Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)
Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: "Інтелектуальна система автоматичного резюмування тексту"

Виконав: студент 6 курсу групи КН-62м
спеціальності
122 "Комп'ютерні науки"
(шифр і назва спеціальності)
Блищак А. Т.
(прізвище та ініціали)

Керівник: Шиманський В. М.
(прізвище та ініціали)

Рецензент: Гецтош М. І.
(прізвище та ініціали)

Львів – 2025

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій

Кафедра комп'ютерних наук

Рівень вищої освіти другий (магістерський)

Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КН

_____ Борецька І.Б.

"10" грудня 2025 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Блищаку Андрію Тарасовичу
(прізвище, ім'я, по батькові)

1. Тема роботи: "*Інтелектуальна система автоматичного резюмування тексту*"

керівник роботи: *Шиманський Володимир Михайлович, кандидат технічних наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "29" квітня 2025 року № С-288

2. Термін подання студентом роботи *10 грудня 2025 р.*

3. Вихідні дані до роботи *Аналіз шляхів вирішення задачі, організаційна структура застосунку*

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

Перелік скорочень та умовних позначень. Вступ.

Розділ 1. Стан проблемної області.

Розділ 2. Інформаційне забезпечення.

Розділ 3. Математичне забезпечення.

Розділ 4. Програмне забезпечення.

Розділ 5. Розроблення стартап-проєкту.

Висновки. Список використаних джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди для доповіді (підготовка матеріалу для доповіді загальним обсягом (10-12 слайдів)

6. Дата видачі завдання *1 травня 2025 року*

Календарний план

№ з/п	Назва етапів дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Аналіз існуючих методів резюмування тексту	20.05.2025	виконано
2.	Планування архітектури системи	24.06.2025	виконано
3	Розробка серверної частини системи	14.08.2025	виконано
4	Імплементація веб-інтерфейсу	17.10.2025	виконано
5	Тестування системи на реальних даних	20.11.2025	виконано

Студент Блищак А. Т. _____
(підпис)

Керівник роботи Шиманський В. М. _____
(підпис)

АНОТАЦІЯ

Магістерська робота містить 67 сторінок пояснювальної записки, 1 додаток, 7 рисунків, 3 таблиці, 15 джерел.

У цій дипломній роботі досліджено методи автоматичного резюмування текстової інформації та розроблено інтелектуальну систему, що дозволяє генерувати узагальнені резюме з вхідного тексту. Основну увагу зосереджено на використанні сучасних моделей глибокого навчання, зокрема великих мовних моделей (LLM), для реалізації абстрактивного резюмування. Запропонована система реалізована у вигляді веб-застосунку з клієнтською частиною, створеною за допомогою Angular, і серверною частиною, побудованою на базі NestJS. Результати резюмування зберігаються у базі даних, що дає змогу аналізувати історію запитів. Система протестована на різних типах текстів, а також проведено оцінку якості сформованих резюме.

Ключові слова: штучний інтелект, автоматичне резюмування, великі мовні моделі, веб-застосунок, NestJS, Angular, OpenAI API.

ABSTRACT

The thesis contains 67 pages of explanatory note, 1 appendix, 7 figures, 3 tables, 15 used literary sources.

This master’s thesis explores the methods of automatic text summarization and presents the development of an intelligent system capable of generating concise summaries from input texts. The focus is placed on the application of modern deep learning approaches, particularly large language models (LLMs), to perform abstractive summarization. The proposed system is implemented as a web application, with a client-side built using Angular and a server-side based on NestJS. The summarization results are stored in a database, enabling further analysis of user requests. The system has been tested on various types of text data, and the quality of the generated summaries has been evaluated.

Keywords: artificial intelligence, text summarization, large language models, web application, NestJS, Angular, OpenAI API.

ТЕХНІЧНЕ ЗАВДАННЯ

Мета розробки:

Створити інтелектуальну систему, здатну автоматично формувати коротке змістовне резюме з вхідного тексту, використовуючи сучасні мовні моделі штучного інтелекту.

Функціональні вимоги:

1. Надання користувачем тексту через веб-інтерфейс.
2. Обробка тексту за допомогою серверної частини системи з використанням OpenAI API або іншої моделі машинного навчання.
3. Генерація абстрактивного резюме відповідного змісту.
4. Відображення результатів резюмування на клієнтській частині у зручному вигляді.

Нефункціональні вимоги:

1. Система має працювати у веб-браузері без потреби встановлення додаткового ПЗ.
2. Інтерфейс повинен бути зручним, адаптивним, з мінімалістичним дизайном.
3. Час генерації резюме, в ідеалі, не повинен перевищувати кількох секунд.
4. Забезпечення збереження даних відповідно до вимог безпеки.

Технічний стек:

- Frontend: Angular 17, Angular Material
- Backend: NestJS, OpenAI API, Mongoose
- База даних: Mongo DB
- Сторонні бібліотеки: openai

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

Позначення	Розшифрування
AI	Artificial Intelligence — штучний інтелект
NLP	Natural Language Processing — обробка природної мови
LLM	Large Language Model — велика мовна модель
API	Application Programming Interface — інтерфейс прикладного програмування
UI	User Interface — інтерфейс користувача
UX	User Experience — досвід користувача
JSON	JavaScript Object Notation — формат обміну даними
CRUD	Create, Read, Update, Delete — основні операції з даними
DB	Database — база даних
REST	Representational State Transfer — архітектурний стиль побудови веб-сервісів
OpenAI	Організація, що розробляє великі мовні моделі, зокрема GPT
AWS	Amazon Web Services — хмарна платформа для розгортання застосунків
S3	AWS Simple Storage Service — сервіс для зберігання файлів
RDS	AWS Relational Database Service — хмарна СУБД від Amazon
GPT	Generative Pre-trained Transformer — архітектура генеративних моделей OpenAI

MongoDB	Документо-орієнтована система керування базами даних (СКБД)
Angular	Веб фреймворк
NestJS	Платформа для створення масштабованих програм Node.js на стороні сервера

ВСТУП

У сучасному інформаційному суспільстві кількість доступної текстової інформації зростає в геометричній прогресії. Щодня люди стикаються з великими обсягами текстів — у науковій літературі, новинах, соціальних мережах, корпоративній документації тощо. Це ускладнює швидке та ефективне сприйняття і засвоєння важливої інформації. Тому автоматичне резюмування текстів — одна з ключових задач у галузі обробки природної мови (NLP), яка має широкий спектр застосування: від персональних асистентів до систем підтримки прийняття рішень.

Завдяки розвитку штучного інтелекту, зокрема великих мовних моделей (LLM), з’явилась реальна можливість створення інтелектуальних систем, здатних не лише виокремлювати ключову інформацію, а й формувати змістовні, граматично правильні резюме. Це відкриває нові горизонти в автоматизації інформаційного аналізу, навчанні, журналістиці, юриспруденції та інших сферах.

Тема створення інтелектуальної системи автоматичного резюмування тексту є надзвичайно актуальною у контексті цифровізації, розвитку мовних моделей та потреби у швидкій обробці великих обсягів інформації.

Об’єктом дослідження є процеси обробки природної мови (Natural Language Processing — NLP), а саме — автоматичне резюмування текстової інформації, що передбачає виявлення ключових думок у великих обсягах тексту для формування короткого та інформативного викладу змісту.

Предметом дослідження є методи та засоби побудови інтелектуальної системи автоматичного резюмування тексту з використанням великих мовних моделей та веб-технологій.

Метою роботи є розробка інтелектуальної системи, яка здатна автоматично генерувати стислий зміст тексту (резюме), базуючись на сучасних моделях глибинного навчання.

Основні завдання, які необхідно вирішити для досягнення мети:

1. Провести огляд і класифікацію сучасних методів автоматичного резюмування.
2. Визначити найефективніші підходи до абстрактивного резюмування з використанням LLM.
3. Спроєктувати архітектуру веб-застосунку для роботи з текстами.
4. Реалізувати серверну та клієнтську частини системи з інтеграцією OpenAI API або аналогів.
5. Реалізувати зберігання вхідних текстів та результатів резюмування у базі даних.
6. Провести тестування системи з різними текстовими даними.
7. Оцінити якість згенерованих резюме з урахуванням мовних критеріїв та відповідності оригіналу.

Наукова новизна роботи полягає у поєднанні сучасних технологій веб-розробки з найновішими підходами до автоматичного абстрактивного резюмування текстів на основі великих мовних моделей (наприклад, GPT-4). Розроблена система дозволяє у реальному часі генерувати резюме, адаптовані до потреб користувача, та зберігати історію обробок, що дає змогу аналізувати якість та ефективність роботи моделі у прикладному середовищі.

Крім того, у роботі досліджено можливості масштабування системи у хмарному середовищі, що є важливим аспектом для подальшого комерційного або інституційного впровадження розробленого програмного продукту.

Практична цінність роботи полягає у створенні працездатної, масштабованої системи автоматичного резюмування, яка може бути використана в освітніх, інформаційних, корпоративних та дослідницьких цілях. Розроблена система може стати основою для інтеграції в корпоративні інформаційні системи, новинні агрегатори, документообіг, системи технічної підтримки, персональні асистенти тощо.

Також робота сприяє поширенню практичного досвіду застосування штучного інтелекту в реальних прикладних задачах, що є актуальним для сучасної ІТ-індустрії.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1 Поняття та мета автоматичного резюмування тексту

Автоматичне резюмування тексту (англ. *text summarization*) — це процес генерації короткої версії довшого тексту, яка зберігає основний зміст вихідного повідомлення. Основною метою резюмування є зменшення обсягу текстової інформації при мінімальних втратах її інформативності. Такий підхід дозволяє користувачам швидко ознайомитися з головними ідеями великого текстового масиву, не витрачаючи час на його повне прочитання.

У сучасних умовах перенасичення інформацією резюмування набуває особливого значення. Його застосовують у сфері медіа, освіти, юриспруденції, технічної документації, електронної комерції, наукових досліджень, підтримки користувачів тощо.

1.2 Класифікація методів автоматичного резюмування

Усі методи автоматичного резюмування поділяються на дві основні категорії:

1.2.1. Екстрактивне резюмування

Екстрактивне резюмування (англ. *extractive summarization*) передбачає вибір найбільш значущих речень або фрагментів оригінального тексту без зміни їх змісту чи структури. Алгоритм визначає важливість фрагментів на основі статистичних показників, таких як частотність слів, позиція речення, співвідношення ключових слів тощо [8].

До найпоширеніших алгоритмів належать:

- TF-IDF (Term Frequency-Inverse Document Frequency)
- TextRank (графовий алгоритм, подібний до PageRank)
- LSA (Latent Semantic Analysis)

1.2.2. Абстрактивне резюмування

Абстрактивне резюмування (англ. *abstractive summarization*) імітує людський підхід до переказу тексту, перефразовуючи інформацію, змінюючи структуру речень та використовуючи синонімію. Це складніше завдання, оскільки вимагає глибшого розуміння семантики, синтаксису, контексту та логічних зв’язків. [8]

Методи абстрактивного резюмування базуються на нейронних мережах:

- Seq2Seq моделі з механізмом attention
- Transformer-моделі
- Pre-trained language models (BART, T5, GPT)

1.3. Роль обробки природної мови (NLP) у резюмуванні

Обробка природної мови є базовим елементом автоматичного резюмування. Основні задачі NLP, які використовуються в цьому процесі: [8]

- Токенізація та нормалізація тексту
- Частиномовний аналіз (POS-tagging)
- Визначення синтаксичної структури речень
- Семантичний аналіз та розпізнавання сутностей (NER)
- Виявлення ключових слів і тематичних кластерів

Інтеграція NLP з машинним навчанням дозволяє досягати високої точності та природності згенерованих резюме.

