МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

ЗВІТ З ПЕРЕДДИПЛОМНОЇ ПРАКТИКИ

Виконала студентка групи ТВ-з11 ШАРАБУРА Еліна Дмитрівна _____
(шифр групи, прізвище, ім'я, по батькові) (підпис)

Керівник практики від НТУУ «КПІ ім.Ігоря Сікорського» МУСІЄНКО Андрій Петрович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник практики від підприємства

ТОВ «СЕО Студіо» ЦУД Віталій Васильович _____
(назва підприємства) (прізвище, ім'я, по батькові) (підпис)

Київ – 2025

ПЛАН ПРАКТИКИ

(14 квітня – 18 травня 2025 року)
студентки 4 курсу групи ТВ-з11

Шарабура Еліна Дмитрівна
Тема практики «Комплексний аналіз настроїв тексту за допомогою штучного інтелекту»

Зміст	Термін виконання
1. Початок проходження практики	14.04.25
2. Вступна бесіда з керівником практики від бази практики і керівником дипломної роботи	14.04.25
3. Виконання завдання переддипломної практики. Підготовка матеріалів за темою дипломної роботи (зі щотижневою перевіркою)	Протягом усієї практики
3.1.Опрацювання літератури за темою дипломної роботи, а саме: дослідження наукових і технічних джерел з методів аналізу тексту, обробки природної мови (NLP) та використання штучного інтелекту в текстовій аналітиці. Розробка алгоритму програмної системи Консультація і звіт керівникові про виконану роботу (понеділок, п’ятниця)	1-й тиждень 14.04 - 20.04.25
3.2. Розробка алгоритму програмної системи Написання коду програмної системи – розробка архітектури програмної системи, визначення принципів її функціонування та взаємодії основних компонентів, а також розробка алгоритмічних рішень для обробки та аналізу даних. Консультація і звіт керівникові про виконану роботу (понеділок, п’ятниця)	2-й тиждень 21.04 - 27.04.25
3.3. Написання коду програмної системи — реалізація базових модулів програмної системи, включаючи обробку вхідних текстових даних та їх аналіз, розробку основних алгоритмів NLP-аналізу та створення структури бази даних. Консультація і звіт керівникові про виконану роботу (понеділок, п’ятниця)	3-й тиждень 28.04 - 04.05.25
3.3. Написання коду програмної системи — інтеграція штучного інтелекту, розробка серверної частини для забезпечення взаємодії між модулями та створення веб-інтерфейсу для взаємодії з користувачем. Консультація і звіт керівникові про виконану роботу (понеділок, п’ятниця)	4-й тиждень 05.05 - 11.05.25
3.4. Комплексне тестування і налагодження програмної системи, виявлення та виправлення помилок, оптимізація продуктивності, перевірка стабільності та безпеки роботи системи Випробування системи на комп'ютері керівника від кафедри Захист на кафедрі програмного продукту Консультація і звіт керівникові про виконану роботу (понеділок, п’ятниця)	5-й тиждень 12.05 - 18.05.25
4. Оформлення щоденника і звіту з переддипломної практики	14.05 - 17.05.25
5. Підготовка і складання заліку з практики	18.05.25

ЗМІСТ

ВСТУП

У інформаційному середовищі зростає потреба в інструментах, які здатні швидко та ефективно аналізувати великі обсяги текстових даних. Соціальні мережі, новинні портали та блоги генерують щоденно сотні тисяч повідомлень, серед яких важливо вчасно виявляти негатив, токсичність або певну тематику. Саме тому розробка систем для автоматизованого аналізу тексту із використанням методів штучного інтелекту є надзвичайно актуальною.

Ця робота присвячена створенню вебсервісу, який дозволяє аналізувати настрій текстової інформації за кількома параметрами: визначення тональності, токсичності, тематичної приналежності, а також автоматичне узагальнення змісту тексту. У системі реалізовано як базовий аналіз без авторизації, так і розширений функціонал для зареєстрованих користувачів, включно з історією аналізів, обробкою файлів та аудіо та можливістю перегенерувати текст з іншим рівнем токсичності.

Метою проєкту є створення зручного, інтуїтивного та технічно якісного вебінструменту, який поєднує сучасні технології штучного інтелекту, зокрема трансформерні моделі, з професійним інтерфейсом користувача.

До основних завдань належать:

- аналіз існуючих рішень та вибір відповідної архітектури системи;
- реалізація серверної частини на Flask із підключенням попередньо завантажених моделей;
- створення адаптивного вебінтерфейсу та інтерактивними елементами;
- забезпечення функціональності збереження результатів, завантаження файлів і трансформації тексту за допомогою OpenAI API.

У результаті реалізовано повнофункціональну інтелектуальну програмну систему, яка демонструє можливості глибокого аналізу текстових даних та підходи до інтеграції штучного інтелекту в реальні вебпродукти.

1 ЗАГАЛЬНА ХАРАКТЕРИСТИКА ПРОБЛЕМИ АНАЛІЗУ ТЕКСТУ

1.1 Поняття та значення аналізу тексту

Аналізом тексту називають процес автоматизованої обробки природномовних даних з метою виявлення семантичної, емоційної, тематичної чи структурної інформації. У контексті сучасних інформаційних систем він широко застосовується в сфері медіа-моніторингу, клієнтської аналітики, контент-модерації та машинного перекладу. Широке розповсюдження соціальних мереж і цифрового контенту зумовлює необхідність точних, масштабованих інструментів для автоматизованого аналізу великих обсягів текстових даних.

1.2 Визначення емоційності, тематики та токсичності тексту

- Емоційність або тональність — класифікація тексту за настроєм: позитивний, негативний або нейтральний. Використовується для аналізу відгуків, коментарів, звернень клієнтів тощо.
- Тематика — визначення предметного напрямку тексту (наприклад: бізнес, політика, спорт і т.д.). Тема дозволяє структурувати велику кількість текстових матеріалів і здійснювати фільтрацію за змістом.
- Токсичність — виявлення у тексті образливих, агресивних або дискримінаційних висловлювань. Це важливий аспект для модерації контенту в онлайн-платформах та месенджерах.

Для автоматичного визначення зазначених характеристик використовуються попередньо навчені трансформерні моделі, які завдяки своїй архітектурі здатні аналізувати семантику та контекст у межах речень або абзаців.

1.3 Завдання та цілі вебсервісу

Метою розробки інтелектуальної програмної системи є створення інструменту для здійснення багаторівневого аналізу настроїв текстової інформації із використанням моделей штучного інтелекту. Система повинна забезпечувати швидку, точну та інформативну обробку текстових даних у зручному для користувача форматі.

Основні завдання вебсервісу:

- класифікація тексту за емоційним забарвленням (тональністю): позитивна, негативна або нейтральна;
- визначення тематичної належності тексту з-поміж кількох основних категорій;
- виявлення токсичності у контексті, тобто образливих, агресивних або соціально неприйнятних висловлювань;
- генерація резюме (узагальнення змісту та витяг основної думки) особливо актуально для довгих текстів;
- переформулювання текстів на основі заданого рівня токсичності (можливість зменшення або підвищення);
- підтримка мультимовності шляхом використання моделей, які були навчені на багатомовних корпусах тексту;
- реалізація двох режимів використання: базового (без авторизації) і повного (після реєстрації або входу в систему);
- можливість аналізу файлів (.txt, .json, .csv, .pdf, .docx, .mp3, .wav, .m4a).
- збереження історії аналізів в базу даних з можливістю фільтрації та перегляду результатів, а також видалення результатів.

Таким чином, вебсервіс забезпечує гнучкий підхід до обробки природної мови та демонструє практичне застосування сучасних моделей трансформерної архітектури для вирішення реальних задач.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ТЕХНОЛОГІЙ

2.1 Моделі на базі BERT та XLM-RoBERTa

BERT (Bidirectional Encoder Representations from Transformers) — це мовна модель від Google, яка стала проривом у сфері обробки природної мови. Моделі типу BERT працюють на основі трансформерної архітектури, що використовує механізм самоуваги (self-attention). Це дозволяє моделі одночасно враховувати всі слова в реченні та встановлювати між ними зв’язки, незалежно від їхнього розташування. У процесі навчання модель передбачає пропущені слова у фрагментах тексту (masked language modeling), завдяки чому вона навчається розуміти контекст.

На вхід модель отримує токенізований текст, де кожне слово або частина слова перетворюється на числовий вектор (ембедінг). Далі ці вектори проходять кілька шарів обробки, де відбувається зважування важливості кожного слова щодо інших. Результатом є векторне представлення, яке можна використовувати для класифікації, генерації чи інших NLP-завдань.

