів для оптимізації витрат	Підтримує управління життєвим циклом BLOB для автоматизованих політик

Джерело: запропоновано автором

Сервіс Amazon S3 вирізняється серед трьох сервісів завдяки своїм великим можливостям, надійній масштабованості та широким можливостям інтеграції в екосистемі AWS. Він пропонує високу надійність, безпеку та доступність, що робить його підходящим для різноманітних застосувань і завдань. Додатково, його цінова політика з оплатою за фактом використання забезпечує ефективність та гнучкість. Тим часом, хоча Google Cloud Storage та Microsoft Azure Blob Storage пропонують подібні можливості, лідерство Amazon S3 очевидно завдяки його комплексному набору функцій, надійності та визнаній репутації в галузі.

Щоб забезпечити ефективне управління даними для вебсервісу, необхідно ретельно вивчити технології баз даних. Існує дві основні категорії баз даних: SQL і NoSQL. Для цього проєкту вибір між SQL та NoSQL залежить від кількох факторів:

- структура даних, для даних із чіткою структурою рекомендується використання SQL-баз даних. У випадках, коли дані мають неструктуровану або напівструктуровану форму, таку як текстові документи, більш підходящими є NoSQL-рішення.

- масштабованість, бази даних NoSQL зазвичай краще масштабуються по горизонталі, що робить їх придатними для великомасштабних додатків, де обсяги даних можуть швидко зростати. Однак, бази даних SQL також можуть обробляти значні обсяги даних при належній оптимізації.

- складні запити, бази даних SQL відмінно справляються зі складними запити, що включають кілька таблиць і зв'язків. Якщо програма вимагає широких можливостей запитів, SQL може бути кращим вибором.

Вибір системи управління базами даних для вебсервісу залежить від різних факторів, таких як характер даних, очікуваний трафік і специфічні вимоги вебсервісу. У контексті запропонованого вебсервісу розглядаються три популярні реляційні системи управління базами даних з відкритим вихідним кодом - MySQL, MySQL, завдяки своєму широкому набору функцій, підтримці з боку спільноти та простому налаштуванню, часто вважається оптимальною універсальною системою для більшості вебдодатків. Вона вирізняється високою продуктивністю, надійністю

та простотою використання. MySQL підтримує широкий спектр типів даних та надає надійні засоби безпеки, що робить її підходящим вибором для вебсервісів, які обробляють конфіденційні дані користувачів, такі як особиста інформація та PDF-документи.

PostgreSQL відома своєю високою здатністю до налаштування та дотриманням стандартів SQL. Вона підтримує широкий спектр типів даних, включаючи ті, яких немає в MySQL. Синтаксис PostgreSQL наближений до стандартного SQL, що може бути корисним для перенесення навичок на інші системи управління базами даних.

SQLite це легка система управління базами даних, яка не потребує серверного процесу, що забезпечує швидшу роботу. Популярний вибір для додатків, які потребують вбудованих баз даних замість клієнт-серверної архітектури. Однак SQLite може бути не найкращим вибором для вебсервісів, що очікують високий трафік або повинні обробляти складні запити, оскільки їй бракує деяких розширених функцій, які надають MySQL та PostgreSQL.

Таблиця 2.4 – Порівняння реляційних баз даних MySQL, PostgreSQL та SQLite [21]

Функціональність	MySQL	PostgreSQL	SQLite
Простота використання	MySQL відомий своєю простотою налаштування та широкою підтримкою спільноти.	PostgreSQL має більш крутий кут навчання порівняно з MySQL.	SQLite простий у використанні, але не має деяких розширених функцій.
Продуктивність	MySQL має перевагу над PostgreSQL для простих операцій зчитування за пропускну здатністю та продуктивністю.	PostgreSQL високо налаштовується і дотримується стандартів SQL.	SQLite швидкий для додатків, яким потрібні вбудовані бази даних.
Масштабованість	MySQL підтримує багатопоточність, що робить його високо масштабованим.	Масштабованість PostgreSQL порівнянна з MySQL.	SQLite не призначений для високої масштабованості.
Розширені функції	MySQL має великий набір функцій.	PostgreSQL підтримує широкий спектр типів даних та розширених функцій.	SQLite не має деяких розширених функцій, які надають MySQL та PostgreSQL.

Продовження таблиці 2.4

Розділення	MySQL підтримує розділення, що покращує продуктивність великої бази даних.	PostgreSQL також підтримує розділення.	SQLite не підтримує розділення.
Типи даних	Стандартні типи даних SQL	Більш просунуті типи даних	Стандартні типи даних SQL

Джерело: [21]

MySQL є лідером серед цих СУБД завдяки простоті використання, високій продуктивності, масштабованості та розширеному набору функцій. Підтримка розбиття на розділи робить її чудовим вибором для роботи з великими базами даних, такими як та, що необхідна для описаного вебсервісу. Крім того, широка підтримка спільноти MySQL гарантує, що допомога буде легкодоступною, коли вона знадобиться. Таким чином, MySQL є підходящим вибором для описуваного вебсервісу, враховуючи його вимоги та характер даних, які він обробляє.

Окрім зберігання текстових даних і метаданих у реляційних базах даних, таких як MySQL, векторні бази даних необхідні для ефективного управління та запитів до високорозмірних семантичних векторів, створених на основі документів. При виборі векторної бази даних необхідно враховувати кілька факторів, зокрема масштабованість, продуктивність запитів і простоту інтеграції з існуючою архітектурою.

Вибір векторної бази даних залежить від кількох факторів, зокрема від конкретних вимог сервісу, масштабу даних і бажаної продуктивності. Три векторні бази даних, що розглядаються - Pinecone, Milvus та Weaviate - мають свої сильні сторони та компроміси.

Pinecone представляє собою кероване рішення, що забезпечує можливості пошуку в реальному часі та відзначається простотою використання. Особливо підходить для стартапів і компаній, які надають перевагу швидкому розвитку та масштабованості. Кероване рішення Pinecone є корисним для вебсервісів, які потребують безперебійної інтеграції з мінімальними зусиллями на налаштування та обслуговування.

Milvus є інструментом з відкритим вихідним кодом, який забезпечує тонкий контроль над продуктивністю та кастомізацією. Розробники, яким зручно працювати з інструментами з відкритим вихідним кодом і які потребують більшого контролю над своєю базою даних, можуть віддати перевагу Milvus. Він може добре підійти для вебсервісу, який вимагає специфічних налаштувань і має розробників, які бажають самостійно керувати базою даних.

Weaviate є ще одним варіантом з відкритим вихідним кодом, який ефективно керує зв'язками між сутностями даних. Вона може бути оптимальним вибором для служб аналізу тексту, що включають не лише документи, але й сутності та їхні взаємозв'язки. Weaviate надає інтерфейс GraphQL для доступу до даних та добре інтегрується з різними фреймворками для frontend розробки. Проте функціонал векторного пошуку у Weaviate може бути менш розвиненим у порівнянні з Pinecone або Milvus.

Вибір між Pinecone, Milvus та Weaviate залежить від конкретних вимог, а саме: простота використання, масштабованість, вартість, кастомізація. Кожна з цих систем пропонує різні можливості і функціонал, що робить їх більш підходящими для різних сценаріїв використання.

Pinecone виявляється найбільш підходящим рішенням для векторної бази даних для запропонованого вебсервісу. Підхід керованого сервісу Pinecone спрощує процес налаштування та обслуговування, що робить його привабливим вибором для швидкого розгортання та інтеграції в існуючий фреймворк Next.js. Простий API добре узгоджується з середовищем розробки та сприяє ефективній інтеграції з моделлю GPT, що використовується у вебсервісі, сприяє безперешкодному обміну даними та дозволяє ефективно використовувати розширені можливості аналізу тексту.

Економічна ефективність добре узгоджується з потенційним зростанням сервісу, а інтеграція з фреймворками машинного навчання розширює можливості розширеного аналізу тексту. Отже, поєднання простоти використання, масштабованості, економічної ефективності та інтеграції з машинним навчанням робить Pinecone доцільним вибором для вебсервісу аналізу текстових документів.

2.3 Проєктування бази даних

Проєктування бази даних є критичним етапом у розробці інформаційних систем, оскільки воно визначає структуру даних, яка забезпечує ефективне зберігання, пошук та маніпуляцію інформацією. Від якості проєктування бази даних залежать її продуктивність, масштабованість, цілісність даних та легкість їх обслуговування. Теоретична база, що лежить в основі проєктування БД, включає концепції моделювання даних, нормалізації, забезпечення цілісності та управління транзакціями.

Процес проєктування БД являє собою послідовність переходів від неформального словесного опису інформаційної структури предметної області до формалізованого опису об'єктів предметної області в термінах деякої моделі та фізичної реалізації бази даних в середовищі вибраної СУБД. В загальному випадку проєктування можна розділити на такі послідовні етапи:

1) перший етап проєктування бази даних полягає у визначенні та зборі вимог до системи. На цьому етапі проводяться зустрічі з зацікавленими сторонами, включаючи кінцевих користувачів, аналітиків та менеджерів проєкту, для з'ясування потреб та очікувань від бази даних. Основними завданнями цього етапу є: визначення інформаційних потреб користувачів, опис бізнес-процесів та функціональних вимог, визначення типів даних, які будуть зберігатися та визначення обсягів даних та очікуваних навантажень на систему;

2) інфологічне моделювання полягає у створенні моделі даних на високому рівні абстракції, що відображає інформаційні потреби організації. Модель є незалежною від конкретних технологій і платформ, фокусуючись на сутностях та їх атрибутах з точки зору бізнесу. Основні завдання: визначення основних інформаційних об'єктів та їх властивостей, опис бізнес-правил, що регулюють взаємодію між інформаційними об'єктами, створення інформаційної моделі, яка відображає структуру та семантику даних на бізнес-рівні;

3) концептуальне моделювання передбачає побудову концептуальної схеми бази даних, що деталізує інфологічну модель. Найпоширенішим інструментом для цього етапу є модель сутність-зв'язок;

4) логічне моделювання полягає у перетворенні концептуальної моделі в логічну структуру, яка може бути реалізована в конкретній системі управління базами даних. На цьому етапі визначаються таблиці, первинні та зовнішні ключі, їх типи даних;

5) фізичне моделювання полягає в оптимізації логічної моделі для конкретного середовища СУБД з урахуванням апаратного забезпечення, операційної системи та вимог до продуктивності. Основними завданнями є: вибір типів зберігання даних та індексів, оптимізація схем таблиць для покращення продуктивності, налаштування параметрів СУБД для забезпечення ефективного зберігання та доступу до даних та розробка стратегії резервного копіювання та відновлення даних.

Кожен з цих етапів є важливим і вимагає ретельного планування та виконання для створення ефективної та надійної бази даних.

Користувачам вебсервісу потрібна система, яка підтримує безпечний вхід, завантаження документів та інтерактивні запити з моделлю GPT-4. Основні визначені інформаційні потреби включають:

- аутентифікація та авторизація користувачів;
- зберігання та пошук завантажених документів;
- генерація та зберігання семантичних векторів для документів і запитів користувачів;
- ефективний пошук та вилучення інформації на основі семантичних векторів;

Основні бізнес-процеси включають реєстрацію та вхід користувачів, завантаження документів, аналіз документів та інтерактивні запити. Функціональні вимоги, що впливають з цих процесів, є наступними:

- безпечна та ефективна система управління користувачами;
- інтеграція з Amazon S3 для зберігання документів;

- використання OpenAI API для створення семантичних векторів та інтерактивних відповідей;

- надійний механізм пошуку для полегшення швидкого отримання релевантного вмісту документів;

Типи даних, що зберігаються в базі даних, поділяються на наступні категорії:

- інформація, що стосується облікових записів користувачів, наприклад, ім'я користувача, адреса електронної пошти та дані профілю користувача;

- метадані про завантажені документи, включаючи URL-адреси з Amazon S3, назви документів, мітки часу завантаження та асоціації користувачів;

- запити користувачів і відповіді з моделі GPT-4, а також пов'язані з ними метадані, такі як часові мітки та ідентифікатори сеансів.

Дизайн бази даних повинен забезпечувати масштабованість, щоб відповідати зростанню кількості користувачів та обсягу документів. Стратегії оптимізації продуктивності включають індексування ключових полів, ефективні алгоритми пошуку даних та належне використання векторних баз даних для зберігання та запитів на вставки.

Перший етап проєктування бази даних для вебсервісу аналізу текстових документів забезпечив чітке розуміння вимог та очікувань. Ці знання будуть використані на наступних етапах розробки схеми бази даних, щоб забезпечити відповідність системи потребам користувачів та її ефективну роботу під очікуваними навантаженнями. Наступні кроки передбачають перетворення цих вимог у детальну схему бази даних та впровадження необхідної інфраструктури для підтримки визначених бізнес-процесів.

Другий етап проєктування бази даних для вебсервісу аналізу текстових документів передбачає інфологічне моделювання. Цей етап зосереджений на створенні високорівневої моделі даних, яка інкапсулює інформаційні потреби організації. Модель є абстрактною і незалежною від конкретних технологій, акцентуючи увагу на сутностях, їхніх атрибутах і бізнес-правилах.

Основними інформаційними об'єктами, визначеними для вебсервісу, є Користувач, Документ та Повідомлення. Кожна сутність представляє критичний аспект бізнес-процесу і має специфічні властивості, пов'язані з ним.

Користувач представляє осіб, які використовують вебсервіс. Атрибути: ідентифікатор користувача, адреса електронної пошти, URL-адреса зображення профілю.

Документ представляє PDF-документ, завантажений користувачем. Атрибути: ідентифікатор документа, назва документа, URL-адреса за якою зберігається документ, унікальний ключ Amazon S3, статус завантаження, дата створення, дата оновлення.

Повідомлення відображає взаємодію між користувачем і моделлю GPT. Атрибути: ідентифікатор повідомлення, текстовий вміст повідомлення, дата створення, дата оновлення, ідентифікатор визначення чи є повідомлення користувача або системи.

Наступним кроком є концептуальне моделювання, яке трансформує інфологічну модель у більш абстрактну та узагальнену структуру, що відображає основні принципи та логіку роботи системи. Основним інструментом для цього етапу є ER-модель, яка чітко відображає сутності, їх атрибути та зв'язки між ними.

Кожен користувач може завантажувати багато PDF-файлів, але кожен файл може бути пов'язаний лише з одним користувачем. Це зберігає цілісність даних, гарантуючи, що файл завжди чітко пов'язаний з конкретним користувачем, який його завантажив та забезпечує ефективний пошук файлів, пов'язаних з певним користувачем. У сутності Документ буде атрибут який міститиме ідентифікатор користувача, який завантажив файл.

Зв'язок один до багатьох між сутностями Користувач та Документ. Кожен користувач може завантажувати багато PDF-файлів, але кожен файл може бути пов'язаний лише з одним користувачем. Це зберігає цілісність даних, гарантуючи, що файл завжди чітко пов'язаний з конкретним користувачем, який його завантажив та забезпечує ефективний пошук файлів, пов'язаних з певним

користувачем. У сутності Документ буде атрибут `userId` (зовнішній ключ), який міститиме ідентифікатор користувача, який завантажив файл.

На рисунку 2.2. наведена узагальнена інфологічна модель бази даних.

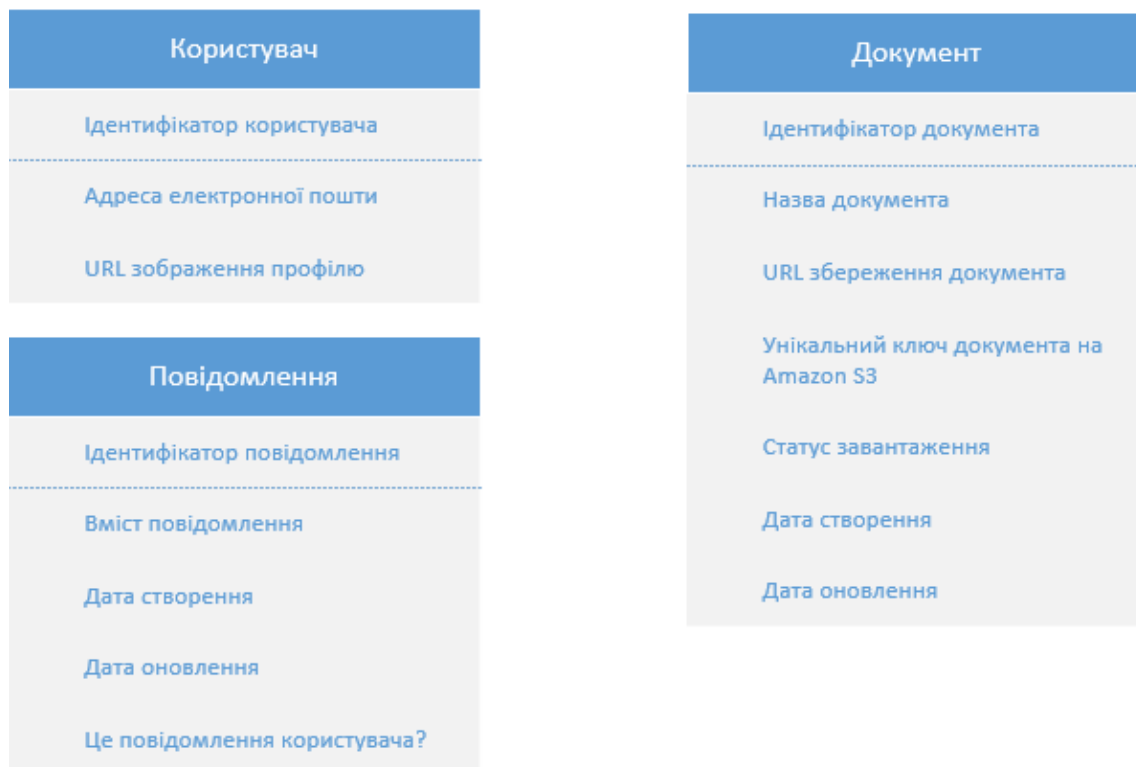


Рисунок 2.2 – Інфологічна модель бази даних

Джерело: запропоновано автором

Зв'язок один до багатьох між сутностями Документ та Повідомлення. Один файл може мати багато пов'язаних з ним повідомлень, але одне повідомлення може бути пов'язане лише з одним файлом. Це дозволяє зберігати історію спілкування користувача з моделлю GPT, пов'язану з певним файлом. Забезпечує контекст для повідомлень, дозволяючи користувачам розуміти, до чого стосується те чи інше повідомлення. У сутності Повідомлення буде атрибут `fileId` (зовнішній ключ), який буде містити ідентифікатор файлу, з яким воно пов'язане.

Зв'язок один до багатьох між сутностями Користувач та Повідомлення. Кожен користувач може створювати багато повідомлень, але одне повідомлення може бути створене лише одним користувачем. Це відповідає основному сценарію використання, де користувачі спілкуються з моделлю GPT. Зберігає цілісність

даних, гарантуючи, що кожне повідомлення чітко пов'язане з користувачем, який його створив. Дозволяє відстежувати активність користувачів та аналізувати їхні взаємодії з моделлю. У сутності Повідомлення буде атрибут `userId` (зовнішній ключ), який буде містити ідентифікатор користувача, який його створив.