1.4. Розвиток великих мовних моделей (LLM)

Поява великих мовних моделей (LLM) таких як GPT-3, GPT-4, BERT, T5, BART суттєво покращила якість абстрактивного резюмування. Ці моделі здатні «читати» великі обсяги текстів, розуміти контекст, і навіть генерувати зв’язні тексти за допомогою навчання на мільярдах прикладів з інтернету [7].

Наприклад, модель GPT-4, розроблена компанією OpenAI, демонструє вражаючу здатність генерувати змістовні та релевантні резюме навіть для спеціалізованих текстів, таких як юридичні, медичні чи технічні документи.

Моделі типу BART або T5, [7] натреновані на специфічних задачах резюмування, також активно використовуються в академічних і прикладних проєктах.

1.5. Огляд існуючих систем та сервісів

На ринку вже існують різноманітні платформи, що використовують автоматичне резюмування:

- OpenAI ChatGPT / API — підтримує функцію узагальнення за допомогою спеціального запиту.
- Hugging Face Transformers — відкриті моделі резюмування (BART, T5, Pegasus).
- Google Cloud Natural Language API — базові можливості обробки тексту.
- SMMRY, Resoomer, Scholarcy — онлайн сервіси для автоматичного стискання тексту.

Проте більшість доступних рішень мають такі обмеження:

- Платні підписки або обмежений безкоштовний доступ
- Відсутність кастомізації для окремих доменів
- Обмежений контроль над процесом резюмування

Саме тому розробка власної системи резюмування на базі LLM з гнучкою архітектурою є доцільною та актуальною.

1.6. Виклики та проблеми автоматичного резюмування

Незважаючи на успіхи, у сфері автоматичного резюмування залишаються відкритими такі проблеми:

- Проблема "галюцинацій" моделей — генерація інформації, якої не було у вихідному тексті.
- Якість стислості vs. інформативність — надто коротке резюме може втратити важливі деталі.
- Проблема багатоаспектності — складно узагальнювати тексти, що містять багато тем або точок зору.
- Контекстні обмеження — навіть GPT-4 має обмеження на обсяг вхідного тексту.
- Мовна підтримка — більшість моделей краще працюють з англійською мовою, ніж з іншими.

Для подолання цих викликів необхідні локалізовані моделі, адаптація до конкретних доменів, а також тестування на прикладних задачах. Зокрема, проблема «галюцинацій» частково вирішується за рахунок використання методів контролю генерації, таких як обмеження температури вибірки або застосування перевірки фактів. [9] А проблема багатоаспектності текстів вирішується за допомогою сегментації вхідного документа на тематичні блоки з подальшим поетапним узагальненням.

Виходячи з моделей, які були використані для вирішення проблем з автоматичного резюмування тексту в даній роботі, а також з тих моделей які є на даний момент, модель GPT-5 виглядає як найбільш перспективна. Вона використовує сильні і швидкі алгоритми, має хорошу гнучкість, здатність працювати з великими та складними текстами. Тоді як, наприклад, GPT-4.1-nano - варто використовувати використовувати лише для простих завдань, де немає потреби в глибокому аналізі або в дуже якісному резюмуванні. [1]

Висновки до розділу

У цьому розділі було проведено огляд сучасного стану проблемної області автоматичного резюмування тексту. Було виявлено, що ця галузь стрімко розвивається завдяки впровадженню глибокого навчання та великих мовних

моделей. Розрізняють екстрактивні та абстрактивні методи, причому останні вимагають складніших алгоритмів і нейромережевих підходів. Сучасні LLM-моделі, такі як GPT-5 або T5, вже демонструють високий рівень продуктивності, однак все ще існують важливі виклики, пов’язані з якістю, достовірністю та контекстною релевантністю згенерованих резюме.

На цьому тлі актуальним є створення спеціалізованої інтелектуальної системи, яка б поєднувала можливості LLM, сучасні веб-технології та можливість адаптації до конкретного типу текстів.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

Інформаційне забезпечення відіграє ключову роль у створенні інтелектуальних систем, оскільки саме від доступності, структури, якості та обсягу даних залежить коректність роботи алгоритмів автоматичного резюмування. У цьому розділі буде розглянуто інформаційні ресурси, формати представлення текстових даних, джерела навчальних і тестових корпусів, а також способи інтеграції сторонніх моделей через API.

2.1 Джерела вхідних текстових даних

В основі роботи системи автоматичного резюмування лежить вхідний текст — як правило, це статті, аналітичні звіти, технічна документація, новини або довільні тексти, що потребують стислого переказу.

Основні джерела текстів, які можуть бути використані:

- Користувацьке введення (через веб-інтерфейс)
- Імпорт документів у форматах .txt, .docx, .pdf (через drag-and-drop або завантаження)
- Веб-сайти / RSS-стрічки (при інтеграції з parser-системами або web scraping)
- API зовнішніх сервісів (наприклад, Wikipedia, news API, корпоративні CMS)

Таким чином, система повинна мати модулі для обробки різних форматів вхідних даних, зокрема для витягування тексту з PDF або DOCX файлів.

2.2 Формати представлення та зберігання даних

Важливою частиною інформаційного забезпечення є визначення форматів представлення вхідної та вихідної інформації, які зручно обробляти програмно. У межах веб-застосунку пропонується використовувати наступні структури:

- Вхідні дані:

```
{
 "inputText": "Довільний текст, який потрібно скоротити...",
 "language": "uk",
 "summaryLength": "short" // або "detailed"
}
```

- Вихідні дані:

```
{
 "summaryText": "Стислий переказ оригінального тексту.",
 "confidence": 0.94,
 "model": "gpt-4"
}
```

Зберігання даних передбачається у нереляційній базі даних MongoDB у вигляді колекції texts:

2.3. Використання навчальних корпусів для тестування і калібрування

Для перевірки якості та коректності роботи системи використовуються відкриті навчальні корпуси текстів та резюме. Наприклад:

- CNN/DailyMail dataset — англомовні новинні статті з резюме
- XSum (Extreme Summarization) — короткі, 1-реченнєві резюме статей BBC
- WikiHow summarization dataset — інструктивні тексти з короткими висновками
- MultiLing 2015 — багатомовний корпус для багатомовного резюмування

Також можна створити власний корпус з навчальних або технічних матеріалів і вручну доданих резюме, що дозволить адаптувати модель до конкретного домену чи мови.

2.4. Інтеграція мовних моделей через API

Основне джерело інтелектуальної обробки — це мовна модель, яку можна використовувати як зовнішній сервіс через API:

Варіанти інтеграції:

- OpenAI API (GPT-3.5, GPT-4) — високоточне абстрактивне резюмування (англійська, українська підтримуються частково)
- Hugging Face Transformers API — доступ до моделей BART, T5, Pegasus
- Google Cloud NLP API — надає аналіз тексту та обмежене резюмування
- Cohere / Mistral / Anthropic Claude — альтернативні LLM-провайдери

Формат взаємодії з API:

- Надсилання тексту через POST-запит у форматі JSON
- Отримання результату з оцінкою якості (якщо підтримується)
- Обмеження по токенам: GPT-4 приймає до 128k токенів, інші — менше

2.5 Попередня обробка текстових даних

Перед подачею вхідних текстів на етап автоматичного резюмування необхідно здійснити їх попередню обробку. Попередня обробка (preprocessing) є обов'язковим етапом в інформаційному забезпеченні інтелектуальних систем, оскільки саме вона значною мірою визначає якість подальшого аналізу та коректність роботи мовних моделей.

До основних етапів попередньої обробки текстових даних належать [12; 14]:

- очищення тексту від службових символів, html-тегів, зайвих пробілів;
- нормалізація регістру (переведення тексту до нижнього або верхнього регістру);
- усунення зайвих повторів, дублікатів фрагментів;
- токенізація — поділ тексту на окремі слова або речення;
- видалення стоп-слів (прийменників, сполучників, часток);
- лематизація або стемінг слів.

У веб-застосунку для резюмування текстів preprocessing може виконуватися як на клієнтській стороні (частково), так і на сервері перед передачею запиту до мовної

моделі. Особливо важливим є коректне вилучення тексту з PDF та DOCX-документів, де можливі проблеми з кодуванням, таблицями, колонтитулами тощо.

Попередня обробка також дозволяє зменшити обсяг передаваних даних, що є критичним з огляду на обмеження мовних моделей за кількістю токенів.

2.6 Лінгвістичні особливості української мови

Однією зі складностей інформаційного забезпечення систем автоматичного резюмування є підтримка природних мов, зокрема української. Українська мова характеризується складною морфологією, відмінюванням, словозмінами та вільним порядком слів у реченні.

До основних лінгвістичних особливостей української мови, які впливають на процес автоматичного резюмування, належать:

- наявність семи відмінків;
- велика кількість словоформ від одного лексичного кореня;
- значна кількість синонімів;
- омонімія та багатозначність слів;
- складні синтаксичні конструкції.

На відміну від англійської мови, для української існує значно менше готових якісних корпусів та моделей для обробки природної мови. Тому використання універсальних багатомовних моделей є доцільним, однак потребує додаткового калібрування. [8]

Інформаційне забезпечення повинно враховувати ці лінгвістичні особливості, зокрема під час очищення, нормалізації та аналізу вхідних текстів українською мовою.

2.7 Оцінювання якості автоматичного резюмування

Важливим компонентом інформаційного забезпечення є система оцінювання якості згенерованих резюме. Якість автоматичного резюмування не може визначатися лише суб'єктивно, тому застосовуються формальні метрики.

До основних метрик оцінювання належать [12]:

- ROUGE-1 — оцінює збіг окремих слів;
- ROUGE-2 — оцінює збіг біґрам;
- ROUGE-L — оцінює найдовшу спільну підпослідовність;
- BLEU — використовується для порівняння текстових зразків.

Також застосовується експертна оцінка, при якій користувачі або фахівці оцінюють:

- інформативність резюме;
- логічність викладу;
- збереження ключових смислових елементів;
- відсутність фактологічних помилок.

Оцінювання якості є необхідним для калібрування системи та корекції параметрів генерації резюме.

2.8 Логування, аудит та зберігання історії обробки

Для забезпечення прозорості та можливості аналізу роботи системи необхідно реалізувати механізми логування та збереження історії обробки запитів.

Історія резюмування може містити:

- ідентифікатор користувача;
- вхідний текст (або його хеш);
- згенероване резюме;
- використану модель;
- час виконання запиту;

- рівень довіри або впевненості (confidence).

Зберігання історії дозволяє:

- відстежувати помилки;
- проводити аналіз якості роботи моделей;
- реалізувати функцію повторного перегляду попередніх резюме;
- здійснювати аудит інформаційної безпеки.

Зберігання журналів подій є частиною інформаційного забезпечення та важливою вимогою до сучасних інтелектуальних систем.

2.9 Масштабованість інформаційного забезпечення

Під час проєктування інформаційного забезпечення необхідно передбачити можливість масштабування системи у разі зростання кількості користувачів та обсягів оброблюваних даних.

Основні аспекти масштабованості:

- горизонтальне масштабування серверів;
- використання хмарних сховищ;
- кешування результатів запитів;
- балансування навантаження;
- асинхронна обробка великих текстів.

Масштабоване інформаційне забезпечення дозволяє підтримувати стабільну роботу системи навіть при значному збільшенні навантаження, що є критично важливим для стартап-проєктів.

2.10 Обмеження та виклики інформаційного забезпечення

Незважаючи на значні можливості сучасних мовних моделей, інформаційне забезпечення систем автоматичного резюмування має низку обмежень [2; 3; 8]:

- залежність від якості вхідних даних;
- похибки у згенерованих резюме;
- можливість втрати важливого контексту;
- обмеження на обсяг вхідного тексту;
- ризики витоку конфіденційної інформації.

До викликів також належать:

- потреба у значних обчислювальних ресурсах;
- складність об’єктивної оцінки якості резюме.