XLM-RoBERTa — багатомовне розширення RoBERTa, яке підтримує понад 100 мов. Її основна перевага — здатність працювати з різними мовами без потреби додаткового донавчання або перекладу, що особливо важливо для мультимовних систем.

Обидві моделі реалізовані у вигляді готових пайплайнів у бібліотеці Hugging Face Transformers, що суттєво спрощує їх інтеграцію у програмні продукти.

2.2 Технології класифікації тексту

Класифікація тексту — це задача віднесення текстового фрагменту до певної категорії на основі його вмісту. Для цього використовуються такі підходи:

- Rule-based (на основі правил): працює з шаблонами, регулярними виразами, словниками. Не гнучкий, складний в підтримці.

- Класичні ML-алгоритми: Naive Bayes, SVM, Decision Trees — потребують ручної інженерії ознак (TF-IDF, Bag-of-Words), менш точні на довгих і складних текстах.
- Нейронні мережі та трансформери: сучасні методи, які автоматично враховують контекст, граматику, порядок слів. Найкращі результати демонструють моделі на базі BERT, XLM-R, DistilBERT тощо.

У проєкті використано саме трансформерні моделі, оскільки вони забезпечують високу точність, працюють із текстами будь-якої довжини й не потребують ручної побудови ознак.

2.3 Порівняльний аналіз рішень та вибір підходу

Для задач класифікації тексту (зокрема визначення тональності, токсичності та тематики) існує кілька категорій моделей, що відрізняються за принципом роботи, точністю, вимогами до ресурсів та гнучкістю використання.

1. Правилозалежні (rule-based) підходи

Це найпростіші системи, які працюють на основі заздалегідь визначених правил, шаблонів або словників. Наприклад, фіксуються ключові слова, яким приписується певна категорія. Такі методи легко реалізувати, але вони не враховують контекст і дуже чутливі до формулювань. Точність у реальних умовах зазвичай низька і недостатня для інтеграції в інтелектуальну систему для отримання професійного результату.

2. Класичні моделі машинного навчання

Сюди належать моделі, які працюють із заздалегідь підготовленими числовими ознаками тексту (наприклад, TF-IDF, n-грами) і використовують алгоритми класифікації — такі як Naive Bayes, SVM або Logistic Regression. Вони здатні до певної міри аналізувати текст, проте не враховують складну структуру мови, порядок слів і контекст. Такі моделі є легкими й швидкими, але менш ефективними на неструктурованих або довгих текстах.

3. Глибокі нейронні мережі на основі рекурентних архітектур

До цього підходу належать моделі типу LSTM і GRU, які були популярними до появи трансформерів. Вони краще враховують послідовність слів і контекст, але мають обмеження в обробці довгих послідовностей та паралельності обчислень.

4. Трансформерні моделі нового покоління

Найбільш потужним і актуальним підходом є трансформерні моделі, зокрема BERT, RoBERTa, DistilBERT, ALBERT, XLM-RoBERTa та інші. Їх особливістю є використання механізму самоуваги (self-attention), як вже було зазначено, який дозволяє моделі аналізувати весь контекст речення одночасно, незалежно від довжини тексту. Це дає високу точність і гнучкість у розумінні мови.

Деякі з них:

- BERT — базова англомовна модель від Google, добре підходить для задач класифікації, але не підтримує мультимовність.
- RoBERTa — покращена версія BERT із розширеним обсягом навчання, вища точність.
- DistilBERT — полегшений варіант BERT, швидший, але менш точний.
- Multilingual BERT (mBERT) — варіант із підтримкою кількох десятків мов, але менш точний за XLM-R.
- XLM-RoBERTa — потужна багатомовна модель, навчена на великих корпусах понад 100 мов; демонструє високу точність і стабільність у задачах класифікації.

Зважаючи на вимоги до високої точності, підтримки кількох мов, можливості легкої інтеграції та відсутності потреби в донавчанні, для реалізації інтелектуальної системи було обрано модель XLM-RoBERTa. Вона забезпечує поєднання якості, продуктивності та масштабованості, що робить її оптимальним рішенням для багатомовного аналізу текстів.

3 СЕРЕДОВИЩЕ ТА ЗАСОБИ РОЗРОБКИ

3.1 Мова програмування Python

За основу було взято мовою розробки серверної частини вебсервісу обрано Python. Вона є однією з найпопулярніших мов у сфері штучного інтелекту та обробки природної мови завдяки своїй читабельності, великій кількості бібліотек (NumPy, Pandas, Transformers, Flask) та активній спільноті розробників. Python також забезпечує зручну інтеграцію з моделями машинного навчання та вебфреймворками.

3.2 Фреймворк Flask

Для реалізації API та серверної логіки використовується Flask — фреймворк для створення вебзастосунків на Python. Його переваги — легкість, мінімалізм, простота розгортання, підтримка REST-архітектури та велика кількість розширень. Flask дозволяє швидко створити надійний сервер для обробки запитів і взаємодії з модельними модулями.

3.3 Інтеграція моделей Hugging Face Transformers

Для виконання аналізу тексту в системі інтегровано попередньо навчені моделі з репозиторію Hugging Face Transformers. Ця бібліотека надає зручні інтерфейси для роботи з сучасними трансформерними моделями (BERT, RoBERTa, XLM-R тощо). З її допомогою були реалізовані функції класифікації тональності, тематики, токсичності, а також узагальнення й переформулювання текстів. Моделі використовуються у вигляді готових об’єктів з інтерфейсом pipeline, що спрощує їх інтеграцію та виклик.

3.4 Використання Firebase для автентифікації та бази даних

Для зберігання історії аналізів користувачів і реалізації системи автентифікації було використано платформу Firebase від Google. Вона забезпечує

просту інтеграцію в вебпроєкти, підтримує реєстрацію, вхід і аутентифікацію користувачів через email або соціальні мережі, а також надає Firestore — хмарну NoSQL-базу даних для зберігання текстів, тегів, дат, типів аналізу тощо.

3.5 Розробка фронтенду: HTML, CSS, JavaScript

Клієнтська частина вебзастосунку реалізована з використанням HTML, CSS та JavaScript. Інтерфейс адаптивний і побудований у мінімалістичному стилі з фокусом на зручність користування. CSS використовується для стилізації, а JavaScript — для обробки подій, виклику API, динамічного оновлення контенту та візуалізації результатів. У проєкті реалізовано кілька сторінок: базовий аналіз без входу, авторизація, розширений аналіз, історія запитів із фільтрацією та модальним переглядом.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Архітектура застосунку

Застосунок побудований за клієнт-серверною архітектурою. Користувач взаємодіє з фронтендом через браузер, який надсилає запити до API, реалізованого на Flask. Серверна частина обробляє вхідний текст, викликає відповідні моделі, агрегує результати за умовою якщо вхідний текст більше за 480 токенів (тоді працює обрізач тексту) та повертає їх у форматі JSON. Для авторизованих користувачів дані зберігаються у Firestore.

Архітектура підтримує масштабування, поділ на логічні модулі (бекенд, фронтенд, моделі, база даних) та дозволяє легко додавати нові функції.

4.2 Реалізація бекенду

Бекенд реалізовано на Python 3.11 з використанням фреймворку Flask. Основні модулі зображені на рисунку 4.1:

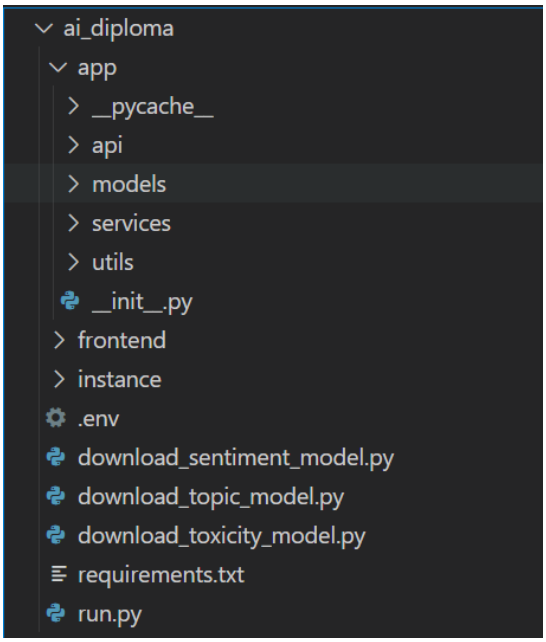

Рисунок 4.1 – Основні модулі

– api/ — реалізація REST-ендпоінтів (/analyze, /analyze_file_full, /analyze_file, /analyze_base, /summary, /regenerate_text, /history і т.д.);

– services/ — логіка виклику моделей, обробки тексту, форматування відповіді;
– utils/ — допоміжні функції (визначення мови, очищення тексту, агрегація);
– models/ — ініціалізація моделей (тональність, токсичність, тематика).
– Передбачена підтримка асинхронного виклику моделей, кешування відповідей та логування помилок.

