МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ І ПРИКЛАДНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДІБРОВА ІВАН СЕРГІЙОВИЧ

Допускається до захисту:
в.о. завідувача кафедри
інформаційних технологій,
д-р техн. наук, професор
_____ Наталія ВЕСЕЛОВСЬКА
«____»_____ 2026 р.

РОЗРОБКА ВЕБДОДАТКУ ДЛЯ АВТОМАТИЧНОГО ОЗВУЧУВАННЯ ТЕКСТУ ТА ГЕНЕРАЦІЇ ВІДЕО

Спеціальність 122 Комп’ютерні науки

Кваліфікаційна (бакалаврська) робота

Керівник:
Роман БАБАКОВ, професор кафедри
інформаційних технологій,
д-р техн. наук, доцент
_____

Оцінка: _____ / _____ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2026

АНОТАЦІЯ

Діброва І. О. Розробка вебдодатку для генерації аудіо та відео на основі тексту. Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні науки». Донецький національний університет імені Василя Стуса, Вінниця, 2026.

У роботі розроблено вебдодаток VidVoice для автоматичного створення аудіо та відео на основі тексту. Реалізовано клієнтську та серверну частини системи, модуль генерації озвучення через ElevenLabs API, механізм створення відео за допомогою FFmpeg, а також систему керування користувацькими проєктами. Вебдодаток підтримує введення та переклад тексту, вибір голосу, завантаження зображень, генерацію аудіо, формування відеофайлів і збереження даних у базі PostgreSQL. Для розгортання системи використано серверне середовище з Nginx, FastAPI та Docker.

Ключові слова: вебдодаток, генерація аудіо, генерація відео, ElevenLabs API, FFmpeg, FastAPI, PostgreSQL, Docker, клієнт-серверна архітектура.

ABSTRACT

Dibrova I. O. Development of a web application for audio and video generation based on text. Specialty 122 "Computer Science", educational program "Computer Science". Vasyl' Stus Donetsk National University, Vinnytsia, 2026.

The thesis presents the development of the VidVoice web application for automatic audio and video generation based on text. The system includes client-side and server-side components, a speech generation module using the ElevenLabs API, a video generation mechanism based on FFmpeg, and a user project management system. The web application supports text input and translation, voice selection, image uploading, audio generation, video composition, and data storage in a PostgreSQL database. The deployment environment includes Nginx, FastAPI, and Docker services.

Keywords: web application, audio generation, video generation, ElevenLabs API, FFmpeg, FastAPI, PostgreSQL, Docker, client-server architecture.

ЗМІСТ

ВСТУП

У сучасну епоху цифрової трансформації мультимедіа стало ключовим інструментом для передачі знань та ідей в інтернеті. Сьогодні відео та аудіо — це не просто розваги, а фундамент для дистанційної освіти, сучасного маркетингу та соціальних платформ. Проте виготовлення якісного продукту зазвичай потребує або дорогих фахівців, або складного софту. Саме цей бар'єр зумовлює попит на автоматизовані рішення - вебзастосунки, що здатні «на льоту» перетворювати звичайний текст у готовий медіафайл.

Завдяки стрімкому прогресу в алгоритмах синтезу голосу та обробки даних, ми отримали можливість створювати системи, які самі озвучують тексти та монтують ролики. Це кардинально змінює правила гри: користувачу більше не потрібно мати студійне обладнання чи навички професійного монтажера. Достатньо лише браузера та ідеї. Хмарні технології роблять ці інструменти доступними з будь-якого пристрою, що значно пришвидшує виробництво контенту.

Ефективність таких вебсистем базується на грамотному поєднанні клієнт-серверної архітектури, зовнішніх API та потужних бібліотек для роботи з медіапотоками. У межах цього проєкту для створення природного звучання було обрано API від ElevenLabs, а для технічного формування відеофайлів - перевірений часом інструментарій FFmpeg. Така комбінація дозволяє побудувати надійний цикл автоматизації: від введення тексту до отримання фінального відео.

Об'єктом дослідження виступає процес автоматизованого створення медіаконтенту в середовищі вебсистем.

Предметом дослідження є технічні методи, інструменти та безпосередньо розробка архітектури застосунку для генерації аудіо та відео на основі текстових даних.

