МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Навчально-науковий інститут новітніх освітніх технологій
Кафедра інформаційно-обчислювальних систем і управління

ГЕРЦІЙ Володимир Вікторович

Прикладний програмний інтерфейс для транскрипції та субтитрів для аудіо та відеофайлів на основі штучного інтелекту / Artificial intelligence powered transcription and subtitling aplication programming Interface for audio or video files

Спеціальність 122 – Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-41
В.В. Герцій

Науковий керівник:
к.т.н., доцент І.В. Турченко

Кваліфікаційну роботу допущено до захисту
«___»_____ 2025 р.

В.о. завідувача кафедри
_____ Н.М. Васильків

Тернопіль – 2025

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
спеціальність 122 – Комп'ютерні науки
освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Н.М. Васильків
«____» _____2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ГЕРЦІЙ Володимир Вікторович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Прикладний програмний інтерфейс для транскрипції та субтитрів для аудіо та відеофайлів на основі штучного інтелекту / Artificial intelligence powered transcription and subtitling Application Programming Interface for audio or video files.
керівник роботи І.В. Турченко к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 20 грудня 2024 р. № 938.
2. Строк подання студентом закінченої кваліфікаційної роботи 25 травня 2025 р.
3. Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- аналіз предметної області;
- аналіз існуючих рішень, які працюють зі транскрипцією та субтитрами;
- визначення переваг і недоліків різних підходів до транскрипції та субтитрування і постановка задачі дослідження;
- розробка модульної організації прикладного програмного інтерфейсу, що дозволить інтегрувати автоматичну транскрипцію в різні програмні рішення;
- відбір та порівняльний аналіз існуючих моделей для обробки природної мови, зокрема українською та англійською, з аудіо та відео для інтеграції в прикладний програмний інтерфейс з метою визначення найбільш ефективної з них, а саме найбільш точно обробляє природну мову;
- алгоритмічне та інформаційне забезпечення прикладного програмного інтерфейсу;
- розробка та тестування прикладного програмного інтерфейсу і порівняння результатів його роботи з існуючими рішеннями.
5. Перелік графічного матеріалу в роботі:
– схема обробки медіафайлів для транскрипції та субтитрування прикладного програмного інтерфейсу;
– структурна схема прикладного програмного інтерфейсу;
– схема алгоритму роботи прикладного програмного інтерфейсу;
– UML-діаграма класів прикладного програмного інтерфейсу;

– UML-діаграма станів прикладного програмного інтерфейсу.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 20 грудня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01. 2025 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.02. 2025 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2025 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05. 2025 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2025 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів	до 25.05.2025 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту	до 30.05.2025 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 10.06.2025 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 10.06. 2025 р.	

Студент_____ В.В. Герцій
(підпис) (прізвище та ініціали)

Керівник кваліфікаційної роботи_____ І.В. Турченко
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Прикладний програмний інтерфейс для транскрипції та субтитрів для аудіо та відеофайлів на основі штучного інтелекту» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп'ютерні науки» освітньої програми «Комп'ютерні науки» написана обсягом в 73 сторінки і містить 24 ілюстрації, 6 таблиць, 12 додатків та 35 використаних джерел.

Метою кваліфікаційної роботи є розробка прикладного програмного інтерфейсу для автоматичної транскрипції та субтитрів для аудіо та відеофайлів з використанням штучного інтелекту.

Об'єктом дослідження є технології отримання та обробки інформації з аудіо і відеоконтенту.

Предмет дослідження – прикладний програмний інтерфейс для транскрипції та субтитрів з аудіо та відеоконтенту на основі штучного інтелекту.

Розроблено прикладний програмний інтерфейс для транскрипції та субтитрів для аудіо та відеофайлів на основі штучного інтелекту, який дозволяє автоматизувати обробку аудіо- та відеоконтенту, підвищити доступність інформації та оптимізувати робочі процеси.

Ключові слова: ПРИКЛАДНИЙ ПРОГРАМНИЙ ІНТЕРФЕЙС, ТРАНСКРИПЦІЯ, СУБТИТРИ, АУДІО, ВІДЕО, ШТУЧНИЙ ІНТЕЛЕКТ.

ANNOTATION

Qualification work on the topic «Artificial intelligence powered transcription and subtitling Application Programming Interface for audio or video files» for obtaining a bachelor's degree in the specialty 122 «Computer Science» of the educational program «Computer Science» is written on 73 pages and it contains 24 figures, 6 tables, 12 annexes and 35 references.

The purpose of the qualification work is to develop an application programming interface for automatic transcription and subtitling of audio and video files using artificial intelligence.

The object of research is technologies for extracting and processing information from audio and video content.

The subject of the study is an application programming interface for transcription and subtitling of audio and video content based on artificial intelligence.

An application programming interface for transcription and subtitling of audio and video files based on artificial intelligence has been developed, which allows you to automate the processing of audio and video content, increase the availability of information, and optimize workflows.

Keywords: APPLICATION PROGRAMMING INTERFACE, TRANSCRIPTION, SUBTITLES, AUDIO, VIDEO, ARTIFICIAL INTELLIGENCE.

ЗМІСТ

ВСТУП

У сучасному світі обробка аудіо- та відеоконтенту стала невід'ємною частиною багатьох сфер діяльності, включаючи медіа, освіту, розваги, бізнес, науку та медицину. Цифрова епоха, в якій ми живемо, характеризується стрімким зростанням популярності відеоформатів та аудіоматеріалів, що, у свою чергу, зумовлює потребу в ефективних інструментах для їх аналізу, обробки, структурування та збереження.

Одна з основних потреб, що виникає у зв'язку з цим, полягає у швидкому та точному перетворенні аудіо- та відеоданих у текстовий формат. Автоматична транскрипція та створення субтитрів є не лише корисною функцією, але й необхідністю для багатьох користувачів, зокрема для людей з вадами слуху, а також для тих, хто споживає контент у шумних або тихих середовищах, де аудіо може бути недоступним чи небажаним. Крім того, субтитрування сприяє глобалізації контенту, дозволяючи людям з різних мовних середовищ отримувати доступ до інформації без необхідності володіння мовою оригіналу.

Тим не менш, традиційні методи обробки аудіо- та відеоматеріалів часто виявляються складними, затратними за часом та неефективними. Процес транскрибування вручну вимагає значних людських ресурсів та є схильним до помилок. Виклики, пов'язані з точністю розпізнавання мовлення, присутністю фонового шуму, різними акцентами, діалектами, а також складною технічною або спеціалізованою термінологією, ускладнюють створення якісних текстових представлень аудіо- та відеоконтенту. Саме тому виникає гостра необхідність у розробці нових автоматизованих рішень, які б змогли значно спростити цей процес та підвищити його ефективність. Отже, тема кваліфікаційної роботи є актуальною.

Метою кваліфікаційної роботи є розробка прикладного програмного інтерфейсу (англ. Application Programming Interface, API) для автоматичної транскрипції та субтитрів для аудіо та відеофайлів з використанням штучного інтелекту.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- аналіз предметної області;

- аналіз існуючих рішень, які працюють зі транскрипцією та субтитрами;
- визначення переваг і недоліків різних підходів до транскрипції та субтитрування і постановка задачі дослідження;
- розробка модульної організації прикладного програмного інтерфейсу, що дозволить інтегрувати автоматичну транскрипцію в різні програмні рішення;
- відбір та порівняльний аналіз існуючих моделей для обробки природної мови, зокрема українською та англійською, з аудіо та відео для інтеграції в прикладний програмний інтерфейс з метою визначення найбільш ефективної з них, а саме найбільш точно обробляє природну мову;
- алгоритмічне та інформаційне забезпечення прикладного програмного інтерфейсу;
- розробка та тестування прикладного програмного інтерфейсу і порівняння результатів його роботи з існуючими рішеннями.

Об’єктом дослідження є технології отримання та обробки інформації з аудіо і відеоконтенту.

Предметною областю виступає прикладний програмний інтерфейс для транскрипції та субтитрів з аудіо та відеоконтенту на основі штучного інтелекту.

Розроблюваний API для транскрипції та створення субтитрів на основі штучного інтелекту матиме широке застосування в багатьох сферах. Основні з них:

- медіа та розваги: автоматичне створення субтитрів для відео на YouTube, TikTok, Instagram тощо; транскрипція подкастів та інтерв’ю; локалізація відеоконтенту через переклад субтитрів;
- освіта: створення транскрипцій лекцій, вебінарів, навчальних курсів; допомога студентам із порушеннями слуху, створення навчальних матеріалів з відео/аудіо;
- бізнес: автоматичний запис і транскрипція зустрічей та аналіз розмов у службах підтримки клієнтів;
- охорона здоров’я: транскрипція аудіозаписів консультацій лікарів та обробка медичних звітів, створених голосом;
- журналістика: транскрипція інтерв’ю, конференцій, публічних виступів;
- технології та розробка програмного забезпечення: інтеграція транскрипцій

у аналіз відео; автоматичне розпізнавання мови для інтерактивних застосунків; використання в чат-ботах і голосових асистентах.

Результати дослідження опубліковано в матеріалах науково-практичної конференції «Інтелектуальні інформаційні технології в прикладних дослідженнях» (IITAR – 2025), м. Тернопіль, 27–29 травня 2025 р. (додаток Н).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

У сучасному світі обробка аудіо- та відеоконтенту є невід'ємною частиною багатьох сфер діяльності, включаючи медіа, освіту, розваги та бізнес. Велика кількість інформації подається у вигляді аудіо- та відеоматеріалів, що створює потребу в ефективних інструментах для їх аналізу, обробки та структурування. Одним із важливих аспектів цієї обробки є автоматична транскрипція та створення субтитрів.

Автоматична транскрипція та субтитрування відіграють важливу роль у багатьох аспектах життя та діяльності. З огляду на зростаючий обсяг цифрового контенту та популяризацію відеоформату, автоматичні інструменти транскрипції та субтитрування стають необхідністю для бізнесу, освітніх установ, контент-мейкерів та технологічних компаній.

Основні переваги цих технологій включають:

- доступність контенту – субтитри та текстові транскрипції роблять аудіо- та відеоматеріали доступними для людей із вадами слуху, а також для тих, хто споживає контент у беззвучному режимі;
- полегшення перекладу та локалізації – автоматично створені субтитри можуть бути швидко перекладені на різні мови, що спрощує адаптацію контенту для міжнародної аудиторії;
- підвищення зручності пошуку та аналізу – текстові версії аудіо- та відеоконтенту дозволяють здійснювати швидкий пошук за ключовими словами, що особливо корисно для медіа-архівів, навчальних платформ і бізнес-інструментів;
- автоматизація робочих процесів – API для транскрипції дозволяє інтегрувати функціональність розпізнавання мовлення в програмні продукти, спрощуючи ведення нотаток, аналіз відеоконференцій, документування зустрічей тощо;
- підвищення взаємодії з користувачем – текстові транскрипції можуть бути використані для створення чат-ботів, віртуальних асистентів та інших інтерактивних рішень, що базуються на голосовому введенні.

Створення ефективного API, який дозволяє точно й швидко конвертувати аудіо у текст, є актуальною задачею, що має велике значення для цифрового середовища.

Прикладний програмний інтерфейс для транскрипції та субтитрування на основі штучного інтелекту (ШІ) дозволяє автоматизувати цей процес, значно підвищуючи швидкість і точність розпізнавання мовлення. Використання технологій машинного навчання та обробки природної мови (NLP) дозволяє створювати якісні текстові представлення аудіоінформації, що відкриває широкі можливості для інтеграції таких рішень у різні програмні продукти.

API, або прикладний програмний інтерфейс, - це набір правил або протоколів, які дозволяють програмним додаткам взаємодіяти один з одним для обміну даними, функціями та функціоналом [1-2].

API можна використовувати у різних галузях: соціальні мережі; електронна комерція; мобільні програми.

До прикладів популярних прикладних програмних інтерфейсів та їх можливостей можна віднести: Google Maps API; Twitter API; Facebook API [2].

Транскрипція – це ретельний процес перетворення усного мовлення в письмовий або текстовий формат. Ця практика має фундаментальне значення для передачі суті аудіо- та відеоконтенту, забезпечуючи збереження інформації, її доступність та використання в різних контекстах. Ефективна транскрипція фіксує не лише вимовлені слова, а й наміри та нюанси мовця, забезпечуючи всебічний запис вербального обміну.

Щоб задовольнити різні потреби та контексти, транскрипцію поділяють на три основні типи: дослівна, відредагована та інтелектуальна. Кожен тип слугує окремій меті та відповідає певним вимогам щодо чіткості, деталізації та презентації [3].

Незважаючи на значні досягнення в області штучного інтелекту та розпізнавання мовлення, автоматична транскрипція все ще стикається з рядом викликів та проблем: акценти та діалекти; фонові шуми; перекривання мовлення; наявність спеціалізованої термінології; емоційне забарвлення та інтонація; якість аудіозапису.

Субтитри – це потужний інструмент, який дозволяє донести якісний контент до аудиторії в усьому світі всіма мовами. Субтитри допомагають глядачам дивитися іноземні фільми рідною мовою, а також роблять онлайн-контент доступним для глобальної аудиторії – вони допомагають кожному зрозуміти відеоконтент [4].

Найпопулярніші формати субтитрів: SubRipper (.srt); MicroDVD (.sub); Sub Station Alpha (.ssa); SAMI (.smi); WebVTT (.vtt).

Найбільш універсальними форматами субтитрів, які би працювали у більшості відеоплеєрів є SRT і VTT. Вони також працюють на більшості популярних відеоплатформ і сторінок у соціальних мережах [5].

Субтитри відіграють ключову роль у забезпеченні доступності відеоконтенту для різних категорій користувачів, а також у розширенні аудиторії за межами мовного бар'єру. Основні аспекти їх важливості включають: інклюзивність для людей із порушеннями слуху; доступність у шумному середовищі; підтримка багатомовності та перекладу; поліпшення розуміння контенту; освітні можливості.

З огляду на ці фактори, використання субтитрів у медіаконтенті стає важливим стандартом, який сприяє його доступності, популярності та розширенню цільової аудиторії.

Штучний інтелект – це набір технологій, які дозволяють комп'ютерам виконувати різноманітні розширені функції, включаючи здатність бачити, розуміти і перекладати усну і письмову мову, аналізувати дані, надавати рекомендації тощо [6].

Штучний інтелект відіграє важливу роль в автоматизації процесу транскрипції та створення субтитрів. Основні методи та технології включають: глибоке навчання та нейромережі; обробка природної мови (NLP); автоматичне визначення мовців; автоматичний переклад та локалізація.

Отже, обробка аудіо- та відеоконтенту важлива для медіа, освіти, розваг і бізнесу. Транскрипція та створення субтитрів на базі ШІ підвищують доступність і точність, автоматизуючи процеси через API. Субтитри допомагають людям з вадами слуху, а також полегшують переклад і аналіз. Незважаючи на виклики, як акценти та шуми, розвиток ШІ відкриває нові можливості для інтеграції

автоматизованих рішень.

Основні процеси, які виконуватиме прикладний програмний інтерфейс: завантаження аудіо- або відеофайлів; обробка відеофайлів; обробка аудіофайлів; відстеження статусу завдання; завантаження результату; додаткові функції.

Схема обробки медіафайлів прикладного програмного інтерфейсу зображена на рисунку 1.1.

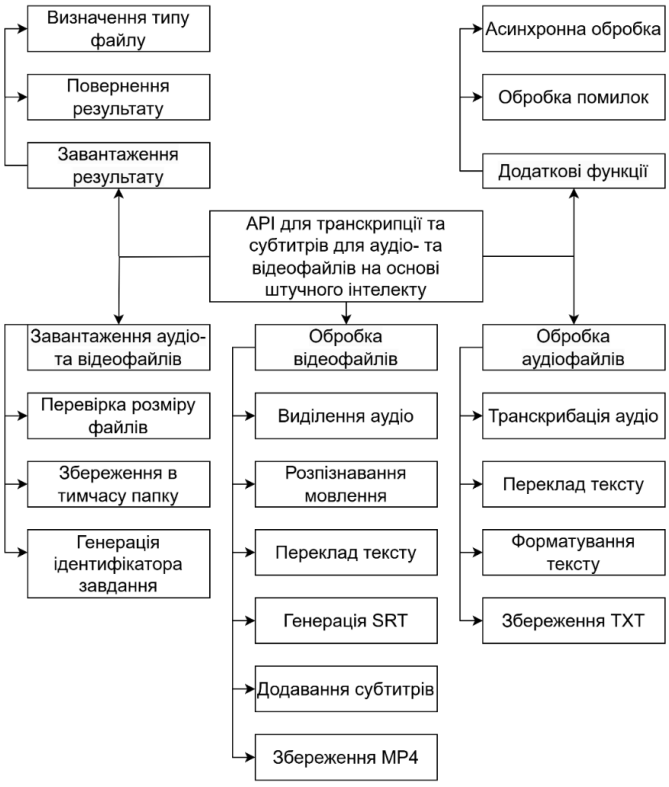


Рисунок 1.1 – Схема обробки медіафайлів для транскрипції та субтитрування

API призначений для обробки аудіо- та відеофайлів, створюючи транскрипції та субтитри. Прикладний програмний інтерфейс перевіряє та зберігає завантажені файли, присвоюючи їм унікальний ідентифікатор. Відеофайли обробляються шляхом виділення аудіодоріжки, розпізнавання мовлення нейромережею Hugging Face API, перекладу (за потреби) та генерації субтитрів у форматі SRT, які вбудовуються у MP4. Аудіофайли транскрибуються Hugging Face API, перекладаються та зберігаються як текст. Користувач отримує готовий файл (відео з субтитрами або транскрипцію). API підтримує логування, асинхронну обробку та захист від помилок для ефективної роботи.

Отже, всі ці компоненти працюють разом, забезпечуючи повний цикл

обробки медіафайлів – від завантаження до отримання готового результату з необхідними субтитрами чи транскрипцією.

У розробці API для транскрипції та субтитрів виділяються транзакційна та аналітична складові. Транзакційна охоплює обробку аудіо- та відеофайлів: завантаження, присвоєння унікального ідентифікатора, екстракцію аудіо, транскрипцію, переклад і генерацію субтитрів. Дані про етапи обробки зберігаються в базі. Аналітична складова дозволяє аналізувати кількість оброблених файлів, час обробки, розподіл за мовами та виявляти тенденції для оптимізації процесів. Інтеграція цих складових забезпечує ефективну обробку запитів, аналіз даних і покращення якості послуг.

Отже, інтеграція транзакційної та аналітичної складових у процеси дозволяє ефективно обробляти запити та вдосконалювати послуги на основі аналізу даних, підвищуючи якість обслуговування і задоволеність клієнтів.

1.2 Огляд і аналіз існуючих рішень

Як існуючі рішення було проаналізовано наступні сервіси:

а) Kapwing;

б) HappyScribe;

в) Sonix.