2.11 Архітектура інформаційних потоків у системі автоматичного резюмування

Інформаційне забезпечення веб-застосунку автоматичного резюмування тісно пов’язане з архітектурою обміну даними між його компонентами. У загальному вигляді система реалізує клієнт-серверну модель з використанням REST API та сервісів мовних моделей.

Основні інформаційні потоки у системі такі [9]:

- потік введення тексту від користувача до серверної частини;
- потік передавання тексту для обробки до зовнішнього API;
- потік повернення згенерованого резюме;
- потік збереження результатів у базі даних;
- потік відображення історії та оцінювання у користувацькому інтерфейсі.

На клієнтській стороні (Angular) формується запит, що містить:

- вхідний текст;

- обрану мовну модель;
- параметри довжини та стилю резюме.

На серверній стороні (NestJS) відбувається:

- валідація вхідних даних;
- журналювання запиту;
- виклик OpenAI API;
- обробка відповіді;
- запис результату у базу даних MongoDB.

Таким чином, інформаційне забезпечення забезпечує безперервний та контрольований рух даних між усіма рівнями системи.

2.12 Інформаційне забезпечення модуля вибору мовної моделі

Однією з ключових функціональних можливостей розробленого веб-застосунку є можливість вибору мовної моделі для обробки тексту. Це значно підвищує гнучкість системи та дозволяє адаптувати процес резюмування під різні типи текстів і вимоги до якості.

З інформаційної точки зору, модуль вибору моделі оперує такими даними:

- ідентифікатор моделі (наприклад, gpt-3.5, gpt-4);
- максимальна кількість токенів;
- орієнтовна вартість одного запиту;
- швидкість обробки;
- рівень точності.

Ці дані можуть зберігатися:

- у конфігураційних файлах серверної частини;
- у середовищних змінних;
- або у спеціальній колекції бази даних.

Користувацький інтерфейс отримує ці дані у вигляді структурованого списку та дозволяє обирати модель перед запуском процесу резюмування. Обрана модель передається у тілі запиту до серверної частини, після чого використовується при формуванні виклику до OpenAI API.

Інформаційне забезпечення модуля вибору моделі повинно гарантувати:

- коректність переданих ідентифікаторів;
- сумісність параметрів з обраною моделлю;
- захист від несанкціонованої зміни конфігурації.

2.13 Інформаційне забезпечення підсистеми оцінювання

У розробленому застосунку реалізовано окрему панель оцінювання якості обробки тексту (Evaluation Panel), яка відображається в інтерфейсі користувача з лівого боку. Інформаційне забезпечення цієї підсистеми ґрунтується на зберіганні та обробці числових і текстових показників якості резюмування.

До інформаційних параметрів оцінювання належать:

- інформативність резюме;
- відповідність змісту оригінальному тексту;
- рівень стислості;
- логічна зв’язність;
- суб’єктивна оцінка користувача.

Ці параметри можуть формуватися двома способами:

1. автоматично (на основі внутрішніх метрик або довіри моделі);
2. вручну користувачем через інтерфейс.

З інформаційної точки зору, кожен запис оцінювання містить:

- ідентифікатор сесії;
- ідентифікатор тексту;

- набір числових показників;
- коментар користувача;
- часову мітку.

У базі даних ці дані зберігаються разом з історією резюмування, що дозволяє здійснювати:

- аналітику якості роботи моделей;
- порівняння результатів різних моделей;
- формування статистичних звітів.

Підсистема історії запитів відіграє важливу роль в інформаційному забезпеченні веб-застосунку. Вона дозволяє користувачеві переглядати всі попередні запити, аналізувати результати, повторно відкривати тексти та резюме.

Інформаційна модель одного запису історії включає:

- оригінальний текст;
- згенероване резюме;
- використану мовну модель;
- дату і час обробки;
- оцінки якості;
- службові параметри.

Ці дані зберігаються у вигляді документів у MongoDB, що дозволяє ефективно виконувати пошук, фільтрацію та сортування. На ктівній стороні історія завантажується у вигляді масиву об’єктів та відображається у вигляді інтерактивного списку.

З інформаційної точки зору, підсистема історії дозволяє:

- реалізувати повторний аналіз текстів;
- порівнювати результати різних моделей;
- відстежувати динаміку якості резюмування;

- здійснювати аудит роботи користувачів.

Таким чином, історія запитів є важливим інформаційним джерелом для подальшого розвитку та оптимізації системи.

2.14 Обмеження OpenAI API та їх вплив на інформаційне забезпечення

Використання зовнішнього OpenAI API як основного обчислювального ядра накладає низку обмежень на інформаційне забезпечення системи автоматичного резюмування.

Основні обмеження:

- обмеження на кількість токенів у запиті;
- обмеження на кількість запитів за одиницю часу (rate limits);
- залежність від доступності зовнішнього сервісу;
- платний доступ до окремих моделей;
- неможливість повного контролю над внутрішнім механізмом роботи моделі.

Ці обмеження впливають на проєктні рішення інформаційного характеру, зокрема:

- необхідність попереднього скорочення великих текстів;
- збереження результатів для повторного використання;
- кешування запитів;
- контроль витрат на API-виклики.

Окрім цього, необхідно враховувати питання передачі конфіденційної інформації третім сторонам, що додатково підвищує вимоги до інформаційної безпеки.

2.15 Специфікація використаних моделей

У межах розробленого застосунку було використано кілька мовних моделей OpenAI, які відрізняються за продуктивністю, точністю та вартістю використання. А саме:

- GPT-4

Ця модель є досить точним мовним інструментом, має орієнтацію на складні завдання обробки. Основні характеристики:

  - висока точність формування підсумків;
  - здатність коректно працювати з великими обсягами тексту;
  - стабільне збереження контексту;
  - доцільність використання для основних, відповідальних операцій підсумування.
- GPT-4.1-nano

Це полегшена та оптимізована модель, призначена для швидкої обробки запитів із мінімальними витратами ресурсів. [1] Основними перевагами є:

  - доволі висока швидкість;
  - низьке споживання обчислювальних ресурсів;
  - достатня точність для простих задач підсумування;
  - ціна - ця модель є найдешевшою серед використовуваних у розробленому застосунку.
- GPT-5.1

Це сучасна мовна модель. Забезпечує покращену якість узагальнення, глибше розуміння контексту та складних семантичних зв’язків. [1] Особливості моделі:

- покращена якість узагальнення інформації;
- здатність працювати зі складними багатотематичними текстами;
- висока стабільність результатів;
- найдорожча модель серед використовуваних у розробленому застосунку.

Висновки до розділу

Інформаційне забезпечення є критичним компонентом системи автоматичного резюмування. У розділі було розглянуто інформаційне забезпечення системи автоматичного резюмування як ключовий елемент її функціонування. Проаналізовано джерела вхідних текстових даних, формати їх представлення, особливості зберігання, використання навчальних корпусів та інтеграцію мовних моделей через API. Окрему увагу приділено питанням попередньої обробки текстів, лінгвістичним особливостям української мови, оцінюванню якості резюмування, логуванню та масштабованості інформаційної інфраструктури. Врахування зазначених аспектів дозволяє створити надійну, безпечну та ефективну систему автоматичного резюмування текстової інформації.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

Математичне забезпечення є фундаментальною складовою інтелектуальних інформаційних систем, що реалізують обробку природної мови. У цьому розділі розглянуто математичні моделі, алгоритми та формалізми, що лежать в основі процесу автоматичного резюмування тексту, зокрема статистичні, евристичні та нейромережеві методи.

3.1 Формалізація задачі резюмування тексту

Задачу автоматичного резюмування можна подати як задачу трансформації вхідного тексту $T = \{s_1, s_2, ..., s_n\}$ у підмножину $S \subseteq T$ або у новий узагальнений текст T', що відображає основний зміст T, при цьому задовольняє обмеження на обсяг [14].

Залежно від підходу:

- Екстрактивне резюмування — вибір найінформативніших речень $S \subset T$
- Абстрактивне резюмування — генерація нового тексту T', що переказує зміст

Формально, задача резюмування полягає в оптимізації функції релевантності:

$$\text{argmax}_S \, f(S, T), \quad \text{де } S \subset T, \; |S| \leq k \tag{3.1}$$

Або генеративної функції:

$$T' = \text{Summarize}(T) = \arg\max_{T'} P(T' \mid T) \tag{3.2}$$

3.2 Статистичні методи резюмування

У ранніх підходах використовувалися частотні та позиційні метрики для вибору важливих речень:

- TF-IDF (Term Frequency–Inverse Document Frequency) — обчислення важливості терміну в контексті документа [13]:

$$\text{TF-IDF}_{t,d} = tf_{t,d} \cdot \log\left(\frac{N}{df_t}\right) \tag{3.3}$$

- TextRank — алгоритм ранжування речень за принципом PageRank на графі подібності:

Створюється граф G=(V,E)G = (V, E)G=(V,E), де вузли — речення, а ребра — схожість між ними [13]:

$$\text{sim}(s_i, s_j) = \frac{|\text{words}(s_i) \cap \text{words}(s_j)|}{\log(|s_i|) + \log(|s_j|)} \tag{3.4}$$

Після побудови графа обчислюється вага кожного речення через ітеративний алгоритм.

3.3 Нейромережеві методи абстрактивного резюмування

Сучасні підходи базуються на глибокому навчанні та трансформерах, таких як BART, T5, Pegasus, GPT.

Основна модель [13]:

$$P(y_1, ..., y_m \mid x_1, ..., x_n) = \prod_{t=1}^{m} P(y_t \mid y_1, ..., y_{t-1}, x) \tag{3.5}$$

де x — токени вхідного тексту, y — токени згенерованого резюме.

Використовується архітектура Seq2Seq з механізмом уваги (Attention) [13]:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.6}$$

У трансформерах це дозволяє моделі враховувати повний контекст тексту.

Популярні моделі:

- BART (Bidirectional and Auto-Regressive Transformers) — двосторонній енкодер + авто-регресивний декодер
- T5 (Text-To-Text Transfer Transformer) — уніфікація всіх NLP задач як текст-у-текст
- GPT-3.5 / GPT-4 — великі мовні моделі з autoregressive decoder-only архітектурою

3.4 Метричне оцінювання якості резюме

Для оцінки якості згенерованих резюме використовуються наступні метрики:

- ROUGE-N (Recall-Oriented Understudy for Gisting Evaluation) — n-грамне перекриття між референтним та згенерованим резюме [9]:

$$\text{ROUGE-N} = \frac{\sum_{S \in \text{Reference}} \sum_{gram_n \in S} \text{Count}_{match}(gram_n)}{\sum_{S \in \text{Reference}} \sum_{gram_n \in S} \text{Count}(gram_n)} \tag{3.7}$$

- BLEU — використовує модифіковане n-грамне співпадіння з покаранням за довжину
- BERTScore — семантична метрика, що використовує векторні подання слів через BERT

3.5 Задачі оптимізації та регуляризації

Під час тренування моделей використовуються [9]:

- Крос-ентропія як функція втрат:

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(y_t \mid y_{<t}, x) \tag{3.8}$$

- Регуляризація: Dropout, weight decay

- Оптимізація: Adam / AdamW

Також можливе застосування reinforcement learning with human feedback (RLHF) — коли система навчається на основі оцінок користувачів [9].

3.6 Ймовірнісна модель процесу генерації резюме

Процес абстрактивного резюмування у великих мовних моделях розглядається як стохастичний процес генерації послідовності токенів. Метою є знаходження такої послідовності слів [13; 15]

$T' = (w_1, w_2, \ldots, w_m)$,

яка максимізує апостеріорну ймовірність:

$P(T' \mid T)$

де T — вхідний текст.