Описані детальні зв'язки між сутностями в концептуальній моделі бази даних забезпечують гнучку та ефективну структуру для зберігання даних вебсервісу. Вибір зв'язків один до багатьох відповідає сценаріям використання системи. Модель є основою для подальшого логічного та фізичного проектування бази даних. На рисунку 2.3 представлено концептуальну модель бази даних вебсервісу.

На основі концептуальної моделі, описаної раніше, тепер можна перейти до логічного моделювання. Цей етап передбачає трансформацію абстрактних сутностей та зв'язків концептуальної моделі в чітку та детальну структуру, яка може бути реалізована в конкретній системі управління базами даних.



Рисунок 2.3 – Концептуальна модель бази даних

Джерело: запропоновано автором

На основі концептуальної моделі, описаної раніше, тепер можна перейти до логічного моделювання. Цей етап передбачає трансформацію абстрактних сутностей та зв'язків концептуальної моделі в чітку та детальну структуру, яка може бути реалізована в конкретній системі управління базами даних (СУБД).

У даній логічній моделі буде створено три таблиці:

1) Користувач (User):

- id: унікальний ідентифікатор користувача - первинний ключ;
- email: адреса електронної пошти - унікальний ключ;
- imageUrl: URL-адреса зображення профілю.

2) Файл (File):

- id: унікальний ідентифікатор файлу - первинний ключ;
- name: назва файлу;
- url: URL-адреса файлу;
- key: унікальний ключ Amazon S3;
- uploadStatus: статус завантаження;
- createdAt: дата створення;
- updatedAt: дата оновлення;
- userId: зовнішній ключ до таблиці User.

3) Повідомлення (Message):

- id: унікальний ідентифікатор повідомлення - первинний ключ;
- text: текстовий вміст повідомлення;
- isUserMessage: чи є повідомленням користувача;
- createdAt: дата створення;
- updatedAt: дата оновлення;
- userId: зовнішній ключ до таблиці User;
- fileId: зовнішній ключ до таблиці File.

Зв'язки між таблицями:

- один до багатьох: користувач може мати багато файлів (userId в File посилається на id в User);

- один до багатьох: файл може мати багато повідомлень (fileId в Message посилається на id в File);
- один до багатьох: користувач може створювати багато повідомлень (userId в Message посилається на id в User).

Типи даних, які використовуються в базі даних:

- varchar: використовується для коротких текстових рядків;
- enum: використовується для перелічуваних значень;
- datetime: використовується для позначення дат і часу;
- text: використовується для довгих текстових рядків;
- boolean: використовується для логічних значень.

На рисунку 2.4 представлена логічна модель бази даних.

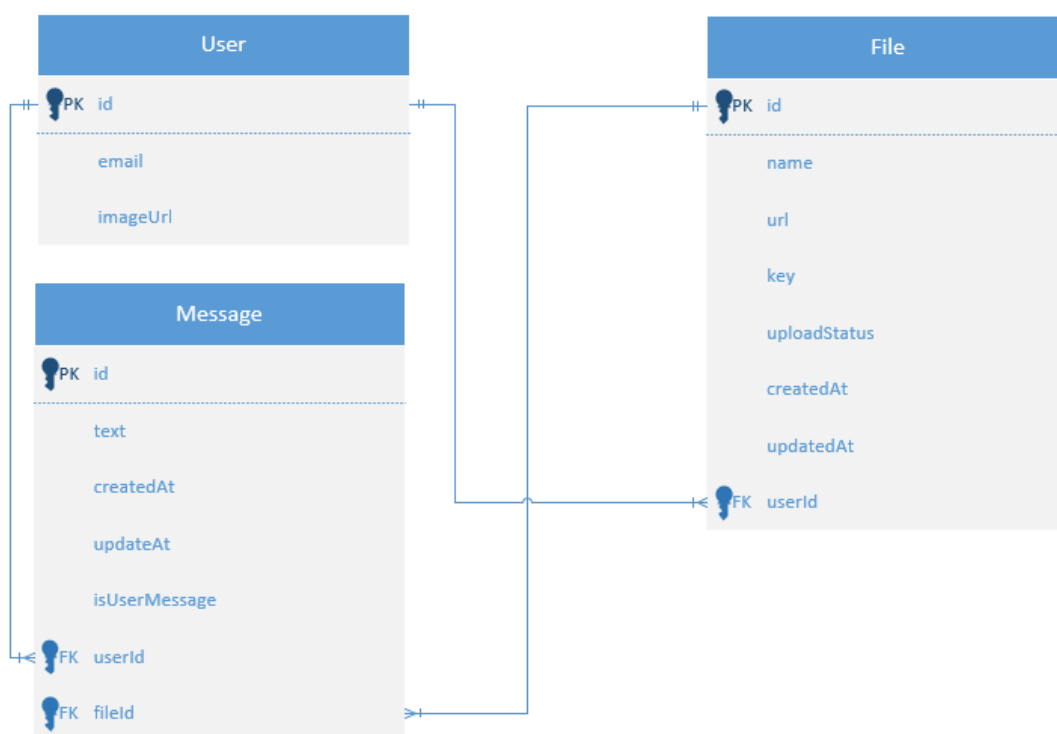


Рисунок 2.4 – Логічна модель бази даних

Джерело: запропоновано автором

Логічна модель бази даних, описана вище, забезпечує детальну та реалістичну структуру для вебсервісу. Вона визначає таблиці, атрибути, їх типи даних та зв'язки між ними. Модель слугує основою для подальшого фізичного

проектування, де будуть визначені конкретні технічні деталі реалізації бази даних в обраній СУБД.

Наступним кроком є фізичне проектування бази даних, де будуть визначені конкретні деталі реалізації. Логічна модель бази даних, описана раніше, слугує основою для фізичного проектування, де будуть визначені конкретні технічні деталі реалізації в обраній СУБД. У цьому проєкті використовується Prisma, це інструмент для роботи з базами даних у TypeScript. Використання Prisma з фізичним проектуванням бази даних забезпечує ряд переваг. Абстракція від бази даних, Prisma дозволяє взаємодіяти з базою даних незалежно від конкретної СУБД, спрощуючи розробку та підтримку. Автоматична генерація коду, генерує код для доступу до даних, CRUD операцій та взаємодії зі зв'язками, що економить час розробки. Безпека типів, допомагає запобігти помилкам під час роботи з даними.

Використовуючи Prisma, можна створити таблиці, відповідно до логічної моделі. Prisma автоматично генерує код для реалізації зв'язків між таблицями, описаних у схемі. Код дозволяє легко отримувати доступ до пов'язаних даних та оновлювати їх. На рисунку 2.5 зображено створену базу даних у графічному інтерфейсі DBeaver.

User	
id	varchar(191)
email	varchar(191)
imageUrl	varchar(191)

Message	
id	varchar(191)
text	text
userId	varchar(191)
fileId	varchar(191)
isUserMessage	tinyint(1)
createdAt	datetime(3)
updatedAt	datetime(3)

File	
id	varchar(191)
name	varchar(191)
url	varchar(191)
key	varchar(191)
uploadStatus	enum('PENDING','PROCESSING','FAILED','SUCCESS')
createdAt	datetime(3)
updatedAt	datetime(3)
userId	varchar(191)

Рисунок 2.5 – База даних в графічному інтерфейсі DBeaver

Джерело: запропоновано автором

Фізичне проектування бази даних з використанням Prisma забезпечує ефективну реалізацію логічної моделі. Вибір СУБД, налаштування Prisma та використання Prisma Client дозволяють створити гнучку та масштабовану базу даних для вебсервісу.

РОЗДІЛ 3

РОЗРОБКА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБСЕРВІСУ ДЛЯ АВТОМАТИЧНОГО АНАЛІЗУ ТЕКСТОВИХ ДОКУМЕНТІВ

3.1 Планування робіт

Розробка та програмна реалізація вебсервісу вимагає комплексного та системного підходу до планування. Процес розробки складається з трьох основних етапів: постановка задачі, розробка та розгортання вебсервісу. Кожен етап включає низку завдань із визначеними термінами виконання.

Перший етап передбачає визначення основних цілей та вимог до вебсервісу. Розпочинається з формулювання основних завдань, які має вирішувати вебсервіс, і триває три дні, з 1 січня 2024 року по 3 січня 2024 року. Після цього, протягом одного дня, 4 січня 2024 року, узгоджуються технічні специфікації вебсервісу.

Етап розробки включає в себе як створення, так і тестування вебсервісу:

- створення ескізів та концептуальних макетів (4-8 січня 2024 року);
- презентація та обговорення цих макетів з клієнтом (9 січня 2024 року);
- внесення необхідних змін та фіналізація макетів (10-17 січня 2024 року).
- налаштування середовища розробки. Встановлення локального серверу, провадження системи контролю версій (18 січня 2024 року);
- верстка цільової сторінки вебсервісу (19-23 січня 2024 року);
- розробка системи навігації (24 січня 2024 року);
- проектування бази даних. Визначення таблиць, зв'язки та обмеження (25-26 січня 2024 року);
- реалізація автентифікації та авторизації користувачів. Розроблення функціоналу реєстрації, входу та виходу з системи (29 січня – 2 лютого 2024 року);
- верстка панелі керування користувача. (5-7 лютого 2024 року);
- інтеграція вебсервісу для завантаження та перевірки документів. Реалізація логіки для зберігання файлів в базі даних (8-15 лютого 2024 року);

- інтеграція з OpenAI API для перетворення вмісту документів у семантичні вектори. Реалізація логіки зберігання векторів у векторній базі даних (16-23 лютого 2024 року);

- реалізація функціоналу чату використовуючи модель GPT. Розроблення логіку зберігання запитів користувача в базі даних (26 лютого – 1 березня 2024 року);

- реалізація семантичного пошуку. Розроблення логіки передачі результатів семантичного пошуку як контексту до моделі GPT (4-8 березня 2024 року);

- проектування та розробка інтерфейсу чату для взаємодії користувачів з моделлю GPT. Забезпечення оновлення чату в режимі реального часу (11-13 березня 2024 року).

Тестування та налагодження програмного забезпечення є критично важливими етапами для забезпечення його коректного функціонування. Ці процеси можуть здійснюватися як під час, так і після розробки програмного забезпечення:

- тестування інтеграції різних модулів і компонентів. Перевірення взаємодії між frontend, backend та зовнішніми API (14-18 березня 2024 року);

- вирішення проблем або помилок, виявлених під час тестування (14-22 березня 2024 року);

- здача сайту замовнику для перевірки (14-26 березня 2024 року);

- усунення зауважень замовника (29 березня – 2 квітня 2024 року).

На останньому етапі відбувається розгортання вебсервісу у виробничому середовищі. Сюди входить:

- розгортання вебсервісу у виробничому середовищі. Налаштування моніторингу і ведення журналів (27-28 березня 2024 року);

- документація архітектури, налаштування та використання вебсервісу. Створення посібнику для користувачів і керівництва для розробників (4-10 квітня 2024 року).

На рисунку 3.1 зображена Діаграма Ганта проєкту розробки вебсервісу.

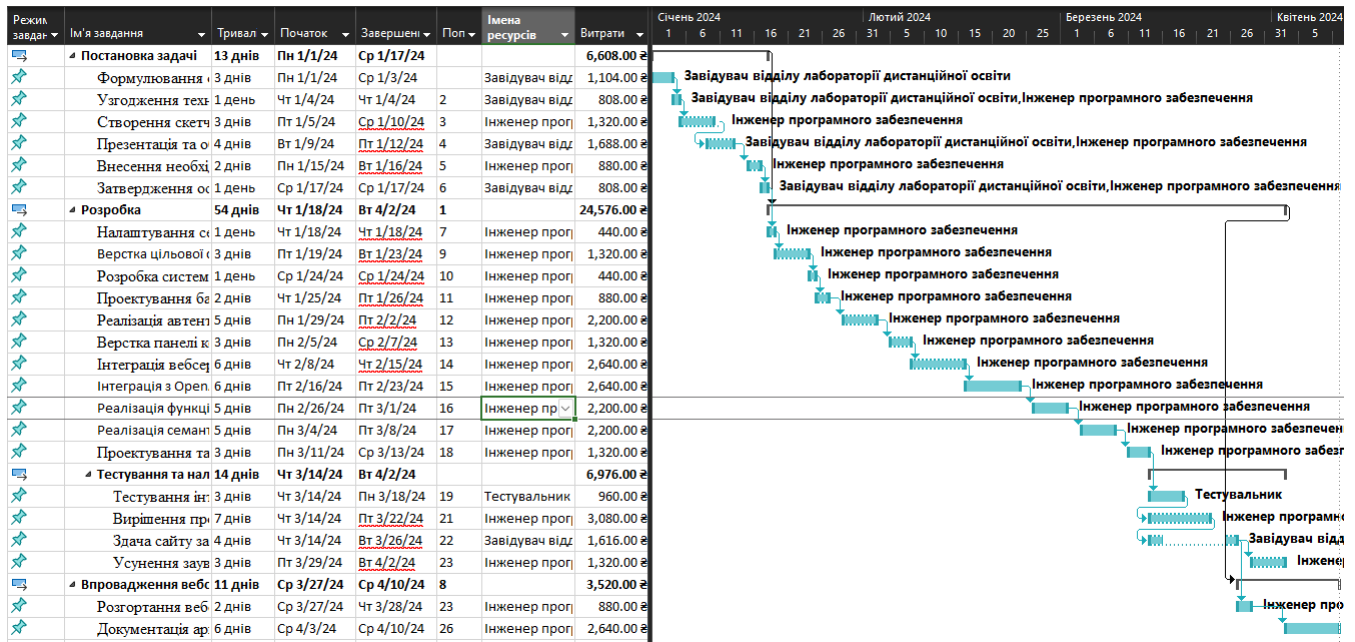


Рисунок 3.1 – Діаграма Ганта проєкту розробки вебсервісу

Джерело: запропоновано автором

Цей графік описує трьох місячний проєкт автоматизації процесів вебсервісу для аналізу текстових даних, який розпочинається 1 січня 2024 року і завершується 10 квітня 2024 року. Проєкт розділений на 3 етапи, кожен з яких має конкретні цілі та завдання. Перший етап присвячений постановці задачі, другий - розробці та тестуванню, третій – впровадженню. До кінця проєкту університет успішно автоматизує свої процеси і матиме дані, які покажуть позитивний вплив проєкту на навчальний процес.

Таким чином, планування роботи з розробки та програмної реалізації вебсервісу - це детальний і динамічний процес, який закладає основу для успіху проєкту. Визначивши чіткі цілі, використовуючи структуровані методи управління проєктами, розумно розподіляючи ресурси, команда може зорієнтуватися в складнощах проєкту і створити функціональний, високоякісний вебсервіс

3.2 Розробка архітектури вебсервісу

У попередньому розділі було описано технології, які були обрані для розробки вебсервісу. Це вебфреймворк Next.js на мові програмування TypeScript,

Amazon S3 для зберігання документів, MySQL для управління реляційними даними та Pinecone для роботи з семантичними векторами. Цей розділ заглиблюється в архітектурний дизайн вебсервісу. Буде досліджено, як ці технології взаємодіють, щоб сформувати цілісну систему для обробки запитів користувачів, аналізу документів і полегшення взаємодії користувачів з системою. Архітектура буде розроблена з урахуванням міркувань масштабованості, безпеки та продуктивності, що забезпечить надійний та ефективний сервіс.

Чітко визначена багаторівнева архітектура має вирішальне значення для створення зручного в обслуговуванні, масштабованого та безпечного вебсервісу. Сервіс аналізу текстових документів буде використовувати багаторівневий підхід, розділяючи функціональні можливості на окремі модулі з чітко визначеними інтерфейсами. Такий підхід сприяє модульності, спрощує розробку та полегшує подальше обслуговування.

Зовнішній шар, рівень подання, слугує інтерфейсом користувача. Розроблений за допомогою вебфреймворку Next.js, цей рівень відповідає за рендеринг елементів користувацького інтерфейсу та обробку взаємодії з користувачем. Фреймворк Next.js пропонує можливості рендерингу на стороні сервера, що покращує початковий час завантаження сторінки та SEO оптимізацію. Рівень подання відображатиме такі функції, як цільова сторінка з описом можливостей, варіанти автентифікації користувача, інтерфейс завантаження документів та вікно чату для взаємодії з моделлю GPT-4. Цей рівень буде взаємодіяти з рівнем бізнес-логіки через API інтерфейс для виконання дій користувача та отримання необхідних даних для візуалізації інтерфейсу користувача.

Рівень бізнес-логіки виконує роль мозку вебсервісу, керуючи основними функціями. Реалізований на мові програмування TypeScript, цей рівень отримує запити користувачів від рівня представлення та організовує взаємодію з іншими рівнями. Після входу користувача рівень бізнес-логіки перевіряє облікові дані та отримує інформацію про користувача. Застосовує правила авторизації, гарантуючи, що користувачі можуть отримати доступ до завантажених ними

документів і функцій лише на основі їхніх дозволів. Коли користувач завантажує PDF-документ, рівень бізнес-логіки взаємодіє з рівнем доступу до даних, щоб зберегти метадані документа в базі даних MySQL. Потім він запускає процес завантаження документа до хмарного сховища Amazon S3, отримуючи URL-адресу по якій зберігається цей документ. Після завантаження документа рівень бізнес-логіки взаємодіє з OpenAI API для перетворення вмісту документа в семантичний вектор за допомогою технології Embeddings. Цей вектор фіксує значення документа у високорозмірному просторі. Потім відбувається взаємодія з сервісним рівнем, щоб зберегти семантичний вектор у векторній базі даних Pinecone для використання семантичного пошуку. Коли користувач ініціює чат з моделлю GPT-4, рівень бізнес-логіки фіксує запити користувача і зберігає їх у базі даних. Перетворює запит користувача на семантичний вектор і використовує семантичний пошук у базі даних Pinecone, щоб знайти відповідні документи на основі їхньої семантичної схожості. Нарешті, він отримує відповідний контекст документа і передає його разом із запитом користувача до моделі GPT-4 через сервісний рівень. Модель GPT-4 генерує відповідь на основі наданого контексту.

Рівень доступу до даних слугує сполучною ланкою між рівнем бізнес-логіки та рівнем подання. Він обробляє всі взаємодії з базою даних MySQL, включаючи зберігання та отримання інформації про користувача, метаданих документів та запити користувачів. Цей рівень забезпечує цілісність даних, застосовуючи типи даних, обмеження та належне управління з'єднаннями з базами даних.

Сервісний рівень слугує рівнем абстракції для взаємодії із зовнішніми сервісами, такими як OpenAI та Pinecone. Він інкапсулює логіку зв'язку для цих сервісів, приховуючи основні деталі реалізації від рівня бізнес-логіки. Це сприяє вільному зв'язку і спрощує майбутні зміни або інтеграцію з альтернативними сервісами. Сервісний рівень використовує відповідні API, надані OpenAI та Pinecone, для виконання таких завдань, як генерація вбудовування документів та зберігання/видобування семантичних векторів.

Завдяки такій багаторівневій архітектурі, вебсервіс досягає модульності, ремонтпридатності та чіткого розподілу обов'язків. Кожен рівень фокусується на

своїй конкретній відповідальності, що призводить до більш надійної та масштабованої системи.