При цьому було сформовано критерії, за якими вони аналізувалися:

- розмір файлу, який можна завантажити (або його тривалість у хвилинах);
- наявність на вихідному відео логотипу сервісу;
- інтерфейс;
- швидкість додавання субтитрів до відео;
- можливість отримати субтитри у текстовому форматі;
- якість вихідного відео.

Kapwing – це сучасна платформа для створення відео, яка допомагає командам швидше створювати чудовий контент. Головне гасло сервісу – створювати більше контенту з меншою витратою часу [7]. Його основні характеристики:

- розмір файлу, який можна завантажити (або його тривалість у хвилинах) – у безкоштовному обліковому запису надається 10 хвилин, які можна використати для додавання субтитрів. Якщо користувач використає ці 10 хвилин, то більше не зможе нічого зробити в цьому обліковому записі, і йому буде запропоновано купувати преміум доступ;
- наявність на відео логотипу сервісу – на відео присутній логотипу сервісу;
- інтерфейс – сучасний, зручний та інтуїтивно зрозумілий для людини, яка працює з сервісами для редагування відео;
- швидкість додавання субтитрів до відео – на цьому сервісі додавання субтитрів зайняло приблизно 2 хвилини, а експорт відео ще хвилини 3, разом весь процес зайняв близько 5 хвилин;
- можливість отримати субтитри у текстовому форматі – можна отримати лише на мові, яка присутня у відео. Для самих субтитрів, потрібно вже мати преміум доступ до сервісу;
- якість вихідного відео – якість відео у вихідному файлі не відповідає якості вхідного, для кращої якості потрібно преміум доступ.

Happyscribe – це універсальна платформа для транскрибації та субтитрів. Головне гасло – штучний інтелект працює пліч-о-пліч з найкращими професіоналами мови [8]. Характеристики платформи:

- розмір файлу, який можна завантажити(або його тривалість у хвилинах) – підтримуються файли великих обсягів та довготривалі, але на один обліковий запис можна зробити одне відео з субтитрами;
- наявність на відео логотипу сервісу – на відео присутній логотипу сервісу;
- інтерфейс – сучасний, зручний, мінімалістичний та інтуїтивно зрозумілий для будь-якої людини. Тут немає додаткових опцій для людей, які займаються відеомонтажем або корекцією звуку;
- швидкість додавання субтитрів до відео – на цьому сервісі додавання субтитрів зайняло приблизно 3 хвилини, а експорт відео ще хвилини 1, разом весь процес зайняв близько 4 хвилин;
- можливість отримати субтитри у текстовому форматі – на цьому сервісі не можливо отримати субтитри у текстовому форматі, бо на це потрібний преміум

доступ;

- якість вихідного відео – якість відео у вихідному файлі таке саме, як якість вхідного.

Sonix – це платформа для перекладу та субтитрів на більше ніж 54 мовах. Головне гасло – швидко, точно та доступно [9]. Ось аналіз цього сервісу:

- розмір файлу, який можна завантажити(або його тривалість у хвилинах) – підтримуються файли великих обсягів, але є обмеження на безкоштовний обліковий запису до 30 хвилин відео;
- наявність на відео логотипу сервісу – на відео відсутній логотипу сервісу;
- інтерфейс – дизайн зовсім не зручний, тому що потрібно пошукати по опціях як додати субтитри до відео, тому що автоматично вони не додаються. Також щоб отримати відео з субтитрами також треба докласти чималих зусиль;
- швидкість додавання субтитрів до відео – на цьому сервісі додавання субтитрів зайняло приблизно 5 хвилин;
- можливість отримати субтитри у текстовому форматі – на цьому сервісі можливо отримати субтитри у різних текстових форматах;
- якість вихідного відео – якість відео у вихідному файлі набагато гірша ніж у вхідному;

Дизайн вищезгаданих сервісів представлено на рисунках 1.2 – 1.4.

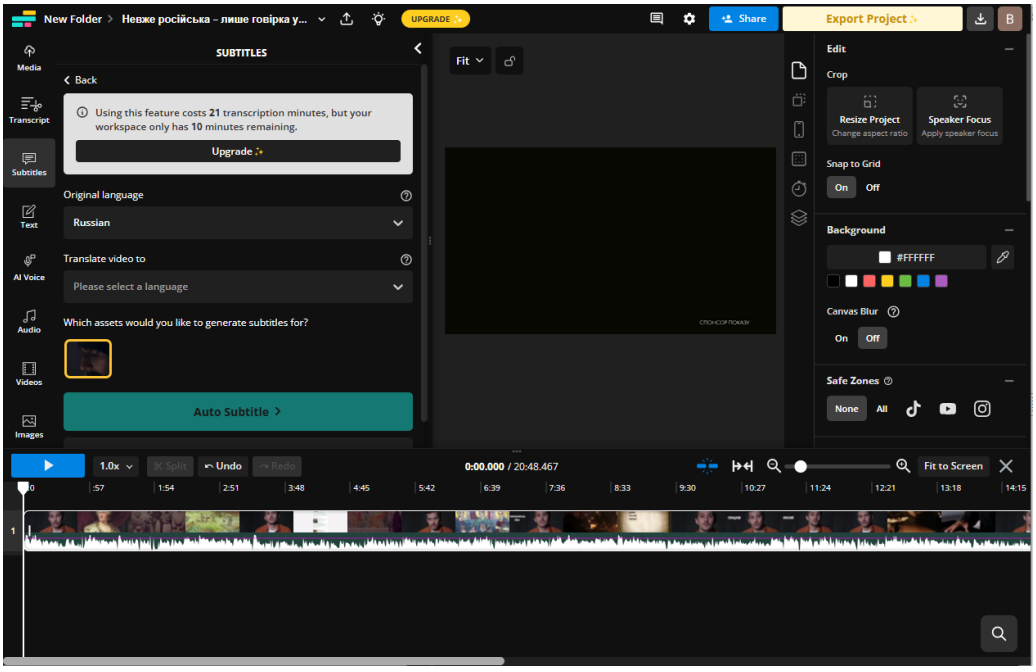


Рисунок 1.2 – Дизайн сервісу Kapwing

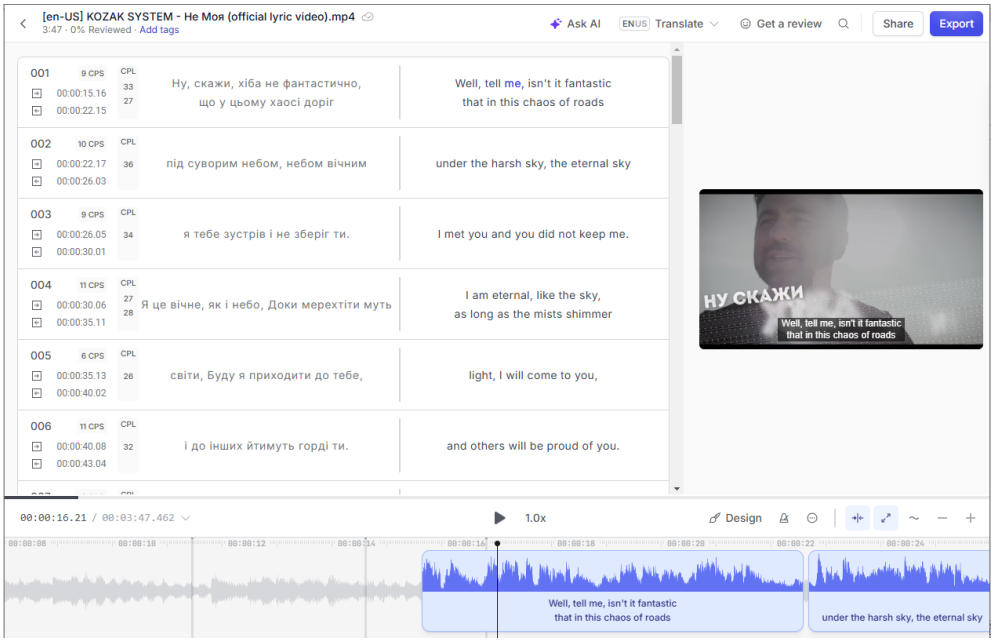


Рисунок 1.3 – Дизайн сервісу Happyscribe

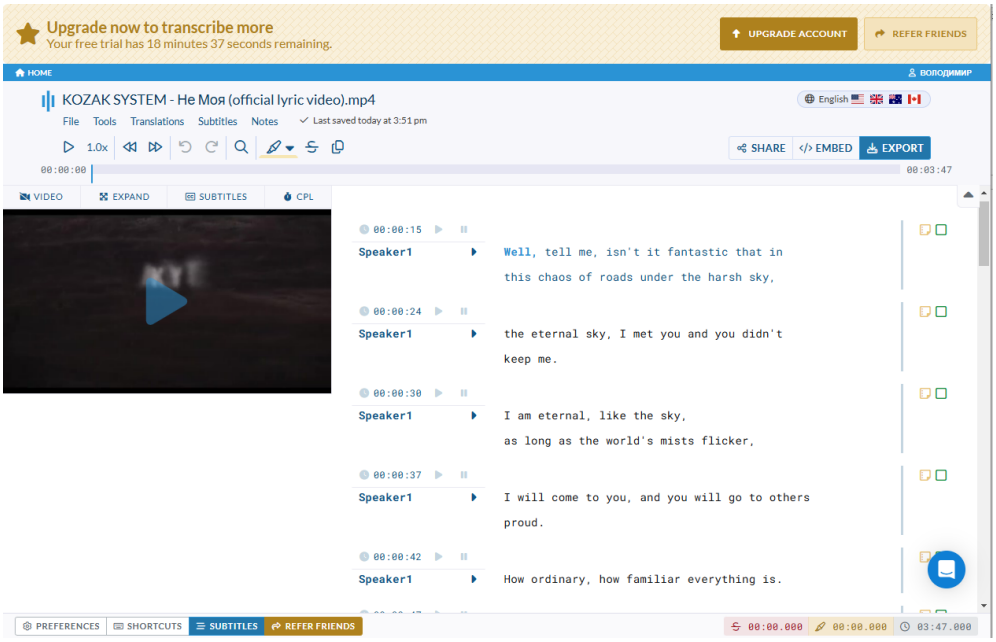


Рисунок 1.4 – Дизайн сервісу Sonix

Після аналізу платформ Kapwing, Happyscribe та Sonix для транскрипції та субтитрів:

- усі мають обмеження у безкоштовних версіях за тривалістю та розміром файлів;
- Sonix не додає водяний знак, інші – додають;
- Happyscribe має найзручніший інтерфейс, Kapwing – середній (більше для відеомонтажу), Sonix – незручний;

- швидкість обробки субтитрів приблизно однакова (~5 хв), залежить від довжини відео;

- Sonix найкраще підходить для текстових субтитрів у різних форматах, Kapwing обмежений мовою відео у безкоштовній версії, Happyscribe не підтримує текстові субтитри безкоштовно;

- найкраща якість відео у Happyscribe, трохи гірша у Kapwing, найгірша – у Sonix;

- усі використовують ШІ для розпізнавання та перекладу тексту.

Отже, проаналізувавши існуючі рішення треба врахувати мінуси цих платформ, зокрема зробити більш гнучку тарифікацію для створення відео; не додавати водяного знаку до відео; сучасний, мінімалістичний та зручний інтерфейс для будь-кого; оптимізація часу на додавання субтитрів; додання можливості отримувати субтитри у текстовому форматі; не змінювати якість вихідного відео на гіршу. Цей перелік функціоналу дозволить проєктованому API мати переваги над існуючими рішеннями.

1.3 Постановка задачі дослідження

На основі проведеного аналізу у попередніх підрозділах було визначено низку обмежень та недоліків, які створюють незручності для користувачів та ускладнюють процес автоматизації обробки відео- та аудіоконтенту.

Серед основних проблем, характерних для існуючих платформ, можна виділити:

а) обмежений функціонал безкоштовних версій (ліміт часу обробки, неможливість отримання текстових субтитрів без оплати, обмеження на кількість файлів);

б) наявність логотипів сервісів на вихідних відеофайлах, що значно знижує якість та придатність контенту для подальшого використання;

в) погіршення якості відео після обробки, особливо у безкоштовних тарифах. Відсутність зручних API-інтерфейсів для автоматичної інтеграції цих сервісів у власні програмні продукти чи бізнес-процеси;

г) недостатня гнучкість у налаштуванні параметрів обробки та формату вихідних даних.

З урахуванням зазначених проблем сформовано загальну концепцію проєктованої кваліфікаційної роботи – створення веб-сервісу у вигляді API, який забезпечуватиме автоматичне розпізнавання мовлення з відео- та аудіофайлів, створення субтитрів та їх подальше збереження у зручних форматах (SRT, VTT, TXT). Важливою особливістю цього API стане його інтеграційна спрямованість, що дозволить стороннім додаткам легко використовувати функціонал сервісу для автоматизації власних процесів.

Актуальність зумовлена зростанням обсягів аудіо- та відеоконтенту в різних сферах, що вимагає ефективних та доступних рішень для автоматичної обробки мовної інформації. Існуючі інструменти транскрипції та субтитрування часто обмежені у функціональності, складні для інтеграції або потребують платного доступу. Розробка відкритого API з підтримкою української та англійської мов і широкими інтеграційними можливостями є актуальним завданням прикладних досліджень у галузі комп’ютерних наук.

Метою кваліфікаційної роботи є розробка прикладного програмного інтерфейсу для автоматичної транскрипції та субтитрів для аудіо та відеофайлів з використанням штучного інтелекту.

Для досягнення поставленої мети необхідно виконати наступні завдання:

а) аналіз предметної області;

б) аналіз існуючих рішень, які працюють зі транскрипцією та субтитрами;

в) визначення переваг і недоліків різних підходів до транскрипції та субтитрування і постановка задачі дослідження;

г) розробка модульної організації прикладного програмного інтерфейсу, що дозволить інтегрувати автоматичну транскрипцію в різні програмні рішення;

д) відбір та порівняльний аналіз існуючих моделей для обробки природної мови, зокрема українською та англійською, з аудіо та відео для інтеграції в прикладний програмний інтерфейс з метою визначення найбільш ефективної з них, а саме найбільш точно обробляє природну мову;

е) алгоритмічне та інформаційне забезпечення прикладного програмного інтерфейсу;

є) розробка та тестування прикладного програмного інтерфейсу і порівняння результатів його роботи з існуючими рішеннями.

Отже, сформульовані мета та завдання дослідження випливають з виявлених проблем існуючих рішень у сфері автоматичної транскрипції та субтитрування. Запропонований підхід передбачає створення функціонального та доступного прикладного програмного інтерфейсу, що ґрунтується на сучасних технологіях штучного інтелекту та відповідає актуальним вимогам цифрової обробки медіаконтенту.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ ПРИКЛАДНОГО ПРОГРАМНОГО ІНТЕРФЕЙСУ

2.1 Загальна структура проєктованого прикладного програмного інтерфейсу

API складається з веб-сервісу (FastAPI) та фронтенду (React.js) для автоматичного створення субтитрів до відео- та аудіофайлів за допомогою моделей розпізнавання мовлення з Hugging Face. Користувач через веб-інтерфейс завантажує файл, вибирає мову та отримує субтитри (SRT, VTT, TXT) або відео з вбудованими субтитрами.

Функції прикладного програмного інтерфейсу наступні:

- обробка HTTP-запитів;
- збереження файлів;
- розпізнавання мовлення через Hugging Face;
- створення та форматування субтитрів;
- накладання субтитрів на відео;
- авторизація, логування, повернення результатів.

Компонентами API є:

а) фронтенд (React.js) – це інтерфейс для завантаження файлів, вибору параметрів, отримання результатів;

б) API-сервер (FastAPI) – це обробка запитів, збереження файлів, взаємодія з Hugging Face, генерація субтитрів;

в) модулі обробки – це розпізнавання мовлення, форматування субтитрів, накладання на відео;

г) сховище та база даних – це тимчасове зберігання файлів, історія обробок, дані користувачів;

д) логування – це моніторинг і аналіз процесів.

Інформаційні зв’язки у API – Користувач → React.js → FastAPI → Сховище/Hugging Face → Модулі обробки → Сховище → React.js.

Прикладний програмний інтерфейс використовує об’єктно- та процесно-орієнтований підхід для чіткої структуризації модулів і послідовності обробки файлів.

На рисунку 2.1 зображена структурна схема прикладного програмного інтерфейсу.

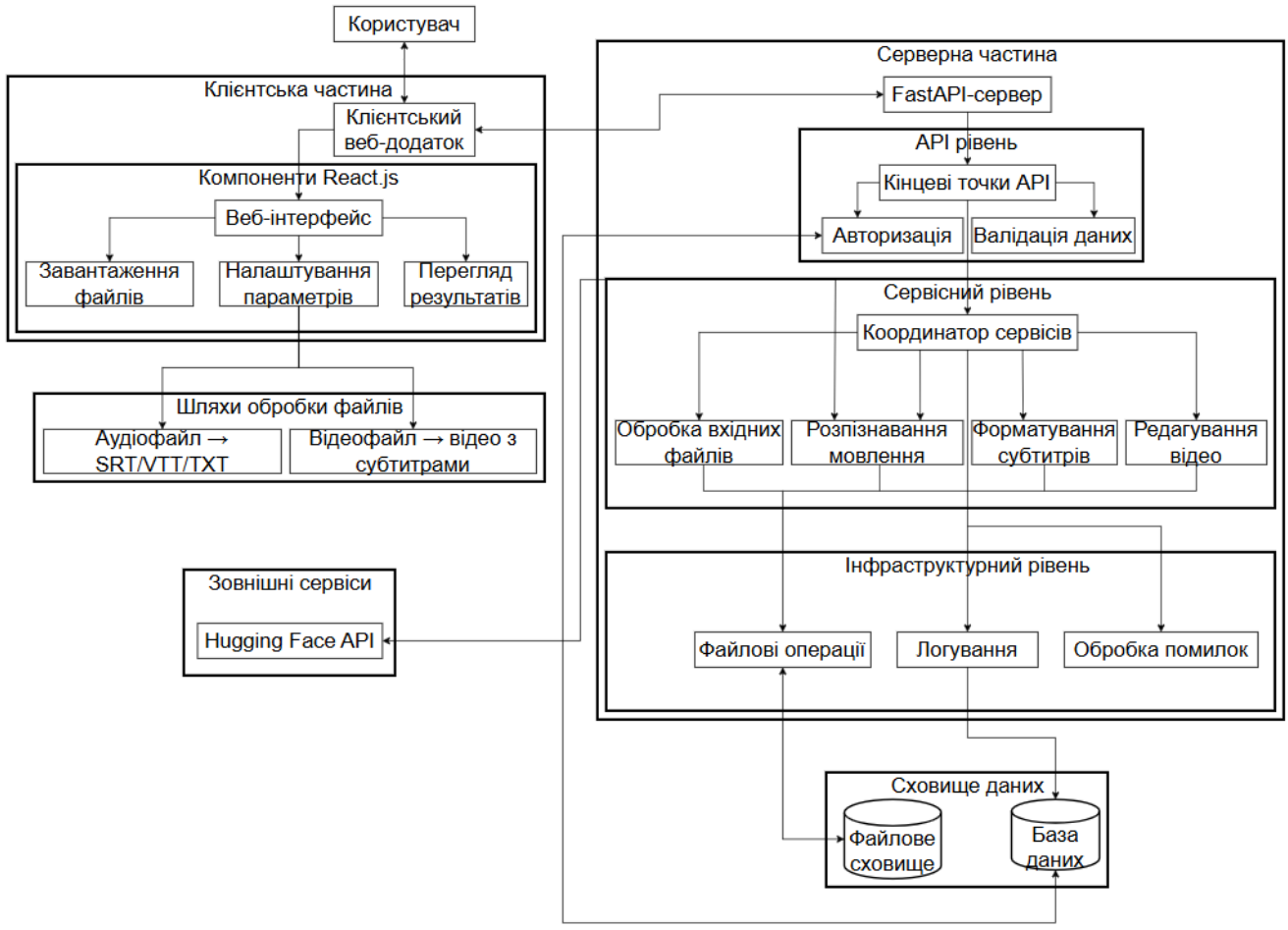


Рисунок 2.1 – Структурна схема прикладного програмного інтерфейсу

Діаграма показує прикладний програмний інтерфейс для автоматичного створення субтитрів, поділену на функціональні блоки. Користувач працює з веб-додатком на React.js для завантаження файлів, налаштування параметрів і перегляду результатів. Серверна частина на FastAPI обробляє аудіо- та відеофайли, виконує розпізнавання мовлення через Hugging Face API, формує субтитри (SRT, VTT, TXT) або відео з субтитрами. Вона включає рівні API, сервісів та інфраструктури, що відповідають за авторизацію, обробку, логування та помилки. Дані зберігаються у файловому сховищі та базі даних. Стрілки на діаграмі відображають потоки даних між компонентами.