Згідно з авторегресивною моделлю[14]:

$$P(T') = \prod_i P(w_i \mid w_1, \ldots, w_{i-1}, T) \quad (3.9)$$

Кожне слово вибирається на основі умовного розподілу ймовірностей, що обчислюється нейронною мережею. Для стабілізації генерації застосовується temperature-параметр τ:

$$P\tau(w_i) = \exp(z_i / \tau) / \Sigma_j \exp(z_j / \tau) \quad (3.10)$$

де z_i — логіт відповідного токена. Менші значення τ роблять генерацію детермінованішою, більші — різноманітнішою.

3.7 Математична модель механізму уваги

Механізм уваги дозволяє моделі фокусуватися на найбільш релевантних частинах тексту. Для кожного кроку декодування обчислюється:

$$e_{ij} = \text{score}(s_i, h_j) \quad (3.11)$$

де s_i — стан декодера, h□ — прихований стан енкодера.

Нормалізація ваг:

$$\alpha_{ij} = \exp(e_{ij}) / \Sigma_k \exp(e_{ik}) \quad (3.12)$$

Контекстний вектор:

$$c_i = \Sigma_j \alpha_{ij} \cdot h_j \quad (3.13)$$

Вектор c_i використовується для генерації наступного слова резюме. Це дозволяє моделі адаптивно зважувати важливість кожного фрагмента тексту. [12; 13]

У трансформерних архітектурах використовується механізм самоуваги (self-attention):

$Q = XW\_Q$
$K = XW\_K$
$V = XW\_V$

де:

- Q — матриця запитів,
- K — ключів,
- V — значень,

Це дозволяє моделі одночасно аналізувати різні семантичні підпростори тексту. [13]

3.8 Математична модель вибору мовної моделі у застосунку

У застосунку користувач може обирати різні мовні моделі. Процес вибору можна формалізувати як задачу багатокритеріальної оптимізації [13]:

$$M^* = \text{argmin} (\alpha \cdot C(M) + \beta \cdot T(M) + \gamma \cdot (1 - Q(M))) \quad (3.14)$$

де:

- C(M) — вартість обчислень,

- T(M) — середній час відповіді,
- Q(M) — оцінка якості резюме,
- α, β, γ — вагові коефіцієнти.

Ця модель дозволяє формально пояснити, чому користувач обирає між швидкістю та якістю генерації.

Панель оцінювання у застосунку базується на агрегації числових показників якості. Узагальнена формула:

$$\text{Score} = w_1 \cdot I + w_2 \cdot C + w_3 \cdot R + w_4 \cdot U \qquad (3.15)$$

де:

- I — інформативність,
- C — когерентність,
- R — ступінь стислості,
- U — оцінка користувача,
- w_i — вагові коефіцієнти, $\Sigma w_i = 1$.

Ця модель дозволяє:

- порівнювати резюме різних моделей;
- будувати середні статистичні показники;
- реалізувати автоматичний рейтинг моделей. [15]

3.9 Оцінювання якості резюмування через ROUGE, BLEU, BERTScore

Розглянемо три алгоритми для оцінювання якості резюмування: ROUGE-N, BLEU, BERTScore

3.9.1. ROUGE-N

ROUGE-N:

$$\text{ROUGE-N} = (\Sigma \text{Count}_{match}(\text{N-gram})) / (\Sigma \text{Count}_{ref}(\text{N-gram})) \qquad (3.16)$$

3.9.2. BLEU

$$BLEU = BP \cdot \exp(\Sigma_n w_n \ln p_n) \quad (3.17)$$

де:

- BP — штраф за довжину,
- p_n — точність n-грам,
- w_n — ваги.

3.9.3. BERTScore

$$Precision = (1/|X|) \Sigma \max \cos(x_i, y_j) \quad (3.18)$$

$$Recall = (1/|Y|) \Sigma \max \cos(y_j, x_i) \quad (3.19)$$

$$F1 = 2PR / (P+R) \quad (3.20)$$

Ці метрики використовуються для внутрішнього аналізу якості роботи моделей у системі. [13; 14]

3.10 Функції втрат, регуляризація та оптимізації

Основна функція втрат:

$$L = -\Sigma_i y_i \log(p_i) \quad (3.21)$$

Регуляризація Dropout:

$$h' = h \odot r,\ r \sim Bernoulli(p) \quad (3.22)$$

Weight decay:

$$\theta = \theta - \lambda\theta \quad (3.23)$$

Регуляризація зменшує перенавчання і стабілізує процес генерації. [15]

Алгоритм Adam:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (3.24)$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \quad (3.25)$$

$$\theta_t = \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon) \quad (3.26)$$

де:

- g_t — градієнт,
- α — швидкість навчання,
- β_1, β_2 — коефіцієнти згладжування. [15]

Середньоквадратична помилка:

$$MSE = (1/n) \Sigma (y_i - \hat{y}_i)^2 \quad (3.27)$$

Абсолютна похибка:

$$MAE = (1/n) \Sigma |y_i - \hat{y}_i| \quad (3.28)$$

Ці показники дозволяють оцінювати стабільність генеративного процесу.

3.11 Математична модель історії запитів у базі даних

Історія обробки описується множиною:

$$H = \{ (T_i, S_i, M_i, E_i, t_i) \} \quad (3.29)$$

де:

- T_i — вхідний текст,
- S_i — згенероване резюме,
- M_i — модель,
- E_i — вектор оцінок,
- t_i — час обробки.

Ця модель дозволяє будувати статистичні розподіли:

$$\bar{Q}(M) = (1/n) \Sigma E_i \quad (3.30)$$

для кожної моделі M. [13; 14]

3.12 Математична модель обмежень та токенізації тексту в OpenAI API

Запит до API формалізується як:

$$R = (T, M, L, \tau) \tag{3.31}$$

де:

- T — текст,
- M — модель,
- L — ліміт токенів,
- τ — temperature.

Ці обмеження враховуються у математичному проєктуванні системи. [12]

Перед подачею тексту до мовної моделі виконується попередня математична обробка — токенізація та нормалізація. Вхідний текст T подається у вигляді послідовності символів:

$T = (c_1, c_2, \ldots, c_n)$

Після процесу токенізації формується послідовність токенів:

$X = (x_1, x_2, \ldots, x_k)$, де $k \leq n$

Оператор токенізації можна подати у вигляді відображення:

$\varphi : T \rightarrow X$

Нормалізація тексту включає:

- перетворення до нижнього регістру,
- усунення службових символів,
- стандартизацію пробілів.

Для оцінювання складності вхідного тексту перед обробкою використовується ентропія Шеннона [14]:

$$H(T) = -\Sigma\, p(w_i) \log_2 p(w_i) \tag{3.32}$$

де:

- $p(w_i)$ — ймовірність появи слова w_i у тексті.

Високе значення H(T) означає:

- високе різноманіття лексики,
- складну семантичну структуру,
- підвищене навантаження на модель.

У системі ентропія може використовуватися для:

- автоматичного вибору моделі;
- адаптації параметра temperature;
- оцінювання складності джерельного матеріалу. [14]

3.13 Математична модель компресії інформації при резюмуванні

Зменшення обсягу тексту при резюмуванні описується коефіцієнтом стискання [14]:

$$K = |S| / |T| \;,\; 0 < K < 1 \tag{3.33}$$

де:

- |T| — довжина оригінального тексту,
- |S| — довжина резюме.

Абстрактивне резюмування формалізується як задача оптимізації:

$$S^* = \operatorname{argmax} P(S \mid T) \tag{3.34}$$

де:

- T — вхідний текст,
- S — резюме.

Таким чином задача зводиться до знаходження такої послідовності токенів, що максимізує умовну ймовірність при заданих обмеженнях за довжиною та стилем. [14; 15]

3.14 Математична модель багатокрокової обробки тексту у системі

Процес обробки в застосунку можна подати як композицію операторів:

$$F(T) = E(G(N(T))) \tag{3.35}$$

де:

- N(T) — нормалізація,
- G(·) — генерація резюме,
- E(·) — оцінювання результату.

Це дозволяє розглядати систему як каскадну математичну модель, що узгоджується з принципами системного аналізу. [12; 13]

Для аналізу історії використовується середнє значення показників:

$$\mu = (1/n) \Sigma x_i \tag{3.36}$$

Дисперсія:

$$\sigma^2 = (1/n) \Sigma (x_i - \mu)^2 \tag{3.37}$$

Ці показники застосовуються для:

- стабільності генерації;
- порівняння різних моделей;
- аналізу якості на великих вибірках.

Також обчислюється довірчий інтервал[13]:

$$CI = \mu \pm t \cdot (\sigma / \sqrt{n}) \quad (3.38)$$

Генерація кожного слова є випадковою величиною:

$$W \sim P(w \mid C) \quad (3.39)$$

де C — контекст.

Дисперсія:

$$D(L) = E(L^2) - (E(L))^2 \quad (3.40)$$

Ці характеристики дозволяють аналізувати передбачуваність результатів генерації для однакових запитів. [14]

3.15 Математична модель затримок у клієнт-серверній архітектурі

Загальний час відповіді:

$$T\_total = T\_net + T\_proc + T\_api + T\_render \quad (3.41)$$

де:

- T_net — час передачі даних;
- T_proc — серверна обробка;
- T_api — обробка OpenAI API;
- T_render — рендеринг у клієнті.

Висновки до розділу

У цьому розділі було детально проаналізовано математичне підґрунтя системи автоматичного резюмування тексту. Розглянуто класичні статистичні методи та сучасні нейронні архітектури, включно з метриками оцінки результатів. Таким чином, математичне забезпечення є основою для коректної реалізації інтелектуальних механізмів обробки текстів у системі. Розширене математичне моделювання дозволило формально описати допоміжні процеси — від токенізації та нормалізації тексту до аналізу затримок і пропускної здатності системи.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

Програмне забезпечення є центральною складовою інтелектуальної системи автоматичного резюмування тексту. Його архітектура повинна забезпечувати зручний інтерфейс для користувача, ефективну обробку текстів, збереження результатів та можливість масштабування. У цьому розділі детально розглянуто архітектуру системи, вибір інструментів, використання моделей штучного інтелекту, інтеграцію з базою даних MongoDB, а також бюджетні витрати на реалізацію.

4.1 Технологічний стек

Система реалізована як веб-додаток за архітектурою **«клієнт-сервер»**, де клієнтська частина відповідає за взаємодію з користувачем, а серверна – за обробку запитів, збереження даних та генерацію резюме за допомогою мовних моделей [5; 6].

Таблиця 4.1 – Технологічний стек

Компонент	**Технологія / Інструмент**
Клієнтська частина	Angular 17
Серверна частина	NestJS (Node.js + TypeScript)
База даних	MongoDB + Mongoose
AI API	OpenAI (GPT-4.1-nano, GPT-4.1, GPT-5)
VCS	Git

4.2 Архітектура системи

Система має модульну архітектуру, що включає:

- Frontend (Angular):
    - Компонент введення тексту
    - Список попередніх запитів

- Вибір моделі (GPT-4.1 / GPT-4.1-nano / GPT-5)

- Backend (NestJS):
  - API /summarize
  - Логіка з опрацювання якості у взаємодії із допоміжними бібліотеками
  - Сервіси взаємодії з OpenAI
  - Модулі збереження текстів у MongoDB
- MongoDB (Mongoose):
  - Колекції: textsummary

4.3 Реалізація резюмування

Користувач вводить текст, вибирає модель (наприклад, GPT-4.1-nano або GPT-4), і після надсилання запиту отримує короткий виклад тексту.

Обробка виглядає так:

1. Angular формує запит:

```
{
   text: "Повний текст для скорочення...",
   model: "gpt-4"
} [5]
```

2. NestJS-API обробляє запит та відправляє його до OpenAI:

```
const response = await openai.chat.completions.create({
        model: dto.gptModel,
        input: dto.text,
}); [6]
```

3. NestJS також відсилає запит на отримання оцінки оброблення поточного тексту до OpenAI:

```
const evaluationPrompt = ` Evaluate the quality of the summary. …`
const response = await this.openApiClient.responses.create({
 model: gptModel,
 input: evaluationPrompt,
});[6]
```

4. Результат зберігається в MongoDB та повертається користувачу.