4.3 Реалізація фронтенду

Фронтенд реалізовано на HTML, CSS і JavaScript. Розроблено п’ять основних сторінок які зображено на рисунку 4.2:

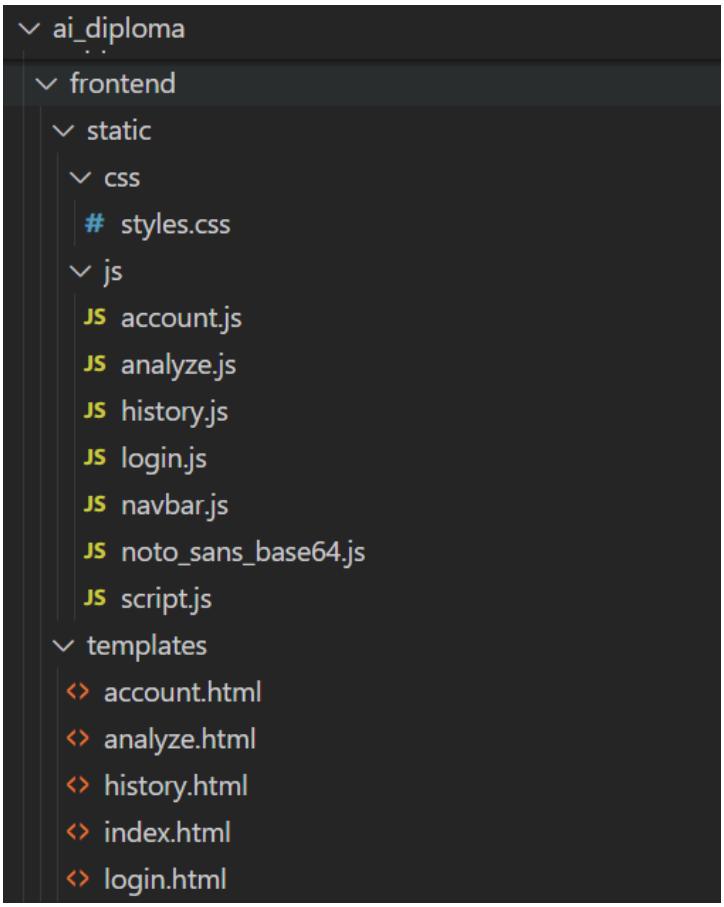

Рисунок 4.2 – Структура файлів фронтенду

– index.html — базовий аналіз (тональність, токсичність) без входу;
– login.html — автентифікація через Firebase;

– analyze.html — повний аналіз із виведенням графіків, резюме, результатів моделей;
– history.html — історія аналізів з фільтрацією, переглядом карток і модальних вікон.
– account.html – перегляд інформації про акаунт користувача, можливість видалити акаунт, або історії аналізів тексту.

Сторінки адаптивні, оформлені у сучасному стилі з плавними анімаціями та підтримкою інтерактивних елементів.

4.4 Інтеграція моделей та API

Інтеграція здійснюється через бібліотеку Hugging Face Transformers. Моделі завантажені локально, що зображено на рисунку 4.3 (для офлайн-роботи) та ініціалізуються при старті сервера. Використовуються:

- XLM-RoBERTa для аналізу тональності, тематики, токсичності;
- OpenAI API для генерації резюме та переформулювання тексту.

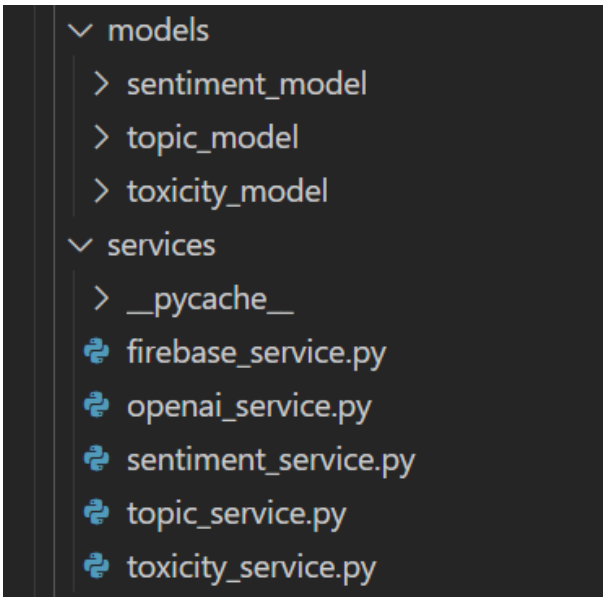

Рисунок. 4.3

Запити обробляються відповідними модулями, результат проходить постобробку та повертається у зручному форматі.

4.5 Обробка тексту: Chunking та агрегація результатів

Оскільки більшість трансформерних моделей мають обмеження на максимальну довжину вхідного тексту (наприклад, 512 токенів для XLM-RoBERTa), у системі реалізовано механізм попередньої перевірки довжини тексту. Якщо вхідний текст містить понад 480 токенів, він автоматично передається на обробку через спеціальний модуль (рис.4.4) для обрізання тексту.

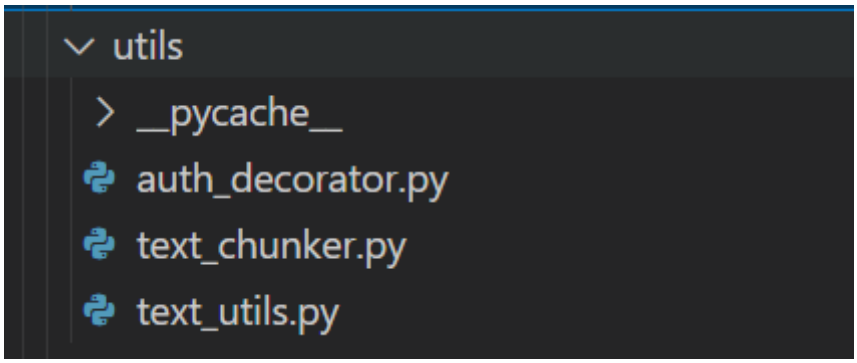

Рисунок 4.4 – модуль обрізання тексту text_chunker

Цей модуль розбиває текст на частини довжиною до 480 токенів, кожна з яких аналізується окремо за всіма доступними моделями. Після цього результати агрегуються відповідно до типу аналізу:

- тональність — визначається переважаючий клас (позитивна, негативна, нейтральна);
- токсичність — обчислюється середнє або максимальне значення;
- тематика — обираються 2–3 найчастіше згадані теми.

Такий підхід дозволяє обробляти тексти довільної довжини без втрати точності та без ризику помилки, пов'язаної з перевищенням обмеження на вхід моделі.

4.6 Генерація резюме та переформулювання тексту

Система підтримує:

1. Генерацію резюме (summary) на основі моделі OpenAI GPT, що стискає довгі тексти до короткого опису;
2. Переформулювання текстів із метою зниження або підвищення токсичності — реалізовано через запит до OpenAI з відповідною інструкцією (prompt engineering).

Функції інтегровані у фронтенд як окремі кнопки з відповідним виводом результату.

5 ІНТЕРФЕЙС КОРИСТУВАЧА ТА РОБОТА ІЗ СИСТЕМОЮ

5.1 Базовий аналіз без авторизації

На головній сторінці (index.html) доступна функція швидкого аналізу (рис. 5.1) без реєстрації. Користувач може вставити або ввести текст, після чого отримає результат аналізу тональності та токсичності.