Метою роботи є створення вебплатформи «VidVoice», яка б автоматизувала озвучення тексту, роботу з аудіо та фінальну збірку відеоряду з візуальних елементів та звуку.

Для реалізації мети було визначено наступний перелік завдань:

1. Дослідити ринок вебтехнологій та актуальні методи генерації медіафайлів.

2. Вивчити внутрішню логіку та принципи роботи сучасних сервісів синтезу мовлення.
3. Спроєктувати надійну архітектуру майбутнього вебдодатку.
4. Написати програмний код для клієнтської та серверної частин.
5. Налаштувати взаємодію з ElevenLabs API для отримання якісного озвучення.
6. Розробити програмний модуль для рендерингу відео через FFmpeg.
7. Побудувати базу даних для адміністрування проєктів користувачів.
8. Провести комплексне тестування розгорнутої системи в реальних умовах.

Технологічний стек проєкту включає: мови HTML, CSS та JavaScript для фронтенду, Python та фреймворк FastAPI для логіки сервера, базу даних PostgreSQL, а також Docker та Nginx для стабільної роботи й розгортання.

Практична цінність проєкту полягає у створенні готового інструменту, який полегшить життя викладачам, блогерам та маркетологам, дозволяючи за лічені хвилини створювати презентації, навчальні ролики чи рекламні креативи.

Робота має чітку структуру: вступ, три основні розділи з висновками та додатки. У першій частині розглядається теорія та існуючі рішення. Друга присвячена безпосередньо проєктуванню та кодуванню «VidVoice». Третя частина описує практичний етап: запуск на сервері, пошук помилок та оцінку того, наскільки добре система справляється зі своїми задачами.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ГЕНЕРАЦІЇ АУДІО ТА ВІДЕО З ТЕКСТУ

1.1 Актуальність теми дослідження

Сьогодні - одним із головних способів передачі інформації, є мультимедійний контент. Особливо відео, які широко застосовуються в різних сферах, починаючи від здоров'я та спорту, закінчуючи освітою та роботою. Значного розвитку за рахунок мультимедійного контенту набула реклама, так як вона є основним джерелом притоку нових клієнтів та покупців, що приносить бізнесу значні кошти та доходи. Поєднання візуальних та звукових елементів роблять сучасні відеоматеріали дуже ефективними та зрозумілими для сприйняття. Сучасні мультимедійні технології дозволяють інтегрувати у відео анімацію, звукові ефекти, що може значно підвищити популярність відео і його клікабельність [1, 2].

Проте, створення дійсно якісного відео, яке матиме охоплення і буде популярним – потребує багато часу та ресурсів, особливо традиційними методами та шляхом ручного монтажу. Навіть створення короткого відео потребує значних витрат: написання сценарію, пошук чи створення візуальних елементів та фото, запис озвучки, ручний монтаж та синхронізація аудіо та відео. Створення будь якого відео, навіть простого, потребує певних технічних знань та навичок роботи із спеціальним програмним забезпеченням.

Для тих, хто кожен день займається створенням подібного відеоматеріалу, ця проблема стає дуже актуальною, особливо коли мова йде про контент, який може повторюватися, або є подібним між собою, наприклад – єдина тематика, жанр переказу чи рекапів. Це може відносити до різної групи людей, наприклад: маркетологи, власники онлайн-проєктів, автори контенту, які публікують свої роботи на YouTube з метою заробітку через партнерство. Зазвичай, створення такого контенту – повторювана дія, яка потребує автоматизації, що може дуже сильно економити час, дасть змогу швидше перетворювати текстову інформацію в аудіо і в подальшому створити готове відео [3].

Перший крок для автоматизації процесу створення відео – це автоматизація процесу озвучення. Наступний етап – формування відео та синхронізація його з аудіо. За допомогою технології синтезу мовлення – процес автоматизації озвучки є реальною задачею, тому що це дозволяє легко перетворити текст в аудіофайл, який можна використовувати як звуковий супровід. Це може значно розширити можливості будь якого автора, який не має можливості записати якісний звук. Сучасні технології синтезу мовлення дозволяють зробити голос, який майже неможливо відрізнити від людського. Такий контент дуже часто використовується в навчальних платформах та інформаційних системах [4].