4.4. Система оцінювання якості (Evaluation System)

Це одна з унікальних складових застосунку. Її функції та структура:

- label
- value
- comment

Оцінка за кількома критеріями:

- Змістовність
- Точність
- Стислий виклад
- Збереження сенсу
- Стилістична грамотність.

Кожна оцінка прив’язана до відповідного запису історії. Структура:

- _id
- summaryId
- evaluations

Система сама створює відповідні метрики для кожного запису з резюмування тексту, може обчислювати середню якість роботи певної моделі.

Після опрацювання запиту, підсумований текст відкривається у спеціальному акордеоні під секцією введення тексту, а також зявляється відповідна оцінка тексту за відповідними критеріями з лівого боку від акордеону, який містить поле для введення тексту.

Це перетворює застосунок з простого інструменту в невелику аналітичну платформу, яка дає змогу опрацювати текст та самому проаналізувати результати по критеріях і, за необхідності, повторити опрацювання.

4.5. Робота з MongoDB

Для зберігання історії запитів, даних користувача та інформації про використані моделі використовується MongoDB через Mongoose.

Приклад схеми збереження резюме:

```
const SummarySchema = new mongoose.Schema({
  originalText: String,
  summary: String,
  gptModel: String,
  createdAt: { type: Date, default: Date.now },
  evaluations: { label: string, value: number, comment: string }
}); [4]
```

Колекція: textsummary

4.6. Використання різних моделей AI

OpenAI надає багато різних моделей для обробки запитів. Ці моделі відрізняються між собою набором характеристик, таких як ціна, швидкість та якість опрацьованих запитів.

Приклад моделей [1; 9]:

- GPT-3.5-turbo — швидка й економна модель, підходить для простих резюме та масових запитів при обмеженому бюджеті.
- GPT-4 — потужніша модель з кращою якістю та глибшим розумінням контексту; рекомендується для складних текстів.
- GPT-4-turbo — версія GPT-4 з оптимізаціями для швидкості та вартості, часто служить «золотою серединою» між ціною й якістю.
- o4-mini / o-series — компактні моделі OpenAI, оптимізовані для швидкого та економного reasoning (корисні для кодування, матаналізу та візуальних задач).
- o3, o3-mini — моделі серії o3 (різні варіанти)—сильні в задачах reasoning, є варіанти «mini» для дешевших/швидших викликів.

- GPT-5 — найновіший на момент написання флагман OpenAI; [1] пропонує покращене логічне мислення, reasoning та розширені можливості (доступні також «міні»/чат-варіанти). При впровадженні в продакшн слід враховувати змінні умови доступу, ціноутворення та обмеження контекстного вікна.

Користувач може самостійно обрати, яка модель буде використана під час обробки, результат і якість буде відрізнятись в залежності від моделі.

4.7. Логіка інтеграції з OpenAI API

У застосунку використано різні режими взаємодії з OpenAI API. Основні принципи інтеграції:

1. Уніфікований сервіс OpenAIService
   Він реалізує:
   - вибір моделі
   - формування параметрів запиту
   - оцінка підсумування тексту
   - обробку помилок.

2. Механізм fallback-моделей
   Якщо основна модель недоступна або перевищено rate limit, API автоматично переходить на запасну модель, наприклад:
   - з GPT-4 → GPT-4-turbo
   - з o3 → o3-mini

Це підвищує стабільність.

3. Використання температури та токенів
   Для резюмування встановлено:
   - temperature: 0.2
   - max_tokens: 1024

Це забезпечує стабільні, не хаотичні відповіді.

4.7.1 Асиметрична обробка токенів

Система окремо рахує вхідні токени та вихідні — це дозволяє точно підраховувати бюджет користування. Такий підхід важливий для контролю витрат, оскільки вхідні токени (тобто текст, який користувач надсилає для обробки) та вихідні токени (результат, який генерує модель) можуть сильно відрізнятися за обсягом. [2] Наприклад, короткий запит може призвести до великого обсягу згенерованого тексту, що вплине на суму спожитих токенів.

Окремий підрахунок також дозволяє аналізувати ефективність використання моделей штучного інтелекту та оптимізувати запити: можна передбачати, скільки токенів буде витрачено на певний тип завдання, а також контролювати витрати при масовому обробленні даних. [9] Крім того, така деталізація важлива для аналітики, оскільки дозволяє окремо оцінювати обсяг вхідних даних, що надаються користувачами, та обсяг згенерованих результатів, що може бути використано для побудови статистики використання та прогнозування бюджетів.

У сучасних системах, які працюють з великими мовними моделями, асиметрична обробка токенів є стандартом для точного відстеження ресурсів і забезпечення прозорості у розрахунках оплати за використання API.

4.8. UI/UX архітектура та особливості клієнтської частини

Оскільки застосунок орієнтований на швидку й інтуїтивну взаємодію користувача з інструментом резюмування тексту, значну увагу було приділено проєктуванню UI/UX. Архітектура клієнтської частини передбачає чітке розділення зон інтерфейсу відповідно до ключових функцій системи.

Основні екрани клієнтської частини:

1. Головний екран обробки тексту
    - Поле введення тексту

- Панель вибору моделі
- Кнопка запуску обробки
- Секція “Результат резюмування”
- Інтерактивна оцінка якості (Evaluation Panel)
- Панель історії попередніх текстів і результати їх обробки, включно з результататами оцінки резюмування цих текстів.

2. Панель історії (History Panel)
    - Відображення минулих запитів
    - Автоматичне збереження кожного резюме
    - Відкриття попередніх результатів у відповідних акордеонах
    - Відображення результаців оцінки резюмування тексту
3. Панель оцінки резюмування (Evaluation Panel)
    - Функціонал виставлення оцінок за критеріями:
        - точність;
        - стисліcть;
        - відповідність темі, чіткість;
        - охоплення;
        - читабельність;
    - Збереження оцінок у БД MongoDB.
4. Налаштування
    - Вибір моделі
    - Обсяг тексту, який зберігати в історії

4.8.1 Застосовані UI рішення:

- Material Design компоненти (<mat-expansion-panel>, <mat-form-field>, <mat-menu>).
- Анімаційні переходи для акордеонів та панелі історії.

Користувач може одночасно редагувати текст, працювати з історією та виконувати оцінку попередніх результатів – і все це завдяки грамотному розділенню компонентів у Angular.

4.9. Можливості масштабування та подальший розвиток

Оскільки застосунок уже має модульну структуру та API-first дизайн, його легко розширити у майбутньому.

Потенційні напрями розвитку:

- Додавання авторизації користувачів
- Введення тарифних планів
- Збереження чорновиків
- Версійність резюмувань
- Підтримка багатомовності (ukr/eng/etc.)
- Покращений UI/UX
- Експорт у PDF / DOCX
- Групування історії за проектами
- Додавання інтерфейсу порівняння моделей ("compare summaries")

Таким чином, створена архітектура дає потужний фундамент для майбутнього розвитку стартапу як комерційного продукту.

4.10. Бюджет проєкту

Для реалізації функціоналу було необхідно поповнити баланс на платформі OpenAI, оскільки використання моделей GPT є платним. Орієнтовна вартість за 1K токенів (включаючи вхідні та вихідні) [1; 2; 3]:

Таблиця 4.2 – Опис та ціни AI моделей на ринку

№	Назва моделі	Доступ через	Ціна за 1000 токенів (вхід / вихід)	Швидкість	Якість	Примітки
1	GPT-3.5-turbo	OpenAI API	$0.0015 / $0.002	4/5	4/5	Ефективна, бюджетна модель
2	GPT-4	OpenAI API	$0.03 / $0.06	2/5	5/5	Висока якість, повільніше
3	GPT-4-turbo	OpenAI API	$0.01 / $0.03	4/5	5/5	Швидша та дешевша версія GPT-4
4	GPT-4o	OpenAI API, Chat GPT	$0.025/$0.1	3/5	3/5	Модель, яка використовується в ChatGPT
5	Claude 3 Haiku	Anthropic	$0.00025 / $0.00125	5/5	4/5	Дешева, але добра якість
6	Claude 3 Sonnet	Anthropic	$0.003 / $0.015	4/5	5/5	Альтернатива GPT-4

Продовження таблиці 4.2

№	Назва моделі	Доступ через	Ціна за 1000 токенів (вхід / вихід)	Швидкість	Якість	Примітки
7	Claude 3 Opus	Anthropic	$0.015 / $0.075	2/5	5/5	Для найскладніших задач
8	Gemini 1.5 Pro	Google AI Studio	$0.0025 / $0.005	4/5	4/5	Хороша альтернатива GPT
9	Gemini 1.5 Flash	Google AI Studio	$0.0005 / $0.001	5/5	3/5	Дуже швидка, підходить для чернеток
10	Mistral 7B Instruct	Open-source (local)	$0.0 (локально)	3/5	3/5	Потрібні власні обчислювальні ресурси
11	Mixtral 8x7B	Open-source (local)	$0.0 (локально)	2/5	4/5	Потрібен потужний GPU (16GB+)

Продовження таблиці 4.2

№	Назва моделі	Доступ через	Ціна за 1000 токенів (вхід / вихід)	Швидкість	Якість	Примітки
12	LLaMA 3 8B	Meta AI (local/API)	$0.0–$0.002	3/5	4/5	Можна запускати локально через Ollama
13	LLaMA 3 70B	Meta AI (API)	$0.002 / $0.006 (через сторонні сервіси)	2/5	5/5	Якість GPT-4 рівня
14	Command R+ (Reka)	Reka API	$0.002 / $0.006	4/5	4/5	Орієнтована на задачі скорочення
15	Cohere Command R	Cohere API	Безкоштовно з обмеженнями	3/5	4/5	Має спеціальний режим резюмування
16	Zephyr 7B	Hugging Face, Local	$0.0 (локально)	2/5	3/5	Тренована спеціально для діалогів

Продовження таблиці 4.2

№	Назва моделі	Доступ через	Ціна за 1000 токенів (вхід / вихід)	Швидкість	Якість	Примітки
17	o3	OpenAI API	$0.01 / $0.04	3/5	4/5	Модель малого міркування, яка забезпечує високий інтелект
18	o3-mini	OpenAI API	$0.01 / $0.03	3/5	4/5	Спрощена версія моделі o3
19	o4-mini	OpenAI API	$0.01 / $0.04	3/5	4/5	Доволі ефективна модель для кодингу та візуальних задач
20	gpt-5	OpenAI API	$0.00125 / $0.0100	3–4 / 5	5/5	Найновіша модель (станом на 2025), висока якість резюме

В даній роботі, найбільш використовуваними моделями є GPT-4 та GPT-5.1 Незважаючи на те, що модель є порівняно дорожчою, вона добре справляється з текстовими задачами і є доволі швидкою. [1]

4.11. Порівняльний аналіз використаних моделей, результати та метрики

У додатку використовуються наступні моделі від OpenAI [1]:

- GPT-4
- GPT-4.1-nano
- GPT-5.1

Використавши один і той самий текст для підсумування, використовуючи кожну модель по черзі, можна проаналізувати, яка модель найкраще справляєтья з сумаризацією тексту, використовуючи такі критерії:

- достовірність (Faithfulness);
- стислість (Brevity);
- чіткість (Clarity);
- охоплення (Coverage);
- читабельність (Readability);

Розгляньмо тестовий текст для порівняння резюмування між моделями:

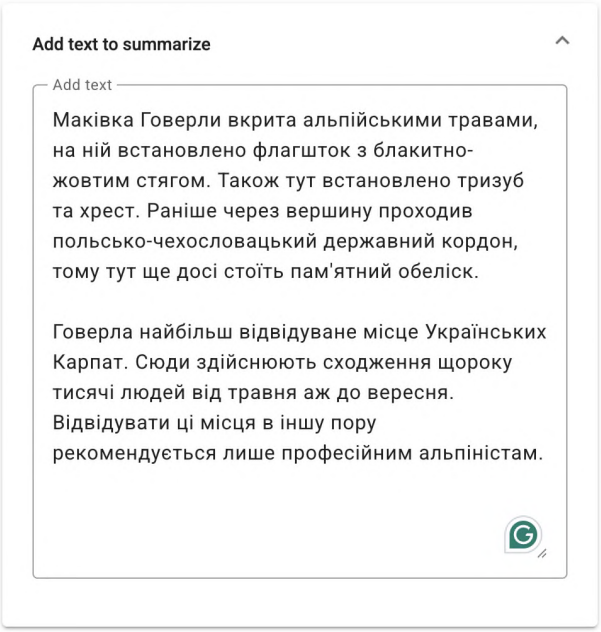

Рисунок 4.1 – Вхідний текст для резюмування

Проаналізуймо резюмований текст в розробленій програмі, а також результати якості підсумування кожною з моделей.