Рисунок 2.1 дає повне уявлення про структуру прикладного програмного інтерфейсу для створення субтитрів, демонструючи її компоненти, їх взаємозв'язки та шляхи обробки даних.

2.2 Алгоритм роботи прикладного програмного інтерфейсу

Щоб розпочати роботу з API, користувач завантажує відео- чи аудіофайл через веб-інтерфейс, файл перевіряється за форматом і розміром, конвертується (з відео витягується аудіо) і відправляється до Hugging Face API для розпізнавання мовлення. Модель повертає текст із часовими мітками, який розбивається на репліки, формується у субтитри (SRT, VTT, TXT). За потреби субтитри накладаються на відео за допомогою FFmpeg. Готовий файл зберігається, користувач отримує посилання. Усі етапи логуються.

Алгоритми API наступні:

- у завантаженні та перевірці файл перевіряється за типом і розміром, зберігається, з відео витягується аудіо;
- на етапі алгоритму розпізнавання мовлення аудіо ділиться на 10-секундні сегменти, обробляється Hugging Face API, результати об’єднуються;
- у створенні субтитрів текст з мітками розбивається на репліки з урахуванням пауз, формується у вибраний формат;
- на етапі накладання субтитрів за запитом субтитри додаються до відео через FFmpeg, результат зберігається.

Схема алгоритму роботи прикладного програмного інтерфейсу наведена у додатку А. Спочатку користувач або модуль управління надсилають запит на транскрипцію до сервісу обробки аудіо/відео. Після цього API визначає доступність операцій транскрипції та субтитрування. Якщо операція доступна, запит направляється до штучного інтелекту для виконання транскрипції. Штучний інтелект обробляє аудіофайл, аналізує його, визначає мову, структуру даних транскрипції та субтитрів, після чого генерує готовий файл транскрипції або субтитрів. У разі успішної обробки результат повертається користувачу або модулі управління. Якщо ж операція недоступна, прикладний програмний інтерфейс повертає повідомлення про помилку. На завершальному етапі результат або помилка надсилаються клієнту через прикладний програмний інтерфейс. Схема також враховує збереження готових субтитрів у сховище даних для подальшого використання.

2.3 Відбір та порівняльний аналіз існуючих моделей розпізнавання тексту для інтеграції в прикладний програмний інтерфейс

Для вибору моделі розпізнавання українськомовного аудіо проаналізовано вісім моделей із Hugging Face[13], відібраних за точністю (>90%) з 20 моделей, результати досліджень щодо використання їх для розпізнавання та синтезу мовлення для української мови, представлені на GitHub-репозиторію [22]. Тестування проводилося на 5-хвилинній аудіодоріжці з YouTube-відео «Реальна Історія» Акіма Галімова [23], оцінюючи точність транскрибування, коректність термінів, власних назв, чисел та збереження граматичних особливостей (кличний відмінок, повноголосся). Опис моделей взятих для порівняльного аналізу:

- nvidia/stt_uk_citrinet_1024_gamma_0_25 (CitriNet) – 18 шарів, 1024 канали. Помилки в термінах («стар слов'янська»), назвах («бостромирова»), датах («тисячі пдесят штого») [24];
- neongeckocom/stt_uk_citrinet_512_gamma_0_25 (CitriNet) – 512 каналів, компактніша. Плутає назви («росії» замість «Русі») [25];
- theodotus/stt_uk_contextnet_512 (ContextNet): 512 каналів. Краща в термінах («старослов'янська»), але помилки в назвах і датах [26];
- nvidia/stt_ua_fastconformer_hybrid_large_pc (FastConformer): 18 шарів, гібридний CTC/RNNT. Висока точність назв («Кирило») і дат («1056-1057»), помилки в термінах («словка») [27];
- theodotus/stt_ua_fastconformer_hybrid_large_pc (FastConformer): найкраща, коректно відтворює терміни («старослов'янська»), назви («Остромирове»), дати («1056-1057»), граматику («друже») [28];
- theodotus/stt_uk_squeezeformer_ctc_sm (Squeezeformer): 12 шарів. Помилки в термінах («ябетки»), середня точність [29];
- theodotus/stt_uk_squeezeformer_ctc_ml (Squeezeformer): 16 шарів. Точніша, але помилки в числах і назвах [30];
- taras-sereda/uk-pods-conformer (Conformer): 18 шарів. Висока точність, незначні помилки («боголожіні») [31].

Узагальнений результат порівняльного аналізу всіх моделей для

розпізнавання тексту з аудіо українською мовою представлено у таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика моделей для розпізнавання тексту з українського аудіо

№	Модель	Точність термінів	Точність назв	Точність дат	Граматичні особливості	Час обробки	
						для аудіо	для відео
1	nvidia/stt_uk_ citrinet _1024_gamma_0_25	Середня	Низька	Низька	Частково збережені	3 хв 9 с	5 хв 14 с
2	neongeckocom/ stt_uk _citrinet_512_ gamma_0_25	Середня	Низька	Низька	Частково збережені	17 с	2 хв 2 с
3	theodotus/stt_uk_ contextnet_512	Вище середньої	Середня	Низька	Добре збережені	27 с	2 хв 15 с
4	nvidia/stt_ua_ fastconformer _hybrid_large_pc	Висока	Висока	Середня	Добре збережені	32 с	2 хв 21 с
5	theodotus/stt_ua_ fastconformer _hybrid_large_pc	Дуже висока	Дуже висока	Висока	Відмінно збережені	30 с	3 хв 3 с
6	theodotus/stt_uk_ squeezeformer _ctc_sm	Середня	Середня	Низька	Частково збережені	16 с	2 хв
7	theodotus/stt_uk_ squeezeformer _ctc_ml	Вище середньої	Середня	Низька	Добре збережені	31 с	2 хв 25 с
8	taras-sereda/uk-pods-conformer	Дуже висока	Висока	Висока	Відмінно збережені	52 с	2 хв 38 с

Отже, модель для обробки україномовного аудіо theodotus/stt_ua_fastconformer_hybrid_large_pc продемонструвала найкращі результати, забезпечуючи дуже високу точність термінів, назв, дат і відмінне збереження граматичних особливостей української мови. Вона оптимально балансує швидкість обробки (30 с для аудіо, 3 хв 3 с для відео) та якість транскрибування. Інші моделі поступаються за точністю або часом обробки, що робить theodotus/stt_ua_fastconformer_hybrid_large_pc найкращим вибором для розпізнавання українськомовного аудіо.

Для розпізнавання англомовного аудіо було обрано модель, яка має найбільше завантажень на сервісі Hugging Face [13]. Для перекладу тексту з української мови на англійську та перекладу тексту з англійської мови на

українську було використано дві моделі, які були відібрані аналогічним чином.

Модель для розпізнавання англомовного аудіо – nvidia/stt_en_conformer_transducer_large, для розпізнавання українськомовного аудіо – theodotus/stt_ua_fastconformer_hybrid_large_pc, для перекладу тексту з української мови на англійську – Helsinki-NLP/opus-mt-uk-en [32] та для перекладу тексту з англійської мови на українську – facebook/nllb-200-distilled-600M [33].

2.4 Інформаційне забезпечення

Проєктований API обробляє вхідну та вихідну інформації.

До вхідної інформації можна віднести:

а) користувацькі дані – ім’я, пароль, JWT-токен;

б) медіафайли – відео (.mp4, .avi, .mov, .mkv), аудіо (конвертуються в WAV);

в) параметри обробки – вихідна та цільова мови (uk/en), формат субтитрів (SRT, VTT, TXT);

г) ідентифікатор завдання (task_id).

Вихідна інформація API включає:

- результати автентифікації – токени доступу, тип токену, статуси запитів;
- результати обробки – відео з субтитрами, файли субтитрів (SRT, VTT, TXT), метадані (статус: processing/completed/failed, прогрес у %, повідомлення про помилки);
- історія та статистика – історія обробки користувача, статистика (кількість файлів, середній час обробки, розподіл за мовами).

У ході концептуального проектування бази даних для прикладного програмного інтерфейсу виділено три основні сутності:

- Користувач (User) – представляє зареєстрованого користувача API з такими полями сутності, як: ID користувача (PK), ім'я користувача, хешований пароль та дата створення облікового запису користувача;
- Завдання обробки (ProcessedFile) – представляє собою процес обробки медіафайлу. Сюди можна віднести такі поля: ID завдання (PK), ID користувача (FK), ідентифікатор завдання (task_id), назва файлу, статус (completed, failed),

вихідна мова, цільова мова, час обробки та мітка часу;

- Статус завдання (TaskStatus) – представляє поточний стан обробки для медіафайлу. Для цієї сутності можна вважати за потрібне наступні поля: ID завдання (PK), статус (processing, completed, failed), прогрес виконання, шлях до вихідного файлу та повідомлення про помилку.

Для інфологічної моделі можна віднести такі зв’язки між сутностями:

- Користувач (User) має зв'язок «один-до-багатьох» з Завданням обробки (ProcessedFile);

- Завдання обробки (ProcessedFile) має зв'язок «один-до-одного» з Статусом завдання (TaskStatus).

На рисунку 2.2 представлено ER-діаграму бази даних прикладного програмного інтерфейсу.

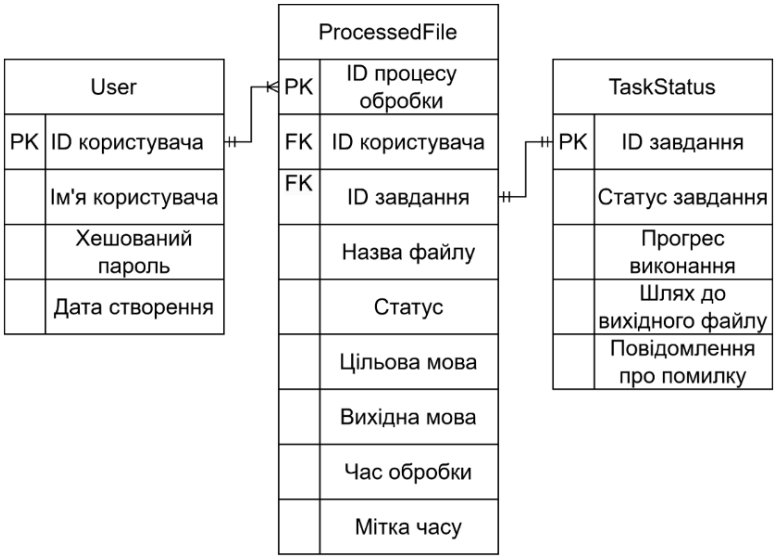


Рисунок 2.2 – ER-діаграма прикладного програмного інтерфейсу

У таблицях 2.2 – 2.4 представлені сутності бази даних прикладного програмного інтерфейсу.

Таблиця 2.2 – Сутність Користувач (User)

Ключ	Поле сутності	Опис сутності
PK	ID користувача	Унікальний ідентифікатор користувача
	Ім'я користувача	Логін або ім'я користувача
	Хешований пароль	Зашифрований пароль користувача
	Дата створення	Дата реєстрації користувача

Таблиця 2.3 Сутність Завдання обробки (ProcessedFile)

Ключ	Поле сутності	Опис сутності
PK	ID процесу обробки	Унікальний ідентифікатор процесу обробки файлу
FK	ID користувача	Зв'язок із користувачем, який запустив обробку
FK	ID завдання	Зв'язок із завданням у таблиці TaskStatus
	Назва файлу	Ім'я оброблюваного файлу
	Статус	Статус процесу обробки (помилка, завершено тощо)
	Цільова мова	Мова, на яку виконується обробка
	Вихідна мова	Мова оригіналу файлу
	Час обробки	Загальний час виконання обробки
	Мітка часу	Дата та час створення запису

Таблиця 2.4 – Сутність Статус завдання (TaskStatus)

Ключ	Поле сутності	Опис сутності
PK	ID завдання	Унікальний ідентифікатор завдання
	Статус завдання	Поточний статус завдання
	Прогрес виконання	Відсоток виконання завдання
	Шлях до файлу	Шлях до обробленого файлу
	Повідомлення про помилку	Текст повідомлення у випадку помилки

Нижче наведено опис зв’язків між таблицями у базі даних:

- зв’язок між таблицями User та ProcessedFile – один користувач може мати багато оброблених файлів. Це зв’язок один до багатьох (1:N);

- зв’язок між таблицями TaskStatus та ProcessedFile – одне завдання може бути пов’язане лише з одним процесом обробки файлу, і навпаки – кожен оброблений файл пов’язаний із певним завданням. Це зв’язок один до одного (1:1).

Таблиці фізичної моделі бази даних представлені кодом на рисунках 2.3 – 2.5.

```
CREATE TABLE users (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    username VARCHAR(255) UNIQUE NOT NULL,
    hashed_password VARCHAR(255) NOT NULL,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_username ON users(username);
```

Рисунок 2.3 – Представлення кодом сутності Users

```
CREATE TABLE processed_files (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    user_id TEXT NOT NULL,
    task_id TEXT NOT NULL,
    file_name TEXT NOT NULL,
    status TEXT NOT NULL,
    source_lang TEXT NOT NULL,
    target_lang TEXT NOT NULL,
    processing_time REAL,
    timestamp TEXT
);

CREATE INDEX idx_user_id ON processed_files(user_id);
CREATE INDEX idx_task_id ON processed_files(task_id);
```

Рисунок 2.4 – Представлення кодом сутності ProcessedFile

```
tasks_status = {
    task_id: {
        "status": status,
        "progress": progress,
        "error": error,
        "result_file": result_file
    }
}
```

Рисунок 2.5 – Представлення кодом сутності TaskStatus

База даних для прикладний програмний інтерфейс використовує SQLite для зберігання даних користувачів та історії обробки. Для взаємодії з базою даних використовується ORM SQLAlchemy для таблиці користувачів та прямі SQL-запити для таблиці історії обробки.

На рисунку 2.6 зображений код, який описує ініціалізацію бази даних у прикладному програмному інтерфейсі.

```
DATABASE_URL = "sqlite:///./users.db"
engine = create_engine(DATABASE_URL, connect_args={"check_same_thread": False})
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
Base = declarative_base()

class User(Base):
    tablename = "users"
    id = Column(Integer, primary_key=True, index=True)
    username = Column(String, unique=True, index=True)
    hashed_password = Column(String)
    created_at = Column(DateTime, default=datetime.utcnow)

Base.metadata.create_all(bind=engine)

def init_history_db():
    conn = sqlite3.connect('processing_history.db')
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS processed_files (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user_id TEXT NOT NULL,
            task_id TEXT NOT NULL,
            file_name TEXT NOT NULL,
            status TEXT NOT NULL,
            source_lang TEXT NOT NULL,
            target_lang TEXT NOT NULL,
            processing_time REAL,
            timestamp TEXT
        )
    ''')
    conn.commit()
    conn.close()

init_history_db()
```

Рисунок 2.6 – Код ініціалізації бази даних

Код ініціалізації бази даних складається з двох частин:

- налаштування бази даних користувачів за допомогою SQLAlchemy: створюється з’єднання з SQLite базою «users.db», визначається модель User (поля:

id, username, hashed_password, created_at). Таблиця створюється автоматично, якщо відсутня;

- ініціалізація бази для історії обробки файлів: функція init_history_db() підключається до «processing_history.db», створює таблицю processed_files (поля: id, user_id, task_id, file_name, status, source_lang, target_lang, processing_time, timestamp) і викликається при запуску програми.

На рисунку 2.7 представлений код, який описує функції для роботи з історією обробки у базі даних у прикладному програмному інтерфейсі.

```
def save_processing_history(user_id, task_id, file_name, status, source_lang, target_lang, processing_time):
    conn = sqlite3.connect('processing_history.db')
    cursor = conn.cursor()

    timestamp = datetime.now().isoformat()
    cursor.execute('''
        INSERT INTO processed_files (user_id, task_id, file_name, status, source_lang, target_lang, processing_time, timestamp)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)
    ''', (user_id, task_id, file_name, status, source_lang, target_lang, processing_time, timestamp))

    conn.commit()
    conn.close()

def get_user_history(user_id):
    conn = sqlite3.connect('processing_history.db')
    cursor = conn.cursor()

    cursor.execute('SELECT task_id, file_name, status, source_lang, target_lang, processing_time, timestamp FROM processed_files
    WHERE user_id = ?', (user_id,))
    history = cursor.fetchall()

    conn.close()
    return [{"task_id": row[0], "file_name": row[1], "status": row[2], "source_lang": row[3], "target_lang": row[4],
    "processing_time": row[5], "timestamp": row[6]} for row in history]

def get_processing_stats():
    conn = sqlite3.connect('processing_history.db')
    cursor = conn.cursor()

    cursor.execute('SELECT COUNT(*) FROM processed_files WHERE status = "completed"')
    total_files = cursor.fetchone()[0]

    cursor.execute('SELECT AVG(processing_time) FROM processed_files WHERE status = "completed" AND processing_time IS NOT NULL')
    avg_time = cursor.fetchone()[0] or 0

    cursor.execute('SELECT source_lang, target_lang, COUNT(*) FROM processed_files WHERE status = "completed" GROUP BY
    source_lang, target_lang')
    lang_distribution = cursor.fetchall()

    conn.close()
    return {
        "total_files": total_files,
        "avg_processing_time": round(avg_time, 2),
        "language_distribution": {f"{row[0]}→{row[1]}": row[2] for row in lang_distribution}
    }
```

Рисунок 2.7 – Код для роботи з історією обробки у базі даних

Ця частина коду визначає функції для роботи з базою даних історії обробки. Функція save_processing_history() додає запис про новий оброблений файл, зберігаючи всі передані параметри разом з поточним часом. Функція get_user_history() повертає історію обробки файлів конкретного користувача у вигляді списку словників, де кожен словник містить інформацію про одне завдання.