Багаторівнева архітектура створила основу для вебсервісу, з окремими шарами, що відповідають за конкретні функції. Щоб забезпечити зв'язок між користувацьким інтерфейсом і рівнем бізнес-логіки, сервіс використовує чітко визначений інтерфейс прикладного програмування. API діє як контракт, визначаючи, як зовнішні додатки можуть взаємодіяти з функціоналом сервісу. Існує кілька підходів до розробки API, кожен з яких має свої сильні та слабкі сторони. Найпоширеніші з них включають: RESTful, SOAP та GraphQL API.

RESTful API дотримуються архітектурного стилю Representational State Transfer. REST визначає набір рекомендацій щодо того, як здійснюються запити, а саме за допомогою дієслів HTTP, таких як GET, POST, PUT, DELETE і як форматуються відповіді, найчастіше у форматі JSON. RESTful API орієнтовані на ресурси, тобто вони розглядають дані як ресурси, ідентифіковані за допомогою URI. Цей підхід добре узгоджується з вебсервісами, де документи і дані користувача можна вважати ресурсами.

SOAP API використовують мову розмітки XML як для запитів, так і для повідомлень-відповідей. Вони пропонують більш структурований підхід у порівнянні з REST, але можуть бути складнішими в реалізації та менш продуктивними.

GraphQL API дозволяють клієнтам запитувати певні поля даних з сервера, що потенційно зменшує обсяг переданих даних. Однак вони можуть бути складнішими в реалізації та безпечнішими.

При виборі дизайну API для вебсервісу аналізу текстових документів RESTful API є найбільш підходящим варіантом через низку переваг:

- стандартизація, RESTful API дотримуються усталеного набору принципів, що полегшує їх розуміння та інтеграцію з різними клієнтськими додатками;

- ресурсно-орієнтований дизайн, узгоджується з функціональністю сервісу, де документи та дані користувача виступають як ресурси, якими можна маніпулювати за допомогою операцій CRUD;
- простота використання, RESTful API зазвичай вважаються простішими у розробці та розумінні порівняно з SOAP або GraphQL, що робить їх гарним вибором для цього проєкту.

Однак реалізація традиційного RESTful API з нуля може бути громіздкою, особливо при роботі зі складними структурами даних і забезпеченні безпеки типів протягом усього процесу розробки. Хоча REST визначає загальну структуру і стиль спілкування, він не забезпечує дотримання типів даних і не пропонує автоматичну генерацію коду. Це може призвести до помилок під час розробки та вимагати ручних зусиль для синхронізації коду користувацького інтерфейсу та бізнес-логіки.

Для вирішення цієї проблеми було розроблено бібліотеку tRPC. tRPC базується на принципах RESTful, пропонує додаткові функції, які спрощують розробку та покращують роботу з API. tRPC спеціально усуває недоліки, згадані вище [26].

tRPC застосовує визначення типів як для даних запиту, так і для даних відповіді. Це діє як захист від помилок під час виконання, спричинених неправильними типами даних, що передаються між frontend і backend. Це як вбудований механізм для виявлення потенційних проблем на ранній стадії процесу розробки.

Бібліотека автоматизує генерацію клієнтського коду на основі визначень API на стороні сервера. Це позбавляє розробників необхідності писати і підтримувати шаблонний код для обох сторін, заощаджуючи час і знижуючи ризик невідповідностей між frontend і backend реалізаціями.

Функціональність представлена у вигляді процедур. Ці процедури безпосередньо відображають функціональність, реалізовану на рівні бізнес-логіки. Це забезпечує чіткий і детальний спосіб взаємодії клієнтських додатків з backend, що робить API простішим для розуміння і використання.

Підсумовуючи, tRPC виступає сполучною ланкою між обраним стилем RESTful API та фактичною реалізацією сервісу. Він використовує переваги RESTful API такі як стандартизація і орієнтація на ресурси, пропонуючи при цьому додаткові функції для більш надійної та зручної для розробників розробки. Забезпечуючи безпеку типів, генерацію коду і процедурний підхід, tRPC допомагає створити добре структурований і підтримуваний API.

Після визначення основних технологій та бібліотек, що будуть використовуватися для розробки вебсервісу, наступним кроком є інтеграція з зовнішніми сервісами, такими як OpenAI API та Pinecone. Ці сервіси відіграють важливу роль у функціонуванні вебсервісу, надаючи можливість аналізувати текстові документи та зберігати семантичні вектори відповідно.

Для взаємодії з OpenAI API використовується бібліотека `openai`. Бібліотека надає простий інтерфейс для використання моделі GPT-4, що дозволяє виконувати запити до API, отримувати відповіді та обробляти помилки. Вона також дозволяє використовувати технологію Embeddings для перетворення тексту документа в семантичний вектор, що використовується для семантичного пошуку.

Взаємодія з Pinecone відбувається через бібліотеку `pinecone-database`. Pinecone є векторною базою даних, що дозволяє зберігати та виконувати пошук семантичних векторів. Бібліотека `pinecone-database` надає простий інтерфейс для взаємодії з Pinecone, дозволяючи зберігати семантичні вектори, виконувати семантичний пошук та отримувати результати.

Інтеграція з цими зовнішніми сервісами відбувається на рівні бізнес-логіки. Коли користувач завантажує документ, вміст документа перетворюється на семантичний вектор за допомогою Embeddings через OpenAI API. Цей вектор потім зберігається в Pinecone за допомогою бібліотеки `pinecone-database` [25].

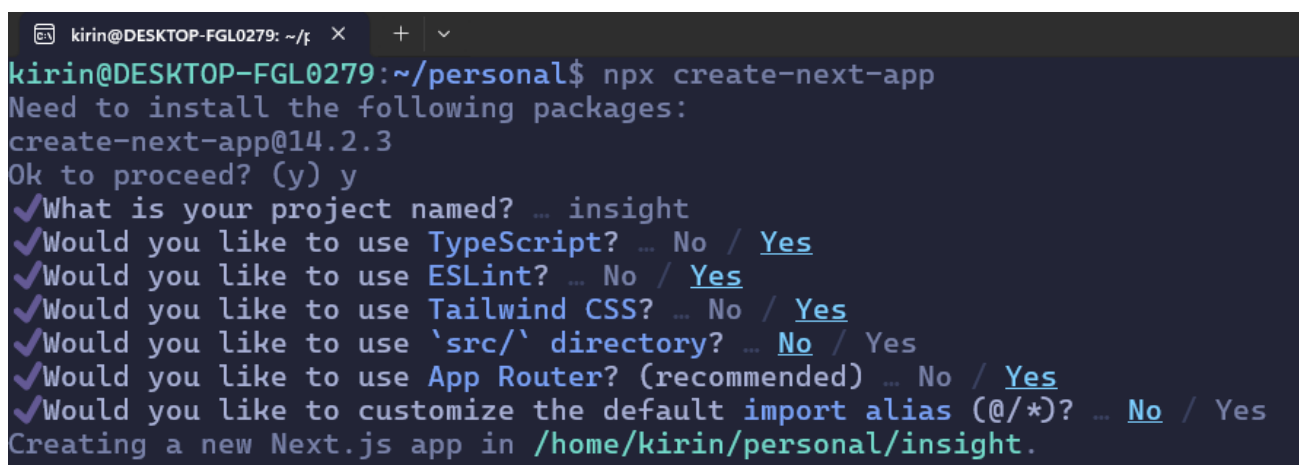
Коли користувач ініціює чат з моделлю GPT-4, запит користувача також перетворюється на семантичний вектор, і за допомогою семантичного пошуку в Pinecone знаходяться відповідні документи. Потім інформація передається моделі GPT-4 як контекст для генерації відповіді.

Таким чином, інтеграція з OpenAI API та Pinecone є важливою частиною архітектури вебсервісу. Вона дозволяє використовувати потужні можливості цих сервісів для аналізу текстових документів та взаємодії з користувачами на основі цього аналізу.

Отже, вебсервіс використовує багаторівневу архітектуру, що забезпечує модульність, масштабованість, безпеку та простоту обслуговування. RESTful API з бібліотекою tRPC використовується для чіткої та лаконічної взаємодії між frontend та backend. Інтеграція з OpenAI API та Pinecone дозволяє аналізувати текстові документи, генерувати семантичні вектори та використовувати їх для семантичного пошуку та чату з моделлю GPT-4.

3.3 Програмна реалізація

Після створення комплексного плану архітектури вебсервісу, настав час перетворити ці плани на функціонуючий додаток. Першим кроком на шляху реалізації програмного забезпечення є налаштування додатку Next.js. Це дуже важливий крок, оскільки він закладає основу для всього вебсервісу. Для створення додатку Next.js використовується інструмент командного рядку `npx create-next-app` (рисунок 3.2).



```
kirin@DESKTOP-FGL0279: ~/personal$ npx create-next-app
Need to install the following packages:
create-next-app@14.2.3
Ok to proceed? (y) y
✓What is your project named? ... insight
✓Would you like to use TypeScript? ... No / Yes
✓Would you like to use ESLint? ... No / Yes
✓Would you like to use Tailwind CSS? ... No / Yes
✓Would you like to use 'src/' directory? ... No / Yes
✓Would you like to use App Router? (recommended) ... No / Yes
✓Would you like to customize the default import alias (@/*)? ... No / Yes
Creating a new Next.js app in /home/kirin/personal/insight.
```

Рисунок 3.2 – Ініціалізація додатку Next.js використовуючи інструмент командного рядку

Джерело: запропоновано автором

Команда `prx create-next-app` запускає інтерактивний процес створення нового проєкту на Next.js. Вона пропонує вибрати назву проєкту та визначити деякі налаштування, такі як використання TypeScript, ESLint, Tailwind CSS, структуру каталогів з директорією `src`, налаштування маршрутизації App Router та імпорту за допомогою псевдонімів.

Після відповідей на запитання створюється директорія з назвою проєкту та встановлюються всі необхідні залежності (рисунк. 3.3). Цей інструмент дозволяє швидко розпочати роботу над новим додатком Next.js з попередньо налаштованими параметрами, що суттєво спрощує початковий етап розробки. Зокрема, налаштування середовища включає конфігурацію Babel для трансформації коду JavaScript, налаштування Webpack для обробки модулів та оптимізації коду, а також підготовку до роботи з CSS та іншими статичними ресурсами.

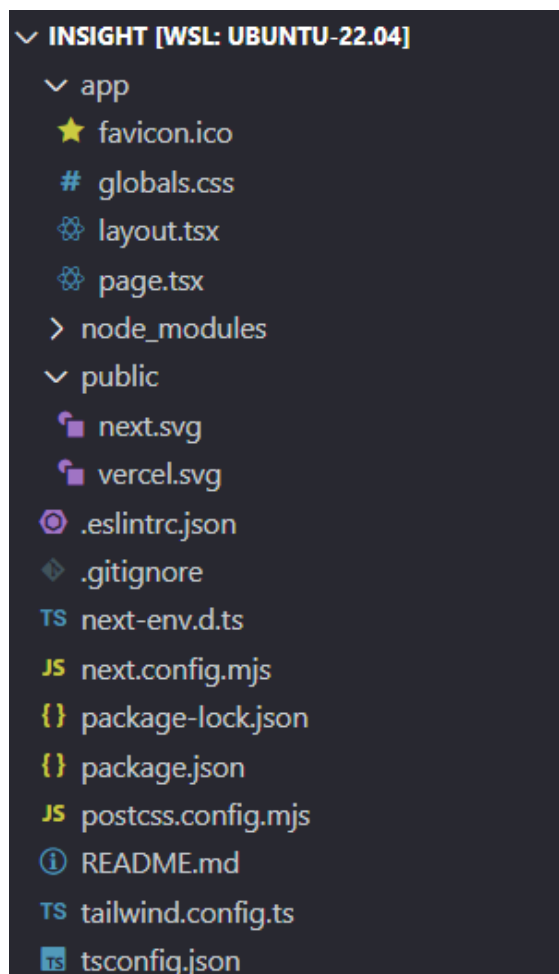
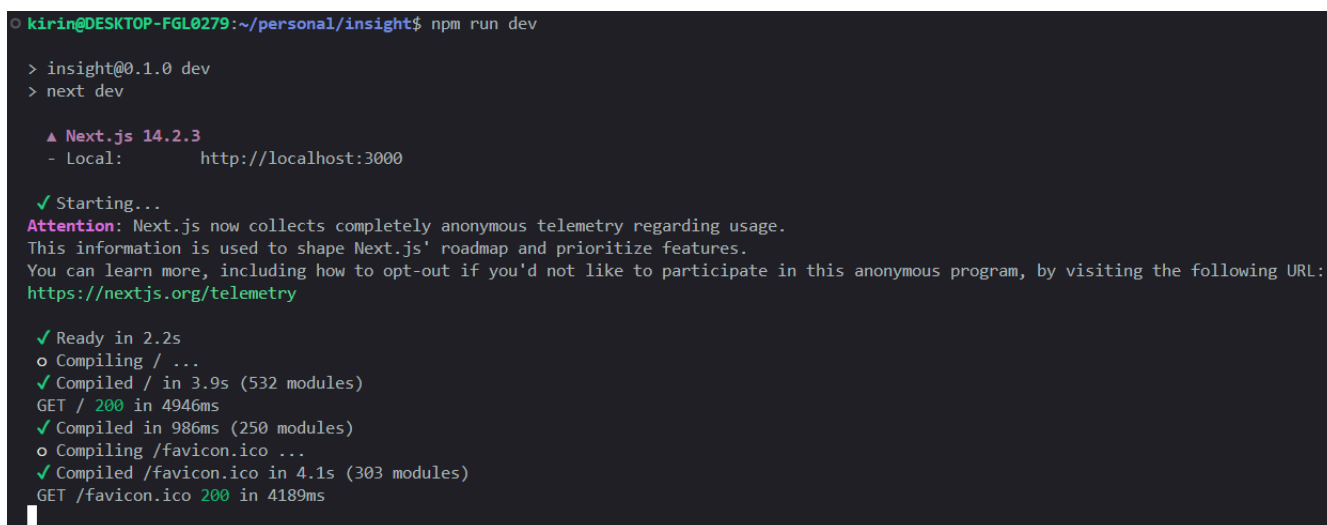


Рисунок 3.3 – Структура стандартного додатку Next.js

Джерело: запропоновано автором

Для перевірки працездатності додатку Next.js його можна запустити в режимі розробника, використовуючи скрипт `npm run dev`. За замовчуванням додаток запускається на локальному сервері, використовуючи порт 3000 (рисунок 3.4). В режимі розробника додаток Next.js автоматично перезавантажується при кожній зміні коду. Це досягається за допомогою функції `hot-reloading`, яка значно підвищує ефективність розробки, оскільки розробникам не потрібно вручну перезапускати сервер або оновлювати сторінку у браузері після внесення змін до коду.



```
kirin@DESKTOP-FGL0279:~/personal/insight$ npm run dev

> insight@0.1.0 dev
> next dev

  ▲ Next.js 14.2.3
  - Local:      http://localhost:3000

✓ Starting...
Attention: Next.js now collects completely anonymous telemetry regarding usage.
This information is used to shape Next.js' roadmap and prioritize features.
You can learn more, including how to opt-out if you'd not like to participate in this anonymous program, by visiting the following URL:
https://nextjs.org/telemetry

✓ Ready in 2.2s
  ○ Compiling / ...
✓ Compiled / in 3.9s (532 modules)
GET / 200 in 4946ms
✓ Compiled in 986ms (250 modules)
  ○ Compiling /favicon.ico ...
✓ Compiled /favicon.ico in 4.1s (303 modules)
GET /favicon.ico 200 in 4189ms
```

Рисунок 3.4 – Запуск додатку в режимі розробника

Джерело: запропоновано автором

Додаток стає доступним для перегляду в браузері за посиланням `http://localhost:3000`, де можна побачити результати роботи в реальному часі (рисунок 3.5). Це дозволяє розробникам легко відстежувати і тестувати зміни, які вони вносять, що забезпечує більш інтерактивний і зручний процес розробки.

Важливо зазначити, що режим розробника призначений виключно для процесів розробки та тестування. Він не оптимізований для продуктивності та безпеки, тому не підходить для використання в реальних умовах експлуатації.

Для того щоб підготувати Next.js додаток до розгортання в продуктивному середовищі, використовується команда `npm run build`. Ця команда запускає процес збірки, який включає в себе мінімізацію та оптимізацію коду, створення оптимізованих версій ресурсів, таких як зображення та CSS, а також генерацію

статичних файлів, якщо додаток використовує статичну генерацію сайтів. Після завершення процесу збірки зібраний додаток буде розміщено в папці `out`, готовий до розгортання на сервері.

Цей підхід забезпечує максимальну продуктивність та надійність додатку, оскільки всі ресурси будуть оптимізовані для швидкого завантаження та ефективної роботи в умовах реального навантаження. Крім того, Next.js підтримує різні стратегії розгортання, включаючи розгортання на платформах без серверного коду, що додає додаткової гнучкості та масштабованості для різних типів вебдодатків.

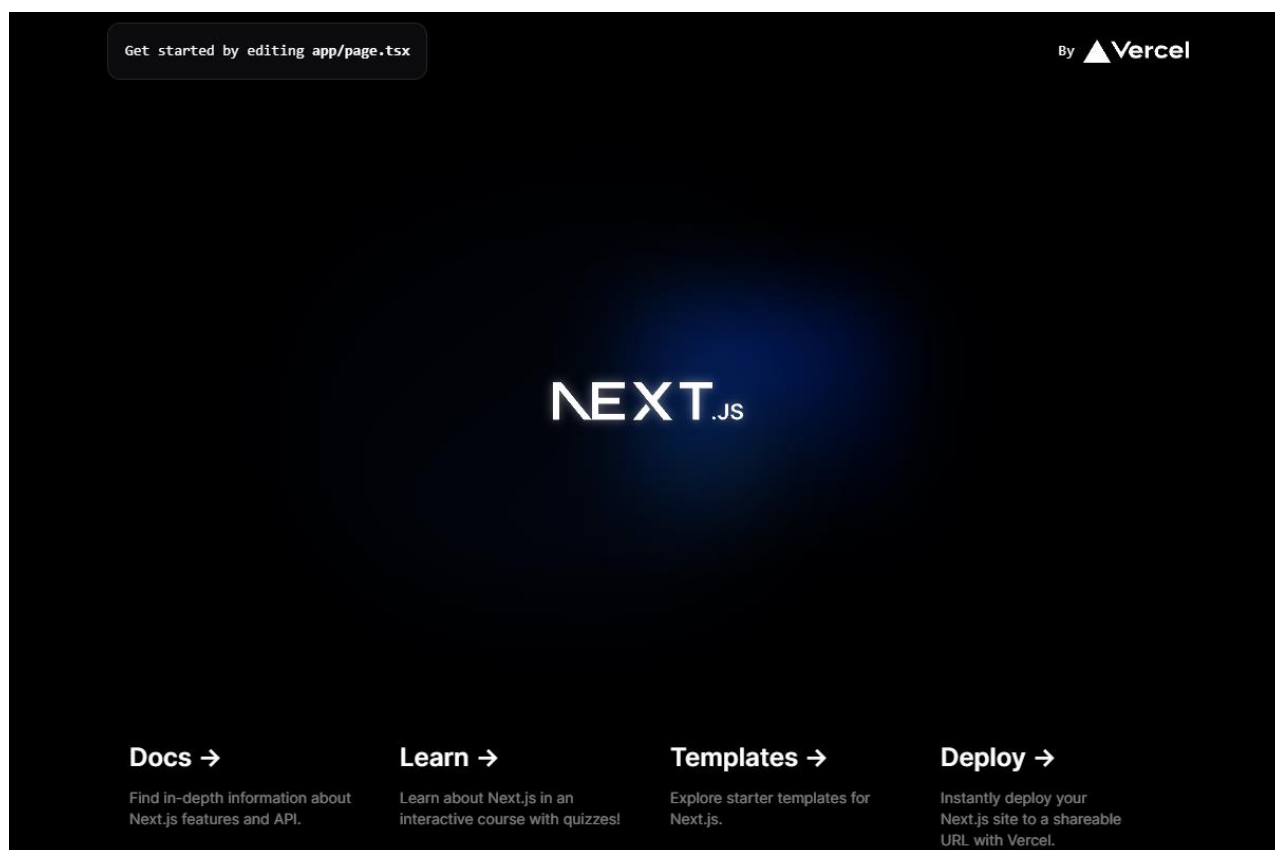


Рисунок 3.5 – Стартова сторінка вебдодатку створеного за допомогою утиліти `create-next-app`

Джерело: запропоновано автором

Наступним кроком є впровадження системи авторизації користувачів вебсервісу. Авторизація є процесом визначення, чи є людина тим, за кого вона себе видає. Після перевірки користувач отримує доступ до безпечної системи.