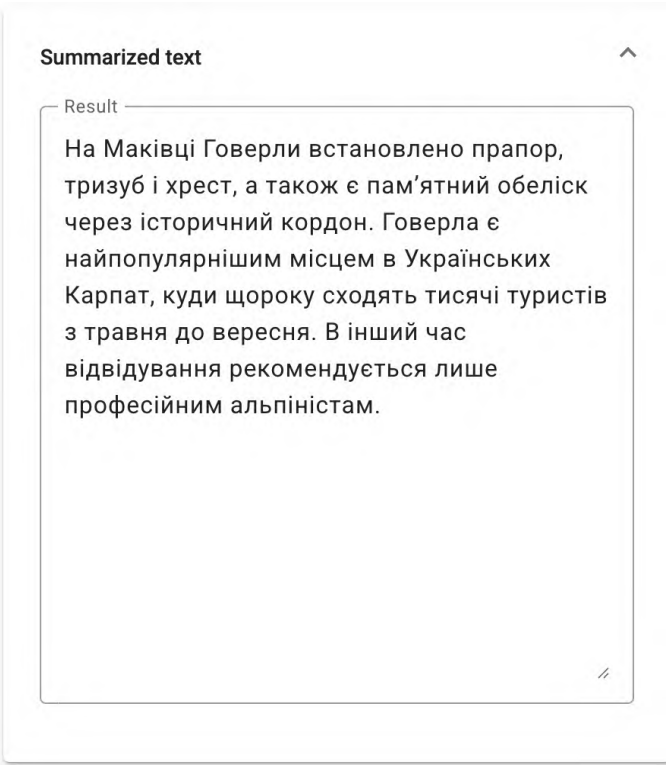

Рисунок 4.2 – Результат резюмування, модель GPT-4.1-nano

Рисунок 4.3 – Якість резюмування, модель GPT-4.1-nano

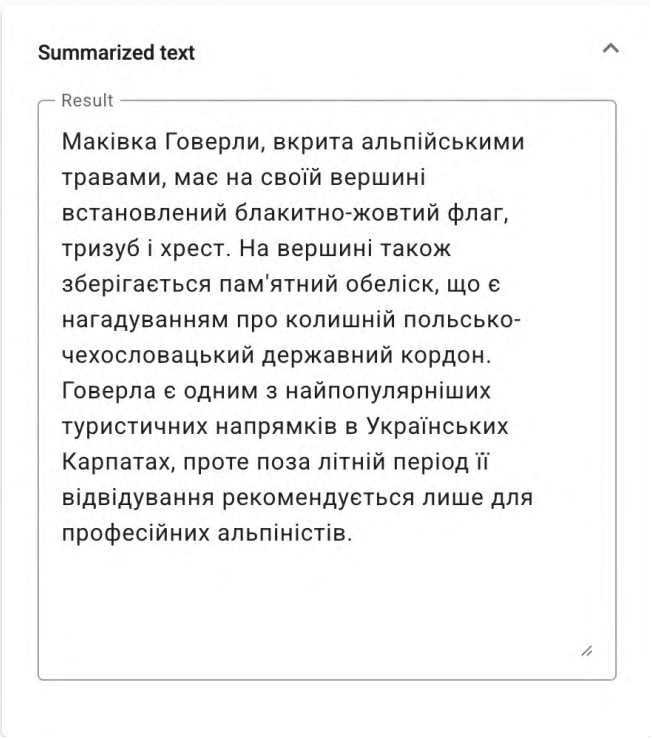

Рисунок 4.4 – Результат резюмування, модель GPT-4

Рисунок 4.5 – Якість резюмування, модель GPT-4

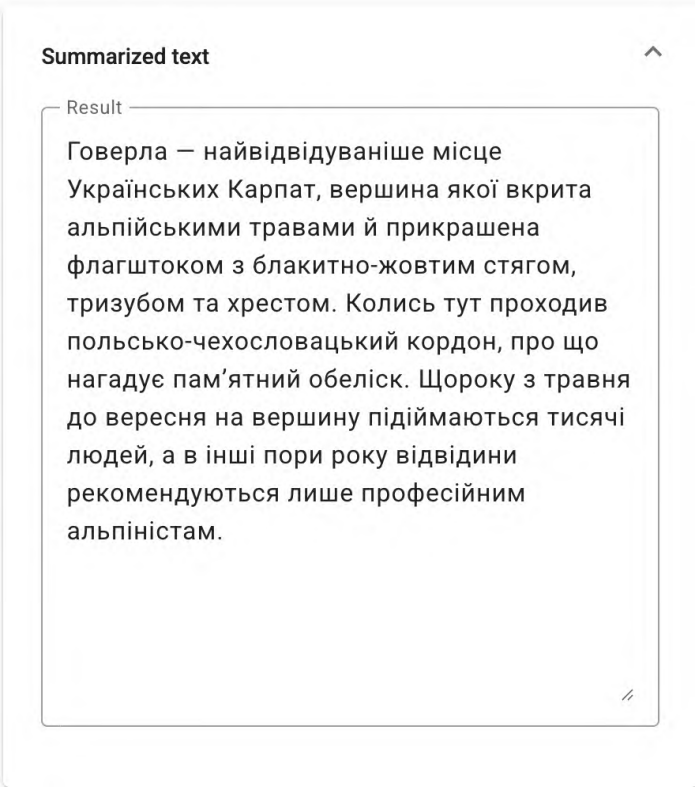

Рисунок 4.6 – Результат резюмування, модель GPT-5.1

Рисунок 4.7 – Якість резюмування, модель GPT-5.1

Складемо таблицю аналізу результатів для кожної з моделей, на основі визначеної якості резюмування:

Таблиця 4.3 – Порівняння якості резюмування тексту GPT моделями

Модель	**Достовірність**	**Стислість**	**Чіткість**	**Охоплення**	**Читабельність**
GPT-4.1-nano	85/100	80/100	90/100	85/100	90/100
GPT-4	100/100	95/100	97/100	97/100	100/100
GPT-5.1	100/100	100/100	98/100	90/100	98/100

На основі отриманих результатів, можна зробити наступні висновки:

- Модель GPT-4.1-nano продемонструвала хороші результати, але видно, що модель простіша і такі характеристики як стислість, достовірність та охоплення можуть відчутно просідати. Як наслідок - не достатньо якісно підсумований текст.
- Модель GPT-4 показала чудові результати резюмування, місцями перевершуючи модель новішого покоління GPT-5.1. Результати підсумування тексту на доволі високому рівні, враховуючи показники вище, особливо - достовірність та читабельність.
- Модель GPT-5.1 також продемонструвала чудові результати з усіх вище вказаних показників. Показник охоплення виявився найменшим, що може казати про деякі недоліки цієї моделі і може вказати на те, що потрібно більш детально проаналізувати дану модель на різних текстах, якщо стоїть питання вибору моделі для резюмування тексту.

Висновки до розділу

Програмне забезпечення системи було реалізовано з урахуванням сучасних підходів до веб-розробки, хмарних сервісів та інтеграції зі штучним інтелектом. Система побудована на стеку Angular + NestJS + MongoDB, із підтримкою OpenAI API та можливістю вибору моделей для резюмування. Реалізація передбачає збереження та відображенняісторії, а також якості резюмування тексту. Архітектура розробленого програмного забезпечення сприятливе для масштабування, а також можна проводити контроль та аудит витрат на використання мовних моделей. Було зроблено порівняння та аналіз трьох моделей, які були використані у застосунку. Розробка та тестування підтвердили надійність і готовність системи до реального використання.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

У сучасних умовах розвитку інформаційних технологій та економіки все більшого значення набувають стартап-проєкти як ефективний інструмент впровадження нових рішень. Особливо актуальними є програмні продукти, які створені на основі штучного інтелекту, які дозволяють автоматизувати складні процеси обробки інформації, а також підвищити продуктивність праці користувачів.

5.1 Опис ідеї проєкту

У сучасних умовах розвитку технологій і постійного зростання кількості інформації, все більш актуальною постає проблема швидкої та ефективної обробки текстових даних. Користувачі щоденно стикаються з необхідністю аналізу великих обсягів інформації: наукових статей, новин, звітів, навчальних матеріалів, технічної документації тощо. [9; 10] Обмеженість часу та перевантаження інформацією зумовлюють потребу в інструментах, які здатні автоматизувати процес стислого викладу основного змісту тексту.

Ідея стартап-проєкту полягає у створенні веб-застосунку для автоматичного резюмування текстів із використанням сучасних великих мовних моделей штучного інтелекту. Користувачеві надається можливість ввести довільний текст, обрати модель для обробки та отримати структурований, стислий і змістовний підсумок. Додатково реалізовано модуль оцінювання якості згенерованого резюме за такими критеріями, як точність, повнота, зрозумілість, лаконічність та читабельність.

Цільовою аудиторією проєкту є:

- студенти та викладачі;
- науковці та дослідники;
- журналісти;
- бізнес-аналітики;
- фахівці, які працюють із великими обсягами текстової інформації.

Основна цінність продукту полягає в економії часу користувача, підвищенні продуктивності обробки інформації та зниженні когнітивного навантаження. Система дозволяє не лише отримати короткий зміст тексту, а й проаналізувати якість результату, що підвищує довіру до використання штучного інтелекту.

Проєкт реалізований як клієнт-серверний веб-застосунок із використанням сучасного стеку технологій: Angular для клієнтської частини, NestJS для серверної логіки та MongoDB для зберігання даних. Для обробки тексту використовується OpenAI API.

Таким чином, ідея стартапу орієнтована на створення програмного продукту в галузі інтелектуальної обробки тексту, що має доволі широкий спектр практичного застосування.

5.2 Аналіз технологічних можливостей реалізації ідей проєкту

Для реалізації ідеї застосовуються сучасні інформаційні та програмні технології.

Виготовлення програмного продукту здійснюється за допомогою веб-технологій та сервісів штучного інтелекту. Клієнтська частина реалізується у вигляді веб-інтерфейсу з полем введення тексту, полем відображення результатів, панеллю якості резюмування та панеллю історії. Серверна частина забезпечує обробку запитів, передачу тексту до мовної моделі, отримання результатів та збереження даних у базі даних.

Ключовою технологією є використання великих мовних моделей для автоматичного аналізу та узагальнення текстів. Дані технології вже існують у вигляді готових хмарних API-сервісів і не потребують створення з нуля. Для реалізації проєкту використовуються наступні технології:

- веб-технології для створення інтерфейсу користувача;
- серверні технології для обробки запитів;
- сервіси мовних моделей для резюмування текстів;

- системи керування базами даних для збереження історії.