Функція get_processing_stats() збирає статистичні дані про всі оброблені файли, включаючи загальну кількість файлів, середній час обробки та розподіл мовних пар.

На рисунку 2.8 зображений код, який описує роботу зі статусом завдання, де використовується глобальний словник tasks_status та допоміжні функції у прикладному програмному інтерфейсі.

```
def update_task_status(task_id: str, status: str, progress: float = None, error: str = None,
result_file: str = None):
    tasks_status[task_id] = {
        "status": status,
        "progress": progress,
        "error": error,
        "result_file": result_file
    }
    logger.info(f"Task {task_id} status updated: {status} (progress: {progress}, error: {error})")

def fetch_task_status(task_id: str):
    return tasks_status.get(task_id, {})
```

Рисунок 2.8 – Код, який описує роботу зі статусом завдання, де використовується глобальний словник tasks_status та допоміжні функції

Остання частина коду відповідає за відстеження статусу завдань. Функція update_task_status() оновлює інформацію про завдання у глобальному словнику tasks_status, зберігаючи його поточний статус, прогрес, помилки та шлях до результуючого файлу. Крім того, функція записує інформацію про оновлення статусу в журнал. Функція fetch_task_status() дозволяє отримати поточний статус завдання за його ідентифікатором.

Загалом усі коди формують основу API, який обробляє файли, перекладаючи їх з однієї мови на іншу, відстежує прогрес обробки та зберігає історію для подальшого аналізу та звітності.

3 РЕАЛІЗАЦІЯ ПРИКЛАДНОГО ПРОГРАМНОГО ІНТЕРФЕЙСУ

3.1 Реалізація програмного забезпечення

Для забезпечення ефективної роботи сервісу було спроектовано модульну організацію API, яка враховує всі етапи обробки запитів та взаємодії з користувачами. Основна увага приділялася безпеці, масштабованості та зручності інтеграції з іншими сервісами.

Схема модульної організації коду API-додатку зображена на рисунку 3.1.

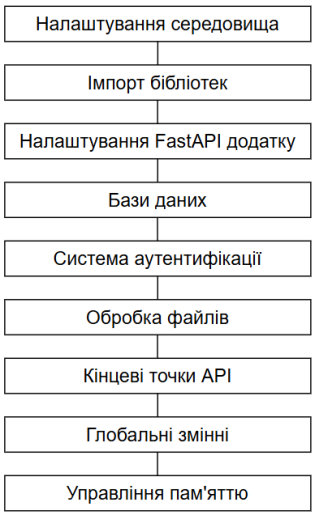


Рисунок 3.1 – Схема модульної організації коду API-додатку

API виконує модульну обробку медіа з налаштованим середовищем і FastAPI для безпечної взаємодії. Аутентифікація через OAuth2 та JWT захищає дані. Основна логіка використовує моделі NeMo і transformers для транскрипції, перекладу та генерації субтитрів. API легко масштабований, гнучкий і керує ресурсами через глобальні змінні та фонові задачі.

Налаштування середовища більш детально зображено на рисунку 3.2.

Налаштування середовища	
1	Встановлення шляхів кешу
2	Перевірка та створення директорій кешу
3	Налаштування шляху до ImageMagick

Рисунок 3.2 – Деталізована схема модуля «Налаштування середовища»

Налаштування середовища забезпечує правильну роботу API – встановлюються змінні для кешування NLP-моделей, створюються необхідні директорії, налаштовується шлях до ImageMagick для роботи з графікою. Всі дії виконуються до імпорту бібліотек для стабільної ініціалізації.

Схема модуля «Імпорт бібліотек» більш детально зображено на рисунку 3.3.

Імпорт бібліотек	
1	Основні: os, signal, pathlib, loggit g, uuid, gc, time
2	FastAPI та його компоненти
3	Обробка медіа: ffmpeg, pydub, mo viepy
4	NLP: transformers, nemo.colle ctions.asr
5	Безпека: OAuth2PasswordBe arer, CryptContext, jwt
6	Бази даних: sqlalchemy, sqlit e3

Рисунок 3.3 – Деталізована схема модуля «Імпорт бібліотек»

Імпорт бібліотек організований модульно: системні бібліотеки, FastAPI для HTTP, ffmpeg, pydub, moviepy та ImageMagick для медіаобробки, transformers і NeMo для транскрипції й перекладу. Безпека забезпечена OAuth2, bcrypt і JWT, дані зберігаються в SQLAlchemy та SQLite для надійності й масштабованості.

Схема модуля «Налаштування FastAPI додатку» більш детально зображено на рисунку 3.4.

Налаштування FastAPI додатку	
1	Створення екзмепляру app
2	Додавання CORS middlware
3	Налаштування логування

Рисунок 3.4 – Деталізована схема модуля «Налаштування FastAPI додатку»

Налаштування FastAPI включає створення екземпляра app, додавання CORS middleware для безпечної взаємодії та налаштування логування у файл і консоль. Це забезпечує безпеку, розширюваність і ефективний моніторинг API.

Схема модуля «База даних» більш детально зображено на рисунку 3.5.

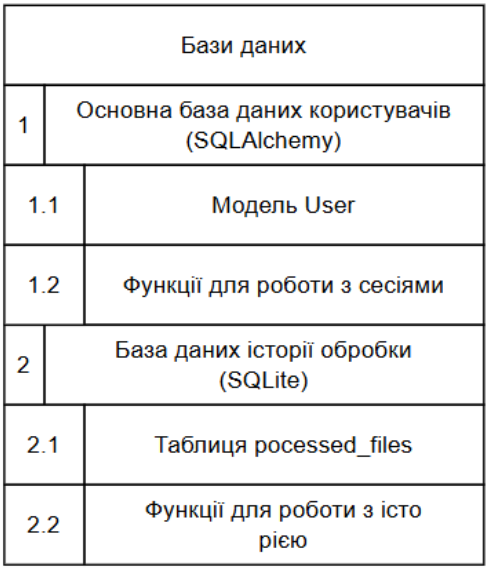


Рисунок 3.5 – Деталізована схема модуля «База даних»

SQLAlchemy ORM керує даними користувачів, SQLite зберігає історію обробки файлів. Модель User використовується для аутентифікації, історія зберігається в таблиці processed_files з метаданими та статусом. Реалізовані функції для збереження, отримання записів і статистики, що забезпечує безпеку та ефективне логування.

Схема модуля «Аутентифікації» більш детально зображено на рисунку 3.6.

Рисунок 3.6 – Деталізована схема модуля «Аутентифікація»

Для аутентифікації використовується OAuth2 і JWT для захисту API. Паролі хешуються через bcrypt, токени мають обмежений термін дії й перевіряються при кожному запиті. Реєстрація та вхід виконуються через окремі кінцеві точки, а FastAPI-залежності автоматизують перевірку доступу до захищених маршрутів.

Схема модуля «Обробка файлів» більш детально зображено на рисунку 3.7.

Обробка файлів	
1	Глобальні налаштування
1.1	BASE_DIR, TEMP_DIR
1.2	MAX_FILE_SIZE
2	Допоміжні функції
2.1	format_time
2.2	generate_subtitles
2.3	update_task_status
3	Основні функції обробки
3.1	process_video
3.1.1	Витягнення аудіо (FFmpeg)
3.1.2	Транскрипція (ASR) - залежить від мови:
3.1.2.1	uk: theodotus/stt_ua_fastconformer_hybrid_large_pc (NeMo)
3.1.2.2	en: nvidia/stt_en_conformer_transducer_large (NeMo)
3.1.3	Переклад (якщо потрібно)
3.1.3.1	uk→en: Helsinki-NLP/opus-mt-uk-en (transformers)
3.1.3.2	en→uk: facebook/nllb-200-distilled-600M (transformers)
3.1.4	Генерація субтитрів (формати: srt, vtt, txt)
3.1.5	Накладання субтитрів на відео (FFmpeg)
4.1	process_audio_to_text
4.1.1	Конвертація аудіо (FFmpeg -> mono WAV 16kHz)
4.1.2	Транскрипція (ASR) - залежить від мови:
4.1.2.1	uk: theodotus/stt_ua_fastconformer_hybrid_large_pc (NeMo)
4.1.2.2	en: nvidia/stt_en_conformer_transducer_large (NeMo)
4.1.3	Переклад (якщо потрібно)
4.1.3.1	uk→en: Helsinki-NLP/opus-mt-uk-en (transformers)
4.1.3.2	en→uk: facebook/nllb-200-distilled-600M (transformers)
4.1.4	Генерація текстових субтитрів(формати: srt, vtt, txt)

Рисунок 3.7 – Деталізована схема модуля «Обробка файлів»

Модуль API автоматизує транскрипцію, переклад і створення субтитрів з аудіо та відео. Вона має окремі функції для обробки відео й аудіо, використовує моделі NVIDIA NeMo, Helsinki-NLP і NLLB для розпізнавання та перекладу, а субтитри генерує у форматах srt, vtt або txt.

Схема модуля «Кінцеві точки API» більш детально зображено на рисунку 3.8.

Кінцеві точки API	
1	/upload-video (POST) - завантаження відео
2	/upload-audio (POST) - завантаження аудіо
3	/task-status/{task_id} (GET) - перевірка статусу
4	/download/{task_id} (GET) - завантаження результату
5	/history (GET) - історія обробки
6	/stats (GET) - статистика

Рисунок 3.8 – Деталізована схема модуля «Кінцеві точки API»

Веб-сервіс автоматизує обробку аудіо- та відеофайлів: завантаження, транскрипцію, переклад і створення субтитрів. API містить ключові кінцеві точки для обробки, перегляду історії та статистики. Використовуються FFmpeg, NeMo, Helsinki-NLP і Facebook-моделі, а результати зберігаються у форматах srt, vtt або txt з відстеженням через task_id.

Схема модуля «Глобальні змінні» більш детально зображено на рисунку 3.9.

Глобальні змінні	
1	tasks_status - словник для відстеження статусів завдань
2	моделі перекладу (translator_uk_en, translator_en_uk)

Рисунок 3.9 – Деталізована схема модуля «Глобальні змінні»

Глобальні змінні координують ключові процеси API. tasks_status відстежує стан завдань, а перекладачі translator_uk_en і translator_en_uk (на основі Helsinki-NLP і Facebook) зберігаються глобально для підвищення продуктивності. Це забезпечує стабільну й ефективну роботу сервісу.

Схема модуля «Управління пам’яттю» більш детально зображено на рисунку 3.10.

Управління пам'яттю	
1	Видалення великих об'єктів (asr_model, translator)
2	Виклик gc.collect()

Рисунок 3.10 – Деталізована схема модуля «Управління пам’яттю»

Для ефективної обробки медіафайлів важливо оптимізувати використання пам’яті, оскільки моделі розпізнавання мовлення та перекладу споживають багато ресурсів. З цією метою реалізовано видалення моделей з пам’яті після використання та виклик gc.collect() у ключові моменти. Це запобігає витокам пам’яті й забезпечує стабільну роботу сервісу, особливо при використанні ресурсомістких моделей NeMo та перекладачів.

Отже, спроєктована модульна організація API є стабільною та масштабованою, з чітким розподілом відповідальностей і сучасними механізмами безпеки. Вона підтримує обробку аудіо, відео та переклад між українською й англійською, ефективно використовує ресурси та придатна для промислового застосування.

UML-діаграма класів, яка наведена у додатку Д, охоплює ключові компоненти API. AuthenticationService відповідає за автентифікацію, взаємодіє з класом User. FastAPI використовується для обробки HTTP-запитів через кінцеві точки API (реєстрація, логін, завантаження, статус, історія). ProcessingService забезпечує обробку медіа, створення субтитрів і відстеження задач, результати яких зберігаються в TaskStatus. DatabaseService та HistoryService відповідають за збереження даних і статистику. Класи Token, UserCreate та ProcessingResponse підтримують автентифікацію й обробку. Прикладний програмний інтерфейс працює з кількома мовами та форматами субтитрів.

UML-діаграма станів, яка наведена у додатку Е, описує процес обробки файлів українською мовою: після завантаження файл зберігається, далі запускається фонове завдання. Обробка розгалужується на відео (витягнення аудіо) та аудіо (конвертація в моно), обидва шляхи зливаються на етапі транскрибації, потім відбувається сегментація, переклад, генерація та рендеринг субтитрів. Завершення – це успішна обробка або перехід у стан помилки при збої на будь-якому етапі.

Для прикладного програмного інтерфейсу було використано наступні програмні засоби: Python [10], FastAPI [11], Nemo ASR [12], Transformers (Hugging Face) [13], FFmpeg [14], SQLite [15], Pydantic [16], SQLAlchemy [17], Pydub [18], MoviePy [19] та SilieroVad [20].

FastAPI [11] забезпечує високопродуктивний API із швидкою обробкою запитів і автоматичною документацією, що сприяє ефективній взаємодії з фронтендом. Для розпізнавання мовлення в аудіофайлах застосовано NVIDIA NeMo [12], яке підтримує українську та англійську мови з високою точністю. Переклад транскрибованих сегментів реалізовано за допомогою бібліотеки Transformers [13]. FFmpeg [14] використовується для витягування аудіо з відео та накладання субтитрів завдяки підтримці численних форматів. SQLite [15], слугує легкою вбудованою базою даних для зберігання інформації про користувачів і оброблені файли, а SQLAlchemy [17] полегшує взаємодію з нею через ORM. Для валідації вхідних даних застосовується Pydantic [17], що підвищує надійність API. Обробку аудіо (наприклад, конвертацію в моно та сегментацію) реалізовано за допомогою Pydub [18], а редагування відео – за допомогою MoviePy [19]. Для виявлення мовлення використано SileroVAD [20], який точно визначає часові межі мовних сегментів.

Отже, для створення API використано сучасні інструменти на базі Python, що дозволили реалізувати обробку мовних, аудіо- та відеоданих і стабільну взаємодію з користувачем.

Основна програмна логіка прикладного програмного інтерфейсу для транскрипції та субтитрів аудіо- та відеофайлів реалізована у функціях process_video і process_audio_to_text, а також у моделях даних, таких як ProcessingResponse, TaskStatus і User. Лістинг відповідного коду подано в додатках Б, В та Г.

Функція process_video (додаток Б) відповідає за обробку відеофайлів. Вона виконує наступні етапи:

- витягнення аудіо відбувається за допомогою бібліотеки FFmpeg із відеофайлу витягується аудіодоріжка у форматі WAV із частотою дискретизації 16 кГц;

- транскрипція на цьому етапі аудіо розбивається на фрагменти тривалістю 30 секунд, які транскрибуються за допомогою моделей Nemo ASR. Для української мови використовується модель theodotus/stt_ua_fastconformer_hybrid_large_pc, для англійської – nvidia/stt_en_conformer_transducer_large. Виявлення мовних сегментів

здійснюється за допомогою бібліотеки Silero VAD;

- переклад відбувається, якщо вихідна мова відрізняється від цільової, текст перекладається за допомогою моделей Transformers (Helsinki-NLP/opus-mt-uk-en для українсько-англійського перекладу або facebook/nllb-200-distilled-600M для англійсько-українського);

- генерація субтитрів тут транскрибований і перекладений текст форматується у файли субтитрів (SRT, VTT або TXT) за допомогою функції generate_subtitles;

- накладання субтитрів на цьому етапі субтитри додаються до вихідного відео за допомогою FFmpeg із використанням кодека libx264 для оптимізації розміру файлу.

Функція інтегрується з FastAPI для асинхронної обробки великих файлів у фоновому режимі, оновлює статус через update_task_status і зберігає історію в базі даних за допомогою save_processing_history (додаток Г). Функція process_audio_to_text (додаток В) генерує текст або субтитри для аудіофайлів, використовуючи ті ж моделі, що й process_video, але без обробки відео, що зменшує ресурсоємність. Моделі Pydantic (ProcessingResponse, TaskStatus) (додаток Г) структурують відповіді API, забезпечуючи валідацію та типізацію для статусу обробки й результатів. Модель User (SQLAlchemy) (додаток Г) зберігає дані користувачів (ідентифікатор, ім’я, хеш пароля, дата створення) для автентифікації через JWT і зв’язку з історією обробки. Функція save_processing_history (додаток Г) фіксує в базі SQLite дані про обробку (користувач, задача, файл, статус, мови, час, дата) для аналізу через кінцеву точку API /history.

3.2 Інтерфейс користувача

React.js [21] використано для створення інтерактивного інтерфейсу з модульною структурою, що забезпечує зручну взаємодію з API та динамічне оновлення UI.

На рисунку 3.11 зображено вигляд сторінки для входу або реєстрації в обліковий запис.

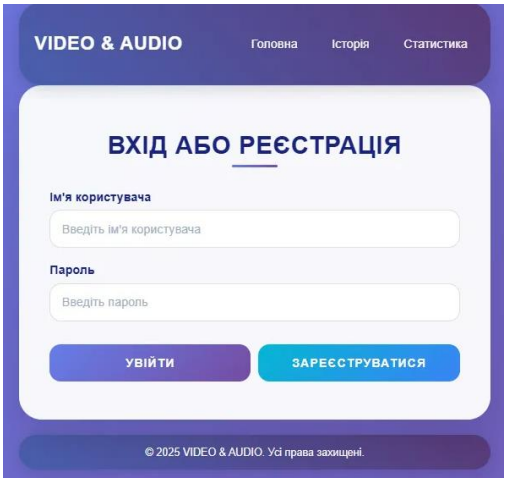


Рисунок 3.11 – Вигляд сторінки входу або реєстрації в обліковий запис

Інтерфейс Video & Audio виконаний у фіолетово-синій гамі з градієнтним фоном і білим контентним блоком із заокругленими кутами. Верхнє меню має фіолетовий фон, білий текст, логотип ліворуч і навігацію праворуч («Головна», «Історія», «Статистика»). Екран входу/реєстрації — мінімалістичний: темно-синій заголовок, світло-сірі поля, кнопки «Увійти» (синя) та «Зареєструватися» (блакитна) з білим текстом і заокругленими кутами. У футері — напис про авторські права.

Як виглядає головна сторінка API можна побачити на рисунку 3.12.