Процес підключення авторизації через сервіс Kinde складався з декількох основних етапів, які забезпечували інтеграцію безпекових механізмів і налаштувань для управління доступом користувачів до вебзастосунку.

Першим кроком було створення облікового запису в Kinde та реєстрація нового додатку. Це вимагало заповнення форми з базовою інформацією про додаток, включаючи його назву, опис та URL-адреси. Після успішної реєстрації було надано унікальні ідентифікатори, такі як Client ID та Client Secret, які використовувалися для подальшої аутентифікації. На рисунку 3.6 зображено процес реєстрації нового додатку Kinde.

The screenshot displays the 'Details' page for a new application in Kinde. On the left, a sidebar lists navigation options: Applications, SNAU AI, Quick start, Details (selected), Authentication, Tokens, and APIs. The main content area is titled 'Details' and contains the following sections:

- Basic information**
 - Name**: A text input field containing 'SNAU AI'.
 - Language / framework**: A dropdown menu showing 'NextJS'.
 - A 'NEXTJS' section with a link: 'Set up NextJS. Get set up in minutes with our [NextJS docs](#)'.
 - Type**: A dropdown menu showing 'Back-end web'.
- App keys**
 - Domain**: A text input field containing 'https://kirinyoku.kinde.com' with a 'Copy' button to its right.

Рисунок 3.6 – Створення нового додатку Kinde

Джерело: запропоновано автором

Унікальні ідентифікатори Client ID та Client Secret, отримані після реєстрації додатку Kinde, а також додаткові зміни середовища додано до файлу локального середовища проєкту – .env.

Другим етапом стало налаштування протоколу OAuth 2.0, який використовувався для авторизації. У цьому контексті необхідно було визначити параметри авторизації, такі як Authorization Endpoint та Token Endpoint. Також налаштовано редирект-URL, який використовувався для повернення користувача до вебзастосунку після успішної аутентифікації (рисуюнок 3.7).

The screenshot shows the configuration page for an application named 'SNAU AI'. On the left is a sidebar with navigation links: 'Applications', 'SNAU AI', 'Quick start', 'Details' (which is underlined), 'Authentication', 'Tokens', and 'APIs'. The main content area is titled 'Callback URLs' and contains several sections:

- Application homepage URI:** A text input field containing 'http://localhost:3000'. Below it, a note states: 'The homepage link to your application. E.g. https://app.yourapp.com. Opens on logo-click and for other back link actions.'
- Application login URI:** A text input field containing 'http://localhost:3000'. Below it, a note states: 'The default login route for resolving session issues. Allows users to redirect back to the application sign-in page via the /auth endpoint. E.g. https://app.yourapp.com/api/auth/login'.
- Allowed callback URLs:** A text input field containing 'http://localhost:3000/api/auth/kinde_callback'. To the right of the field is a small icon of three diagonal lines. Below the field, a note states: 'These are the redirect URIs where the user can be sent after authenticating'.
- Allowed logout redirect URLs:** A text input field containing 'http://localhost:3000'. To the right of the field is a small icon of three diagonal lines. Below the field, a note states: 'These are the URLs where the user can be sent after signing out'.

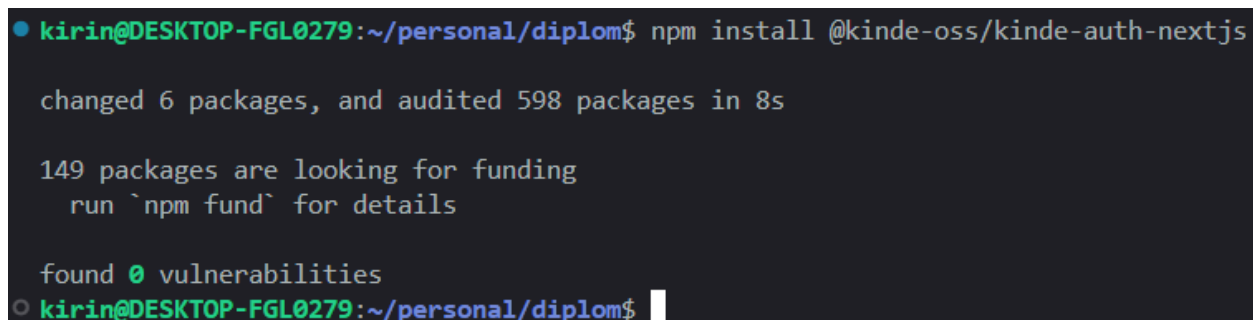
Рисуюнок 3.7 – Налаштування протоколу OAuth 2.0

Джерело: запропоновано автором

Наступним кроком була інтеграція з вебзастосунком. Це включало додавання необхідних бібліотек та модулів для роботи з OAuth 2.0 (рисуюнок 3.8), а також реалізацію логіки аутентифікації. Створено сторінки для входу та виходу з системи, а також налаштовано обробку токенів доступу та оновлення.

Для реалізації логіки аутентифікації було створено відповідні обробники маршрутів у вебзастосунку. Обробник входу перенаправляє користувача на

сторінку авторизації Kinde. Це досягається шляхом генерації URL з відповідними параметрами.



```

kirin@DESKTOP-FGL0279:~/personal/diplom$ npm install @kinde-oss/kinde-auth-nextjs
changed 6 packages, and audited 598 packages in 8s

149 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
kirin@DESKTOP-FGL0279:~/personal/diplom$

```

Рисунок 3.8 – Встановлення бібліотеки @kinde-oss/kinde-auth-nextjs

Джерело: запропоновано автором

Лістинг коду обробника входу:

```

import { NextRequest } from "next/server";
import { handleAuth } from "@kinde-oss/kinde-auth-nextjs/server";
export async function GET(request: NextRequest, { params }: any) {
  const endpoint = params.kindeAuth;
  return handleAuth(request, endpoint);
}

```

Обробник редиректу обробляє відповідь від Kinde після успішної аутентифікації. Цей обробник отримував авторизаційний код, який використовувався для отримання токенів доступу та оновлення через Token Endpoint. Лістинг коду обробника редиректу:

```

export const appRouter = router({
  authCallback: publicProcedure.query(async () => {
    const { getUser } = getKindeServerSession();
    const user = getUser();
    if (!user?.id || !user?.email)
      throw new TRPCError({ code: "UNAUTHORIZED" });
    const dbUser = await db.user.findFirst({
      where: {

```

```

      id: user.id,
    },
  });
  if (!dbUser) {
    await db.user.create({
      data: {
        id: user.id,
        email: user.email,
      },
    });
  }
  return { success: true };
}),
})

```

Для захисту певних маршрутів вебзастосунку була реалізована перевірка наявності та валідності токену доступу. Це дозволяло обмежити доступ до ресурсів лише авторизованим користувачам. У разі відсутності дійсного токену, користувач перенаправлявся на сторінку входу. Лістинг коду:

```

trpc.authCallback.useQuery(undefined, {
  onSuccess: ({ success }) => {
    if (success) {
      router.push(origin ? `${origin}` : "/dashboard");
    }
  },
  onError: (error) => {
    if (error.data?.code === "UNAUTHORIZED") {
      router.push("/sign-in");
    }
  },
});

```

Для входу та реєстрації в систему використовуються компоненти `LoginLink` та `RegisterLink` з бібліотеки `kinde-oss/kinde-auth-nextjs`. Лістинг коду для входу та реєстрації:

```
<div className="flex gap-3 items-center">
  <LoginLink className={buttonVariants({ variant: "ghost", size: "sm" })}>
    {i18n("navbar.auth.sign-in")}
  </LoginLink>
  <RegisterLink className={buttonVariants({ size: "sm" })}>
    {i18n("navbar.auth.sign-up")}
  </RegisterLink>
</div>
```

На рисунках 3.9 та 3.10 зображено авторизацію в систему з використанням сервісу Kinde.



Ласкаво просимо!

Введіть свою адресу електронної пошти, і ми надішлемо вам код для входу.

Електронна пошта

[Продовжити](#)

Або

 [Продовжити з Google](#)

Немає облікового запису? [Створити](#)

Рисунок 3.9 – Сторінка авторизації

Джерело: запропоновано автором

Таким чином, процес підключення авторизації через сервіс Kinde складався з послідовних етапів, кожен з яких був спрямований на забезпечення надійної та безпечної інтеграції авторизаційних механізмів у вебзастосунок.

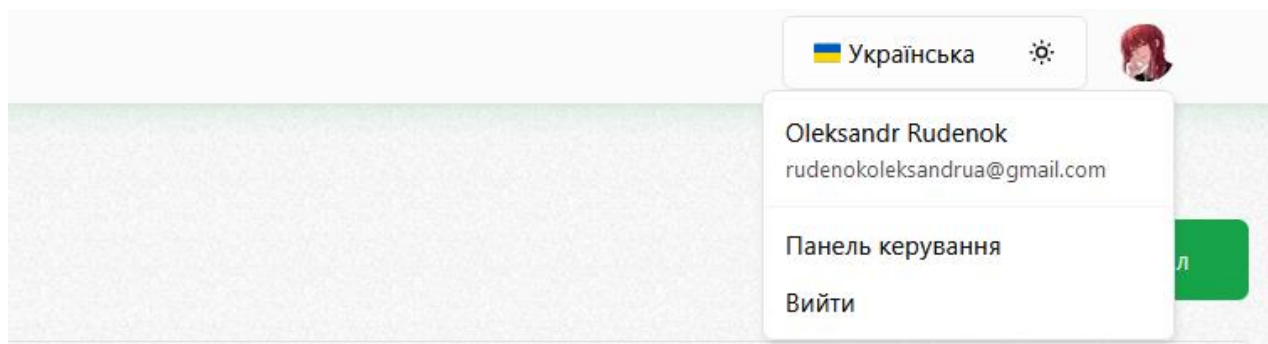


Рисунок 3.10 – Результат успішної авторизації

Джерело: запропоновано автором

Наступний крок це запровадження системи завантаження файлів. Але спочатку потрібно налаштувати векторну базу даних Pinecone. Спочатку необхідно зареєструватися на їхньому вебсайті і створити обліковий запис. Після цього сервіс надає доступ до інтерфейсу керування та API ключів. Після реєстрації необхідно створити новий індекс у Pinecone. Індекс визначає структуру, за якою будуть зберігатися та опрацьовуватися вектори даних (рисунок 3.11).

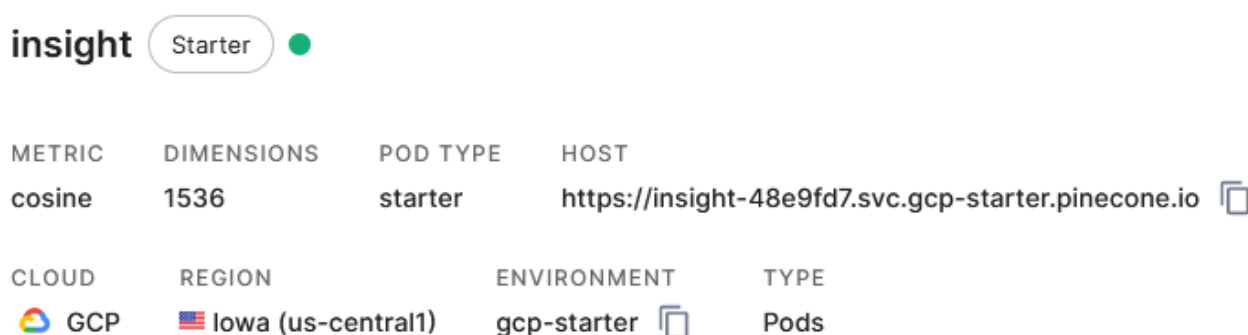


Рисунок 3.11 – Індекс векторної бази даних Pinecone

Джерело: запропоновано автором

Для підключення до Pinecone зазвичай використовуються SDK або бібліотеки, які надають зручний інтерфейс для роботи з векторами. Наприклад,

бібліотека `pinecone-database/pinecone`, що підтримуються мовами програмування JavaScript та TypeScript, дозволяють інтегрувати Pinecone з додатком безпосередньо через API. Лістинг коду для інтеграції Pinecone з вебсервісом:

```
import { Pinecone } from "@pinecone-database/pinecone";
export const pinecone = new Pinecone({
  apiKey: process.env.PINECONE_API_KEY!,
  environment: "gcp-starter",
});
export const pineconeIndex = pinecone.Index("insight");
```

Процес завантаження PDF-документів складається з двох частин: клієнтська та серверна. Для реалізації клієнтської частини використовується бібліотека `react-dropzone`. Лістинг коду функції `handleDrop` яка обробляє процес завантаження в компоненті `UploadFile` наведений нижче, а повний лістинг коду компонента `UploadFile` наведено в додатку Б.

```
async function handleDrop(file: File[]) {
  setIsUploading(true);
  const progressInterval = simulateProgress();
  const res = await startUpload(file);
  if (!res) {
    setIsUploadingError(true);
    return toast({
      title: i18n("toast-error.title"),
      description: i18n("toast-error.description"),
      variant: "destructive",
    });
  }
  const [fileResponse] = res;
  const { key } = fileResponse;
  if (!key) {
    setIsUploadingError(true);
```

```

return toast({
  title: i18n("toast-error.title"),
  description: i18n("toast-error.description"),
  variant: "destructive",
});
}
clearInterval(progressInterval);
setUploadProgress(100);
polling({ key });
}

```

Функція `handleDrop` є асинхронною функцією, яка виконує процес завантаження файлу, а також обробляє можливі помилки під час цього процесу. Вона складається з декількох етапів, кожен з яких відповідає за певний аспект завантаження та обробки файлів. Нижче подано детальний опис кожного кроку цієї функції.

Функція викликає `setIsUploading(true)` для встановлення стану "завантаження", що може бути використано для відображення індикатора завантаження користувачу.

Викликається функція `simulateProgress()`, яка створює інтервал для імітації прогресу завантаження файлу. Цей крок необхідний для покращення користувацького досвіду шляхом відображення візуального індикатора прогресу.

Викликається асинхронна функція `startUpload(file)`, яка ініціює процес завантаження файлу на сервер. Вона очікує на відповідь від сервера, яка зберігається у змінній `res`.

Якщо відповідь від сервера `res` є невизначеною або помилковою, встановлюється стан помилки завантаження за допомогою `setIsUploadingError(true)` і відображається повідомлення про помилку через функцію `toast()`.

Якщо відповідь від сервера містить інформацію про файл, з неї витягується ключ файлу `key`. У випадку, якщо ключ відсутній, встановлюється стан помилки і відображається відповідне повідомлення про помилку. Після успішного отримання

ключа файлу зупиняється інтервал, який відповідав за імітацію прогресу завантаження. Встановлюється прогрес завантаження на 100%, що вказує на завершення процесу завантаження. Викликається функція `polling`, яка починає опитування сервера для отримання статусу завантаженого файлу, використовуючи отриманий ключ файлу.

Ця функція забезпечує повний цикл завантаження файлу, включаючи інформування користувача про прогрес і обробку можливих помилок, що робить її важливою складовою частиною механізму завантаження файлів у вебзастосунку.

На серверній частині створюється об'єкт маршрутизатора файлів `ourFileRouter`, який визначає маршрут для завантаження PDF-файлів з обмеженням на розмір файлу до 2 МБ. `Middleware` виконує перевірку авторизації користувача за допомогою сесії `Kinde`. Якщо користувач не авторизований, викликається помилка `"Unauthorized"`.

Після завершення завантаження перевіряється, чи вже існує файл у базі даних. Якщо файл існує, нічого не робиться. Також перевіряється, чи користувач не перевищив ліміт у 2 файли. Якщо ліміт перевищено, викидається помилка. У випадку успішного завантаження файл додається до бази даних зі статусом `"PROCESSING"`.

Файл завантажується з S3-бакета і завантажується як PDF за допомогою `PDFLoader`. Кожна сторінка PDF-файлу обробляється окремо, створюючи масив документів. Кожен документ позначається метаданими та зберігається у векторній базі даних `Pinecone` за допомогою `OpenAIEmbeddings`. Після успішного зберігання статус файлу оновлюється на `"SUCCESS"`. У випадку помилки статус оновлюється на `"FAILED"`.

Таким чином, цей код забезпечує комплексний процес завантаження, обробки та зберігання PDF-файлів, забезпечуючи безпеку та управління даними користувачів. Повний лістинг коду завантаження PDF-документа наведений у додатку В.

На рисунку 3.12 зображено вікно завантаження файлу PDF-документа з індикатором завантаження, який показує поточний процес завантаження файлу.