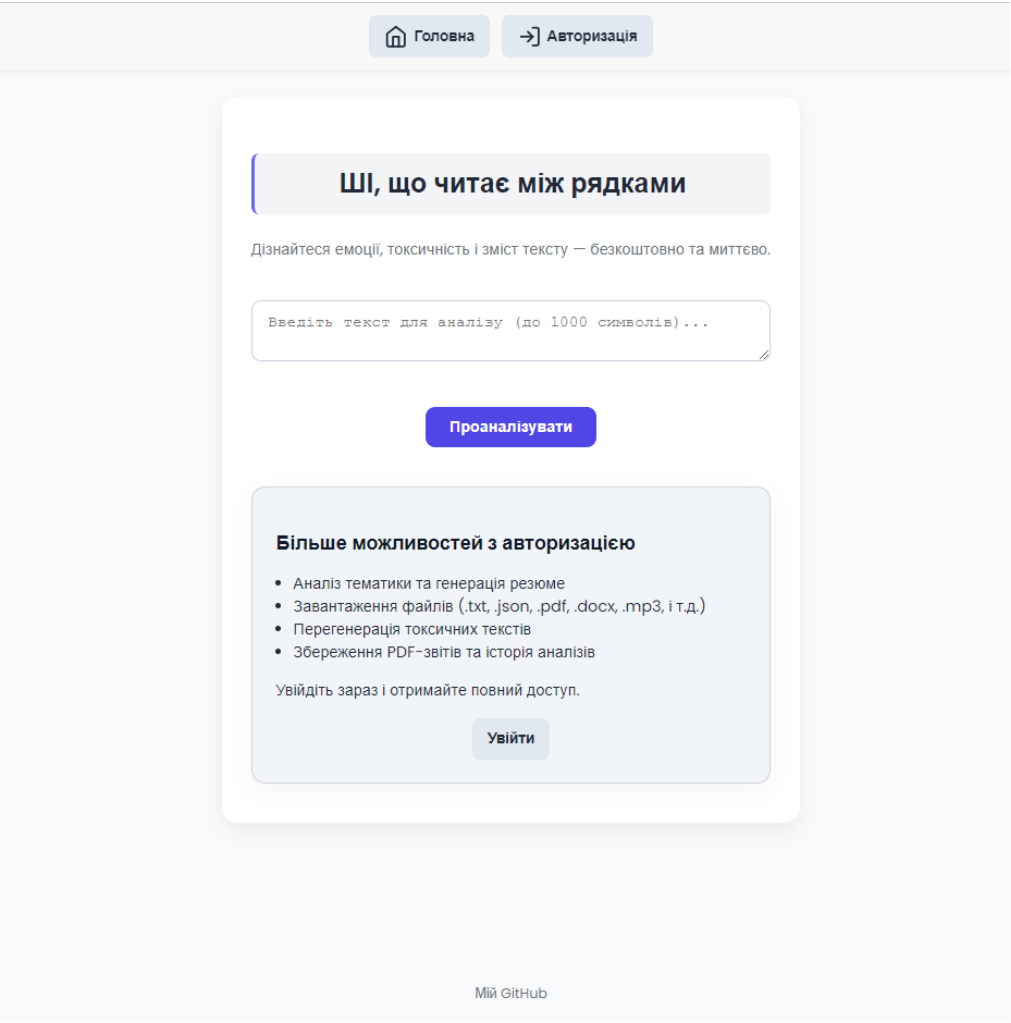

Рисунок 5.1 – Головна сторінка без реєстрації

Якщо виявлено високий рівень токсичності, система виводить попередження (alert). Проте рівень токсичності не вказується у відсотках на відміну від повної версії. У цьому режимі результати не зберігаються, і функціонал обмежений для збереження приватності та простоти використання.

5.2 Повний функціонал після авторизації

Після авторизації через Firebase, яка реалізована на сторінці login.html (рис. 5.2) користувач отримує доступ до повного функціоналу вебсервісу.

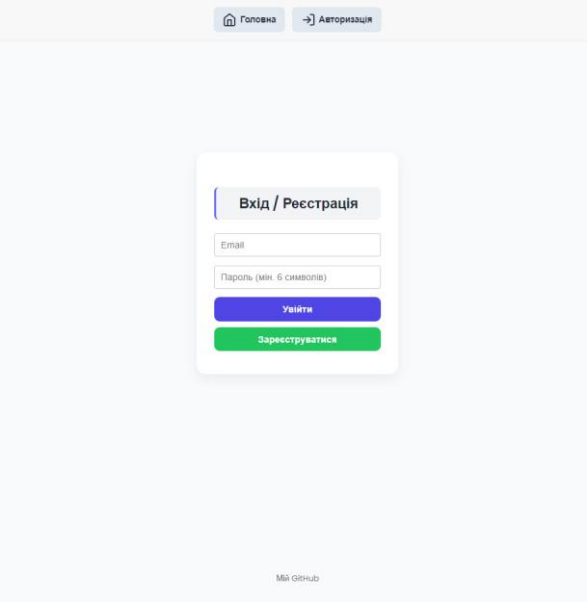

Рисунок 5.2 – Екран авторизації

Три основні вкладки повного функціоналу:

analyze.html — Аналіз тексту (рис. 5.3).

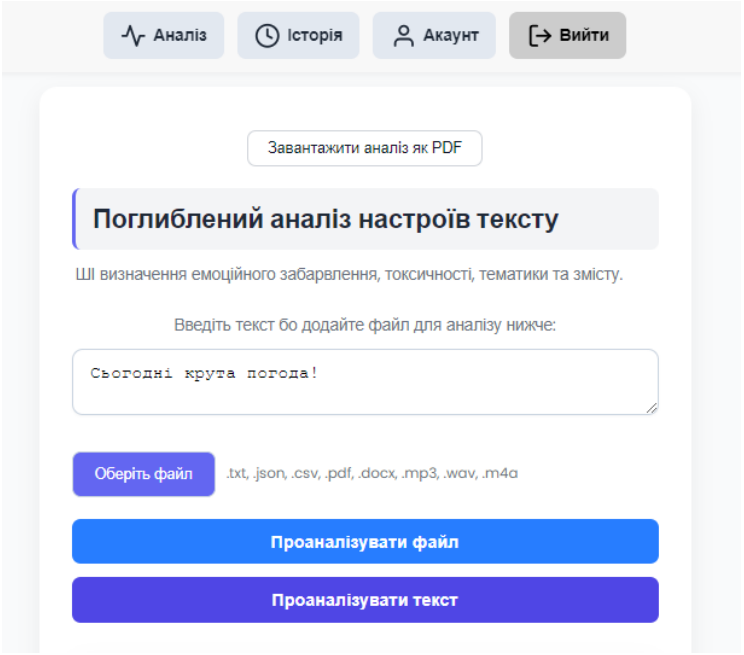

Рисунок 5.3 – Сторінка аналізу тексту повного функціоналу

Це головна вкладка після входу. Вона надає доступ до:

– повного аналізу тексту за тональністю (5.5.), токсичністю (рис. 5.6) та тематикою (рис. 5.4);