Ці технології є повністю доступними, мають відкриту документацію та підтримуються сучасними програмними середовищами розробки. Всі необхідні інструменти є наявними на ринку програмного забезпечення.

Ідея не потребує створення нових технологій, а базується на поєднанні вже існуючих та доступних програмних рішень.

5.3 Аналіз ринкових можливостей запуску стартап-проєкту

Ринок інтелектуальних інформаційних систем, зокрема систем обробки текстової інформації, демонструє стабільну тенденцію до зростання. Збільшення обсягів цифрового контенту, наукових публікацій, навчальних матеріалів та бізнес-документації формує стійкий попит на інструменти автоматичного резюмування.

Основними потенційними групами клієнтів є:

- студенти та викладачі закладів вищої освіти;
- наукові працівники;
- журналісти та контент-менеджери;
- бізнес-аналітики та менеджери.

Для кожної з груп ключовими вимогами до системи є швидкість обробки, зрозумілість інтерфейсу, точність резюмування, можливість оцінювання якості та збереження історії запитів.

Факторами, що сприяють впровадженню проєкту, є:

- активний розвиток ринку штучного інтелекту;
- цифровізація освіти та бізнес-процесів;
- зростання попиту на автоматизацію аналітичної роботи.

Факторами, що можуть перешкоджати впровадженню, є:

- наявність конкурентних сервісів;
- висока залежність від сторонніх API-провайдерів;
- необхідність фінансових витрат на використання мовних моделей. [11]

До конкурентів належать як спеціалізовані сервіси резюмування, так і багатофункціональні платформи обробки тексту. Перевагами запропонованої системи є простота інтерфейсу, наявність панелі оцінювання якості та збереження історії роботи користувача.

Можна зробити висновок, що ринок є перспективним для запуску стартап-проєкту, однак потрібно думати над певними інноваціями, які зроблять цей продукт більш привабливим для цільової аудиторії.

5.4 Розроблення ринкової стратегії проєкту

Для даного стартап-проєкту можна обрати стратегію диференційованого маркетингу, оскільки система може використовуватись різними сегментами ринку з певними відмінностями у вимогах:

- для освітнього сегмента — акцент на навчальних текстах;
- для бізнес-сегмента — на аналітичних і звітних матеріалах.

Проєкт буде позиціонуватися на ринку як зручний, швидкий та доступний інструмент для автоматичного резюмування з можливістю контролю якості результату.

Основний акцент робиться на:

- зручності та простоті використання;
- хорошій якості результатів;
- наявності історії запитів;
- наявності результатів якості обробки.

5.5 Маркетингова програма стартап-проєкту

Маркетингова програма стартапу спрямована на формування впізнаваності бренду, залучення користувачів та утримання клієнтів. Основними інструментами маркетингового просування визначено цифрові канали комунікації.

Основні елементи маркетингової програми:

- створення офіційного сайту та лендингу продукту;
- активна присутність у соціальних мережах (Instagram, LinkedIn тощо);
- контент-маркетинг (статті, відеоогляди, навчальні матеріали);
- реклама у пошукових системах;
- співпраця з освітніми закладами.

Для залучення перших користувачів передбачено:

- безкоштовний доступ до повного функціоналу на тестовий період;
- проведення презентацій та вебінарів;
- участь у студентських стартап-конкурсах.

Регулярний аналіз показників (кількість користувачів, активність, середній час сесії) дозволить коригувати маркетингову стратегію в режимі реального часу. [10; 11]

Висновки до розділу

У даному розділі було розглянуто процес розроблення стартап-проєкту веб-застосунку для автоматичного резюмування текстів із використанням технологій штучного інтелекту. Описано ідею продукту, визначено цільову аудиторію та конкурентні переваги. Сформовано ринкову стратегію та маркетингову програму. Запропонований стартап має хороший потенціал комерціалізації, відповідає сучасним тенденціям та може бути успішно використаний у різних сферах діяльності.

ВИСНОВКИ

У межах дипломного проекту була розроблена та досліджена інтелектуальна система автоматичного резюмування тексту, що дозволяє автоматизувати процес скорочення великих обсягів текстової інформації до найбільш інформативного змісту. Робота охоплює всі ключові аспекти створення сучасного програмного застосунку, який базується на технологіях штучного інтелекту.

У результаті виконання роботи:

1. Досліджено стан проблемної області: було проаналізовано існуючі підходи до автоматичного резюмування тексту, зокрема екстрактивні та абстрактивні методи, а також сучасні лінгвістичні та нейромережеві моделі, що використовуються в подібних системах.
2. Розроблено інформаційне, математичне та програмне забезпечення системи:
    - Інформаційне забезпечення охоплює текстові дані користувача, формат введення/виведення та структуру даних.
    - Математичне забезпечення включає використання мовних моделей великого обсягу (LLM), таких як GPT-5.1, GPT-4
    - Програмне забезпечення побудоване з використанням Angular (frontend), NestJS (backend), MongoDB (база даних), а також OpenAI API для отримання результатів генерації резюме.
3. Реалізовано веб-застосунок з інтуїтивно зрозумілим інтерфейсом, де користувач може ввести текст, обрати модель для резюмування, переглянути результат та скористатися історією обробки текстів для подальшого аналізу, використовуючи також панель з результатами якості обробки. Забезпечено зручність у роботі з API та асинхронну обробку запитів.
4. Проведено тестування системи: оцінено точність, якість, швидкість і вартість генерації резюме для різних моделей. Виявлено, що GPT-4 та GPT-5.1 демонструють найвищу якість, однак GPT-4.1-nano забезпечує хороше співвідношення ціна/якість для типових текстів.

5. Розраховано бюджет використання комерційних мовних моделей та розглянуто альтернативні варіанти з локальними LLM. Обґрунтовано доцільність комбінування платних і безкоштовних рішень залежно від завдань.
6. Підтверджено практичну значимість розробленої системи — вона може бути інтегрована в різні корпоративні рішення (CRM-системи, електронну пошту, документообіг) та застосована в освітньому, журналістському й дослідницькому середовищах.
7. Розроблено програму стартап-проєкту для подальшого розвитку і просування розробленого продукту.

Розроблений застосунок демонструє сучасний підхід до побудови AI-рішень і має перспективи подальшого розвитку, зокрема, шляхом розширення використання сучасніших мовних моделей на великих даних незалежно від області використання, а такожрозширення функціоналу (наприклад, багатомовне резюмування чи класифікація резюме).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. OpenAI (2023). GPT-4 Technical Report. *OpenAI Documentation*. https://openai.com/research/gpt-4
2. Google DeepMind (2024). Gemini 1.5 Technical Overview. https://deepmind.google/technologies/gemini/
3. Meta AI (2024). LLaMA 3 Model Card and Capabilities. https://www.llama.com/
4. MongoDB Inc. (2024). MongoDB Documentation: Mongoose ODM. https://mongoosejs.com/docs/
5. NestJS (2023). NestJS Framework Documentation. https://docs.nestjs.com/
6. Angular Team (2023). Angular 17+ Official Guide. https://angular.io/docs
7. Medium. https://medium.com
8. Flashcards World. https://flashcards.world/flashcards/sets/18989487-f9ee-4bab-937c-ba9fe1bc5ee9/
9. Liu, Y., & Lapata, M. (2020) / Text Summarization with Pretrained Encoders. *Transactions of the Association for Computational Linguistics (TACL)*, 8, 328–345. https://aclanthology.org/2020.tacl-1.17/
10. The Startup Owner's Manual. Wiley, (2020) / 9781119690726, https://www.perlego.com/paid/book/1425741/the-startup-owners-manual-the-stepby step-guide-for-building-a-great-company-pdf
11. Котлер Ф., Картаджая Г., Сетіаван І. Маркетинг 4.0: Від традиційного до цифрового. – КМ-БУКС, 2018. https://www.management.com.ua/books/view-books.php?id=2205
12. Speech and Language Processing / Dan Jurafsky, James H. Martin. — *Foundations of NLP, Sentence Ranking, TF-IDF, TextRank, Entropy*. https://web.stanford.edu/~jurafsky/slp3/ed3bookaug20_2024.pdf

13. Foundations of Statistical Natural Language Processing / Manning, Schütze. https://icog-labs.com/wp-content/uploads/2014/07/Christopher_D._Manning_Hinrich_Sch%C3%BCtze_Foundations_Of_Statistical_Natural_Language_Processing.pdf
14. A Method for Stochastic Optimization / Diederik P. Kingma, Jimmy Ba. https://www.intel.com/content/dam/www/public/us/en/ai/documents/1412.6980.pdf
15. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer / Colin Raffel et al. (2019). https://jmlr.org/papers/volume21/20-074/20-074.pdf

ДОДАТКИ

ДОДАТОК А

1. Код Front-end частини проекту:

Батьківський компонент:

```
import { Component } from '@angular/core';
import { TextSummaryComponent } from "./components/text-summary.component";

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [TextSummaryComponent],
  templateUrl: './app.component.html',
  styleUrl: './app.component.scss',
})
export class AppComponent {}
```

HTML розмітка:

```
<text-summary></text-summary>
```

1.2 Основний компонент з логікою:

```
 import {Component, inject, OnInit} from '@angular/core';
import {HttpClient, HttpClientModule} from "@angular/common/http";
import {TextFieldModule} from '@angular/cdk/text-field';
import {MatInputModule} from "@angular/material/input";
import {MatFormFieldModule} from "@angular/material/form-field";
import {MatButtonModule} from '@angular/material/button';
import {MatRadioModule} from '@angular/material/radio';
import {FormControl, ReactiveFormsModule} from "@angular/forms";
import {MatExpansionModule} from '@angular/material/expansion';
import {TextSummaryApiService} from "../services/text-summary-api.service";
import {MAT_RADIO_OPTIONS} from "../constants/mat-radio-options.constant";
import {MatCard, MatCardTitle} from "@angular/material/card";
import {MatProgressBar} from "@angular/material/progress-bar";
import {NgForOf, NgIf, SlicePipe} from "@angular/common";
import {IEvaluationScores} from "../interfaces/evaluation-scores.interface";
import {GptModel} from "../enums/gpt-model.enum";
import {ITextSummaryResponse} from "../interfaces/text-summary-response.interface";
import {MatListItem, MatNavList} from "@angular/material/list";
```

```
@Component({
  selector: 'text-summary',
  standalone: true,
  imports: [
    HttpClientModule,
    TextFieldModule,
    MatInputModule,
    MatFormFieldModule,
    MatButtonModule,
    MatRadioModule,
    ReactiveFormsModule,
    MatExpansionModule,
    MatCard,
    MatCardTitle,
    MatProgressBar,
    NgForOf,
    NgIf,
    MatNavList,
    MatListItem,
    SlicePipe,
  ],
  templateUrl: './text-summary.component.html',
  styleUrl: './text-summary.component.scss',
  providers: [HttpClient, TextSummaryApiService, MAT_RADIO_OPTIONS],
})
export class TextSummaryComponent implements OnInit {
  private readonly textSummaryApiService = inject(TextSummaryApiService);

  public readonly textControl = new FormControl<string>('');
  public readonly summarizedTextControl = new FormControl<string>('');
  public readonly gptModelControl = new FormControl<string>(GptModel.GPT_4_1_NANO);

  public summaryPanelOpenState = false;
  public textSummaryOpenState = true;
  public activeItem = {};
  public summaryHistory: ITextSummaryResponse[] | null = null;
  public evaluationScores: IEvaluationScores[] | null = null;

  protected readonly GptModel = GptModel;

  public ngOnInit(): void {
```

```
  this.setSummaryHistory();
}

public selectHistoryItem(item: ITextSummaryResponse): void {
  this.textControl.setValue(item.originalText);
  this.summarizedTextControl.setValue(item.processedText);
  this.evaluationScores = item.evaluations;
  this.openTextInputAccordeon()
}

public summarizeText() {
  const input = {
    text: this.textControl.value,
    gptModel: this.gptModelControl.value,
  };

  this.textSummaryApiService.summarize(input).subscribe(res => {
    this.summarizedTextControl.setValue(res.resultText);
    this.evaluateText(res.evaluation);
    this.summaryPanelOpenState = true;
    this.setSummaryHistory();
    this.openTextSummaryAccordeon();
  });
}

private openTextInputAccordeon(): void {
  this.textSummaryOpenState = true;
  this.summaryPanelOpenState = false;
}

private openTextSummaryAccordeon(): void {
  this.textSummaryOpenState = false;
  this.summaryPanelOpenState = true;
}

private setSummaryHistory(): void {
  this.textSummaryApiService.getSummaries().subscribe(result => {
    this.summaryHistory = result;
  });
}

private evaluateText(evaluatedText: string): void {
```

```
    try {
      this.evaluationScores = JSON.parse(evaluatedText);
    } catch (e) {
      console.error("Error occurred while parsing evaluated text", e);
    }
  }
}
```