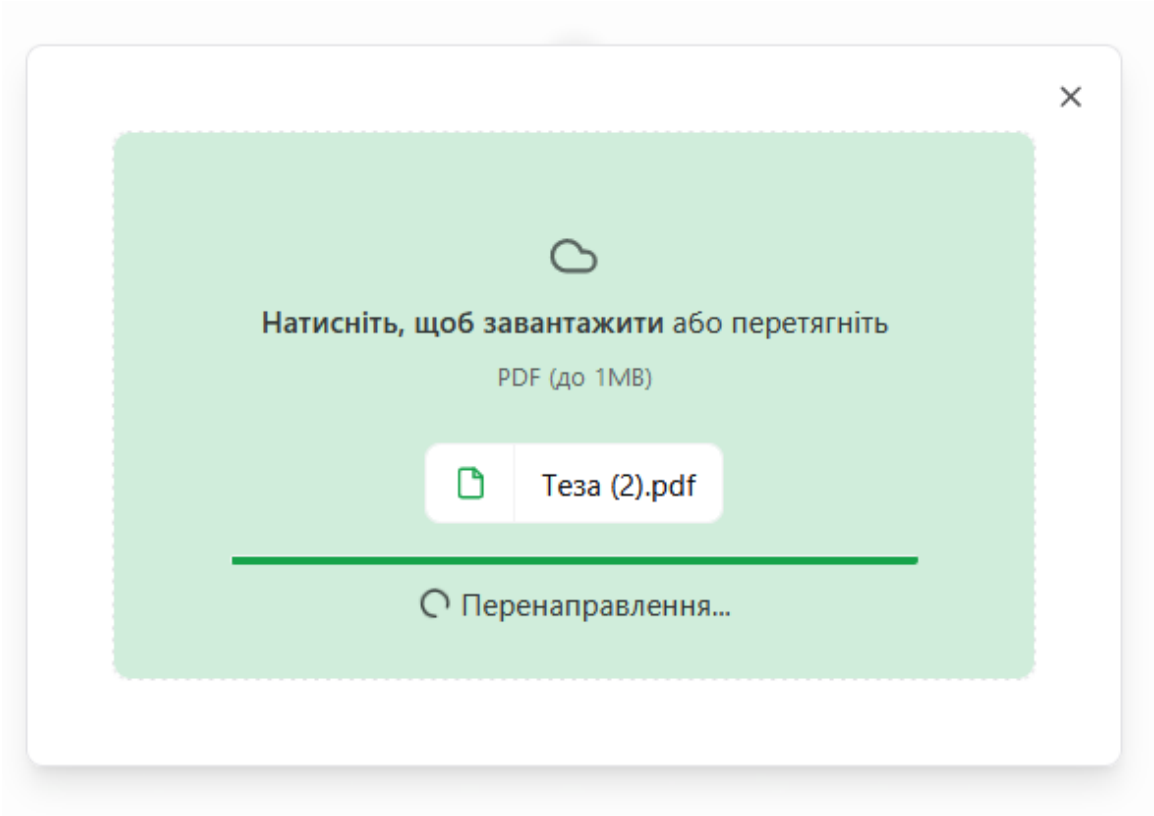


Рисунок 3.12 – Процес завантаження PDF-документа
Джерело: запропоновано автором

Відкривши індекс векторної бази даних на сайті Pinescone, можна спостерігати два семантичні вектори, які були сформовані з успішно завантаженого PDF-документа (рисунок 3.13) [24].

1	ID	VALUES
	469ae519-24...	-0.00479211472, 0.000525163254, 0.0118783638, -0.0297408234, 0.00484394, 0.00926636718, -0.0300172251,
	METADATA	
SCORE	fileId: "clw1z3qb30001z7zwi4jlev1k"	
-0.0247	text: "МІНІМІЗАЦІЯ ЗАТРИМОК У ВЕБДОДАТКАХ ВИКОРИСТОВУЮЧИ ГРАНИЧНІ ТЕХНОЛОГІЇ\n\nРуденко О. А., студ. 2 с.т. курсу ІСТ	

2	ID	VALUES
	71c920e8-e1a...	-0.00479211472, 0.000525163254, 0.0118783638, -0.0297408234, 0.00484394, 0.00926636718, -0.0300172251,
	METADATA	
SCORE	fileId: "clwavv4hs000128cnlus4h8w9"	
-0.0247	text: "МІНІМІЗАЦІЯ ЗАТРИМОК У ВЕБДОДАТКАХ ВИКОРИСТОВУЮЧИ ГРАНИЧНІ ТЕХНОЛОГІЇ\n\nРуденко О. А., студ. 2 с.т. курсу ІСТ	

Рисунок 3.13 – Семантичні вектори утворені з завантаженого PDF-документа
Джерело: запропоновано автором

Семантичний вектор складається з унікального ідентифікатора, числового значення та метаданих, які в свою чергу мають унікальний ідентифікатор PDF-документа з якого було утворено вектор та текстовий вміст цього документа.

Наступним етапом розробки вебсервісу є реалізація обробки повідомлень користувача та надавання інтерактивних відповідей. Лістинг коду для обробки повідомлень користувача:

```
const results = await vectorStore.similaritySearch(message, 1, {
  fileId: file.id,
});
const prevMessages = await db.message.findMany({
  where: { fileId },
  orderBy: { createdAt: "asc" },
  take: 5,
});
const formattedPrevMessages = prevMessages.map((msg) => ({
  role: msg.isUserMessage ? "user" : "assistant",
  content: msg.text,
}));
const context = `PREVIOUS CONVERSATION:${formattedPrevMessages.map(
  (msg) => {
    if (msg.role === "user") return `User:${msg.content}\n`;
    return `Assistant:${msg.content}\n`;
  }
)}CONTEXT:${results.map((r) => r.pageContent)
  .join("\n\n")}USER INPUT:${message}`;
const response = await openai.chat.completions.create({
  // model: "gpt-3.5-turbo-0125",
  model: "gpt-4o",
  temperature: 0.5,
  stream: true,
```

```

messages: [
  {
    role: "system",
    content: "Your name is SNAU AI. Use the following context snippets (or the
previous conversation, if necessary) to answer the user's question in markup format.
Answer in the language of the user's question.",
  },
  {
    role: "user",
    content: context,
  },
],
));

const stream = OpenAIStream(response, {
  async onCompletion(completion) {
    await db.message.create({
      data: { text: completion, isUserMessage: false, fileId, userId },
    });
  },
});

return new StreamingTextResponse(stream);
} catch (error) {
  console.error("Error searching for similar messages:", error);
  return new Response("InternalServerError", { status: 500 });
}

```

Даний код обробляє POST-запити для додавання повідомлень користувача до бази даних, виконує пошук за схожістю векторів у файлі, використовуючи Pinecone, і генерує відповідь за допомогою моделі GPT-4 на основі попередніх розмов та введеного користувачем тексту (рисунок 3.14).

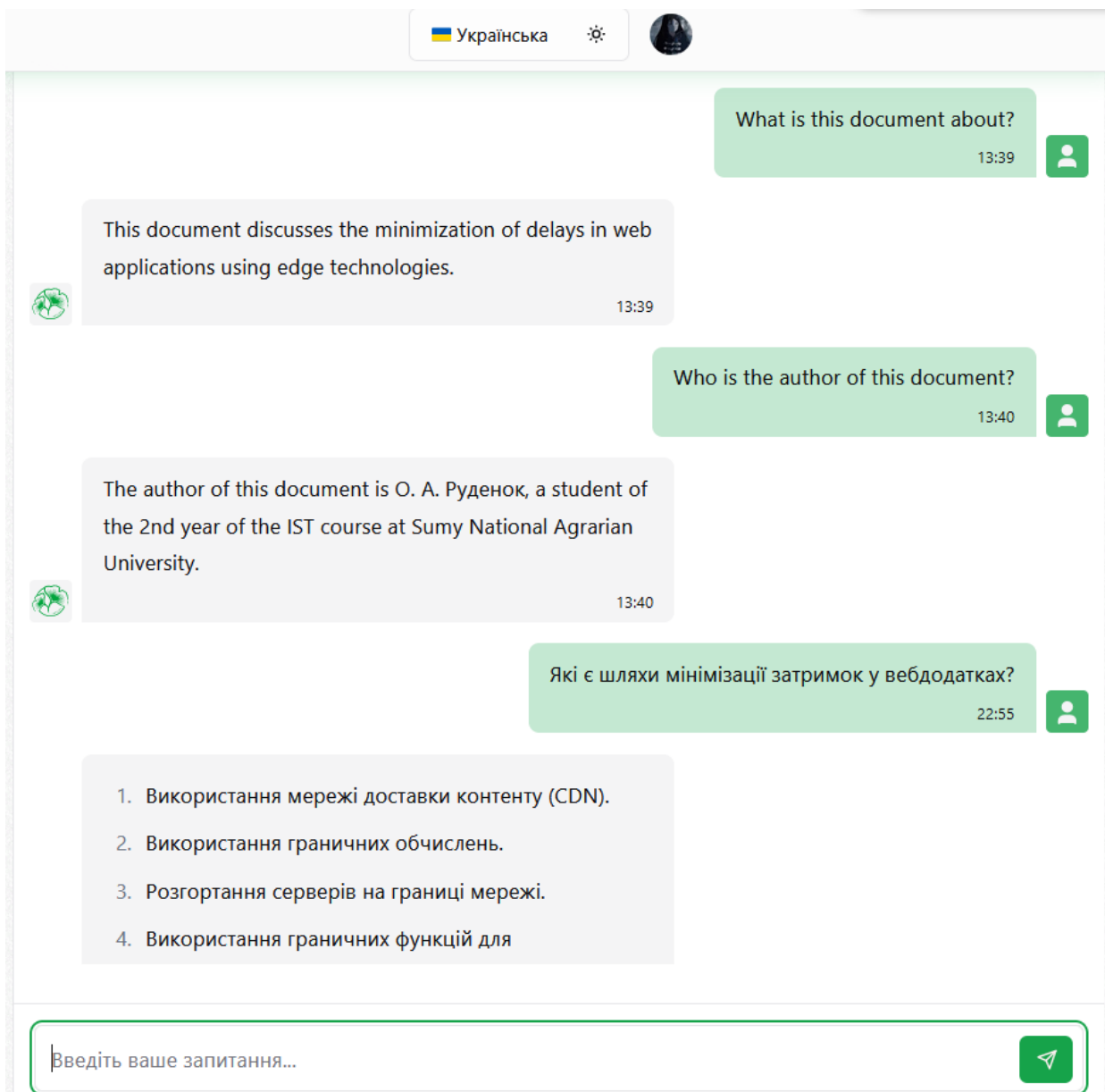


Рисунок 3.14 – Отримання інтерактивних відповідей на питання користувача стосовно інформації в завантаженому файлі

Джерело: запропоновано автором

3.4 Розгортання вебсервісу із використанням граничних технологій

У сучасній веброзробці важливе значення має вибір оптимальної платформи для розгортання додатків. Це зумовлено необхідністю забезпечення високої продуктивності, надійності та доступності вебдодатків, а також спрощенням процесів розробки та підтримки. Однією з популярних платформ для розгортання є Vercel, яка здобула визнання завдяки своїй інтеграції з фреймворком Next.js [28].

Next.js надає широкий спектр можливостей для побудови сучасних вебдодатків, включаючи підтримку статичної та динамічної генерації сторінок, оптимізацію для SEO та покращення продуктивності за рахунок серверного рендерингу. Проте для повноцінного використання цих можливостей необхідна відповідна платформа для розгортання, яка б забезпечила всі переваги Next.js.

Vercel використовує граничні технології, що значно підвищує швидкість та ефективність доставки контенту до кінцевих користувачів. Edge-технології передбачали розміщення обчислювальних ресурсів і кешованого контенту ближче до користувачів, що знижувало затримки під час завантаження сторінок і покращувало загальний користувацький досвід.

Розглянуто доцільність розгортання Next.js додатка на платформі Vercel, а також описано основні кроки цього процесу. Описаний процес розгортання покликаний забезпечити ефективне використання можливостей як Next.js, так і Vercel, що сприяло підвищенню якості та продуктивності вебдодатків.

Перед початком розгортання необхідно створити або вибрати існуючий проєкт Next.js. Це передбачає інтеграцію з системою контролю версій. Ініціалізація репозиторію Git та додавання файлів проєкту до репозиторію (рисуюнок 3.15).

```

● kirin@DESKTOP-FGL0279:~/personal/git/diplom$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/kirin/personal/git/diplom/.git/
● kirin@DESKTOP-FGL0279:~/personal/git/diplom$ git add .
● kirin@DESKTOP-FGL0279:~/personal/git/diplom$ git commit -m "Initial commit"
[master (root-commit) dbbcc44] Initial commit
80 files changed, 19594 insertions(+)
create mode 100644 .env.example
create mode 100644 .eslintrc.json
create mode 100644 .gitattributes
create mode 100644 .gitignore
create mode 100644 README.md
create mode 100644 components.json

```

Рисуюнок 3.15 – Інтеграція з системою контролю версій Git

Джерело: запропоновано автором

Далі необхідно завантажити проєкт на платформу GitHub або інший сервіс для управління версіями [9]. На рисунку 3.16 зображено процес створення приватного Github репозиторію, використовуючи графічний інтерфейс.

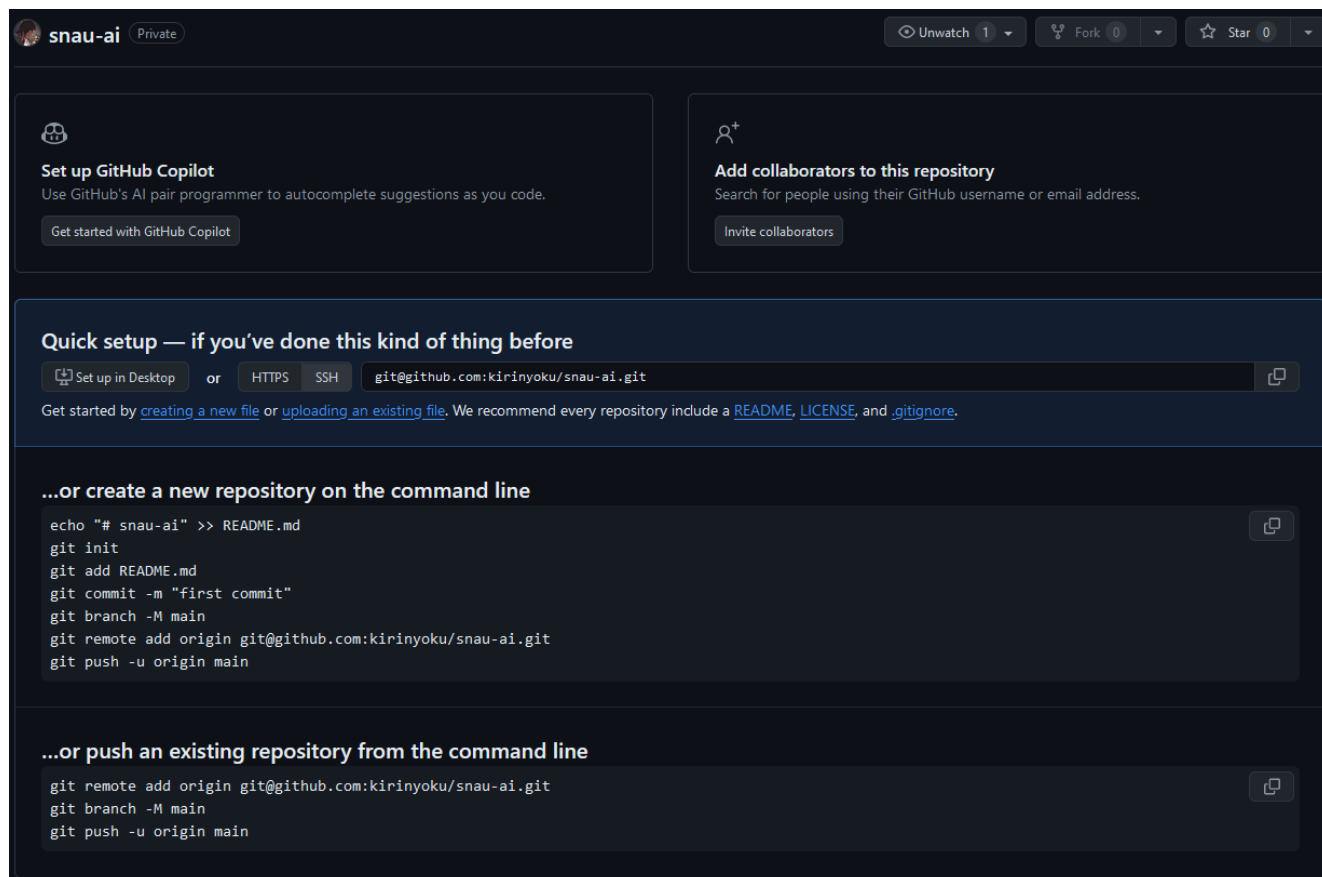


Рисунок 3.16 – Створення приватного Github репозиторію

Джерело: запропоновано автором

Завантаження проєкту на репозиторій GitHub починається з встановлення зв'язку між локальним та віддаленим репозиторіями (рисунок 3.17). Це досягається за допомогою команди “git remote add origin” URL, де URL вказує на URL-адресу віддаленого репозиторію на GitHub. Після встановлення зв'язку можна завантажити локальні коміти на віддалений репозиторій за допомогою команди “git push -u origin main”, де main вказує на основну гілку проєкту. Цей процес забезпечує зручне управління версіями коду та співпрацю з іншими розробниками на платформі GitHub.

```

● kirin@DESKTOP-FGL0279:~/personal/git/diplom$ git remote add origin git@github.com:kirinyoku/snau-ai.git
● kirin@DESKTOP-FGL0279:~/personal/git/diplom$ git branch -M main
● kirin@DESKTOP-FGL0279:~/personal/git/diplom$ git push -u origin main
Enumerating objects: 112, done.
Counting objects: 100% (112/112), done.
Delta compression using up to 12 threads
Compressing objects: 100% (92/92), done.
Writing objects: 100% (112/112), 355.61 KiB | 5.74 MiB/s, done.
Total 112 (delta 0), reused 0 (delta 0), pack-reused 0
To github.com:kirinyoku/snau-ai.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
○ kirin@DESKTOP-FGL0279:~/personal/git/diplom$

```

Рисунок 3.17 – Завантаження проєкту на платформу GitHub

Джерело: запропоновано автором

Наступним кроком є створення облікового запису на платформі Vercel або вхід у вже існуючий обліковий запис. Після авторизації на платформі Vercel здійснюється імпорт проєкту. Вибір опції «New Project» на головній сторінці консолі Vercel. Здійснення імпорту проєкту з GitHub шляхом вибору відповідного репозиторію зі списку доступних. Вказання основних параметрів проєкту, таких як назва та гілка репозиторію, яку необхідно розгорнути (рисунок 3.18).