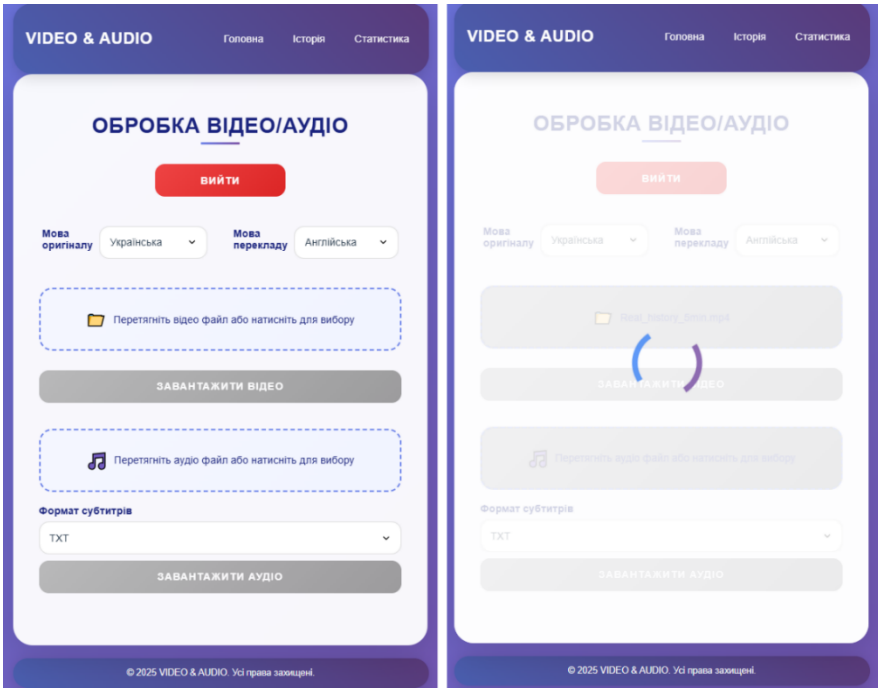


Рисунок 3.12 – Дизайн головної сторінки API (зліва сторінка без обробки файлів; справа – під час обробки файлів)

Екран обробки медіа має інтуїтивний дизайн у фіолетово-синій гамі. Містить заголовок, червону кнопку виходу, випадаючі списки для вибору мов і формату субтитрів, області завантаження відео та аудіо з відповідними іконками, сірі кнопки завантаження, а також анімований індикатор прогресу. Після завершення з'являється голуба кнопка «Завантажити результат». Інтерфейс адаптивний, зручний для сенсорних екранів і включає візуальні ефекти наведення та фокусу. Дизайн сучасний, простий і естетичний.

Отже, дизайн API VIDEO & AUDIO базується на REST-принципах із групуванням кінцевих точок за функціональними блоками: аутентифікація, обробка медіа, історія, статистика. Підтримує безпечний обмін даними, асинхронність, фільтрацію та багатомовність. Навігація відображає структурований поділ API, а дизайн відповідає сучасним стандартам і єдиному стилю.

3.3 Тестування розробленого прикладного програмного інтерфейсу

Весь функціонал прикладного програмного інтерфейсу було перевірено на наступних сценаріях тестування: реєстрація користувача; автентифікація користувача; завантаження відео з транскрипцією та перекладом; завантаження аудіо для транскрипції; перевірка історії обробки та тестування на помилку (завантаження занадто великого файлу).

Тестування за сценарієм «реєстрація користувача»:

- мета – перевірити, чи кінцева точка API /register/ створює нового користувача;
- вхідні дані – «username»: «testuser2025», «password»: «testpass2025»;
- очікуваний результат – код 201, відповідь із username і id, запис у базі users.db;
- фактичний результат – користувач успішно зареєстрував обліковий запис;
- статус – пройдено.

Детальні результати цього тестування представлено на рисунку Ж.1.

Тестування за сценарієм «автентифікація користувача»:

а) мета – перевірити, чи кінцева точка API /token/ повертає JWT-токен;

б) вхідні дані – «username»: «testuser2025», «password»: «testpass2025»;

в) очікуваний результат – код 200, відповідь із access_token;

г) фактичний результат – користувач успішно авторизувався на веб-сторінці;

д) статус – пройдено.

Детальні результати цього тестування представлено на рисунку И.1.

Тестування за сценарієм «завантаження відеофайлу з транскрипцією та перекладом»:

- мета – перевірити обробку відеофайлу через /upload-video/;
- вхідні дані – MP4-файл (українська аудіодоріжка), параметри source_lang='uk', target_lang='en', subtitle_format='srt';
- очікуваний результат – код 200, відповідь із task_id і повідомленням «Video uploaded successfully»; після завершення обробки – відео з англійськими субтитрами у форматі SRT, запис у processing_history.db;
- фактичний результат – користувач отримав відео з англійськими субтитрами. Час обробки тривав 2 хвилини та 32 секунди;
- статус – пройдено.

Детальні результати цього тестування представлено на рисунку К.1-К.4.

Тестування за сценарієм «завантаження аудіофайлу для транскрипції»:

а) мета – перевірити обробку аудіофайлу через /upload-audio/;

б) вхідні дані – MP3-файл (англійська мова), параметри source_lang='en', target_lang='uk', subtitle_format='txt';

в) очікуваний результат – код 200, відповідь із task_id; після обробки – текстовий файл із українськими субтитрами. Час обробки тривав 1 хвилину та 6 секунд;

г) фактичний результат – користувач отримав текстовий файл з англійськими субтитрами у форматі .txt;

д) статус – пройдено.

Детальні результати цього тестування представлено на рисунках Л.1-Л.4.

Тестування за сценарієм «перевірка історії обробки»:

- мета – перевірити, чи кінцева точка API /history повертає історію обробки;

- вхідні дані – GET-запит із токеном користувача;
- очікуваний результат – код 200, список оброблених файлів із task_id, file_name, status, source_lang, target_lang;
- фактичний результат – на сторінці «Історія» користувач може переглянути усі записи з обробки файлів;
- статус – пройдено.

На рисунку 3.13 зображено результат проходження тестування за сценарієм «перевірка історії обробки».

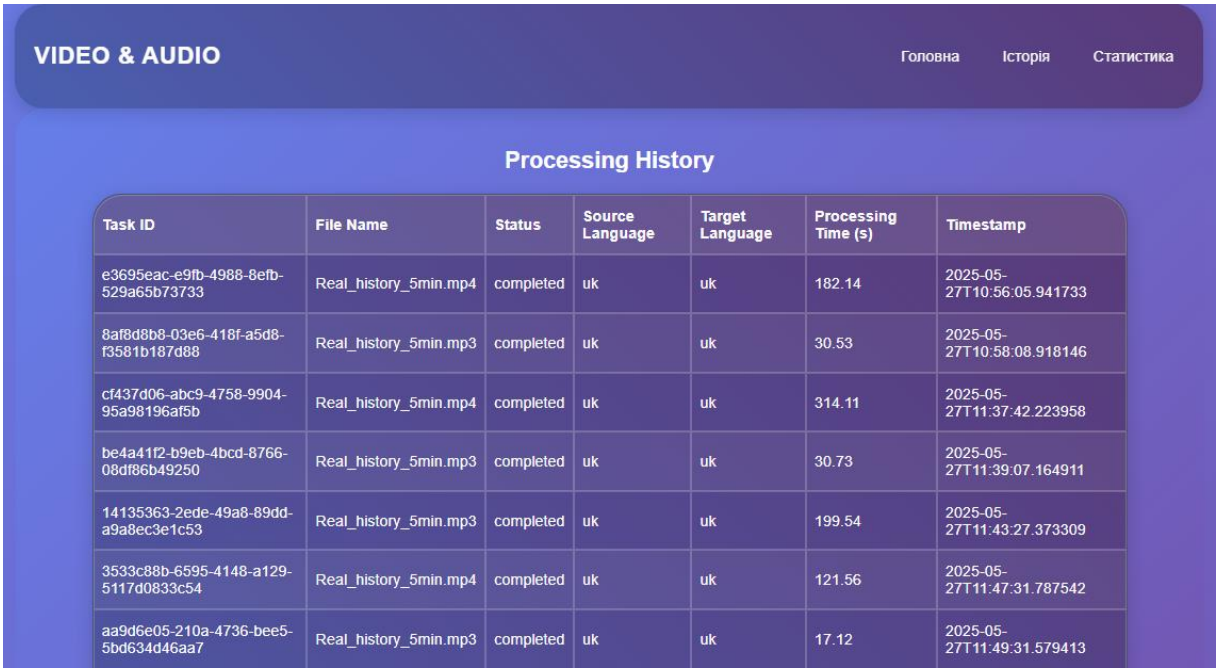


Рисунок 3.13 – Сторінка з історією усіх оброблених файлів користувачем

Тестування за сценарієм «тестування на помилку (завантаження занадто великого файлу)»:

а) мета – перевірити обробку файлу, що перевищує ліміт (5 ГБ);

б) вхідні дані – MP4-файл розміром більше 5 ГБ;

в) очікуваний результат – код 413, повідомлення «File too large!»;

г) фактичний результат – у веб-інтерфейсі користувач отримав помилку «File too large!» з кодом 413;

д) статус – пройдено.

Детальні результати цього тестування представлено на рисунку М.1.

Узагальнені результати тестування сценаріїв можна побачити у таблиці 3.1.

Таблиця 3.1 – Узагальнені результати тестування API

№	Назва сценарію тестування	Мета	Вхідні дані	Очікуваний результат	Фактичний результат
1	Реєстрація користувача	Перевірити, чи кінцева точка API /register/ створює нового користувача	«username»: «testuser2025», «password»: «testpass2025»	Код 201, відповідь із username і id, запис у базі users.db	Користувач успішно зареєстрував обліковий запис
2	Автентифікація користувача	Перевірити, чи кінцева точка API /token/ повертає JWT-токен	«username»: «testuser2025», «password»: «testpass2025»	Код 200, відповідь із access_token	Користувач успішно авторизувався на веб-сторінці
3	Завантаження відео з транскрипцією та перекладом	Перевірити обробку відеофайлу через /upload-video/	MP4-файл (українська аудіодоріжка), параметри source_lang='uk', target_lang='en', subtitle_format='srt'	Код 200, відповідь із task_id і повідомленням «Video uploaded successfully»; після завершення обробки – відео з англійськими субтитрами у форматі SRT, запис у processing_history.db	Користувач отримав відео з англійськими субтитрами. Час обробки тривав 2 хвилини та 32 секунди
4	Завантаження аудіо для транскрипції	Перевірити обробку аудіофайлу через /upload-audio/	MP3-файл (англійська мова), параметри source_lang='en', target_lang='uk', subtitle_format='txt'	Код 200, відповідь із task_id; після обробки – текстовий файл із українськими субтитрами. Час обробки тривав 1 хвилину та 6 секунд	Користувач отримав текстовий файл з англійськими субтитрами у форматі .txt
5	Перевірка історії обробки	Перевірити, чи кінцева точка API /history повертає історію обробки	GET-запит із токеном користувача	Код 200, список оброблених файлів із task_id, file_name, status, source_lang, target_lang	На сторінці «Історія» користувач може переглянути усі записи з обробки файлів;
6	Тестування на помилку (завантаження занадто великого файлу)	Перевірити обробку файлу, що перевищує ліміт (5 ГБ)	MP4-файл розміром більше 5 ГБ	код 413, повідомлення «File too large!»	У веб-інтерфейсі користувач отримав помилку «File too large!» з кодом 413

Отже, тестування прикладний програмний інтерфейс підтвердило його працездатність і відповідність вимогам. Успішно виконані сценарії реєстрації, автентифікації, обробки відео- та аудіофайлів, збереження історії та обробки помилок (наприклад, для файлів >5 ГБ). API коректно транскрибує українську та англійську мови, перекладає текст, створює субтитри (SRT, TXT) і накладає їх на відео. Час обробки: 1 хв 6 с для аудіо, 2 хв 32 с для відео. Обмеження – тривалість обробки великих файлів, що потребує оптимізації. Прикладний програмний інтерфейс готовий для автоматизації транскрипції та субтитрування.

3.4 Результати порівняння роботи прикладного програмного інтерфейсу з існуючими аналогами

Порівнюємо спроектований прикладний програмний інтерфейс з сервісами-аналогами. Результати цього аналізу для україномовного аудіо:

а) сервіс HappyScribe – висока точність, правильні терміни («старослов'янська»), назви («Кирило»), граматика збережена, помилка в даті («2022» замість «865»);

б) сервіс Sonix – найвища точність, мінімальні помилки, коректні терміни і назви, помилка в даті («2022»);

в) сервіс Kapwing – низька точність, помилки («Росії» замість «Русі», «виникіт»), але дата «865» правильна;

г) спроектований прикладний програмний інтерфейс – висока точність, правильна дата («865»), помилки в словах («братки», «Мифоді»).

Результат порівняння прикладного програмного інтерфейсу та сервісів-аналогів можна побачити у таблиці 3.2.

Таблиця 3.2 – Порівняльна характеристика спроектованого API та сервісів-аналогів для розпізнавання тексту з українського аудіо

Критерій	HappyScribe	Sonix	Kapwing	Спроектований API
Точність слів і фраз	Висока, дрібні помилки в пунктуації	Дуже висока, мінімальні помилки	Низька, багато помилок («Росії», «виникіт»)	Висока, але є помилки («братки», «Владір»)

Продовження таблиці 3.2

Критерій	HappyScribe	Sonix	Kapwing	Спроектований API
Коректність термінів	Правильні, але помилка в даті («2022»)	Правильні, помилка в даті («2022»)	Неконсистентні («слов'янська», «великомуравський»)	Правильні, коректна дата («865»)
Відсутність помилок у назвах/числах	Помилка в даті («2022»), назви правильні	Помилка в даті («2022»), назви правильні	Помилки в «Кирилу», «Фокко», але дата «865» правильна	Помилки в «Мифоді», «Константинок», числа точні
Збереження граматики (кличний, повноголосся)	Повністю збережено	Повністю збережено	Частково збережено, помилки («Фокко»)	Збережено, але помилка («фоко»)
Час обробки для відео	2 хв 38 с	4 хв 50 с	2 хв 47 с	3 хв 3 с
Час обробки для аудіо	1 хв 38 с	3 хв 5 с	2 хв 10 с	30 с

Отже, спроектований API демонструє високу точність у відтворенні слів, термінів і чисел, перевершуючи HappyScribe і Kapwing, але поступаючись Sonix. Основні недоліки – окремі помилки у власних назвах і граматиці. Водночас API значно швидший у обробці аудіо та відео за аналог Sonix AI, що робить його ефективним рішенням для швидкого розпізнавання контенту. Для досягнення точності Sonix потрібне подальше вдосконалення.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було:

1. Проаналізовано предметну область, що дозволило визначити актуальність задачі автоматичної транскрипції та субтитрування у контексті сучасного цифрового середовища. Виявлено ключові переваги таких технологій, як підвищення доступності контенту, полегшення пошуку та автоматизація рутинних завдань.

2. Досліджено існуючі рішення (Kapwing, HappyScribe, Sonix), що дозволило виявити їхні недоліки: обмежений безкоштовний доступ, наявність водяних знаків, зниження якості відео, відсутність текстових субтитрів. Вказані обмеження були враховані при формуванні функціоналу власного API.

3. Узагальнено переваги і недоліки різних підходів до транскрипції та субтитрування. Встановлено, що моделі ШІ забезпечують високу точність, але залишаються чутливими до шумів, акцентів та спеціалізованої термінології. На основі аналізу сформульовано концепцію створення API з багатомовною підтримкою, орієнтацією на інтеграцію та відкритим доступом до субтитрів.

4. Реалізовано модульну організацію прикладного програмного інтерфейсу з використанням FastAPI та React.js. API включає модулі для обробки відео та аудіо, генерації субтитрів, авторизації користувачів, логування та збереження історії. Такий підхід забезпечує масштабованість, асинхронність і зручність інтеграції.

5. Відібрано оптимальні моделі для обробки природної мови: для української мови – theodotus/stt_ua_fastconformer_hybrid_large_pc, для англійської – nvidia/stt_en_conformer_transducer_large, для перекладу – Helsinki-NLP/opus-mt-uk-en та facebook/nllb-200-distilled-600M. Це забезпечило високу точність транскрипції й перекладу.

6. Розроблено алгоритмічне та інформаційне забезпечення API: створено ER-діаграму бази даних, описано структуру таблиць, реалізовано функції для обробки статусів завдань, логування та проведення статистичного аналізу.

7. Розроблено API на FastAPI з підтримкою транскрипції, перекладу, субтитрів, логування та авторизації. Реалізовано повний цикл обробки медіа

(FFmpeg, SileroVAD, NeMo, Transformers), систему керування ресурсами, кешування моделей, логіку обробки завдань та базу даних із ER-діаграмою.

8. Створено веб-інтерфейс на React.js з адаптивним дизайном для завантаження медіа, відстеження прогресу та перегляду результатів. Забезпечено інтеграцію з API, збереження історії обробок і логування через SQLite.

9. Проведено тестування API, яке підтвердило ефективність реалізованого рішення. Здійснено порівняльний аналіз з аналогами, що виявив переваги розробленого прикладного програмного інтерфейсу за точністю розпізнавання, зручністю використання, гнучкістю форматів виводу та відсутністю обмежень безкоштовної версії. Результати апробації були представлені на науково-практичній конференції (додаток Н).

Виконана кваліфікаційна робота засвідчила ефективність обраного підходу до реалізації системи автоматичної транскрипції та субтитрування. Розроблений API відповідає сучасним вимогам цифрової обробки мультимедійного контенту, поєднує точність, зручність, відкритість та гнучкість, що робить його придатним для подальшого практичного використання та розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jensen C. T. APIs for Dummies. Hoboken, NJ: John Wiley & Sons, Inc., 2015. 56 p.

2. Cooksey B. An Introduction to APIs. Zapier, Inc., 2014. 60 p.

3. Dresing T., Pehl T., Schmieder C. Manual (on) Transcription: Transcription Conventions, Software Guides and Practical Hints for Qualitative Researchers. 3rd English ed. Marburg: Self-published, 2015. 92 p.

4. Ayonghe L. S. Fundamentals of Subtitling: Concepts, Theory and Practice. Yaoundé: Éditions CLÉ, 2021. 150 p.

5. Subtitle File Formats Overview – Clideo. URL: https://clideo.com/resources/subtitle-file-formats-overview (дата звернення: 15.03.2025).

6. What is Artificial Intelligence (AI) | Google Cloud. URL: https://cloud.google.com/learn/what-is-artificial-intelligence (дата звернення: 18.03.2025).

7. Kapwing – Create more content in less time. URL: https://www.kapwing.com/ (дата звернення: 22.03.2025).

8. Happy Scribe: Audio Transcription & Video Subtitles. URL: https://www.happyscribe.com/ (дата звернення: 22.03.2025).

9. Automatically convert audio and video to text. URL: https://sonix.ai/ (дата звернення: 22.03.2025).

10. Das U., Lawson A., Mayfield C., Norouzi N. Introduction to Python Programming. Houston: OpenStax, 2024. 406 p.

11. Lubanovic B. FastAPI: Modern Python Web Development. Sebastopol: O’Reilly Media, 2023. 277 p.

12. Overview – NVIDIA NeMo Framework User Guide. URL: https://docs.nvidia.com/nemo-framework/user-guide/latest/overview.html (дата звернення: 05.04.2025).