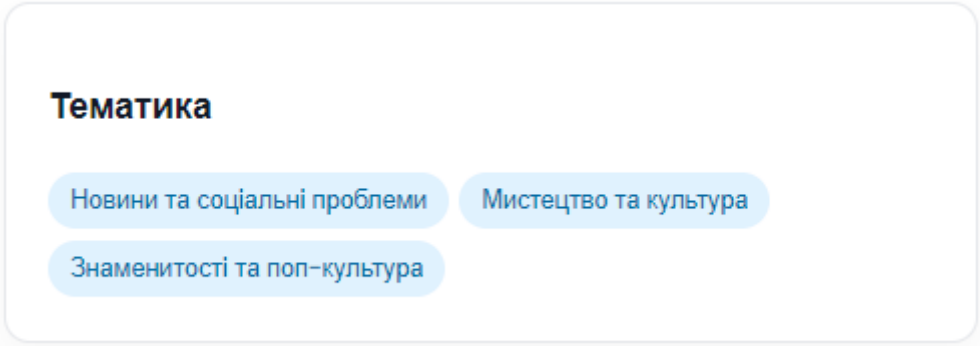

Рисунок 5. 4 – Результат визначення токсичності

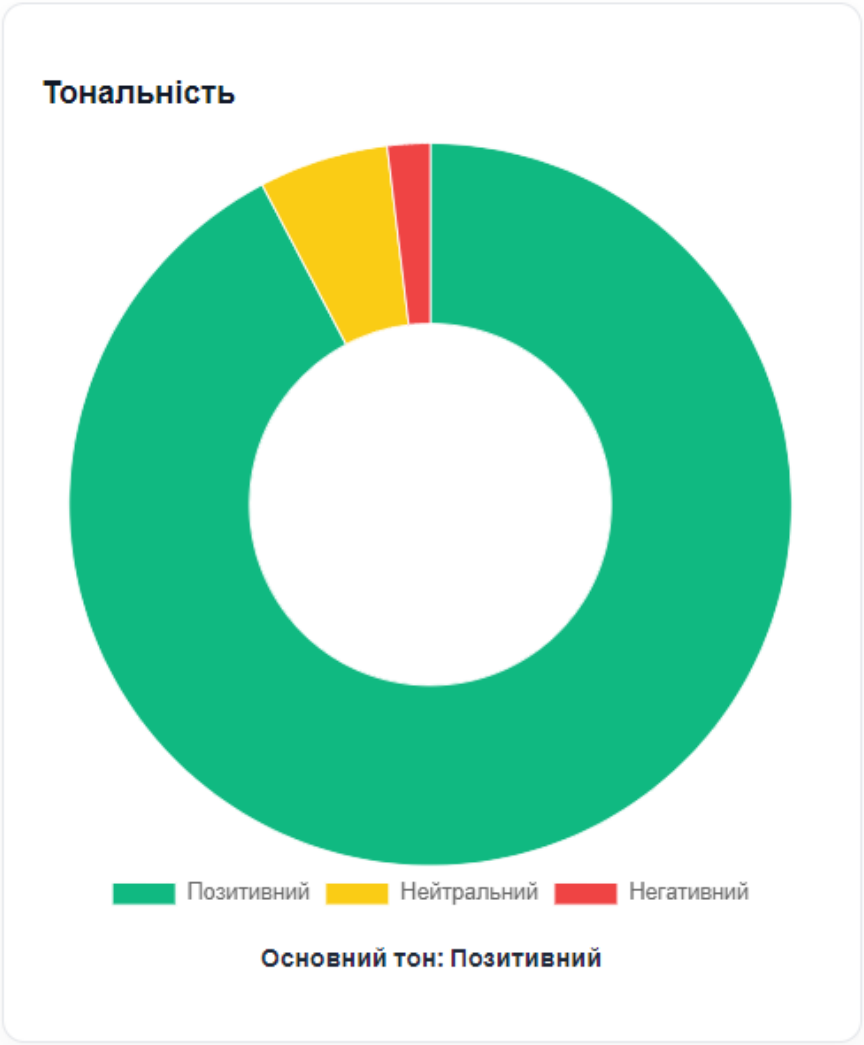

Рисунок 5.5 – Діаграма тональності

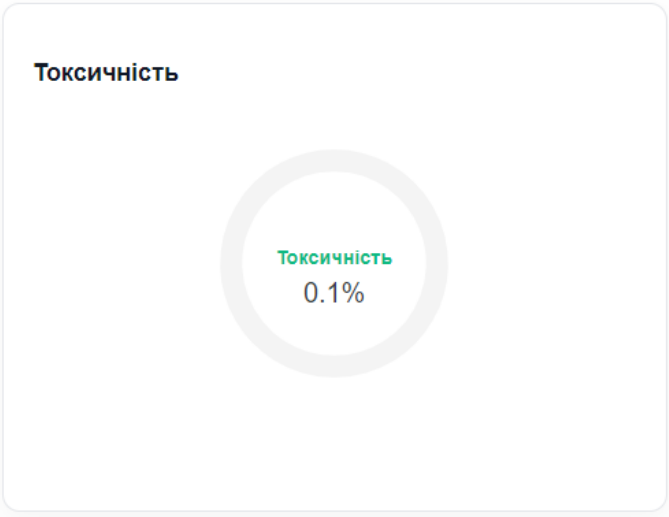

Рисунок 5.6 – Результат аналізу токсичності

– генерації короткого резюме (рис. 5.7) для тексту;

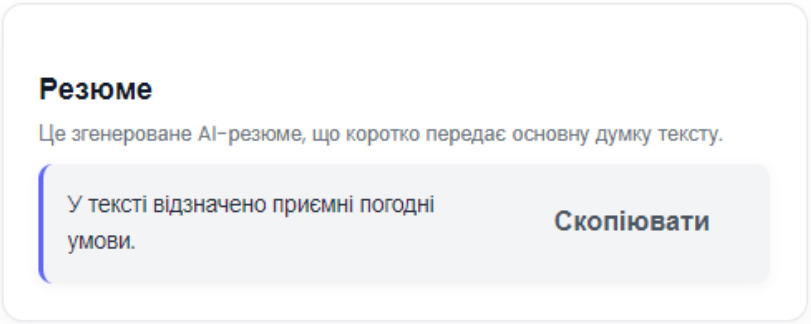

Рисунок 5.7 – Згенероване резюме тексту

– переформулювання тексту (рис. 5.8) з урахуванням токсичності (зменшення або підвищення);

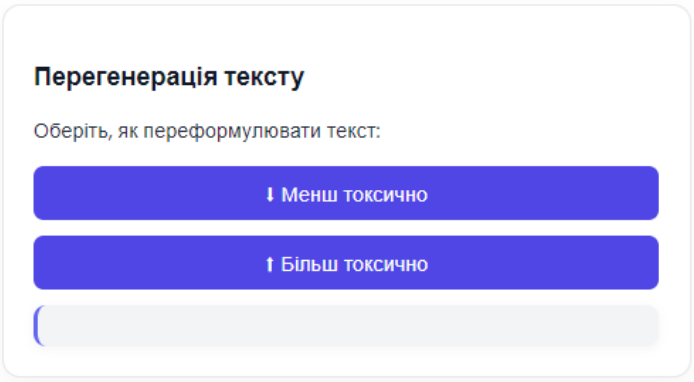

Рисунок 5.8 – Модуль перегенерації тексту

– візуалізацій результатів у вигляді графіків, кольорових індикаторів, тегів тощо;
– збереження результатів в особисту історію користувача (рис. 5.9).

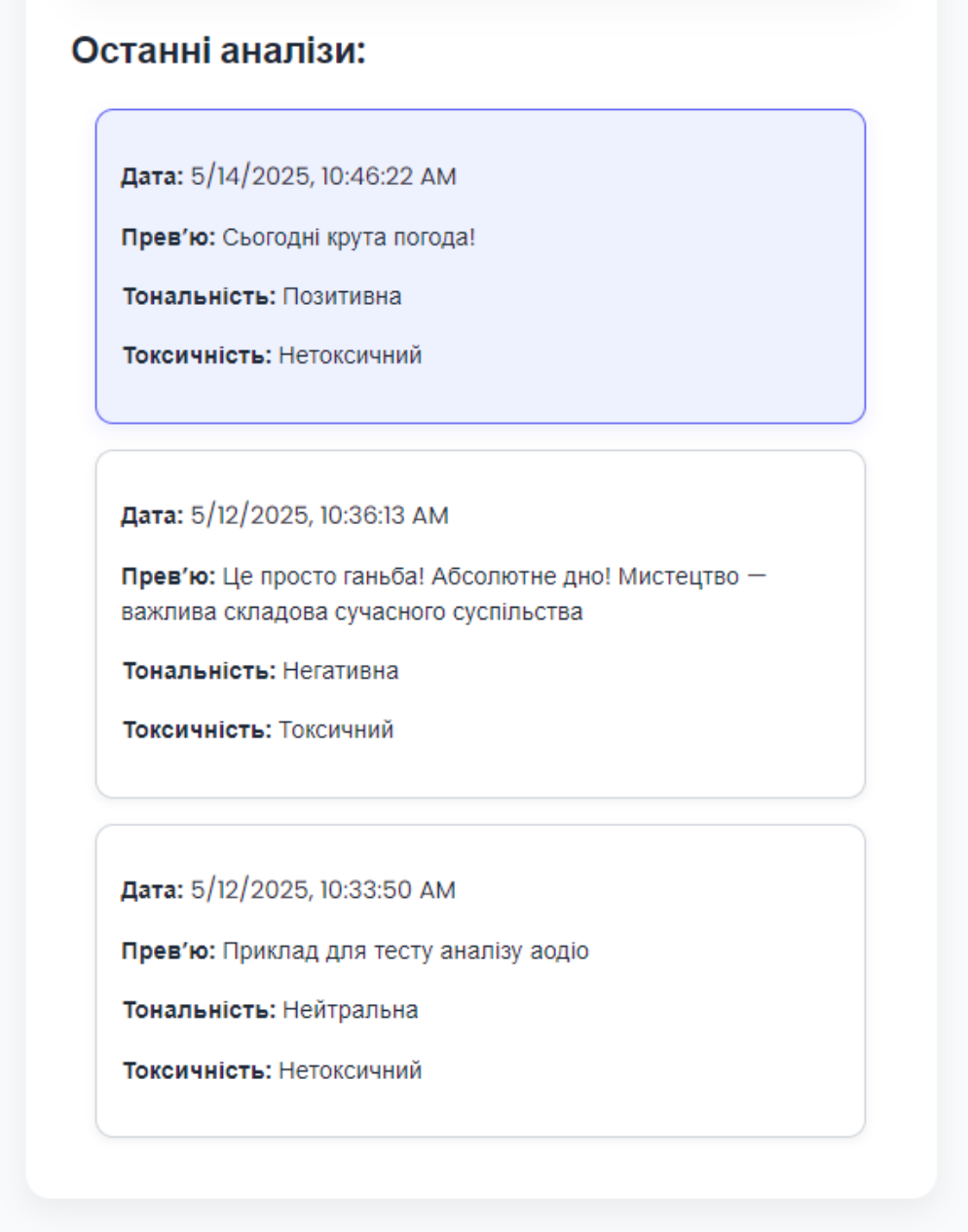

Рисунок 5. 9 – Останні результати аналізів

Інтерфейс передбачає зручне введення тексту або завантаження файлів для аналізу.