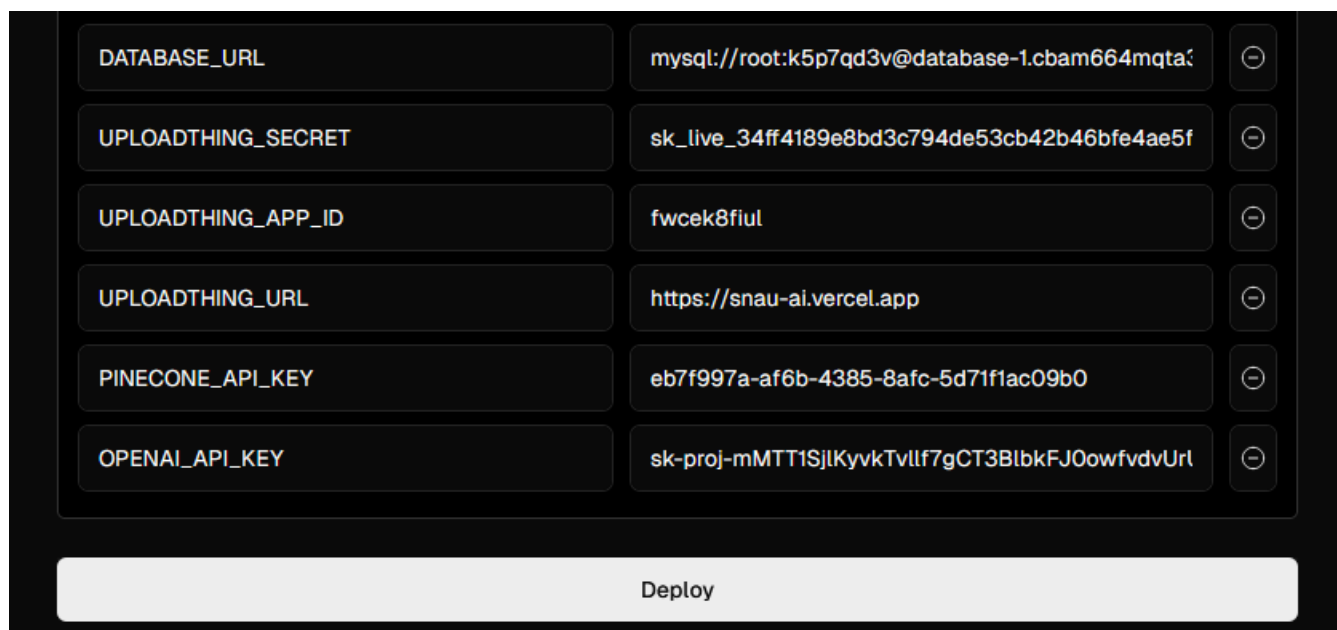
The screenshot displays the Vercel web interface for configuring a new project. On the left, a sidebar shows the account 'snau-ai' and a navigation menu with 'Configure Project' and 'Deploy'. The 'Configure Project' section is active, showing a list of Git repositories with 'kirinyoku/snau-ai' selected on the 'main' branch. The main configuration area includes a 'Project Name' field set to 'snau-ai', a 'Framework Preset' dropdown set to 'Next.js', and a 'Root Directory' field set to '/'. There are also expandable sections for 'Build and Output Settings' and 'Environment Variables'. A large 'Deploy' button is located at the bottom right of the configuration area.

Рисунок 3.18 – Вказання основних параметрів проєкту

Джерело: запропоновано автором

Перед початком розгортання необхідно визначити та налаштувати параметри середовища та конфігураційні змінні (рисунок 3.19).

Змінні середовища використовуються для налаштування програмного забезпечення та іншого програмного середовища залежно від умов роботи. Вони дозволяють розділити конфігурацію програми від коду, що полегшує налагодження, розгортання та масштабування додатків. Змінні середовища часто використовуються для збереження конфіденційних даних, таких як паролі чи ключі API, щоб вони не зберігались у відкритому вигляді у вихідному коді. Це дозволяє однаково налаштовувати додатки для різних середовищ, таких як розробка, тестування та виробництво, забезпечуючи при цьому узгодженість у поведінці програми.



DATABASE_URL	mysql://root:k5p7qd3v@database-1.cbam664mqta:	⊖
UPLOADTHING_SECRET	sk_live_34ff4189e8bd3c794de53cb42b46bfe4ae5f	⊖
UPLOADTHING_APP_ID	fwcek8fiul	⊖
UPLOADTHING_URL	https://snau-ai.vercel.app	⊖
PINECONE_API_KEY	eb7f997a-af6b-4385-8afc-5d71f1ac09b0	⊖
OPENAI_API_KEY	sk-proj-mMTT1SjlKyvkTvllf7gCT3BlbkFJ0owfvdvUrl	⊖

Deploy

Рисунок 3.19 – Додавання змін середовища до проєкту

Джерело: запропоновано автором

Завершивши налаштувань розпочинався процес автоматичного розгортання. Платформа Vercel автоматично запускає процес збірки та розгортання проєкту, використовуючи визначені налаштування. На рисунку 3.20 представлено результат розгортання проєкту.

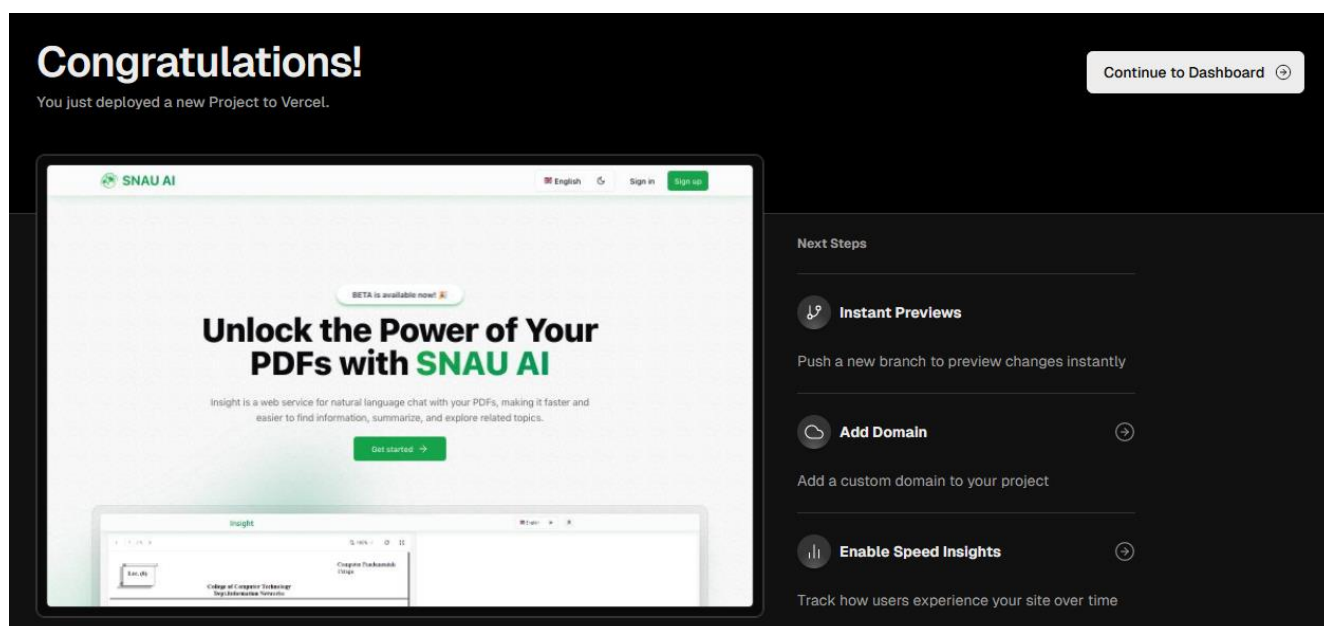


Рисунок 3.20 – Результат успішного розгортання проєкту

Джерело: запропоновано автором

Після успішного розгортання додаток був доступний за наданою Vercel URL, а саме <https://snau-ai.vercel.app>. Платформа також забезпечувала інструменти для подальшого управління та моніторингу додатка. У розділі «Deployments» можна переглядати логи збірки та роботи додатка для виявлення та усунення можливих проблем (рисунок 3.21). Будь-які зміни, внесені до репозиторію автоматично ініціювали новий процес розгортання, що забезпечувало актуальність додатка.

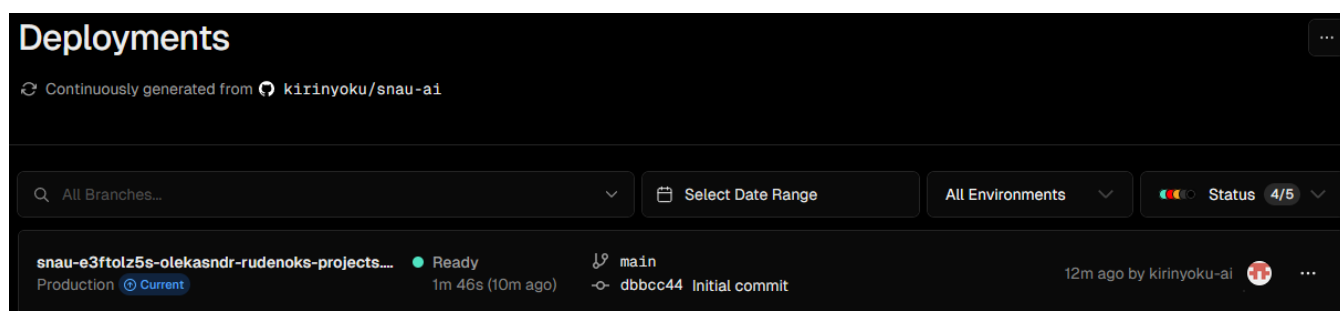


Рисунок 3.21 – Інструмент керування та моніторингу проєкта

Джерело: запропоновано автором

У третьому розділі роботи виконано комплексний підхід до розробки та програмної реалізації вебсервісу для автоматичного аналізу текстових документів.

Перш за все, було детально сплановано роботи, що дозволило чітко визначити етапи розробки, розподілити завдання та ресурси, а також встановити часові рамки для кожного етапу проєкту.

На основі ретельного аналізу вимог і цілей створено архітектуру вебсервісу, що забезпечує його надійність, масштабованість та гнучкість. Використання сучасних технологій і архітектурних рішень дозволило створити систему, здатну ефективно обробляти великі обсяги текстових даних і забезпечувати високу продуктивність.

Програмна реалізація вебсервісу включає розробку та інтеграцію основних модулів, зокрема алгоритмів для автоматичного аналізу текстів, механізмів обробки запитів та генерації результатів. Застосування сучасних мов програмування і фреймворків дозволило досягти високої ефективності, надійності та зручності використання вебсервісу.

На завершальному етапі проведено розгортання вебсервісу з використанням передових граничних технологій для забезпечення безперервної інтеграції та доставки контенту. Це забезпечило високу доступність, безпеку та масштабованість системи, що є критично важливим для її стабільної роботи у виробничому середовищі.

Розроблений вебсервіс готовий до впровадження у системи дистанційного навчання для надання допомоги студентам в аналізі їх електронних лекцій та конспектів. Це сприятиме підвищенню якості освітнього процесу та полегшенню навчальної діяльності.

ВИСНОВКИ

У кваліфікаційній роботі проведено дослідження та розробку вебсервісу для автоматизованого аналізу текстових документів з використанням глибоких нейронних мереж.

Проведено детальний аналіз предметної області, що включає розгляд останніх досліджень та публікацій, пов'язаних з аналізом текстових даних за допомогою глибоких нейронних мереж. Особлива увага була приділена сучасним досягненням у галузі обробки природної мови та впровадженню методів машинного навчання для аналізу текстових даних. Виявлено основні тенденції та напрями розвитку цієї галузі, що дозволило визначити актуальність і доцільність цього дослідження.

Аналіз існуючих програмних продуктів та аналогів виявив їхні переваги та недоліки. На основі цього аналізу було сформульовано основні вимоги до вебсервісу та визначено ключові завдання, які необхідно вирішити в процесі розробки. Розглянуто різні моделі глибоких нейронних мереж, що використовуються для аналізу текстових документів, та проведено порівняння їхньої ефективності. Обрано найоптимальнішу модель для подальшої реалізації. Проєктування вебсервісу здійснювалося з урахуванням вимог до його функціональності, надійності та масштабованості. Розроблено модель бази даних для зберігання та ефективного управління текстовими даними.

На етапі розробки та програмної реалізації створено архітектуру вебсервісу, яка включає серверну частину, клієнтську частину та базу даних. Здійснено програмну реалізацію основних модулів вебсервісу, включаючи модулі для авторизації, завантаження документів, аналізу даних та генерації відповідей.

В результаті проведеного дослідження та розробки створено функціональний вебсервіс для автоматизованого аналізу текстових документів, який може бути використаний у різноманітних прикладних задачах, пов'язаних з обробкою та аналізом текстової інформації. Розроблений вебсервіс готовий до впровадження у системи дистанційного навчання для надання допомоги студентам в аналізі їх

електронних лекцій та конспектів. Це сприятиме підвищенню якості освітнього процесу та полегшенню навчальної діяльності.

Для підвищення ефективності та розширення можливостей вебсервісу, необхідно впровадити декілька ключових покращень. Зокрема, доцільно розширити функціональність шляхом додавання модулів для більш детального аналізу тексту, таких як виявлення емоцій, автоматичне узагальнення та класифікація текстових даних. Крім того, оптимізація алгоритмів та використання більш потужних апаратних засобів дозволить значно підвищити швидкість обробки даних. Інтеграція з іншими програмними продуктами та сервісами забезпечить сумісність та розширить можливості використання вебсервісу, що є важливим аспектом для його подальшого розвитку.

Додатково, важливим аспектом є забезпечення безпеки даних користувачів, що вимагає впровадження сучасних методів шифрування та захисту інформації. Також, планується розробка мобільного додатку для зручного доступу до функцій вебсервісу з будь-якого пристрою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Брикалова В. В., Петрашенко А. В. Метод векторного подання природомовних текстів на основі нейромережевої архітектури Transformer. International scientific e-journal. 2020. № 9.
2. Онищенко К., Данієль Я., Каменєв Р. Аналіз Методів Обробки Природної Мови. Інформаційні системи та технології ICT-2020 : Міжнар. наук. конф., м. Коблеве, 12 верес. 2020 р. Харків, 2020. С. 186–190.
3. Ткаченко К., Зуєнко О. Використання багатошарової LSTM-неймережі в процесі розпізнавання друкованих текстів. Цифрова платформа: інформаційні технології в соціокультурній сфер. 2022. Т. 5, № 1.
4. Amazon S3 - Cloud Object Storage - AWS. Amazon Web Services, Inc. URL: <https://aws.amazon.com/s3/> (дата звернення: 13.05.2024).
5. Analyze insights in text with Amazon Comprehend. Amazon Web Service. URL: <https://aws.amazon.com/getting-started/hands-on/analyze-sentiment-comprehend/> (дата звернення: 07.05.2024).
6. Azure Blob Storage. Microsoft Azure. URL: <https://azure.microsoft.com/en-us/products/storage/blobs> (дата звернення: 12.05.2024).
7. Cloud Storage. Google Cloud. URL: <https://cloud.google.com/storage> (дата звернення: 13.05.2024).
8. Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions / G. Yenduri та ін. ResearchGate. URL: <http://surl.li/tngoq> (дата звернення: 07.05.2024).
9. GitHub: Let's build from here. GitHub. URL: <https://github.com/> (дата звернення: 18.05.2024).
10. GPT-4. OpenAI. URL: <https://openai.com/index/gpt-4-research/> (дата звернення: 07.05.2024).
11. Horev R. BERT Explained: State of the art language model for NLP. Towards Data Science. URL: <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270> (дата звернення: 10.05.2024).

12. Introducing text and code embeddings. OpenAI. URL: <https://openai.com/index/introducing-text-and-code-embeddings> (дата звернення: 07.05.2024).
13. Koubaa A. GPT-4 vs. GPT-3.5: A Concise Showdown. ResearchGate. URL: <https://shorturl.at/hHNT4> (дата звернення: 08.05.2024).
14. Li H. Deep learning for natural language processing: advantages and challenges. National Science Review. 2018. Т. 5, № 1. С. 24–26.
15. Natural Language AI. Google Cloud. URL: <https://cloud.google.com/natural-language/> (дата звернення: 07.05.2024).
16. Next.js by Vercel - The React Framework. Next.js by Vercel - The React Framework. URL: <https://nextjs.org> (дата звернення: 10.05.2024).
17. Nuxt: The Intuitive Vue Framework. Nuxt. URL: <https://nuxt.com/> (дата звернення: 12.05.2024).
18. Raghav P. Understanding NLP Word Embeddings – Text Vectorization. Towards Data Science. URL: <https://shorturl.at/deiV5> (дата звернення: 09.05.2024).
19. Rezaeenour J., Ahmadi M., Jelodar H. Systematic review of content analysis algorithms based on deep neural networks. Springer Link. URL: <https://link.springer.com/article/10.1007/s11042-022-14043-z> (дата звернення: 07.05.2024).
20. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions / L. Alzubaidi та ін. Journal of Big Data. URL: <https://journalofbigdata.springeropen.com/articles/10.1186/s40537-021-00444-8> (дата звернення: 07.05.2024).
21. SQLite vs MySQL vs PostgreSQL (Detailed Comparison). RunCloud Blog. URL: <https://runcloud.io/blog/sqlite-vs-mysql-vs-postgresql> (дата звернення: 14.05.2024).
22. Suissa O., Elmalech A., Zhitomirsky-Geffet M. Text analysis using deep neural networks in digital humanities and information science. ResearchGate. URL: https://www.researchgate.net/publication/352904410_Text_analysis_using_deep_neural

_networks_in_digital_humanities_and_information_science (дата звернення: 07.05.2024).

23. SvelteKit • Web development, streamlined. SvelteKit • Web development, streamlined. URL: <https://kit.svelte.dev> (date of access: 10.05.2024).

24. Text Analytics. Lexalytics. URL: <https://www.lexalytics.com/technology/text-analytics/> (дата звернення: 07.05.2024).

25. The vector database to build knowledgeable AI | Pinecone. The vector database to build knowledgeable AI | Pinecone. URL: <https://www.pinecone.io/> (дата звернення: 18.05.2024).

26. tRPC - Move Fast and Break Nothing.End-to-end typesafe APIs made easy. | tRPC. tRPC - Move Fast and Break Nothing.End-to-end typesafe APIs made easy. | tRPC. URL: <https://trpc.io/> (дата звернення: 17.05.2024).

27. Tripathi R. What are Vector Embeddings. Pinecone. URL: <https://www.pinecone.io/learn/vector-embeddings/> (дата звернення: 07.05.2024).

28. Vercel: Build and deploy the best Web experiences with The Frontend Cloud. Vercel. URL: <https://vercel.com> (дата звернення: 18.05.2024).

ДОДАТКИ

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

На розробку вебсервісу для автоматичного аналізу текстових документів на основі глибоких нейромережевих моделей

Погоджено:

Кандидат технічних наук, доцент кафедри
кібернетики та інформатики

Олександр В'ЮНЕНКО

СУМИ 2024

1. Призначення й мета створення вебсервісу

1.1 Призначення вебзастосунку

Вебсервіс, який дозволить користувачам аналізувати та інтерактивно пояснювати текстові документи за допомогою сучасної мовної моделі GPT-4.

1.2 Мета створення вебзастосунку

Мета цього вебсервісу - надати користувачам потужний інтерактивний інструмент для глибшого розуміння текстових документів. Він дозволить користувачам ставити відкриті запитання та отримувати глибокі пояснення на основі змісту завантажених ними документів.

1.3 Цільова аудиторія

Цільова аудиторія цього веб-сервісу включає:

- Студенти та науковці: особи, які прагнуть проаналізувати та зрозуміти наукові роботи, академічні статті та навчальні матеріали.
- Дослідники: Користувачі, яким потрібно витягувати і розуміти інформацію з технічних звітів, галузевих довідників і юридичних документів.

2. Вимоги до вебзастосунку

2.1 Системні вимоги

Frontend:

- Цільова сторінка з чітким описом функцій та переваг сервісу, включаючи зображення.
- Функціонал для входу користувачів.

- Безпечна панель користувача для завантаження документів та взаємодії з сервісом.

- Інтерактивний чат, де користувачі можуть задавати питання про завантажені документи.

- Інструменти візуалізації для представлення вмісту документів та зв'язків між ними.

Backend:

- Аутентифікація користувачів: Реалізація безпечної системи входу для контролю доступу користувачів.

- Зберігання документів: Інтеграція з Amazon S3 для безпечного зберігання завантажених PDF-документів.