13. Transformers. URL: https://huggingface.co/docs/transformers/index (дата звернення: 05.04.2025).

14. About FFmpeg. URL: https://www.ffmpeg.org/about.html (дата звернення: 05.04.2025).

15. SQLite Home Page. URL: https://sqlite.org/ (дата звернення: 05.04.2025).

16. Welcome to Pydantic – Pydantic. URL: https://docs.pydantic.dev/latest/ (дата звернення: 05.04.2025).

17. SQLAlchemy – The Database Toolkit for Python. URL: https://www.sqlalchemy.org/ (дата звернення: 05.04.2025).

18. Exploring the Pydub Library: A Comprehensive Guide to Audio. URL: https://mjghadge9007.medium.com/exploring-the-pydub-library-a-comprehensive-guide-to-audio-manipulation-in-python-46a09c96f69b# (дата звернення: 05.04.2025).

19. MoviePy documentation - MoviePy documentation. URL: https://zulko.github.io/moviepy/ (дата звернення: 05.04.2025).

20. Github - snakers4/silero-vad: Silero VAD - pre-trained enterprise-grade Voice Activity Detector. URL: https://github.com/snakers4/silero-vad (дата звернення: 05.04.2025).

21. Minnick, C. Beginning ReactJS Foundations: Building User Interfaces with ReactJS. – Hoboken: John Wiley & Sons, 2022. – 512 с. – ISBN 978-1-119-68554-8.

22. speech-recognition-uk UA Speech Recognition & Synthesis. URL: https://github.com/egorsmkv/speech-recognition-uk?tab=readme-ov-file (дата звернення: 14.04.2025).

23. Невже російська – лише говірка української мови? Реальна історія. URL: https://www.youtube.com/watch?v=kJVmZ7WVVWo&t=8s (дата звернення: 21.04.2025).

24. nvidia/stt_uk_citrinet_1024_gamma_0_25 – Hugging Face. URL: https://huggingface.co/nvidia/stt_uk_citrinet_1024_gamma_0_25 (дата звернення: 21.04.2025).

25. neongeckocom/stt_uk_citrinet_512_gamma_0_25 – Hugging Face. URL: https://huggingface.co/neongeckocom/stt_uk_citrinet_512_gamma_0_25 (дата звернення: 21.04.2025).

26. theodotus/stt_uk_contextnet_512 – Hugging Face. URL: https://huggingface.co/theodotus/stt_uk_contextnet_512 (дата звернення: 21.04.2025).

27. nvidia/stt_ua_fastconformer_hybrid_large_pc – Hugging Face. URL: .https://huggingface.co/nvidia/stt_ua_fastconformer_hybrid_large_pc (дата звернення: 21.04.2025).

28. theodotus/stt_ua_fastconformer_hybrid_large_pc – Hugging Face. URL: https://huggingface.co/theodotus/stt_ua_fastconformer_hybrid_large_pc (дата звернення: 21.04.2025).

29. theodotus/stt_uk_squeezeformer_ctc_sm – Hugging Face. URL: https://huggingface.co/theodotus/stt_uk_squeezeformer_ctc_sm (дата звернення: 21.04.2025).

30. theodotus/stt_uk_squeezeformer_ctc_ml – Hugging Face. URL: https://huggingface.co/theodotus/stt_uk_squeezeformer_ctc_ml (дата звернення: 21.04.2025).

31. taras-sereda/uk-pods-conformer – Hugging Face. URL: https://huggingface.co/taras-sereda/uk-pods-conformer (дата звернення: 21.04.2025).

32. Helsinki-NLP/opus-mt-uk-en – Hugging Face. URL: https://huggingface.co/Helsinki-NLP/opus-mt-uk-en (дата звернення: 29.04.2025).

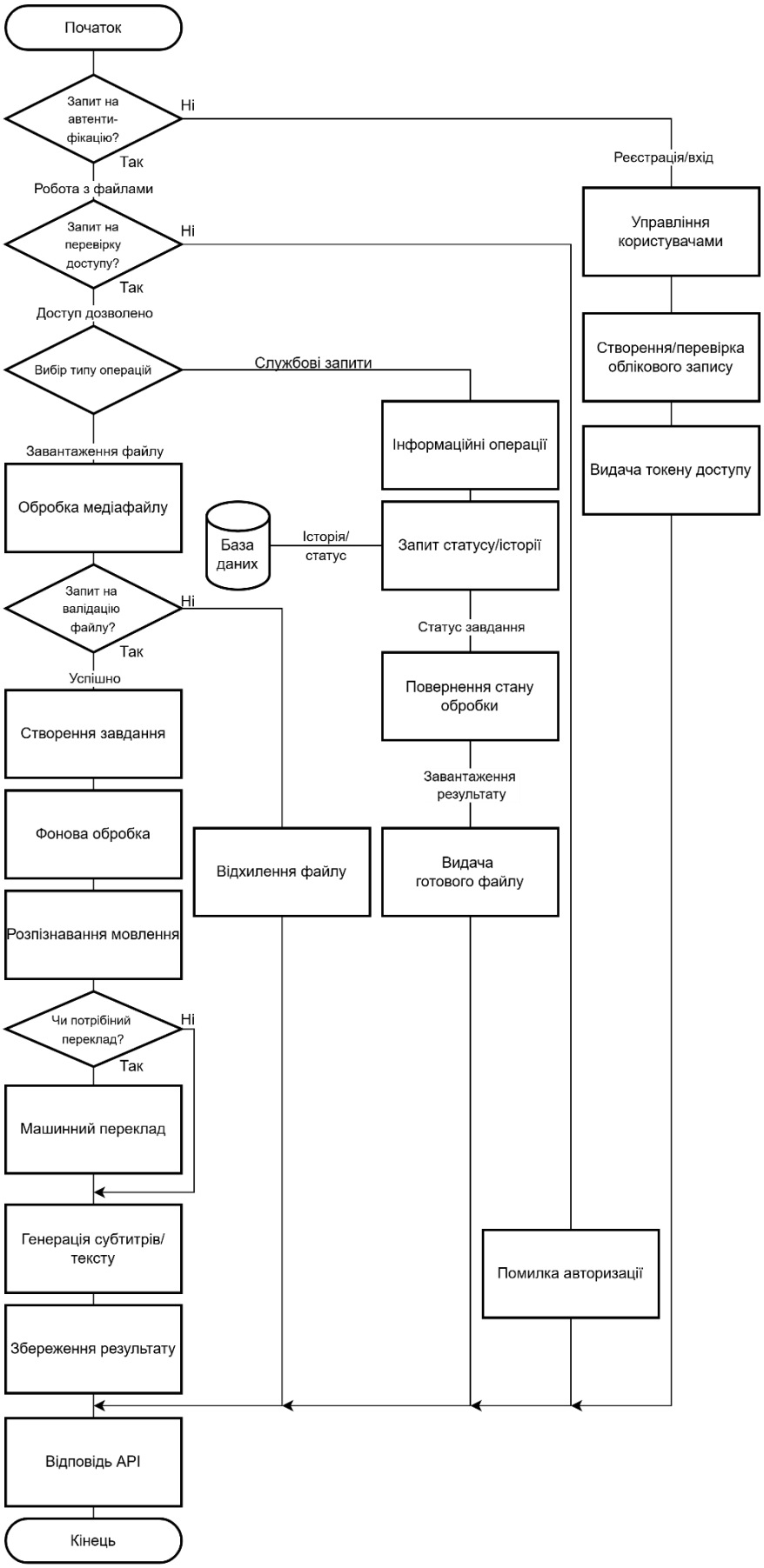
33. facebook/nllb-200-distilled-600M – Hugging Face. URL: https://huggingface.co/facebook/nllb-200-distilled-600M (дата звернення: 29.04.2025).

34. Герцій В.В., Турченко І.В. Прикладний програмний інтерфейс для транскрипції та субтитрів для аудіо та відеофайлів на основі штучного інтелекту. *Інтелектуальні інформаційні технології в прикладних дослідженнях* (*IITAR-2025*): збірник тез доповідей студентської науково-практичної конференції м. Тернопіль: ЗУНУ, 2025. 138-141 с.

35. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

Додаток А

Схема алгоритму роботи прикладного програмного інтерфейсу



Додаток Б

Код модуля обробки відео

```
async def process_video(task_id: str, video_path: str, source_lang: str = 'uk', target_lang: str = 'en',
subtitle_format: str = 'srt', user_id: str = None):
    """
    Асинхронна функція для обробки відеофайлу з транскрипцією, перекладом і накладанням субтитрів.
    Функція витягує аудіо з відео за допомогою FFmpeg, транскрибує його за допомогою моделей Nemo ASR,
    перекладає текст (якщо потрібно) за допомогою моделей Transformers, генерує субтитри у вибраному форматі
    (SRT, VTT або TXT) і накладає їх на відео через FFmpeg.
    Args:
        task_id (str): Унікальний ідентифікатор задачі.
        video_path (str): Шлях до відеофайлу.
        source_lang (str): Мова джерела ('uk' для української, 'en' для англійської).
        target_lang (str): Цільова мова ('uk' для української, 'en' для англійської).
        subtitle_format (str): Формат субтитрів ('srt', 'vtt' або 'txt').
        user_id (str): Ідентифікатор користувача, який ініціював обробку.
    Returns:
        None: Функція зберігає оброблене відео з субтитрами у тимчасовій папці та оновлює статус задачі.
    Raises:
        Exception: Якщо виникають помилки під час обробки (наприклад, проблеми з FFmpeg або моделями AI).
    """
    global translator_uk_en, translator_en_uk
    start_time = time.time()
    try:
        # Ініціалізація моделей перекладу
        if translator_uk_en is None or translator_en_uk is None:
            initialize_translators()
        update_task_status(task_id, "processing", 0)
        temp_path = Path(TEMP_DIR) / task_id
        temp_path.mkdir(exist_ok=True)
        video_path = Path(video_path)
        audio_path = temp_path / "audio.wav"
        subtitle_path = temp_path / f"subtitles.{subtitle_format.lower()}"
        output_video_path = temp_path / f"output{video_path.suffix}"
        logger.info(f"Processing video {video_path} (task {task_id})")

        # Витягнення аудіо з відео
        update_task_status(task_id, "processing", 20, "Extracting audio")
        stream = ffmpeg.input(str(video_path)).output(str(audio_path), acodec='pcm_s16le', ac=1, ar=16000)
        ffmpeg.run(stream, overwrite_output=True, quiet=True)

        # Транскрипція аудіо
        update_task_status(task_id, "processing", 40, "Transcribing")
        if source_lang == 'en':
            asr_model = nemo_asr.models.ASRModel.from_pretrained("nvidia/stt_en_conformer_transducer_large")
        else:
            asr_model = nemo_asr.models.ASRModel.from_pretrained("theodotus/stt_ua_fastconformer_hybrid_large_pc")
        asr_model = asr_model.to('cpu')
        audio = AudioSegment.from_file(audio_path)
        all_segments = []
        chunk_size_ms = 30000
        for i, chunk in enumerate(audio[::chunk_size_ms]):
            chunk_path = temp_path / f"chunk_{i}.wav"
            chunk.export(str(chunk_path), format="wav", codec='pcm_s16le')
            result = asr_model.transcribe([str(chunk_path)])
            chunk_text = result[0].text if result else ""
            chunk_words = chunk_text.split()
            wav = read_audio(str(chunk_path), sampling_rate=16000)
            vad_model = load_silero_vad()
            speech_timestamps = get_speech_timestamps(wav, vad_model, sampling_rate=16000)
            word_idx = 0
            for ts in speech_timestamps:
                segment_duration = (ts['end'] - ts['start']) / 16000
```

```
                segment_words = int(segment_duration * 5)  # Спрощена оцінка
                segment_text = " ".join(chunk_words[word_idx:word_idx + segment_words])
                if segment_text:
                    all_segments.append({
                        "start": ts['start'] / 16000 + i * 30,
                        "end": ts['end'] / 16000 + i * 30,
                        "text": segment_text
                    })
                word_idx += segment_words
        if not all_segments:
            raise Exception("No segments were transcribed.")

        # Переклад тексту (якщо потрібно)
        if source_lang != target_lang:
            update_task_status(task_id, "processing", 60, "Translating")
            if source_lang == 'uk' and target_lang == 'en':
                final_segments = [{"start": s["start"], "end": s["end"], "text": translator_uk_en(s["text"])
                [0]['translation_text']} for s in all_segments]
            elif source_lang == 'en' and target_lang == 'uk':
                final_segments = [{"start": s["start"], "end": s["end"], "text": translator_en_uk(s["text"],
                src_lang="eng_Latn", tgt_lang="ukr_Cyrl")[0]['translation_text']} for s in all_segments]
            else:
                raise Exception("Unsupported language pair.")
        else:
            final_segments = all_segments

        # Генерація субтитрів
        subtitle_file = generate_subtitles(final_segments, subtitle_path, subtitle_format)

        # Накладання субтитрів на відео
        update_task_status(task_id, "processing", 80, "Rendering subtitles")
        subtitle_file_str = str(subtitle_file).replace('\\', '/').replace(':', '\\:')
        stream = ffmpeg.input(str(video_path)).output(str(output_video_path), vf=f"subtitles='{subtitle_file_str}'",
        c:v="libx264", c:a="copy", crf="23", preset="veryfast")
        ffmpeg.run(stream, overwrite_output=True, quiet=True)
        processing_time = time.time() - start_time
        update_task_status(task_id, "completed", 100, result_file=str(output_video_path))
        save_processing_history(user_id, task_id, video_path.name, "completed", source_lang, target_lang, processing_time)
    except Exception as e:
        processing_time = time.time() - start_time
        logger.error(f"Error processing task {task_id}: {e}", exc_info=True)
        update_task_status(task_id, "failed", error=str(e))
        save_processing_history(user_id, task_id, video_path.name, "failed", source_lang, target_lang, processing_time)
        raise
```

Додаток В

Код модуля обробки аудіо

```
async def process_audio_to_text(task_id: str, audio_path: str, source_lang: str = 'uk', target_lang: str = 'en',
subtitle_format: str = 'txt', user_id: str = None):
    """
    Асинхронна функція для обробки аудіофайлу з транскрипцією та перекладом.
    Функція конвертує аудіо в моноформат за допомогою FFmpeg, транскрибує його за допомогою моделей Nemo ASR,
    перекладає текст (якщо потрібно) за допомогою моделей Transformers і генерує текстовий файл із субтитрами
    у вибраному форматі (SRT, VTT або TXT).
    Args:
        task_id (str): Унікальний ідентифікатор задачі.
        audio_path (str): Шлях до аудіофайлу.
        source_lang (str): Мова джерела ('uk' для української, 'en' для англійської).
        target_lang (str): Цільова мова ('uk' для української, 'en' для англійської).
        subtitle_format (str): Формат субтитрів ('srt', 'vtt' або 'txt').
        user_id (str): Ідентифікатор користувача, який ініціював обробку.
    Returns:
        None: Функція зберігає текстовий файл із субтитрами у тимчасовій папці та оновлює статус задачі.
    Raises:
        Exception: Якщо виникають помилки під час обробки (наприклад, проблеми з FFmpeg або моделями AI).
    """
    global translator_uk_en, translator_en_uk
    start_time = time.time()
    try:
        if translator_uk_en is None or translator_en_uk is None:
            initialize_translators()
        update_task_status(task_id, "processing", 0)
        temp_path = Path(TEMP_DIR) / task_id
        temp_path.mkdir(exist_ok=True)
        mono_audio_path = temp_path / "mono_audio.wav"
        output_path = temp_path / f"output.{subtitle_format.lower()}"

        # Конвертація аудіо в моноформат
        update_task_status(task_id, "processing", 20, "Converting to mono")
        stream = ffmpeg.input(str(audio_path)).output(str(mono_audio_path), acodec='pcm_s16le', ac=1, ar=16000)
        ffmpeg.run(stream, overwrite_output=True, quiet=True)
        audio = AudioSegment.from_file(mono_audio_path)

        # Транскрипція аудіо
        update_task_status(task_id, "processing", 40, "Transscribing")
        if source_lang == 'en':
            asr_model = nemo_asr.models.ASRModel.from_pretrained("nvidia/stt_en_conformer_transducer_large")
        else:
            asr_model = nemo_asr.models.ASRModel.from_pretrained("theodotus/stt_ua_fastconformer_hybrid_large_pc'
        asr_model = asr_model.to('cpu')
        chunk_length_s = 30
        chunk_size_ms = chunk_length_s * 1000
        chunks = [audio[i:i + chunk_size_ms] for i in range(0, len(audio), chunk_size_ms)]
        all_segments = []
        for i, chunk in enumerate(chunks):
            chunk_path = temp_path / f"chunk_{i}.wav"
            chunk.export(str(chunk_path), format="wav", codec='pcm_s16le')
            result = asr_model.transcribe([str(chunk_path)])
            chunk_text = result[0].text if result else ""
            chunk_words = chunk_text.split()
            wav = read_audio(str(chunk_path), sampling_rate=16000)
            vad_model = load_silero_vad()
            speech_timestamps = get_speech_timestamps(wav, vad_model, sampling_rate=16000)
            word_idx = 0
            for ts in speech_timestamps:
                segment_duration = (ts['end'] - ts['start']) / 16000
                segment_words = int(segment_duration * 5)
                segment_text = " ".join(chunk_words[word_idx:word_idx + segment_words])
```

```
                if segment_text:
                    all_segments.append({
                        "start": ts['start'] / 16000 + i * 30,
                        "end": ts['end'] / 16000 + i * 30,
                        "text": segment_text
                    })
                word_idx += segment_words
        if not all_segments:
            raise Exception("No segments were transcribed.")

        # Переклад тексту
        if source_lang != target_lang:
            update_task_status(task_id, "processing", 60, "Translating")
            if source_lang == 'uk' and target_lang == 'en':
                final_segments = [{"start": s["start"], "end": s["end"], "text": translator_uk_en(s["text"])
                [0]['translation_text']} for s in all_segments]
            elif source_lang == 'en' and target_lang == 'uk':
                final_segments = [{"start": s["start"], "end": s["end"], "text": translator_en_uk(s["text"],
                src_lang="eng_Latn", tgt_lang="ukr_Cyrl")[0]['translation_text']} for s in all_segments]
            else:
                raise Exception("Unsupported language pair.")
        else:
            final_segments = all_segments

        # Генерація субтитрів
        output_file = generate_subtitles(final_segments, output_path, subtitle_format)
        update_task_status(task_id, "completed", 100, result_file=str(output_file))
        save_processing_history(user_id, task_id, Path(audio_path).name, "completed", source_lang,
        target_lang, time.time() - start_time)
    except Exception as e:
        update_task_status(task_id, "failed", error=str(e))
        save_processing_history(user_id, task_id, Path(audio_path).name, "failed", source_lang,
        target_lang, time.time() - start_time)
        raise
```