Стилі:

```
.text-summary {
  display: flex;
  padding-top: 20px;

  &-container {
    display: flex;
    flex-direction: row;
    flex-grow: 0.3;
    margin: auto;

    &-evaluation-info {
      padding: 10px 0;

      &-card {
        width: 100%;
        max-width: 360px;
        padding: 16px;
      }

      &-score-row {
        margin-bottom: 18px;
      }

      &-label {
        display: flex;
        justify-content: space-between;
        font-weight: 600;
        margin-bottom: 4px;
      }

      &-score {
        font-size: 12px;
        opacity: 0.7;
```

```
  }

  &-comment {
    font-size: 12px;
    opacity: 0.75;
    margin-top: 4px;
  }
}

&-input {
  display: flex;
  flex-direction: column;
  flex-grow: 0.3;
  margin: auto;

  &-model-label {
    padding: 10px 0;
  }

  &-textarea {
    width: 100%;
  }
}

&-history-card {
  padding: 10px 15px;
  max-height: 50vh;
  overflow: auto;

  &-title {
    font-weight: 400;
  }

  &-meta {
    font-size: 12px;
    opacity: 0.6;
  }

  mat-list-item.active {
    background: rgba(98, 0, 238, 0.08);
    border-left: 3px solid #6200ee;
  }
```

```
  }
 }
}
```

HTML розмітка:

```
<div class="text-summary">
 <div class="text-summary-container">

  <div class="text-summary-container-evaluation-info">
   <mat-card *ngIf="evaluationScores" class="text-summary-container-evaluation-info-card">
    <mat-card-title>Summary Quality</mat-card-title>

    <div class="text-summary-container-evaluation-info-score-row" *ngFor="let item of evaluationScores">
     <div class="text-summary-container-evaluation-info-label">
      {{ item.label }}
      <span class="text-summary-container-evaluation-info-score">{{ item.value }}/100</span>
     </div>

     <mat-progress-bar
      mode="determinate"
      [value]="item.value">
     </mat-progress-bar>

     <div class="text-summary-container-evaluation-info-comment">
      {{ item.comment }}
     </div>
    </div>
   </mat-card>
  </div>

  <div class="text-summary-container-input">
   <mat-accordion>
    <mat-expansion-panel
     [expanded]="textSummaryOpenState"
    >
     <mat-expansion-panel-header>
      <mat-panel-title>
       Add text to summarize
      </mat-panel-title>

     </mat-expansion-panel-header>
```

```
          <mat-form-field appearance="outline" class="text-summary-container-input-textarea">
            <mat-label>Add text</mat-label>

            <textarea [formControl]="textControl" matInput rows="15"></textarea>
          </mat-form-field>
        </mat-expansion-panel>

        <mat-expansion-panel
          [expanded]="summaryPanelOpenState"
        >
          <mat-expansion-panel-header>
            <mat-panel-title>
              Summarized text
            </mat-panel-title>
          </mat-expansion-panel-header>

          <mat-form-field appearance="outline" class="text-summary-container-input-textarea">
            <mat-label>Result</mat-label>

            <textarea [formControl]="summarizedTextControl" matInput rows="15" readonly></textarea>
          </mat-form-field>
        </mat-expansion-panel>
      </mat-accordion>

      <div class="text-summary-container-input-model-label">
        <span>Select a gpt model:</span>

        <mat-radio-group [formControl]="gptModelControl" aria-label="Select a model">
                                    <mat-radio-button [checked]="'true'" value="{{ GptModel.GPT_4_1_NANO
}}">gpt-4.1-nano</mat-radio-button>
          <mat-radio-button value="{{ GptModel.GPT_4 }}">gpt-4</mat-radio-button>
          <mat-radio-button value="{{ GptModel.GPT_5_1 }}">gpt-5.1</mat-radio-button>
        </mat-radio-group>
      </div>

      <div>
        <button mat-flat-button color="primary" (click)="summarizeText()">Summarize Text</button>
      </div>
    </div>

    <div *ngIf="summaryHistory">
      <mat-card class="text-summary-container-history-card">
```

```
        <mat-card-title>History</mat-card-title>

        <mat-nav-list>
          <mat-list-item
            *ngFor="let item of summaryHistory; let i = index"
            (click)="selectHistoryItem(item)"
            [class.active]="item === activeItem">

            <div mat-line class="text-summary-container-history-card-title">
              {{ item.originalText | slice:0:40 }}...
            </div>

            <div mat-line class="text-summary-container-history-card-meta">
              {{ item.gptModel }}
            </div>

          </mat-list-item>
        </mat-nav-list>
      </mat-card>
    </div>
  </div>
</div>
```

Константи і додаткові типи:

```
import { MAT_RADIO_DEFAULT_OPTIONS } from "@angular/material/radio";

export const MAT_RADIO_OPTIONS = {
  provide: MAT_RADIO_DEFAULT_OPTIONS,
  useValue: { color: 'primary' },
};

export interface TextSummaryPayload {
  text: string | null,
  gptModel: string | null,
}

export interface TextSummaryPayload {
  text: string | null,
  gptModel: string | null,
}

export enum GptModel {
```

```
  GPT_4 = 'gpt-4',
  GPT_5_1 = 'gpt-5.1',
  GPT_4_1_NANO = 'gpt-4.1-nano',
}

export interface IEvaluationScores {
  label: string;
  value: number;
  comment: string;
}

import {IEvaluationScores} from "./evaluation-scores.interface";

export interface ITextSummaryResponse {
  originalText: string;
  processedText: string;
  gptModel: string;
  evaluations: IEvaluationScores[];
}
```

Сервіс для відправки запитів на сервер:

```
import { inject, Injectable } from "@angular/core";
import { HttpClient } from "@angular/common/http";
import { Observable } from "rxjs";
import { environment as env } from "../../environments/environment";
import {TextSummaryPayload} from "../dtos/text-summary-payload.dto";

@Injectable()
export class TextSummaryApiService {
  private readonly httpClient = inject(HttpClient);

  public summarize(input: TextSummaryPayload): Observable<any> {
    const url = `${env.openAiApiWrapperUri}/summarize`;
    const body = {
      text: input.text,
      gptModel: input.gptModel,
    };

    return this.httpClient.post(url, body);
  }
}
```

2. Серверна частина:

Основний клас:

```
import { Module } from '@nestjs/common';
import { OpenAiApiWrapperComponent } from './openAiApiWrapperComponent';
import { OpenAiApiService } from './services/open-ai-api.service';
import { MongooseModule } from '@nestjs/mongoose';
import { TextSummary, TextSummarySchema } from './schemas/text-summary.schema';
import { TextSummaryService } from './services/text-summary.service';
import { TextSummaryRepository } from './repositories/text-summary.repository';

@Module({
 imports: [
  MongooseModule.forRoot('<mongodb_uri>'),
  MongooseModule.forFeature([{ name: TextSummary.name, schema: TextSummarySchema}]),
 ],
 controllers: [OpenAiApiWrapperComponent],
 providers: [OpenAiApiService, TextSummaryService, TextSummaryRepository],
})
export class AppModule {}
```

Контроллер:

```
@Controller()
export class OpenAiApiWrapperComponent {
 constructor(private readonly openAiService: OpenAiApiService) {}

 @Post('summarize')
 public summarizeText(@Body() body: ITextSummaryPayload): Promise<IResult> {
  return this.openAiService.summarizeText(body);
 }
}
```

Сервіс для відправки запитів на Open AI

```
import OpenAi from "openai";
import { ITextSummaryPayload } from '../dtos/input.dto';
import { IResult } from '../dtos/result.dto';
import { TextSummaryService } from './text-summary.service';
import { Injectable } from '@nestjs/common';

@Injectable()
export class OpenAiApiService {
 private readonly openApiClient = new OpenAi();
```

```
  constructor(private readonly textSummaryService: TextSummaryService) {}

  public async summarizeText(payload: ITextSummaryPayload): Promise<IResult> {
    try {
      const response = await this.openApiClient.responses.create({
        model: payload.gptModel,
        input: payload.text,
      });

      await this.textSummaryService.createTextSummary({
        originalText: payload.text,
        processedText: response.output_text,
        gptModel: payload.gptModel,
      });

      return { resultText: response.output_text };
    } catch (error) {
      throw error;
    }
  }
}
```

Сервіс для запису даних в БД для подальшого аналізу:

```
import { TextSummaryRepository } from '../repositories/text-summary.repository';
import { ITextSummary } from '../dtos/text-summary.dto';
import { Injectable } from '@nestjs/common';
import { TextSummary } from '../schemas/text-summary.schema';

@Injectable()
export class TextSummaryService {
  constructor(private readonly textSummaryRepository: TextSummaryRepository) {};

  public async create(textSummaryDto: ITextSummary): Promise<void> {
    await this.textSummaryRepository.create(textSummaryDto)
  }

  public async getLastHundred(): Promise<TextSummary[]> {
    return this.textSummaryRepository.getLastHundred();
  }
}
```

Репозиторій для зв’язку з БД:

```
import { Injectable } from '@nestjs/common';
import { InjectModel } from '@nestjs/mongoose';
import { TextSummary } from '../schemas/text-summary.schema';
import { Model } from 'mongoose';
import { ITextSummary } from '../dtos/text-summary.dto';

@Injectable()
export class TextSummaryRepository {
 constructor(
  @InjectModel(TextSummary.name) private textSummaryModel: Model<TextSummary>
 ) {}

 public async create(textSummaryDto: ITextSummary): Promise<void> {
  await this.textSummaryModel.create(textSummaryDto);
 }

 public async getLastHundred(): Promise<TextSummary[]> {
  return this.textSummaryModel
   .find()
   .sort({ _id: -1 })
   .limit(100)
   .lean<TextSummary[]>();
 }
}
```

Допоміжні типи для обміну даними між класами:

```
export interface ITextSummary {
 originalText: string;
 processedText: string;
 gptModel: string;
}

export interface IResult {
 resultText: string;
}

export interface ITextSummaryPayload {
 text: string
 gptModel: string
}
```

```
export interface IEvaluation {
  label: string;
  value: number;
  comment: string;
}
```

БД схема:

```
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { HydratedDocument } from 'mongoose';

export type TextSummaryDocument = HydratedDocument<TextSummary>;

@Schema({ _id: false })
export class Evaluation {
  @Prop({ required: true })
  label: string;

  @Prop({ required: true, min: 0, max: 100 })
  value: number;

  @Prop()
  comment: string;
}

export const EvaluationSchema = SchemaFactory.createForClass(Evaluation);

@Schema()
export class TextSummary {
  @Prop()
  originalText: string;

  @Prop()
  processedText: string;

  @Prop()
  gptModel: string;

  @Prop({ type: [EvaluationSchema], default: [] })
  evaluations: Evaluation[];
}

export const TextSummarySchema = SchemaFactory.createForClass(TextSummary);
```