- База даних: Реляційна база даних типу MySQL з таблицями:

- 1) Таблиця User: Зберігає інформацію про користувача.

- 2) Таблиця File: Зберігає інформацію про документ.

- 3) Таблиця Message: Зберігає інформацію про запити користувача та відповіді системи.

2.2 Технічні характеристики

- мови програмування: сучасний та безпечний фреймворк Next.js на мові TypeScript.

- безпека: впровадження стандартної практики безпеки для автентифікації користувачів, шифрування даних та контролю доступу.

- масштабованість: система повинна впоратися зі збільшенням кількості користувачів та обсягу документів.

- ведення журналів: впровадьте комплексне ведення журналу для усунення несправностей та моніторингу.

3. Завершення проєкту

3.1 Результати

- повністю функціональний веб-сервіс з усіма зазначеними функціями.
- модульні тести для критично важливих функцій.
- інструкції з розгортання та документація для легкого розгортання та обслуговування.

1.2 Критерії успіху

- сервіс має бути зручним та інтуїтивно зрозумілим для користувачів з різним рівнем технічної підготовки.
- система повинна надавати точні та змістовні пояснення на запити користувачів.
- сервіс повинен бути масштабованим для обробки зростаючої бази користувачів.
- система повинна бути безпечною та захищати дані користувачів.

ДОДАТОК Б

ЛІСТИНГ КОДУ КОМПОНЕНТА UPLOADFILE

```

"use client";

import Dropzone from "react-dropzone";
import { useState } from "react";
import { Button } from "./ui/button";
import { Icons } from "./ui/icons";
import { Progress } from "./ui/progress";
import { Dialog, DialogContent, DialogTrigger } from "./ui/dialog";
import { useUploadThing } from "@/lib/uploadthing";
import { useToast } from "./ui/use-toast";
import { trpc } from "@/app/_trpc/client";
import { useRouter } from "next/navigation";
import { useTranslations } from "next-intl";

export default function UploadFile() {
  const router = useRouter();
  const i18n = useTranslations("UploadFile");
  const [isOpen, setIsOpen] = useState(false);
  const [isUploading, setIsUploading] = useState(false);
  const [isUploadingError, setIsUploadingError] = useState(false);
  const [uploadProgress, setUploadProgress] = useState(0);
  const { toast } = useToast();
  const { startUpload } = useUploadThing("pdfUploader");
  const { mutate: polling } = trpc.getFile.useMutation({
    onSuccess: (file) => {
      router.push(`/dashboard/${file.id}`);
    },
  },

```

```

    retry: true,
    retryDelay: 1000,
  });

  // Simulated progress bar during file upload for a better user experience.
  function simulateProgress() {
    setUploadProgress(0);

    const interval = setInterval(() => {
      setUploadProgress((prev) => {
        if (prev >= 95 || isUploadingError) {
          clearInterval(interval);
          return prev;
        }

        return prev + 5;
      });
    }, 500);

    return interval;
  }

  // Called when the user starts uploading a file.
  async function handleDrop(file: File[]) {
    setIsUploading(true);
    const progressInterval = simulateProgress();
    const res = await startUpload(file);

    if (!res) {
      setIsUploadingError(true);
    }
  }

```

```

return toast({
  title: i18n("toast-error.title"),
  description: i18n("toast-error.description"),
  variant: "destructive",
});
}

const [fileResponse] = res;
const { key } = fileResponse;

if (!key) {
  setIsUploadingError(true);
  return toast({
    title: i18n("toast-error.title"),
    description: i18n("toast-error.description"),
    variant: "destructive",
  });
}

clearInterval(progressInterval);
setUploadProgress(100);

polling({ key });
}

return (
  <Dialog
    open={isOpen}
    onOpenChange={(visible) => {
      if (!visible) {

```

```

        setIsOpen(visible);
    }
}
>
<DialogTrigger
  onClick={() => {
    setIsOpen(true);
    setUploadProgress(0);
    setIsUploadingError(false);
  }}
  asChild
>
  <Button size="lg">{i18n("upload-file")}</Button>
</DialogTrigger>
<DialogContent>
  <Dropzone multiple={false} onDrop={handleDrop}>
    ({({ getRootProps, getInputProps, acceptedFiles }) => (
      <div
        {...getRootProps()}
        className="border border-border border-dashed h-64 m-4 rounded-lg"
      >
        <div className="flex items-center justify-center h-full w-full">
          <label
            htmlFor="dropzone-file"
            className={`flex flex-col items-center justify-center w-full h-full rounded-
lg cursor-pointer bg-accent/30 hover:bg-accent/70 ${
              uploadProgress === 100
                ? "bg-primary/20 hover:bg-primary/50"
                : isUploadingError
                  ? "bg-destructive/30 hover:bg-destructive/50"

```

```

      : null
    } transition-colors` }
  >
  <div className="flex flex-col items-center justify-center pt-5 pb-6">
    <Icons.cloud className="h-6 w-6 text-foreground/60 mb-2" />
    <p className="mb-2 text-sm text-foreground/80">
      <span className="font-semibold">
        {i18n("dropzone.instruction.1st")}
      </span>{" "}
      {i18n("dropzone.instruction.2nd")}
    </p>
    <p className="text-xs text-foreground/60">
      PDF ({i18n("dropzone.limit")} 1MB)
    </p>
  </div>
  {acceptedFiles && acceptedFiles[0] ? (
    <div className="max-w-xs bg-background flex items-center rounded-md
overflow-hidden outline outline-[1px] outline-accent divide-x divide-accent">
      <div className="px-3 py-2 h-full grid place-items-center">
        <Icons.page className="h-4 w-4 text-primary" />
      </div>
      <div className="px-3 py-2 h-full text-sm truncate">
        {acceptedFiles[0].name}
      </div>
    </div>
  ) : null}

  {isUploading ? (
    <div className="w-full mt-4 max-w-xs mx-auto">
      {!isUploadingError ? (

```

```

        <Progress
          value={uploadProgress}
          className="h-1 w-full bg-accent"
        />
      ): null}
      {uploadProgress === 100 ? (
        <div className="flex gap-1 items-center justify-center text-sm text-
foreground/80 text-center pt-2">
          <Icons.spinner className="w-4 h-4 animate-spin" />{" "}
          {i18n("dropzone.redirect")}
        </div>
      ): null}
    </div>
  ): null}
  <input
    type="file"
    id="dropzone-file"
    className="hidden"
    {...getInputProps()}
  />
</label>
</div>
</div>
  )}
</Dropzone>
</DialogContent>
</Dialog>
);
}

```

ДОДАТОК В

ЛІСТИНГ КОДУ ЗАВАНТАЖЕННЯ ДОКУМЕНТА СЕРВЕРНОЇ ЧАСТИНИ

```
import { db } from "@db";
import { getKindeServerSession } from "@kinde-oss/kinde-auth-nextjs/server";
import { createUploadthing, type FileRouter } from "uploadthing/next";
import { PDFLoader } from "langchain/document_loaders/fs/pdf";
import { PineconeStore } from "langchain/vectorstores/pinecone";
import { OpenAIEmbeddings } from "langchain/embeddings/openai";
import { pineconeIndex } from "@lib/pinecone";

const f = createUploadthing();

// FileRouter for your app, can contain multiple FileRoutes
export const ourFileRouter = {
  // Define as many FileRoutes as you like, each with a unique routeSlug
  pdfUploader: f({ pdf: { maxFileSize: "1MB" } })
  // Set permissions and file types for this FileRoute
  .middleware(async ({ req }) => {
    // Whatever is returned here is accessible in onUploadComplete as `metadata`
    const { getUser } = getKindeServerSession();
    const user = getUser();

    // The user will not be able to upload
    if (!user || !user?.id) throw new Error("Unauthorized");

    return { userId: user.id };
  })
  .onUploadComplete(async ({ metadata, file }) => {
```

```

// Checking if the file already exists in the database
const isFileExist = await db.file.findFirst({
  where: {
    key: file.key,
  },
});
if (isFileExist) return;

// Checking the number of user files
const userFiles = await db.file.findMany({
  where: {
    userId: metadata.userId,
  },
});

// User can have no more than 2 files.
if (userFiles.length >= 2)
  throw new Error("The file upload limit has been exceeded.");

// Adding a PDF file to the database
const createdFile = await db.file.create({
  data: {
    key: file.key,
    name: file.name,
    userId: metadata.userId,
    // URL of the S3 bucket where the PDF file is stored
    url: `https://uploadthing-prod.s3.us-west-2.amazonaws.com/${file.key}`,
    // "PROCESSING" means that the PDF has been added to the database but not yet
    vectorized:
      uploadStatus: "PROCESSING",

```

```

    },
  });

  // Vectorize a PDF file and save it to the Pinecone vector database.
  try {
    // Fetches a file from an S3 bucket
    const response = await fetch(
      `https://uploadthing-prod.s3.us-west-2.amazonaws.com/${file.key}`
    );

    // Loads the fetched file as a PDF
    const blob = await response.blob();
    const loader = new PDFLoader(blob);
    const pageLevelDocs = await loader.load(); // Array containing individual pages of
a document

    // Add a 'dataset' field to the data to distinguish the source
    const combinedData = pageLevelDocs.map((document) => {
      return {
        ...document,
        metadata: {
          fileId: createdFile.id,
        },
        dataset: "pdf", // Use a field to indicate the source dataset (e.g., 'pdf')
      };
    });

    // Initializes OpenAI embeddings with an API key
    const embeddings = new OpenAIEmbeddings({
      openAIApiKey: process.env.OPENAI_API_KEY,

```

```
});
```

```
// Stores the vectorized documents in Pinecone
```

```
await PineconeStore.fromDocuments(combinedData, embeddings, {
```

```
  /*
```

```
    There should be a namespace property with file.key data here,
```

```
    but the "Starter" Pinecone plan does not support it.
```

```
    -----
```

```
    namespace: file.key
```

```
  */
```

```
  pineconeIndex,
```

```
});
```

```
// Updates the upload status of the file in the database to "SUCCESS"
```

```
await db.file.update({
```

```
  data: {
```

```
    uploadStatus: "SUCCESS",
```

```
  },
```

```
  where: {
```

```
    id: createdFile.id,
```

```
  },
```

```
});
```

```
} catch (error) {
```

```
  // Error handling.
```

```
  console.error(error);
```

```
  await db.file.update({
```

```
    data: {
```

```
      uploadStatus: "FAILED",
```

```
    },
```

```
    where: {
```

```
        id: createdFile.id,  
      },  
    });  
  }  
  }),  
} satisfies FileRouter;  
  
export type OurFileRouter = typeof ourFileRouter;
```

ДОДАТОК Г
КОПІЇ ПУБЛІКАЦІЙ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

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

(13 - 17 листопада 2023 р., м. Суми)

Затара А. М. РОЗРОБКА СИСТЕМИ МОНИТОРИНГУ РИЗИКОВИХ ПОДІЙ	309
Кригін В.В. МЕТОДОЛОГІЯ АНАЛІЗУ СОЦІАЛЬНО-ЕКОНОМІЧНОГО РОЗВИТКУ ТЕРИТОРІАЛЬНОЇ ГРОМАДИ	310
Литвиненко О. М. РОЗВИТОК КООПЕРАЦІЇ В АГРАРНОМУ СЕКТОРІ	311
Неледва А. О. ЕВОЛЮЦІЯ МОТИВАЦІЇ : ПЕРЕЛІК СТРАТЕГІЙ ДЛЯ АДАПТАЦІЇ ДО СУЧАСНИХ УМОВ.....	312
Неледва А.О. МОТИВАЦІЯ ПЕРСОНАЛУ В УМОВАХ ЗРОСТАННЯ ВПЛИВУ ШТУЧНОГО ІНТЕЛЕКТУ	313
Новіков Д.С. ДОСЛІДЖЕННЯ ЕКОНОМІЧНИХ ПЕРЕДУМОВ ТА ДОЦІЛЬНОСТІ ВИКОРИСТАННЯ АГРОДРОНІВ.....	314
Пігуль О.В. МЕТОДОЛОГІЧНІ ЗАСАДИ ОЦІНКИ СТАЛОСТІ ЖИТЛОВИХ ПРОЄКТІВ	315
Поздирко В.С. НАПРЯМИ ПІДВИЩЕННЯ ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ГАЛУЗІ ЗЕРНОВИРОБНИЦТВА	316
Поздирко В.С. НАПРЯМИ РОЗВИТКУ ГАЛУЗІ РОСЛИННИЦТВА	317
Сьомушкін В.О. УПРАВЛІННЯ ЖИТТЄВИМ ЦИКЛОМ БУДІВЕЛЬНОЇ ОРГАНІЗАЦІЇ ЯК ІНСТРУМЕНТ СТАЛОГО РОЗВИТКУ	318
Щенякін Д.О. ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ ВИКОРИСТАННЯ СОНЯЧНИХ ПАНЕЛЕЙ НА НАУКОВО-ДОСЛІДНОМУ ПІДПРИЄМСТВІ	319
Бєлюков П.А. СУЧАСНІ АВТОМАТИЗОВАНІ ІНФОРМАЦІЙНІ СИСТЕМИ РЕКРУТИНГУ	320
Стебляк В.В. АНАЛІТИЧНИЙ ОГЛЯД ІТ-ТЕХНОЛОГІЙ ЩОДО ОБЛІКУ ВИТРАТ НА ВИРОБНИЦТВО СІЛЬСЬКО-ГОСПОДАРСЬКОЇ ПРОДУКЦІЇ. ПОСТАНОВКА ЗАДАЧІ.....	321
Агєєв О.А. ОГЛЯД ПЛАТФОРМ ЕЛЕКТРОННОГО НАВЧАННЯ.....	322
Блаженко Д.О. МЕТОДИ ТА ПРОБЛЕМИ ПРОЕКТУВАННЯ ІНФОРМАЦІЙНИХ ПІДСИСТЕМ ВЕДЕННЯ ЗАМОВЛЕНЬ ІЗ ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ TRIZ.....	323
Гончар Д.О. АВТОМАТИЗАЦІЯ РОЗРАХУНКУ БАЛАНСУ ГУМУСУ ТА ПОЖИВНИХ РЕЧОВИН В ГРУНТІ.....	324
Доля А.Р. СУЧАСНІ ТЕХНОЛОГІЇ РОЗРОБКИ АДМІНІСТРАТИВНИХ МОДУЛІВ.....	325
Дубінчин В.С. РОЗРОБКА ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ОЦІНЮВАННЯ ПРАЦЕЗДАТНОСТІ КОМП'ЮТЕРИЗОВАНИХ РОБОЧИХ МІСЦЬ	326
Крамський А.С., Бусенко І.В. ПРОГНОЗУВАННЯ КІЛЬКОСТІ СТУДЕНТІВ В УКРАЇНІ З ВИКОРИСТАННЯМ ЕКОНОМЕТРИЧНИХ МЕТОДІВ.....	327
Олійник А.В. ПОСТАНОВКА ЗАДАЧІ ТА ПРОЕКТУВАННЯ ПІДСИСТЕМИ ОБЛІКУ ПАЛИВНО- МАСТИЛЬНИХ МАТЕРІАЛІВ В ТЕРИТОРІАЛЬНІЙ ГРОМАДІ	328
Руденко О.А. ПОШУК ІНФОРМАЦІЇ В ТЕКСТОВИХ ДОКУМЕНТАХ З УРАХУВАННЯМ КОНТЕКСТУ.....	329
Санжаков А. А. АНАЛІТИЧНИЙ ОГЛЯД ВИКОРИСТАННЯ ІТ- ТЕХНОЛОГІЙ У МОБІЛЬНІЙ РОЗРОБЦІ ДЛЯ ОС АНДРОІД	330
Сіденко А.С. ПРОБЛЕМИ РЕГІСТРАЦІЇ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ МОНИТОРИНГУ НАВКОЛИШНЬОГО СЕРЕДОВИЩА.....	331
Терещенко С.С. ПОСТАНОВКА ЗАДАЧІ ТА ПРОЕКТУВАННЯ ПІДСИСТЕМИ АВТОМАТИЗАЦІЇ ОБЛІКОВИХ ПРОЦЕСІВ В АТЕЛЬЄ З ВИГОТОВЛЕННЯ ОДЯГУ	332
Чубун М.П. СУЧАСНІ ТЕНДЕНЦІЇ У СТВОРЕННІ ВЕБ-САЙТІВ ДЛЯ ІНТЕРНЕТ-МАГАЗИНІВ КОМП'ЮТЕРИЗОВАНИХ РОБОЧИХ МІСЦЬ	333
Шелковський С.В. СУЧАСНІ ТЕНДЕНЦІЇ ПРОЕКТУВАННЯ ПІДСИСТЕМИ ОБЛІКУ ПІДПРИЄМСТВ ПЕРЕРОБКИ СІЛЬСЬКОГОСПОДАРСЬКОЇ ПРОДУКЦІЇ.....	334
Горбань С.В. ТЕОРЕТИЧНІ АСПЕКТИ ФОРМУВАННЯ ПОЗИТИВНОГО ІМІДЖУ ПІДПРИЄМСТВА.....	335
Miao Yinhai INFLUENCE OF MANAGEMENT DECISIONS ON THE ENTERPRISE'S INNOVATION STRATEGY.....	336
Zhao Min THE EFFICIENT MANAGEMENT OF TALENTS FOR THE DEVELOPMENT OF ENTERPRISE	337
Заболотна А.І. УПРАВЛІННЯ ПРОЦЕСАМИ ПІДВИЩЕННЯ ЯКОСТІ ЖИТТЯ РЕГІОНУ ЗА МАТЕРІАЛАМИ ДРУЖБІВСЬКОЇ ТГ	338
Глек О.А. ОРГАНІЗАЦІЯ КЕЙТЕРИНГОВОГО ОБСЛУГОВУВАННЯ	339
Малюк Н.М. СТАЛИЙ РОЗВИТОК ЛІСОВОГО ГОСПОДАРСТВА.....	340
Дутов М.М. СТІЙКИЙ РОЗВИТОК ПІДПРИЄМСТВА.....	341
Бондар В.М. ТЕОРЕТИЧНІ ОСНОВИ КОНКУРЕНТОСПРОМОЖНОСТІ ДІЯЛЬНОСТІ ПІДПРИЄМСТВ.....	342
Макарчук П.В. ОРГАНІЗАЦІЙНА КУЛЬТУРА ЯК ОБ'ЄКТ УПРАВЛІНСЬКОЇ ДІЯЛЬНОСТІ.....	343
Могиленець В.М. ТЕОРЕТИЧНІ ОСНОВИ УПРАВЛІННЯ ПЕРСОНАЛОМ В СИСТЕМІ МЕНЕДЖМЕНТУ ПІДПРИЄМСТВА	344
Борщов А.О. ЗАСТОСУВАННЯ ДІАГНОСТИЧНИХ ЗАСОБІВ У ЛОГІСТИЦІ ТА ЇХ ПОШИРЕННЯ НА СИСТЕМИ ТРАНСПОРТУ	345
Гризодуб Д.Г. АСПЕКТИ УПРАВЛІННЯ ЯКІСТЮ ПОСЛУГ В СУЧАСНИХ УМОВАХ.....	346

ПОШУК ІНФОРМАЦІЇ В ТЕКСТОВИХ ДОКУМЕНТАХ З УРАХУВАННЯМ КОНТЕКСТУ

Руденок О.А., студ. 2 с.т. курсу ФЕiМ, спец. «Інформаційні системи та технології»
Науковий керівник: доцент О.Б.В'юненко
Сумський НАУ

Пошук інформації становить фундаментальне завдання у сучасному інформаційному суспільстві. Зі зростанням обсягу текстової інформації, доступної в Інтернеті та з інших джерел, пошукові системи стали важливим інструментом для знаходження необхідних даних серед безмежної кількості інформації. Однак, пошукові системи не завжди надають релевантні результати, і це пов'язано з тим, що не всі з них враховують семантичні зв'язки між словами та фразами в тексті.