Додаток Г

Код для моделей користувачів, задач і збереження історії обробки

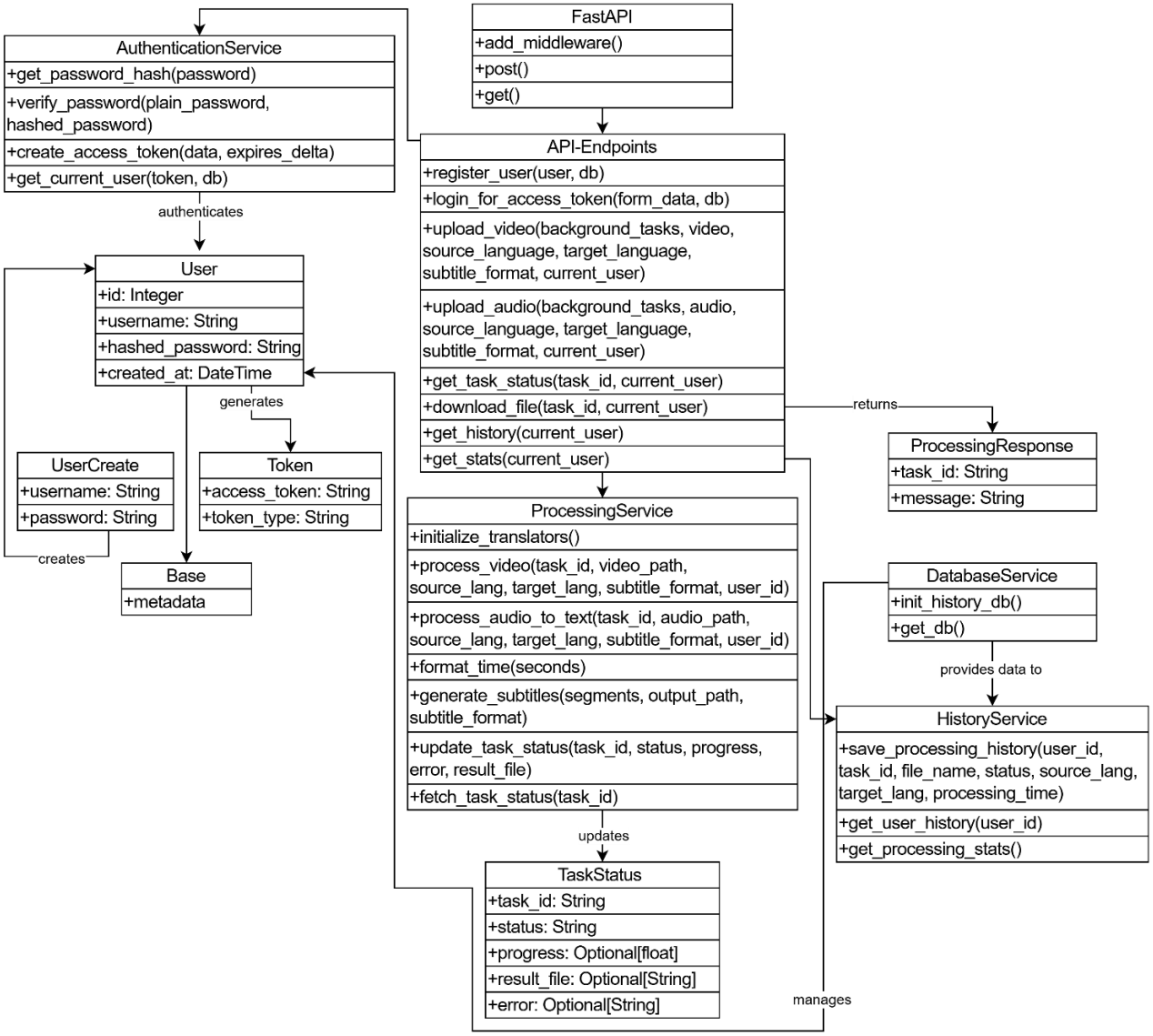
```python
# Код схеми моделей Pydantic для відповіді сервера та статусу задачі обробки
class ProcessingResponse (BaseModel):
    """
    Модель для відповіді API після завантаження файлу.
    Використовується для повернення ідентифікатора задачі та повідомлення про початок обробки.
    """
    task_id: str
    message: str

class TaskStatus (BaseModel):
    """
    Модель для відображення статусу задачі обробки файлу.
    Містить інформацію про поточний стан, прогрес, результат або помилку.
    """
    task_id: str
    status: str
    progress: Optional [float] = None
    result_file: Optional [str] = None
    error: Optional [str] = None
```

```python
# Код моделі таблиці користувачів у базі даних SQLAlchemy
class User (Base):
    """
    ORM-модель для збереження інформації про користувачів у базі даних SQLite.
    Використовується для автентифікації та збереження історії обробки файлів.
    """
    _ tablename _ = "users"
    id Column (Integer, primary_key=True, index=True)
    username = Column (String, unique=True, index=True)
    hashed_password = Column (String)
    created_at = Column (DateTime, default=datetime.utcnow)
```

```python
# Код функції збереження історії обробки файлів у SQLite
def save_processing_history(user_id, task_id, file_name, status, source_lang, target_lang, processing_time):
    """
    Зберігає історію обробки файлу в базі даних SQLite.
    Функція додає запис до таблиці processed_files, який містить інформацію про задачу,
    користувача, файл, мови, статусі час обробки.
    Args:
        user_id (str): Ідентифікатор користувача.
        task_id (str): Унікальний ідентифікатор задачі.
        file_name (str): Назва обробленого файлу.
        status (str): Статус обробки ('completed' або 'failed').
        source_lang (str): Мова джерела.
        target_lang (str): Цільова мова.
        processing_time (float): Час обробки в секундах.
    Returns:
        None: Функція додає запис до бази даних.
    """
    conn = sqlite3.connect('processing_history.db')
    cursor = conn.cursor()
    timestamp datetime.now().isoformat()
    cursor.execute('''
        INSERT INTO processed_files (user_id, task_id, file_name, status, source_lang,
        target_lang, processing_time, timestamp)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?)
    ''', (user_id, task_id, file_name, status, source_lang, target_lang, processing_time, timestamp))
    conn.commit()
    conn.close()
```

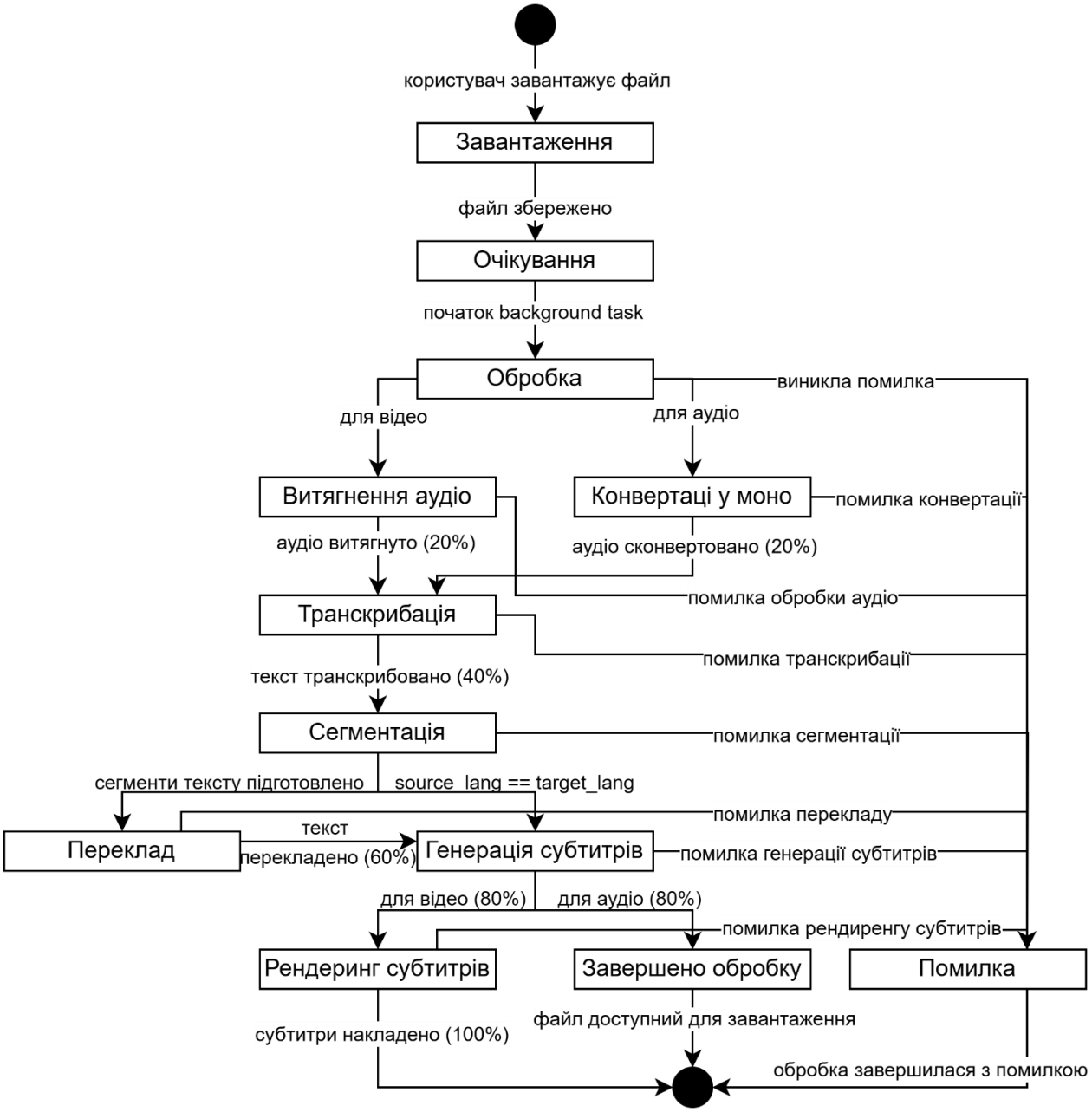
Додаток Д

UML-діаграма класів



Додаток Е

UML-діаграма станів



Додаток Ж

Результати тестування за сценарієм реєстрація користувача

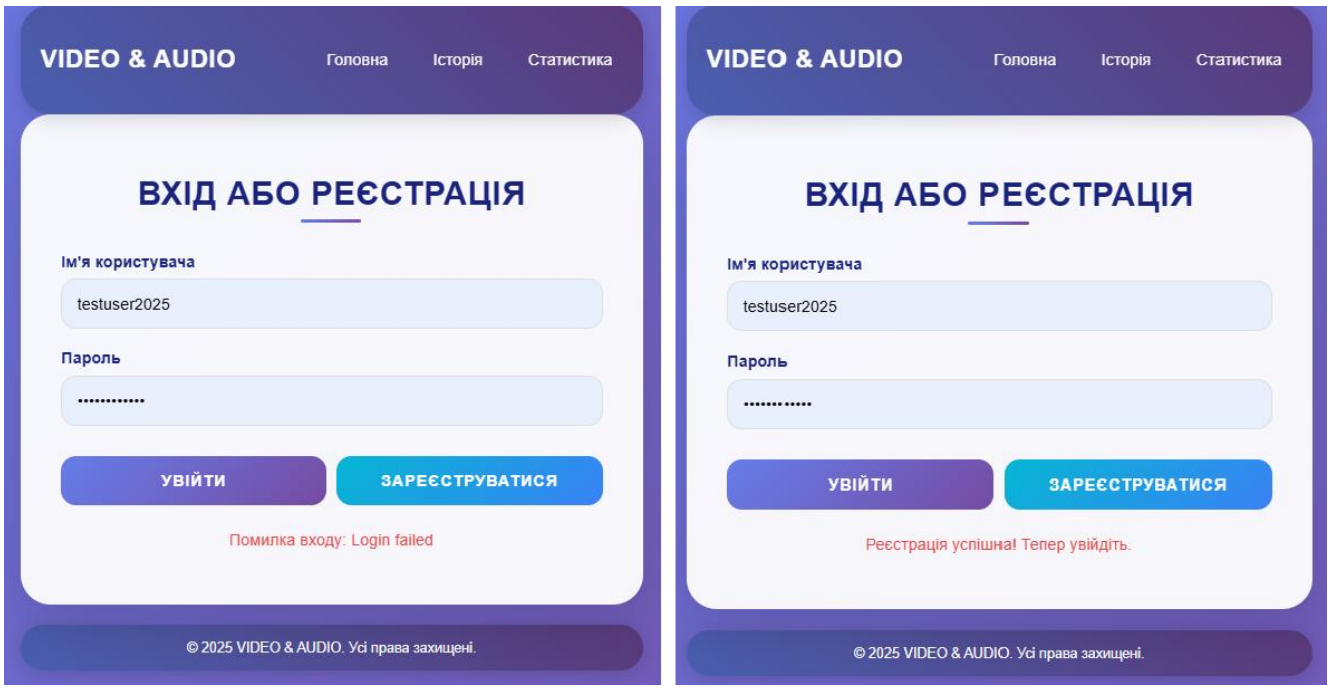


Рисунок Ж.1 – Сторінка реєстрації або входу в обліковий запис (зліва користувача з «username»: «testuser2025» ще не існує, а справа – користувач успішно пройшов реєстрацію)

Додаток И

Результати тестування за сценарієм автентифікація користувача

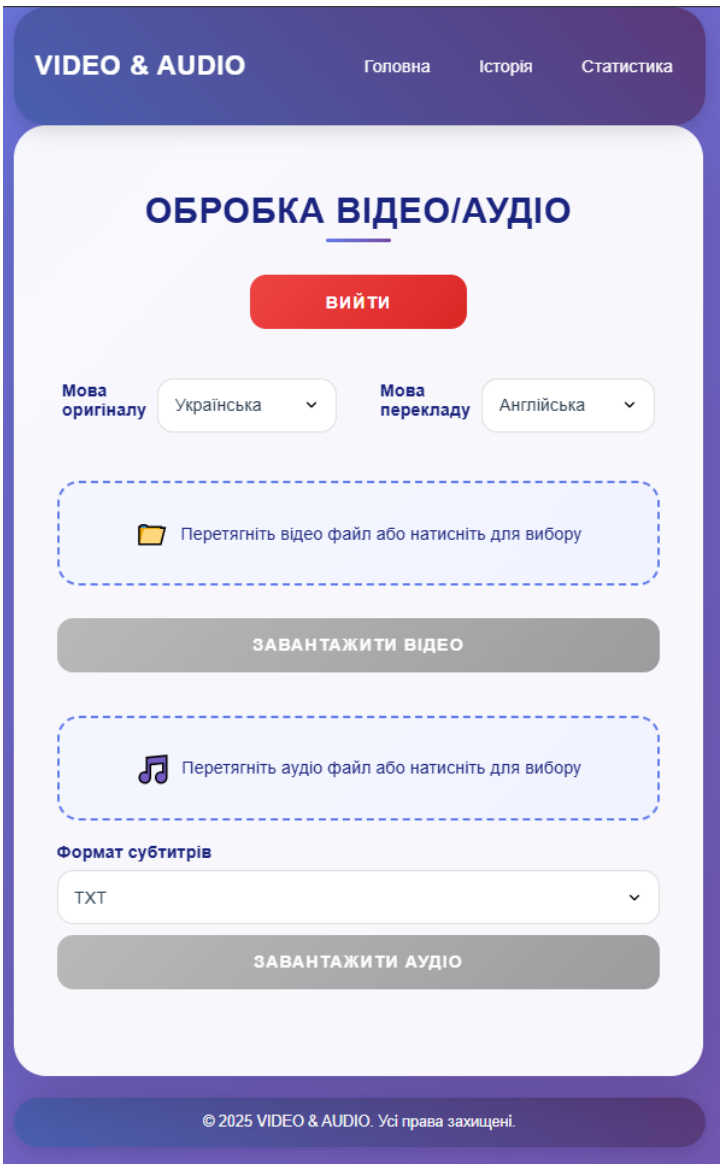


Рисунок И.1 – Головна сторінка API (користувача з «username»: «testuser2025» увійшов в обліковий запис)

Додаток К

Результати тестування за сценарієм завантаження відео з транскрипцією та перекладом

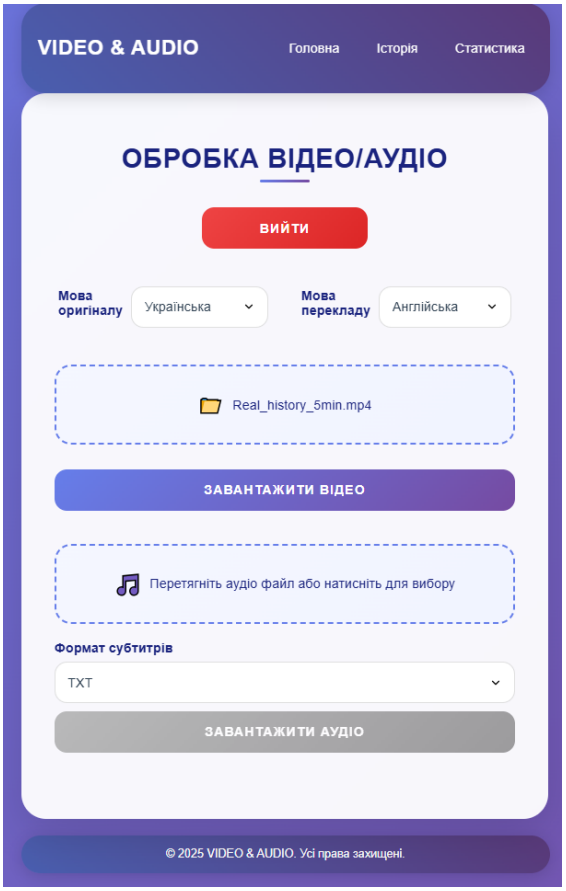


Рисунок К.1 – Головна сторінка API (користувача з «username»: «testuser2025» завантажує відео для обробки)

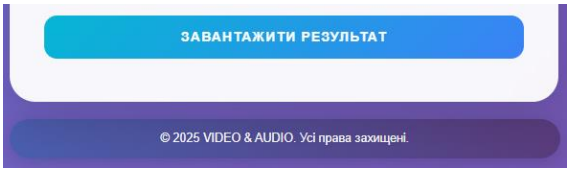


Рисунок К.2 – Головна сторінка API (обробка відео завершилася – з’явилася кнопка «Завантажити результат»)