history.html — Історія аналізів (рис. 5.10)

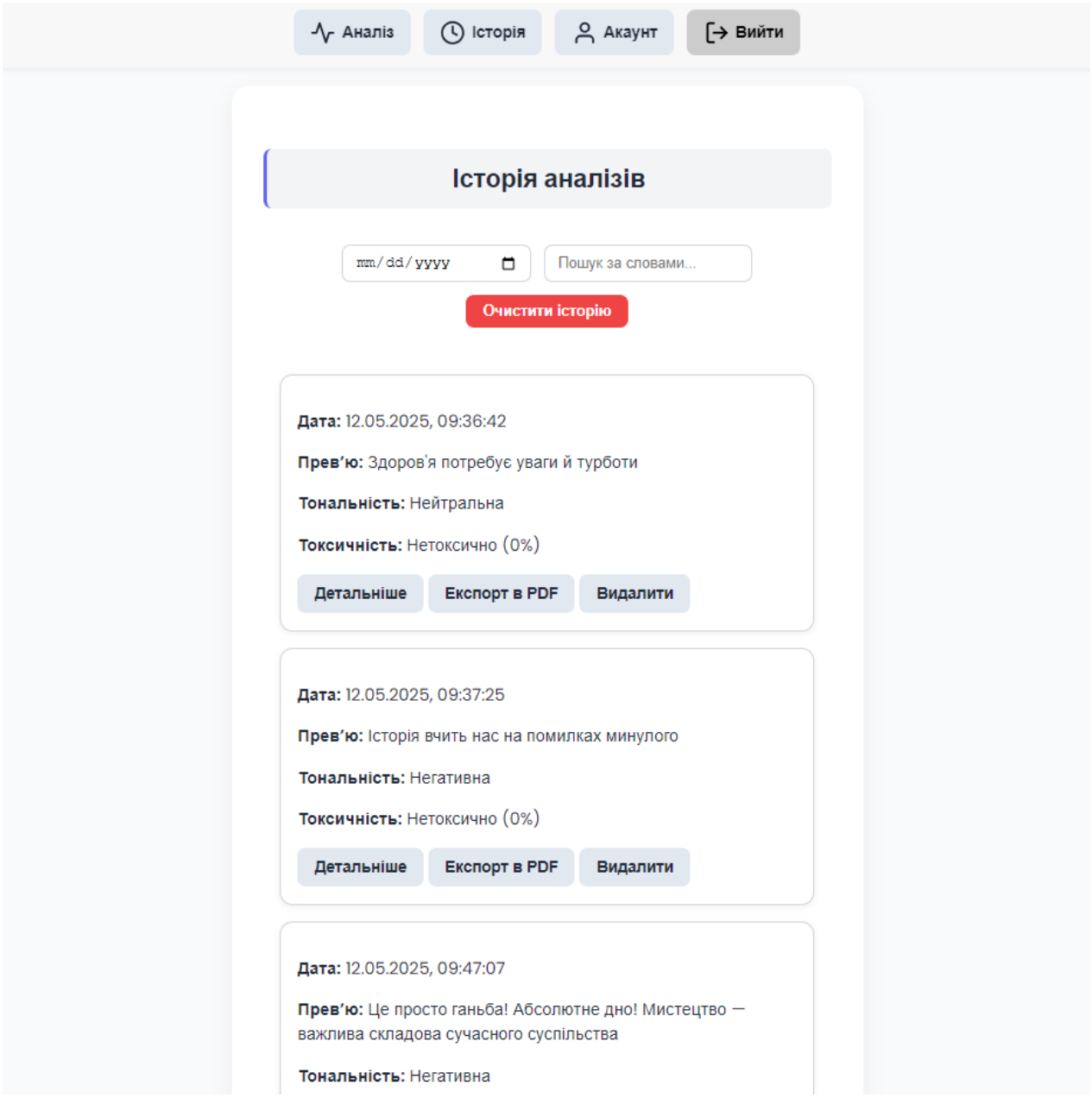

Рисунок 5.10 – Сторінка історії аналізів

Ця вкладка дає змогу працювати з попередніми результатами аналізу:

– відображення історії у вигляді карток (рис. 5.11), де кожна картка — окремий аналіз;

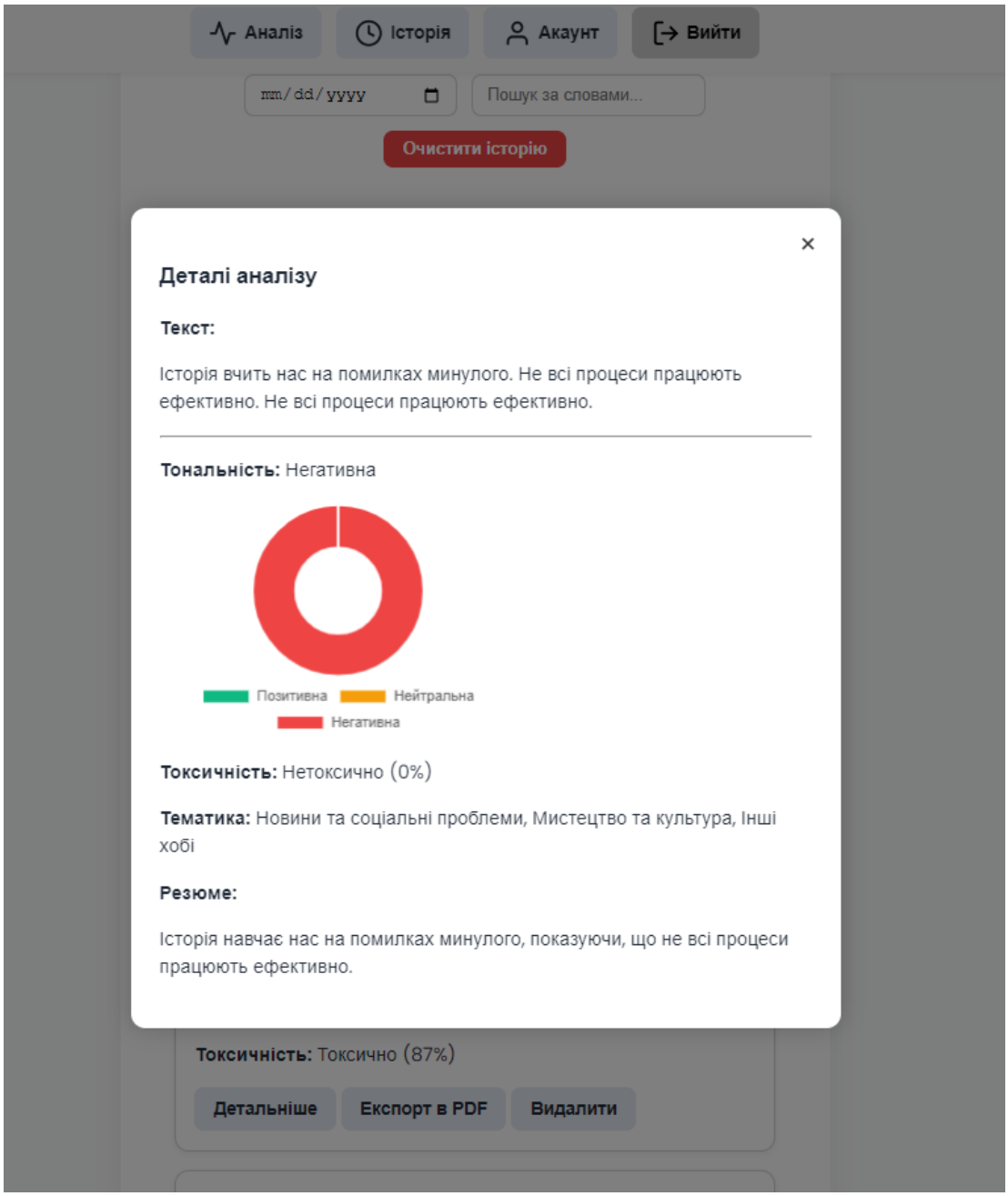

Рисунок 5. 11 – Модальне вікно аналізу з історії

– фільтрація за датою, тональністю, темою;
– пошук за ключовими словами;
– модальне вікно перегляду деталей кожного аналізу;
– можливість видалити окремий запис або очистити всю історію.