Існують два основних підходи до пошуку інформації в тексті: лексичний і семантичний. Лексичний підхід заснований на пошуку слів і словосполучень, які містяться в запиті. Цей підхід є відносно простим і ефективним, але він не завжди може забезпечити релевантні результати пошуку. З іншого боку, семантичний підхід ґрунтується на пошуку семантичних зв'язків між словами й словосполученнями в тексті. Цей підхід складніший, але він дозволяє отримати більш точну і релевантну інформацію.

Одним із популярних методів реалізації семантичного підходу є використання семантичних векторів, отриманих за допомогою нейромережових моделей. Семантичні вектори представляють слова та фрази у вигляді числових векторів, які відображають їхнє семантичне значення. Інтеграція семантичних векторів у процес пошуку дозволяє пошуковим системам враховувати семантичні зв'язки між словами та фразами в тексті, що призводить до підвищення точності пошуку.

Семантичні вектори - це представлення слів або фраз у векторному просторі, де кожен вимір представляє семантичну ознаку. Ці вектори можна отримати за допомогою різних методів, зокрема за допомогою нейронних мереж. Нейромережові моделі здатні вивчати семантичні зв'язки між словами й фразами, аналізуючи великі обсяги текстових даних.

Наприклад, GPT (*Generative Pre-trained Transformer*) – це велика мовна модель розроблена компанією OpenAI, яка здатна вивчати семантичні зв'язки між словами та словосполученнями. GPT створює семантичні вектори за допомогою техніки, яка називається Вбудовування (*англ. Embeddings*). Вбудовування - це числове представлення понять, перетворене на послідовність чисел, що полегшує комп'ютеру розуміння взаємозв'язків між цими поняттями. Вбудовування, які є чисельно подібними, є також семантично подібними.

Семантичні вектори використовуються для покращення продуктивності інформаційно-пошукових систем, враховуючи семантичні зв'язки між словами та словосполученнями. Наприклад, якщо користувач запитує в пошуковій системі термін "центральний процесор", пошукова система може використовувати семантичні вектори, щоб знайти слова, які семантично пов'язані з "центральний процесор", такі як "мікропроцесор", "ЦП", "серце комп'ютера".

Подібність між двома векторами можна виміряти за допомогою різних методів, наприклад, косинусної подібності. Косинусна подібність - це міра подібності між двома векторами, яка базується на куті між ними. Косинус кута між двома векторами визначається як точковий добуток двох векторів, поділений на добуток їхніх величин. Косинусна подібність між двома векторами може коливатися від -1 до 1. Якщо два вектори ідентичні тоді значення дорівнює 1, тоді як -1 вказує на те, що два вектори протилежні один одному, 0 вказує на те, що два вектори ортогональні один одному.

На основі цих знань розробляється алгоритм пошуку інформації в текстових документах. Початковий крок передбачає перетворення документа, в якому буде здійснюватися пошук, у векторне представлення. Згодом запит користувача також повинен пройти аналогічний процес векторизації. Наступним важливим завданням є оцінка ступеня подібності між вектором запиту і векторами, що представляють вміст документа. Вектор, що демонструє найвищий ступінь схожості з вектором запиту, стає найбільш релевантним результатом пошуку. Ця методологія орієнтує пошукову систему на наміри користувача, гарантуючи, що семантично пов'язані терміни будуть включені в результати.

Отже, використання семантичного пошуку інформації має потенціал для значного покращення релевантності результатів. Враховуючи семантичні зв'язки між словами і словосполученнями, семантичні вектори дозволяють пошуковим системам знаходити інформацію, яка з більшою ймовірністю релевантна запиту користувача, навіть якщо запит не містить точних слів, які використовуються в текстових документах.

Використання нейромережових моделей для отримання семантичних векторів є перспективним підходом до розв'язання цієї проблеми. Нейромережові моделі здатні вивчати семантичні зв'язки між словами й фразами, аналізуючи великі обсяги текстових даних. Це дозволяє їм створювати семантичні вектори, які є більш точними та повними, ніж ті, що можуть бути створені за допомогою інших методів.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ НАЦІОНАЛЬНИЙ
АГРАРНИЙ УНІВЕРСИТЕТ

МАТЕРІАЛИ

науково-практичної конференції
викладачів, аспірантів та студентів
Сумського НАУ

(14-16 травня 2024 р.)

Luo Hao. A STUDY ON CONSUMER BEHAVIOUR MANAGEMENT THROUGH ADVANCED MARKETING TOOLS	414
Wang Baichao. MANAGEMENT CRISIS IN AGRICULTURAL COMPANIES IN CURRENT SITUATIONS	415
Wang Ruijie. HUMAN RESOURCES MANAGEMENT OPTIMIZATION AND INNOVATION BASED ON ARTIFICIAL INTELLIGENCE	416
Yu Hang. RESEARCH ON THE INTRODUCTION OF MODERN MANAGEMENT TECHNOLOGY INTO ENTERPRISES	417
Zhou Kunming. PECULIARITIES OF MANAGING HIGH-TECH ENTERPRISES IN A TRANSPARENT ECONOMY	418
Бусенко І.В. АНАЛІЗ ОСНОВНИХ РІЗНОВИДІВ SQL-ІН'ЄКЦІЙ, АНАЛІТИКА ЗАСТОСУВАННЯ ТА СПОСОБИ ЗАХИСТУ ВІД НИХ	419
Белоконь П.А. РОЗРОБКА ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ПІДТРИМКИ РОБОТИ HR-МЕНЕДЖЕРА	420
Стеблянко В. В. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ОБЛІКУ ВИТРАТ НА ВИРОБНИЦТВО ПРОДУКЦІЇ. РЕАЛІЗАЦІЯ ПРОЄКТУ	421
Фідірко Т.Р. РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ. ПОСТАНОВКА ЗАДАЧІ	422
Фідірко Т.Р. РОЗРОБКА ВЕБЗАСТОСУНКУ ОБЛІКУ ТЕХНІЧНОГО СТАНУ ЛЕГКОВИХ АВТОМОБІЛІВ. РЕАЛІЗАЦІЯ ПРОЄКТУ	423
Биченко Є.В. РОЗРОБКА ПРОТОТИПУ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА ВЕБОРІЄНТОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ПІДТРИМКИ УПРАВЛІННЯ ДІЯЛЬНІСТЮ ПІДПРИЄМСТВА	424
Корецький В.А. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ВЕБІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ МОНІТОРИНГУ ІНТЕРНЕТ-ТРАФІКУ	425
Лапченко І. В. ТА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ОРГАНІЗАЦІЇ ДИСТАНЦІЙНИХ КУРСІВ НЕФОРМАЛЬНОЇ ОСВІТИ	426
Хоменко Р.М. ПРОЄКТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА ЧАТ-БОТА ДЛЯ ЗРУЧНИХ ТА ЕФЕКТИВНИХ ОНЛАЙН-ЗАМОВЛЕНЬ ЕЛЕКТРОНІКИ	427
Блаженко Д. О. ПРОБЛЕМИ ПРОЄКТУВАННЯ БАЗИ ДАНИХ ПІДСИСТЕМИ ВЕДЕННЯ ЗАМОВЛЕНЬ ДЛЯ ВИГОТОВЛЕННЯ СІЛЬСЬКОГОСПОДАРСЬКОЇ ПРОДУКЦІЇ	428
Ковальов Р. С. ПРАКТИЧНЕ ПРОГРАМУВАННЯ ГЕНЕРАТОРІВ 3D КОНТЕНТУ НА ПРИКЛАДІ ГЕНЕРАТОРУ МЕША РЕЛЬЄФУ МІСЦЕВОСТІ	429
Ковальов Р. С. ПРАКТИЧНЕ ПРОГРАМУВАННЯ ГЕНЕРАТОРІВ 3D КОНТЕНТУ НА ПРИКЛАДІ ГЕНЕРАТОРУ МЕША РЕЛЬЄФУ МІСЦЕВОСТІ	430
Руденко О. А. МІНІМІЗАЦІЯ ЗАТРИМОК У ВЕБДОДАТКАХ ВИКОРИСТОВУЮЧИ ГРАНИЧНІ ТЕХНОЛОГІЇ	431
Сіденко А. С. ПРОБЛЕМИ РЕАЛІЗАЦІЇ ПІДСИСТЕМИ ОБЛІКУ ПРИРОДООХОРОННИХ ЗАХОДІВ	432
Терещенко С.С. АВТОМАТИЗАЦІЯ ОБЛІКОВИХ ПРОЦЕСІВ: КЛЮЧ ДО ЕФЕКТИВНОГО УПРАВЛІННЯ В АТЕЛЬЄ З ВИГОТОВЛЕННЯ ОДЯГУ	433
Михайлюк О.С. ОГРУНТУВАННЯ НЕОБХІДНОСТІ РОЗРОБКИ WEB-ОРІЄНТОВАНОЇ ПІДСИСТЕМИ ПІДТРИМКИ ОБЛІКОВИХ ПРОЦЕСІВ РЕГІОНАЛЬНОГО ІНТЕРНЕТ-ПРОВАЙДЕРА	434
Олійник А.В. ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПІДСИСТЕМИ ОБЛІКУ ПАЛИВНО-МАСТИЛЬНИХ МАТЕРІАЛІВ В ТЕРИТОРІАЛЬНІЙ ГРОМАДІ	435
Чубун М.П. АНАЛІЗ ТА ВИЗНАЧЕННЯ ВИМОГ ДО СУЧАСНОГО ВЕБ-САЙТУ	436
Дубінін В. С. ВИЗНАЧЕННЯ ФУНКЦІОНАЛЬНИХ ВИМОГ ДО ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ ОЦІНЮВАННЯ ПРАЦЕЗДАТНОСТІ КОМП'ЮТЕРИЗОВАНИХ РОБОЧИХ МІСЦЬ	437
Санжаков А. А. НОВИЙ ПІДХІД РОЗРОБКИ ДИЗАЙНУ ВЕБ-ЗАСТОСУНКІВ – MATERIAL 3	438
Федірко Д.Д. ПРОБЛЕМИ ПРОЄКТУВАННЯ ВЕБ-ЗАСТОСУНКІВ ОЦІНКИ ТАРИФНИХ ПРОПОЗИЦІЙ МОБІЛЬНИХ ОПЕРАТОРІВ УКРАЇНИ	439
Федірко Д.Д. РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ АНАЛІЗУ МОБІЛЬНИХ ОПЕРАТОРІВ УКРАЇНИ.....	440
Маслак О. М. ДЕРЖАВНЕ РЕГУЛЮВАННЯ ПРОЦЕСІВ АГРОПРОМИСЛОВОГО ВИРОБНИЦТВА НА РЕГІОНАЛЬНОМУ РІВНІ (НА ПРИКЛАДІ СУМСЬКОЇ ОБЛАСТІ)	441
Горбань С.В. ІМІДЖ ПІДПРИЄМСТВА: ВПЛИВ, ФОРМУВАННЯ ТА ОЦІНКА	442
Заболотна А.І. НОРМАТИВНО-ПРАВОВЕ ЗАБЕЗПЕЧЕННЯ ЕКОНОМІЧНОГО РОЗВИТКУ ДРУЖБІВСЬКОЇ ТЕРИТОРІАЛЬНОЇ ГРОМАДИ	443
Гулініна О.Ю. ШЛЯХИ ОПТИМІЗАЦІЇ ОРГАНІЗАЦІЙНОЇ СТРУКТУРИ УПРАВЛІННЯ ЗАКЛАДУ ОХОРОНИ ЗДОРОВ'Я	444
Сущенко В. О. ПРОЦЕС РОЗВИТКУ ПЕРСОНАЛУ, ЯК ЧИННИК КОНКУРЕНТОЗДАТНОСТІ ПІДПРИЄМСТВА В РИНКОВИХ УМОВАХ	445
Каплун В.М. УПРАВЛІННЯ СІЛЬСЬКОГОСПОДАРСЬКИХ ПІДПРИЄМСТВ ІЗ ЗАСТОСУВАННЯ ПРОГРАМНОГО КОНТРОЛЮ І МОНІТОРИНГУ	446
Бондар А.В. ТЕОРЕТИКО-МЕТОДИЧНІ ПІДХОДИ ДО ВПРОВАДЖЕННЯ ТА ОЦІНКИ СТРАТЕГІЧНОГО УПРАВЛІННЯ РОЗВИТКОМ СІЛЬСЬКОГОСПОДАРСЬКИХ ПІДПРИЄМСТВ	447
Романенко Т.А. СУЧАСНІ НАПРЯМИ ІНТЕНСИФІКАЦІЇ ВИРОБНИЦТВА ПРОДУКЦІЇ РОСЛИННИЦТВА В КОНТЕКСТІ ПРОДОВОЛЬЧОЇ БЕЗПЕКИ	448
Мишенін Є. В., Ярова І. Є. ОРГАНІЗАЦІЙНО ЕКОНОМІЧНИЙ МЕХАНІЗМ ЗАБЕЗПЕЧЕННЯ СТАЛОГО ПРОСТОРОВНОГО РОЗВИТКУ ЛІСОВОГО ГОСПОДАРСТВА	449
Hu Guannan CRISIS MANAGEMENT THEORY	450
Zhang Liming HOW DOES BYD COMPANY IMPROVE ITS COMPETITIVENESS?	451
Zhao Jing THE INFLUENCE OF CORPORATE CULTURE ON EMPLOYEE SATISFACTION	452
Cui Xiaoping DEVELOPMENT PATH OF ENTERPRISE MANAGEMENT IN THE ERA OF ARTIFICIAL INTELLIGENCE	453

МІНІМІЗАЦІЯ ЗАТРИМОК У ВЕБДОДАТКАХ ВИКОРИСТОВУЮЧИ ГРАНИЧНІ ТЕХНОЛОГІЇ

Руденко О. А., студ. 2 с.т. ФЕІМ, спец. «Інформаційні системи та технології»
Науковий керівник к.е.н., доц. О.Б. В'юненко
Сумський НАУ

Вебдодатки стають дедалі складнішими і динамічнішими, а користувачі очікують від них швидкої та оперативної роботи. Однак традиційні вебархітектури можуть страждати від затримок, особливо для користувачів, які знаходяться далеко від вихідного сервера.

Зменшення затримок має вирішальне значення для вебдодатків з кількох причин. Перш за все, від того, наскільки швидким і чуйним є додаток, значною мірою залежать задоволеність користувачів та їх утримання. Дослідження Google показало, що навіть пів секундної затримки в завантаженні сторінки може призвести до падіння залученості користувачів на 20%. Як наслідок, забезпечення низької затримки має важливе значення для залучення користувачів і підвищення конверсії. Крім того, затримка також впливає на користувацький досвід. Повільне отримання даних може призвести до затримок у завантаженні контенту, наприклад, зображень, відео або інформації в режимі реального часу, що може розчаровувати користувачів. У випадку електронної комерції або платформ для потокового передавання контенту проблеми затримок можуть призвести до втрат продажів або зменшення кількості переглядів.

Одним з інтуїтивно зрозумілих підходів є розміщення сервера в безпосередній близькості до користувача. Однак це, здавалося б, просте рішення стає складним, якщо врахувати глобальну та різноманітну природу користувачів. Це не один користувач, а безліч, і вони можуть бути розкидані по різних регіонах світу. Для ефективного розв'язання цієї проблеми потрібна більш комплексна стратегія. Замість того, щоб покладатися на єдиний сервер, необхідна мережа розподілених серверів, стратегічно розміщених для забезпечення рівномірного покриття по всьому світу.

Одним зі способів рівномірного розподілу серверів по всьому світу є використання мережі доставлення контенту (*англ. Content Delivery Network, CDN*). CDN - це мережа серверів, розташованих у різних частинах світу. Коли користувач запитує контент з CDN, він доставляється з сервера, який знаходиться найближче до користувача. Це може значно зменшити затримку, оскільки інформація не повинна подорожувати так далеко.

Ще один спосіб зменшити затримку - використання граничних обчислень. Граничні обчислення - це сучасна технологія, яка може допомогти зменшити затримку, наблизивши обчислення і дані до користувача. Це можна зробити шляхом розгортання серверів на границі мережі, наприклад, у центрах обробки даних провайдерів або на базі веж стільникового зв'язку. Граничні обчислення можна використовувати для кешування динамічного контенту, персоналізації контенту для кожного користувача і перенесення дорогих в обчислювальному плані завдань з центрального сервера на периферію.

Потужне рішення, що поєднує в собі переваги CDN та граничних обчислень, гранична мережа розроблена з метою мінімізації затримок та покращення взаємодії з користувачем, вона забезпечує безпосередню близькість контенту та обчислень до користувача. Однією з таких реалізацій є технологія Edge Network від компанії Vercel. Гранична мережа Vercel призначена для зберігання контенту та виконання обчислень у регіонах, близьких до клієнтів або даних, тим самим зменшуючи затримки та покращуючи продуктивність кінцевого користувача. Вона слугує як CDN, так і глобально розподіленою платформою для виконання граничних обчислень.

Гранична мережа Vercel використовує вебфреймворк Next.js з відкритим вихідним кодом для JavaScript бібліотеки інтерфейсу користувача React. Одним із ключових компонентів цієї архітектури є граничні функції (*англ. Edge Function*). Вони дозволяють обслуговувати вміст із найближчого до користувача сервера CDN. Маючи лише трохи JavaScript або TypeScript коду, розробники можуть використовувати граничні функції, щоб змінювати мережеві запити на границі, щоб вони були ближчими до відвідувачів вебсайту по всьому світу. Це дає розробникам можливість локалізувати вміст, показувати відповідну рекламу, автентифікувати відвідувачів, додавати A/B-тестування і багато іншого.

Граничні функції виконуються у середовищі виконання Edge Runtime. Хоча Edge Runtime не розкриває весь API Node.js, він надає доступ до API вебстандарту, які мають сенс на сервері. Він побудований на основі рушія V8 - високопродуктивного рушія JavaScript і вебасемблера з відкритим вихідним кодом, написаного мовою програмування C++.

Завдяки граничним функціям, розробники мають можливість налаштовувати та оптимізувати обробку запитів на границі, що дозволяє показувати користувачам відповідний контент, прискорювати завантаження сторінок та забезпечувати персоналізацію для кожного відвідувача. Гранична мережа та подібні технології є інструментами, які допомагають вирішити ці завдання та забезпечити задоволення користувачів. Разом із розвитком інтернет-технологій, подібні інновації продовжують впливати на спосіб, яким ми споживаємо вебконтент та взаємодіємо з вебдодатками.