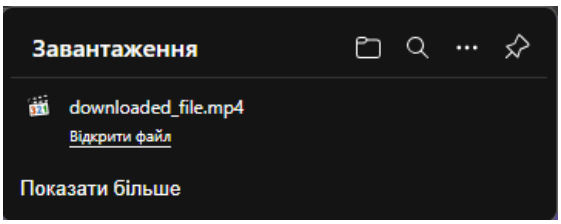


Рисунок К.3 – Користувач завантажив оброблене відео собі на пристрій

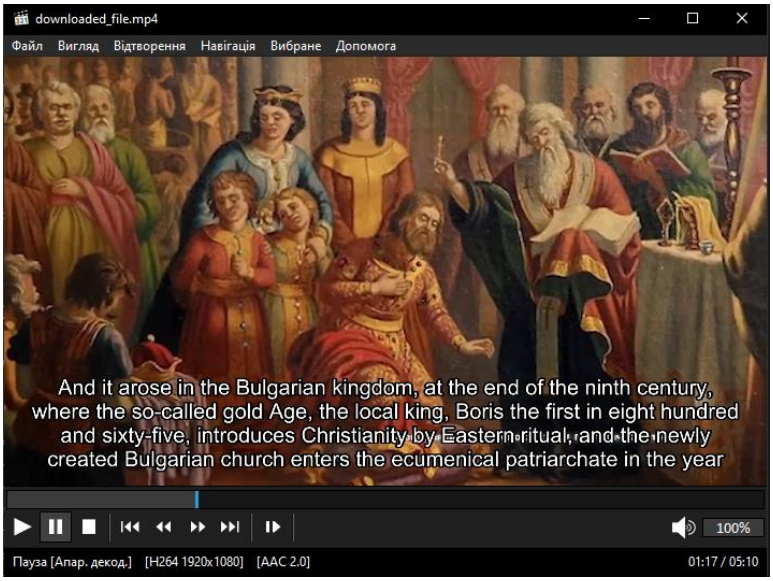


Рисунок К.4 – Приклад обробленого відео з субтитрами

Додаток Л

Результати тестування за сценарієм завантаження аудіо для транскрипції

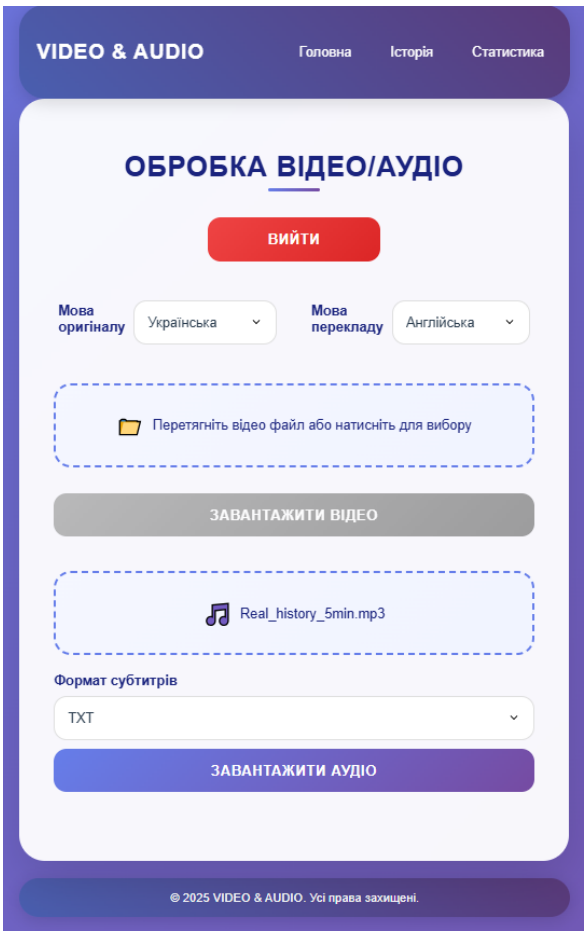


Рисунок Л.1 – Головна сторінка API (користувача з «username»: «estuser2025» завантажує аудіо для обробки)

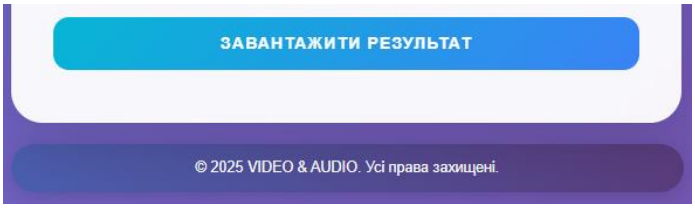


Рисунок Л.2 – Головна сторінка API (обробка аудіо завершилася – з’явилася кнопка «Завантажити результат»)

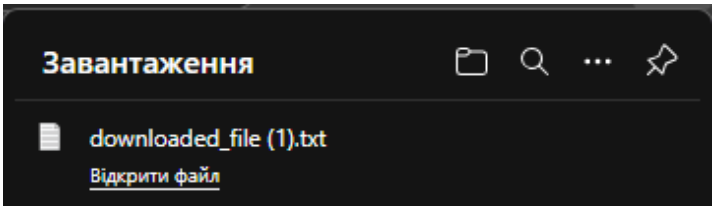


Рисунок Л.3 – Користувач завантажив файл з субтитрами собі на пристрій

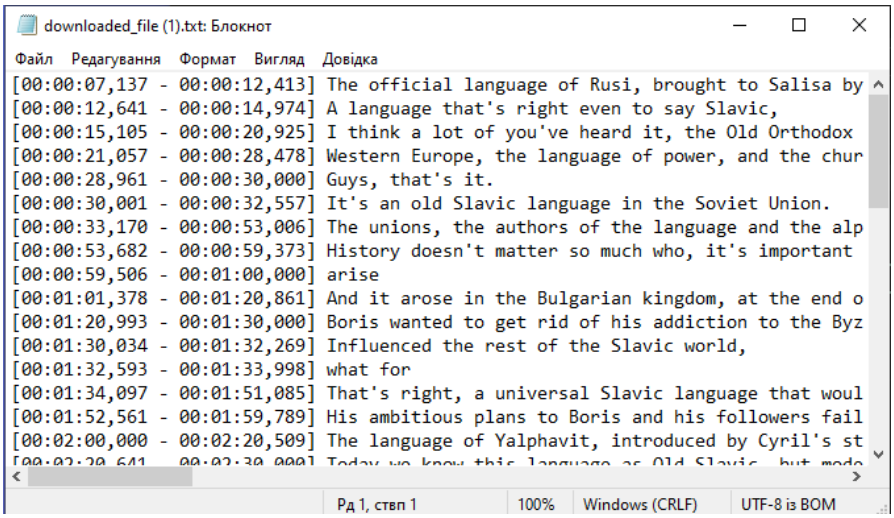


Рисунок Л.4 – Приклад файлу з субтитрами

Додаток М

Результати тестування за сценарієм перевірки об’єму файлу

Рисунок М.1 – Користувач завантажив відео розміром більше ніж 5 ГБ та отримав помилку

Додаток Н

Копії опублікованих результатів

Західноукраїнський національний університет

Факультет комп'ютерних інформаційних технологій

Кафедра інформаційно-обчислювальних систем і управління

ЗБІРНИК ТЕЗ ДОПОВІДЕЙ

Студентської науково-практичної конференції

ІНТЕЛЕКТУАЛЬНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В ПРИКЛАДНИХ ДОСЛІДЖЕННЯХ

(IITAR-2025)

27-29 травня 2025 року

Тернопіль

2025

Герцій Володимир
студент групи КН-41
gertsiyvolodymyr2004@gmail.com
Турченко Ірина
к.т.н., доцент
itu@wunu.edu.ua
Західноукраїнський національний університет
Тернопіль, Україна

ПРИКЛАДНИЙ ПРОГРАМНИЙ ІНТЕРФЕЙС ДЛЯ ТРАНСКРИПЦІЇ ТА СУБТИТРІВ ДЛЯ АУДІО ТА ВІДЕОФАЙЛІВ НА ОСНОВІ ШТУЧНОГО ІНТЕЛЕКТУ

У сучасному світі аудіо- та відеоконтент відіграє важливу роль у багатьох сферах: від розваг до освіти й бізнесу [1]. Зростання обсягу такого контенту зумовлює потребу в ефективних інструментах для його обробки, зокрема автоматичної транскрипції та субтитрування [2]. Ці технології забезпечують доступність інформації для людей із вадами слуху, полегшують локалізацію контенту для міжнародної аудиторії та автоматизують аналіз і структурування даних [3]. Традиційні методи транскрипції, що базуються на ручній праці, є повільними, дорогими та схильними до помилок, особливо за наявності фонового шуму, акцентів чи спеціалізованої термінології.

Користувачі відомих існуючих сервісів, таких як Kapwing, HappyScribe та Sonix, стикаються із наступними проблемами: наявність на вихідному відео логотипу сервісу, обмежений функціонал безкоштовних версій і зниження якості вихідного відео. Запропоноване рішення покликане подолати їх.

Для розробки прикладного програмного інтерфейсу (англ. Application Programming Interface (API)) для транскрипції та субтитрів для аудіо та відеофайлів на основі штучного інтелекту використано FastAPI для серверної частини та React.js для веб-інтерфейсу. API підтримує завантаження файлів, їх обробку (видобування даних з аудіо, транскрипція, переклад, генерація субтитрів) і повернення результатів у форматах SRT, VTT, TXT. Структурна схема API включає фронтенд, сервер, модулі обробки та сховище даних, як показано на рисунку 1. На етапі тестування перевірено функціональність на різних сценаріях, включаючи обробку відео й аудіо, автентифікацію та обробку помилок.

Для розробки API використано сучасні бібліотеки і інструменти: FFmpeg для обробки медіа, NeMo ASR і Transformers для розпізнавання та перекладу, SQLite для зберігання даних, а також Pydantic і SQLAlchemy для валідації та роботи з базою даних.

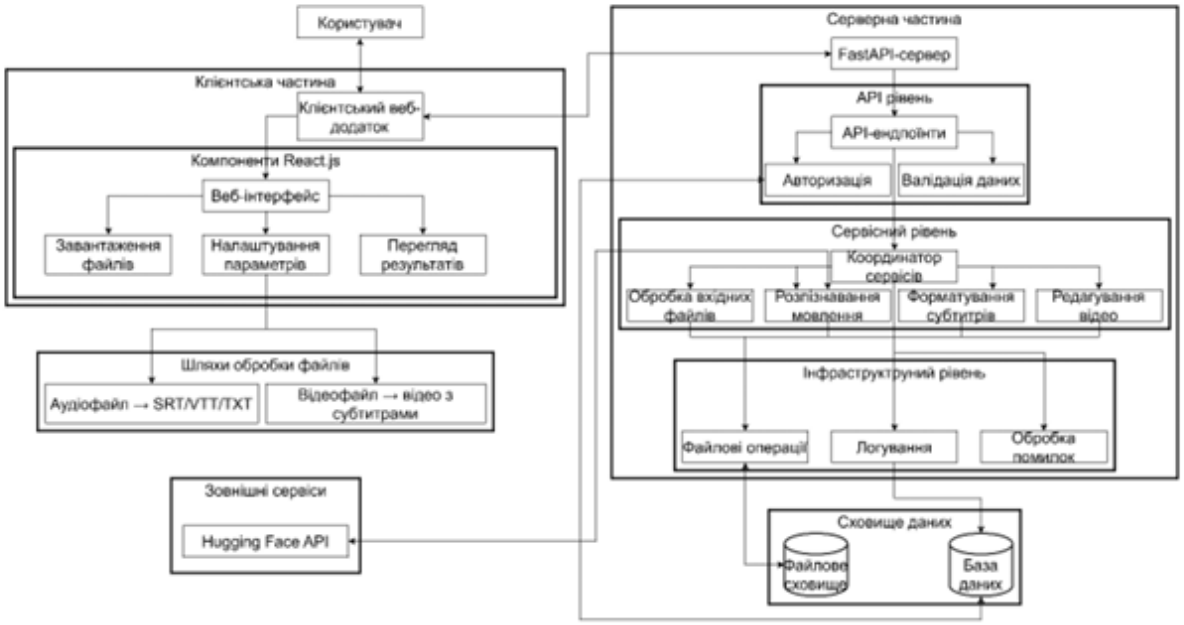


Рисунок 1 – Структурна схема прикладного програмного інтерфейсу

Для розпізнавання україномовного аудіо було проаналізовано вісім моделей для обробки природної мови з Hugging Face, відібраних з двадцяти, у яких точність більше 90%. Інформація про їх використання для розпізнавання та синтезу української мови доступна на GitHub [4].

Для розпізнавання англомовного аудіо було обрано модель, яка має найбільше завантажень на сервісі Hugging Face. Для перекладу тексту з української мови на англійську та перекладу тексту з англійської мови на українську було використано дві моделі, які були відібрані аналогічним чином.

Розроблений API використовує моделі theodotus/stt_ua_fastconformer_hybrid_large_pc [5] (для обробки україномовного аудіо), nvidia/stt_en_conformer_transducer_large [6] (для обробки англомовного аудіо), Helsinki-NLP/opus-mt-uk-en (для перекладу тексту з української мови на англійську) та facebook/nllb-200-distilled-600M (для перекладу тексту з англійської мови на українську). Час обробки становить 1 хв 6 с для аудіо та 2 хв 32 с для відео (на прикладі 5-хвилинного файлу з YouTube [7]).

Функціональність API підтримує популярні формати (MP4, MP3, WAV), автоматично витягує аудіо з відео, генерує субтитри у форматах SRT, VTT, TXT і вбудовує їх у відео без втрати якості. Логотип сервісу відсутній на вихідному відео.

Зручність використання забезпечується веб-інтерфейсом, створеним на React.js. Інтерфейс має інтуїтивний дизайн, дозволяє вибирати мови, формати та переглядати історію обробок, як зображено на Рисунку 2. API добре підходить для інтеграції в сторонні системи завдяки зрозумілій REST-архітектурі [8].

Проведене тестування підтвердило стабільну і надійну роботу API в різних сценаріях. Зокрема, вона коректно обробляє великі файли обсягом до 5 ГБ і належним чином повідомляє про помилки при перевищенні лімітів.

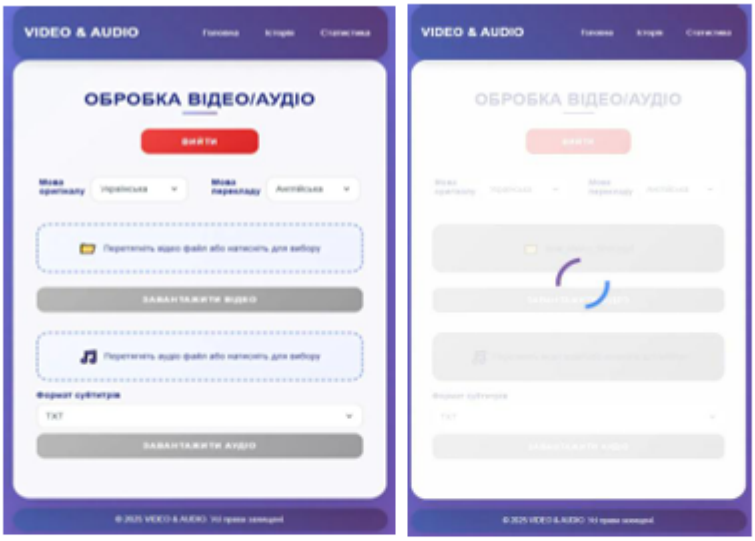


Рисунок 2 – Вигляд головної сторінки API (зліва сторінка без обробки файлів; справа – під час обробки файлів)

Для оцінки ефективності роботи розробленого API та сервісів-аналогів було проведено аналіз обробки україномовного аудіо, результати якого наведено в таблиці 1, яка ілюструє порівняльну характеристику за критеріями точності, коректності термінів, відсутності помилок у назвах/числах та збереження граматики.

Таблиця 1 – Порівняльна характеристика розробленого API та сервісів-аналогів для україномовного аудіо

Критерій	Happy Scribe	Sonix	Kapwing	Розроблений API
Точність слів і фраз	Висока, дрібні помилки в пунктуації	Дуже висока, мінімальні помилки	Низька, багато помилок ("Росії", "виникіт")	Висока, але є помилки ("братки", "Владір")
Коректність термінів	Правильні, але помилка в даті ("2022")	Правильні, помилка в даті ("2022")	Неконсистентні ("слов'янська", "великомуравський")	Правильні, коректна дата ("865")
Відсутність помилок у назвах/числах	Помилка в даті ("2022"), назви правильні	Помилка в даті ("2022"), назви правильні	Помилки в "Кирилу", "Фокко", але дата "865" правильна	Помилки в "Мифоді", "Константинок", числа точні
Збереження граматики (кличний, повноголосся)	Повністю збережено	Повністю збережено	Частково збережено, помилки ("Фокко")	Збережено, але помилка ("фоко")
Час обробки для аудіо та відео	1 хв. 25 с. для аудіо та 2 хв 15 с. для відео	1 хв. для аудіо та 2 хв 38 с. для відео	1 хв. 6 с. для аудіо та 2 хв 32 с. для відео	1 хв. 6 с. для аудіо та 2 хв 32 с. для відео

Розроблений API має конкурентні переваги завдяки відсутності водяних знаків, підтримці якості відео та гнучкості обробки.

У майбутньому планується оптимізація обробки великих файлів, розширення мовної підтримки, покращення точності розпізнавання та інтеграція з хмарними сервісами.

Отже, розроблений прикладний програмний інтерфейс для транскрипції та субтитрів для аудіо та відеофайлів на основі штучного інтелекту забезпечує точність, гнучкість і зручність інтеграції в інші системи, роблячи його цінним інструментом для широкого кола користувачів.

Список використаних джерел

1. Jensen C. T. APIs for Dummies. Hoboken, NJ: John Wiley & Sons, Inc., 2015. 56 с.
2. Ayonghe L. S. Fundamentals of Subtitling: Concepts, Theory and Practice. Yaoundé: Éditions CLÉ, 2021. 150 с.
3. Dresing T., Pehl T., Schmieder C. Manual (on) Transcription: Transcription Conventions, Software Guides and Practical Hints for Qualitative Researchers. 3rd English ed. Marburg: Self-published, 2015. 92 с.
4. speech-recognition-uk UA Speech Recognition & Synthesis. URL: https://github.com/egorsmkv/speech-recognition-uk?tab=readme-ov-file (дата звернення: 20.05.2025).
5. theodotus/stt_ua_fastconformer_hybrid_large_pc – Hugging Face. URL: https://huggingface.co/theodotus/stt_ua_fastconformer_hybrid_large_pc (дата звернення: 21.05.2025).
6. nvidia/stt_ua_fastconformer_hybrid_large_pc – Hugging Face. URL: .https://huggingface.co/nvidia/stt_ua_fastconformer_hybrid_large_pc (дата звернення: 24.04.2025).
7. Невже російська – лише говірка української мови? Реальна історія [Електронний ресурс]. URL: https://www.youtube.com/watch?v=kJVmZ7WVVWo&t=8s
8. Cooksey B. An Introduction to APIs. Zapier, Inc., 2014. 60 с.