Дані історії зберігаються у Firestore і прив’язані до конкретного облікового запису.

account.html – Управління акаунтом (рис. 5.12)

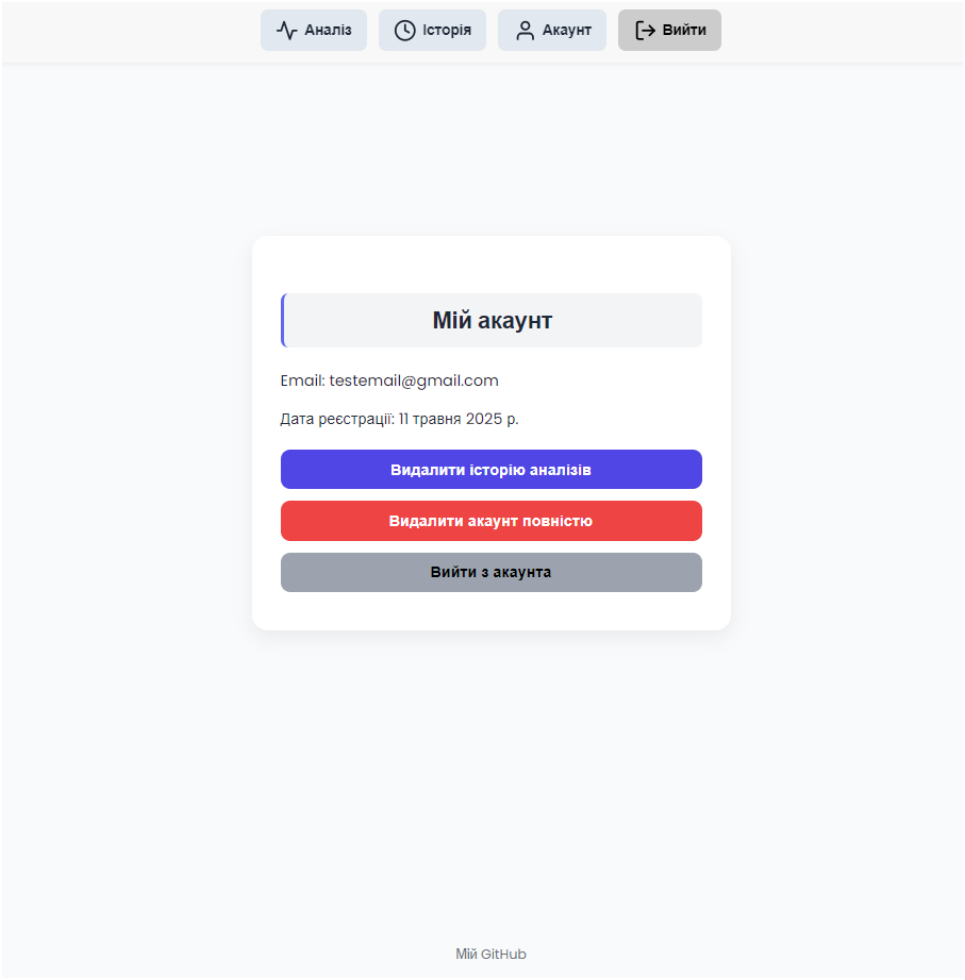

Рисунок 5.12 – Сторінка акаунта користувача

У шапці інтерфейсу доступний обліковий запис користувача, де:

- відображається ім’я або email;
- реалізовано вихід із системи (logout);
- можна перемикатись між вкладками, не втрачаючи авторизацію.

Таким чином, після входу користувач отримує персоналізований доступ до всіх можливостей системи, з історією дій та збереженням контексту роботи.

5.3 Історія аналізів, перегляд та фільтрація

Вкладка history.html стає доступною після входу до системи. На ній реалізовано зручний інтерфейс для перегляду та керування попередніми

результатами аналізу, які були збережені під час використання вкладки analyze.html.

Функціональність вкладки включає:

- виведення всіх збережених аналізів у вигляді адаптивної сітки карток. Кожна картка містить короткий опис запиту (частина тексту, дата, тональність, токсичність, тематика);
- фільтрацію результатів за заданими параметрами — тональністю (позитивна/негативна/нейтральна), темами, рівнем токсичності та датою;
- пошук за ключовими словами, що дає змогу швидко знайти потрібний запис;
- перегляд деталей у модальному вікні — при натисканні на картку відкривається повна інформація про проведений аналіз;
- видалення окремих записів за допомогою кнопки "Видалити";
- очищення всієї історії через відповідну кнопку, яка доступна в інтерфейсі.

Усі записи зберігаються у хмарній базі Firestore та асоціюються з конкретним обліковим записом користувача, що дозволяє забезпечити приватність, синхронізацію та збереження історії навіть після виходу з системи чи зміни пристрою.

5.4 Завантаження файлів та транскрипція аудіо

У вебсервісі реалізовано підтримку завантаження текстових та аудіофайлів, що дозволяє користувачу працювати з інформацією в різних форматах, не вводячи текст вручну.

Користувач може обрати та завантажити один із підтримуваних форматів (рис.5.13) : .txt, .json, .csv, .pdf, .docx, .mp3, .wav, .m4a

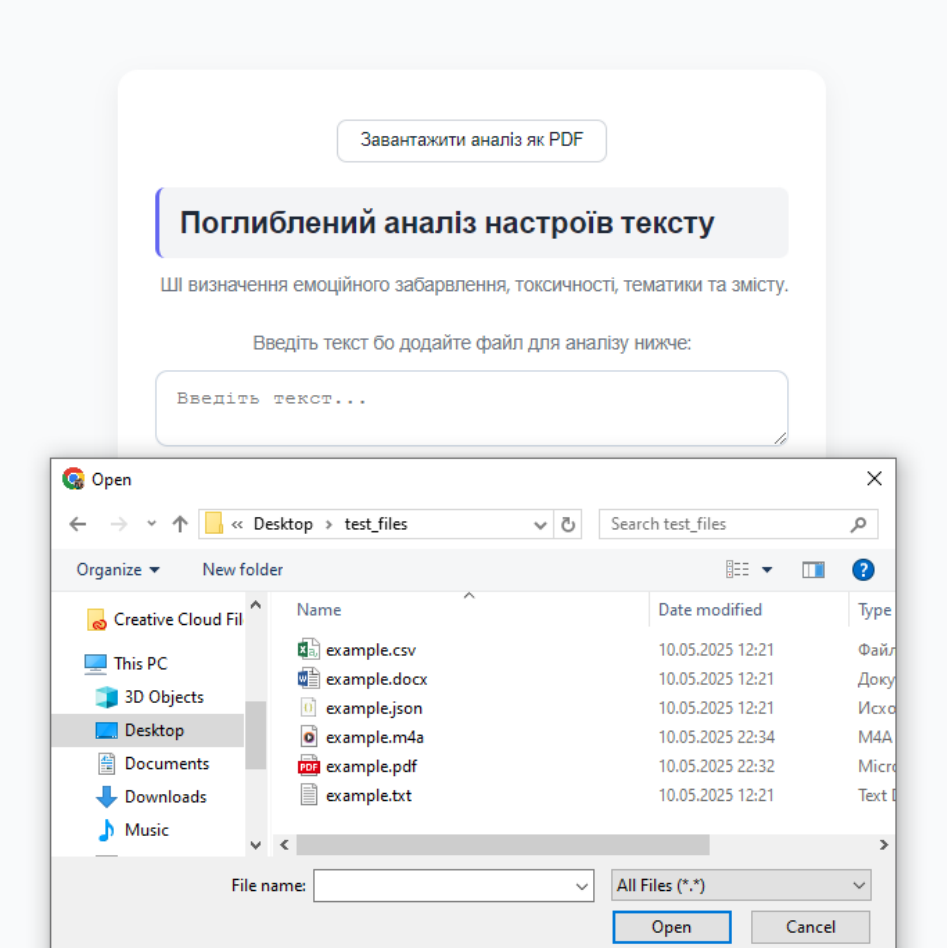

Рисунок 5.13 – Вибір файлу для аналізу

Після завантаження файл автоматично зчитується, його вміст передається у відповідний API, далі — обробляється за тими ж правилами, що й звичайний текст. Для великих документів застосовується модуль чанкінгу (розбиття на частини), що дозволяє уникнути обмежень за кількістю токенів і зберегти точність аналізу.

Сервіс підтримує обробку аудіофайлів (формати — .mp3, .wav, .m4a), які надсилаються на обробку через API на основі моделі Whisper від OpenAI. Ця модель виконує розпізнавання мовлення й повертає текстову транскрипцію, яка одразу передається в аналізатор, що зображено на рисунку 5.14.

Таким чином, користувач може:

– завантажити голосовий фрагмент;
– отримати текстову версію мовлення;

– виконати повноцінний аналіз — визначення тональності, токсичності, тематики, резюме тощо.

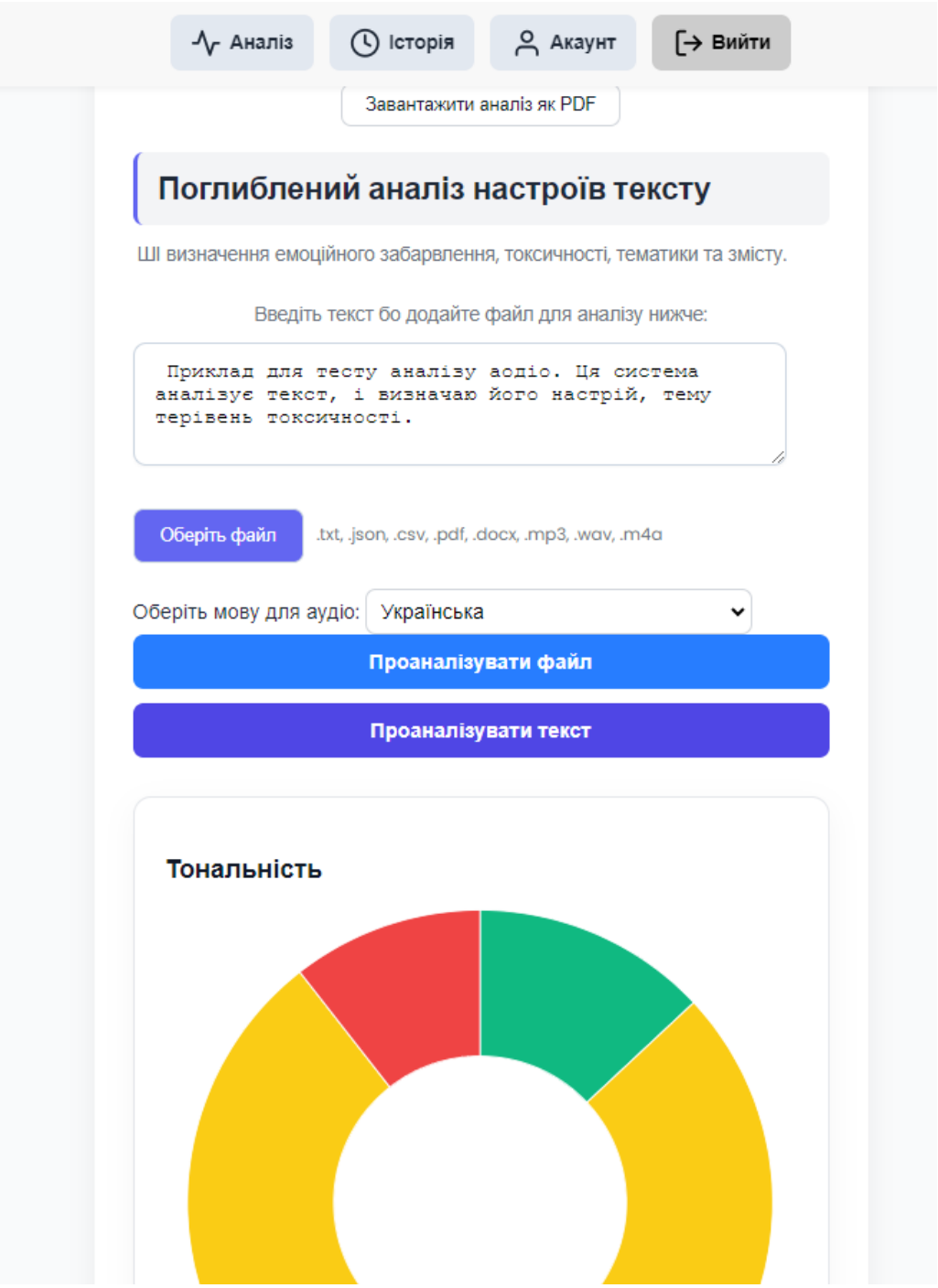

Рисунок 5.14 – Аналіз аудіофайлу

Ця функціональність розширює сферу застосування сервісу — від аналізу листів і документів до оцінки усного мовлення (подкастів, повідомлень, відео тощо).

ВИСНОВКИ

У межах цієї роботи було реалізовано вебсервіс для комплексного аналізу текстів із використанням сучасних технологій обробки природної мови. Основною метою було створення зручного інструменту, здатного визначати тональність, тематику та рівень токсичності тексту, а також виконувати генерацію коротких резюме та зміну стилістики тексту залежно від рівня токсичності.

Для досягнення цієї мети було:

- проаналізовано сучасні методи класифікації текстів та обґрунтовано вибір трансформерних моделей;
- обрано XLM-RoBERTa як основну модель для багатомовного аналізу завдяки її високій точності та широкій мовній підтримці;
- реалізовано серверну частину застосунку на Flask з чіткою структурою ендпоінтів, інтеграцією моделей і обробкою вхідних даних;
- розроблено адаптивний фронтенд з інтуїтивним інтерфейсом та інтеграцією з Firebase для авторизації і збереження історії;
- впроваджено підтримку завантаження текстових файлів, а також транскрипцію аудіо за допомогою моделі Whisper;
- реалізовано систему розбиття великих текстів на чанки з подальшою агрегацією результатів для забезпечення коректної роботи моделей.

Сервіс дозволяє користувачу виконувати як швидкий базовий аналіз без входу, так і поглиблений аналіз з доступом до додаткових функцій після авторизації. Таким чином, проєкт об'єднує штучний інтелект і практичну користь, демонструючи реальне застосування NLP-моделей у вебсередовищі.

У майбутньому систему можна масштабувати шляхом додавання нових мов, типів аналізу (наприклад, визначення фейкових новин), інтеграції з месенджерами або створення мобільної версії застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hugging Face. Transformers documentation. [Електронний ресурс] – Режим доступу до ресурсу: URL: https://huggingface.co/docs/transformers (дата звернення: 15.05.2025).
2. Hugging Face. XLM-RoBERTa. [Електронний ресурс] – Режим доступу до ресурсу: URL: https://huggingface.co/xlm-roberta-base (дата звернення: 15.05.2025).
3. Рассел С. Дж. Штучний інтелект: сучасний підхід : навч. посіб. 4-те вид. Бостон : Пірсон, 2021. 1136 с.
4. OpenAI. Whisper Speech Recognition. [Електронний ресурс] – Режим доступу до ресурсу: URL: https://platform.openai.com/docs/guides/speech-to-text (дата звернення: 15.05.2025).
5. OpenAI. Text completion and prompting with GPT models. [Електронний ресурс] – Режим доступу до ресурсу: URL: https://platform.openai.com/docs/guides/gpt (дата звернення: 15.05.2025).
6. Firebase. Firebase Authentication. [Електронний ресурс] – Режим доступу до ресурсу: URL: https://firebase.google.com/docs/auth (дата звернення: 15.05.2025).
7. Хеберт Б., Грін С. Applied Natural Language Processing with Python : навч. посіб. Бірмінгем : Packt Publishing, 2020. 334 с.
8. Firebase. Cloud Firestore documentation. [Електронний ресурс] – Режим доступу до ресурсу: URL: https://firebase.google.com/docs/firestore (дата звернення: 15.05.2025).
9. Flask. Flask Documentation. [Електронний ресурс] – Режим доступу до ресурсу: URL: https://flask.palletsprojects.com/ (дата звернення: 15.05.2025).
10. Python Software Foundation. Python 3.11 Documentation. [Електронний ресурс] – Режим доступу до ресурсу: URL: https://docs.python.org/3.11/ (дата звернення: 15.05.2025).

11. Чоллет Ф., Аллана Г., Браун Д., Воллес А. Deep Learning with Python : навч. посіб. 2-ге вид. Нью-Йорк : Manning Publications, 2021. 504 с.
12. Бьорн Х., Лю Й., Кернс Е., Крузі М. Natural Language Processing in Action : навч. посіб. Нью-Йорк : Manning Publications, 2019. 391 с.
13. Бішоп К. та ін. Pattern Recognition and Machine Learning : підручник. Нью-Йорк : Springer, 2006. 738 с.