На сьогоднішній день існує досить багато інструментів, які дозволяються створити якісний звук чи сформувати відео, але, всі вони працюють по окремості і виконують окремі завдання. Одні сервіси фокусуються на синтезі голосу, інші пропонують набір функцій для його редагування чи створення відео. В контексті масового і частого створення контенту – це можу бути дуже не зручно для автора, тому що це змушує його перемикатися та залучати кілька програм одразу, через що, не є виключенням створення ним помилок. Такий підхід може значно ускладнити процес, тому що все доводиться робити вручну.

Тому, розробка вебдодатку, який об’єднує в собі ці процеси і дозволяє якісно озвучити текст та сформувати відео в одному місці – значно спростить процес створення відеоконтенту. В результаті – користувач має змогу одразу ввести текст, або завантажити його із файлу, обрати параметри голосу, додавати потрібні зображення і створювати на їх основі відео без необхідності кількох різних програм чи сервісів. Такий підхід значно скорочує час на роботу, є зручнішим та мінімізує виникнення помилок. Завдяки такому вебдодатку, процес створення контенту стає доступнішим для категорії людей які не мають спеціальних технічних знань в монтажі.

Формат вебдодатку – є найкращим для реалізації подібних функцій. Це дозволить багатьом користувачам мати доступ до системи саме через браузер, без необхідності встановлювати щось на свій пристрій. За рахунок сучасних вебтехнології – можна створити інтуїтивно зрозумілий дизайн, з яким зможе

розібратися людина, яка ніколи не займалася монтажем. Крім того, завдяки адаптивності, вебдодатком можна користуватися на різних типах пристроїв [5]. Під час розробки такої програми, важливо враховувати різні аспекти, такі як: архітектура програми, зручність користування та експлуатації, готовність до маштабування та можливість збереження даних [6].

Таким чином, тема дослідження стає дуже актуальної завдяки в потребі створення контенту. Це спрощує процес створення відео, мінімізує ручну працю та помилки, залучає більше людей і скорочує час виконання завдань, що в результаті може підвищити доступність мультимедійного контенту для ширшого кола людей.

1.2 Основні підходи до синтезу мовлення та генерації відео

Ключовий напрямок роботи з природною мовою та мультимедійними технологіями – синтез мовлення. Він потрібен для того, аби перетворити текст на аудіо, що дуже схоже на людське мовлення. На сьогоднішній день такі системи набули дуже широкого використання. Вони використовуються в навчальних матеріалах, електронних помічниках, навігаційних системах, системах для людей із вадами зору а також в різних вебсервісах які озвучують контент [3, 4].

Синтез мовлення це процес, який складається із кількох послідовних етапів. На першому етапі — виконується попередня обробка тексту, спочатку із нього видаляються зайві пробіли та інші символи, суцільні речення розбиваються на менші частини, визначаються дати, числа та інші елементи, які потрібно вимовляти правильно. Після цього виконується лінгвістичний аналіз, система визначає структуру тексту, усі наголоси, інтонацію та паузи. На останньому етапі відбувається створення звукового сигналу, який можна назвати озвученим текстом.

Існує кілька способів створення мовлення. Перший спосіб — конкатенативний метод, який використовує фрагменти людського голосу, які були записані раніше. В такому випадку з’єднуються окремі звукові частини у необхідному порядку і утворюються нові повні речення. Особливістю цього

підходу є те, що фрагменти звучать природньо, тому що за основу був взятий реальний голос людини. Цей підхід містить також ряд недоліків, серед яких складність створення великої кількості звукових одиниць. Також його дуже важко адаптувати для озвучення різних текстів.

Інший підхід – параметричний синтез. У цьому підході синтез мовлення відбувається не шляхом склеювання готових фрагментів, а за допомогою математичних моделей, які описують характеристики голосу. За допомогою цього підходу можна змінювати параметри звучання, налаштовувати висоту голосу, тональність та тембр. Проте, якість такого звучання набагато нижча, у порівнянні із першим методом, тому що згенерований голос дуже часто звучить штучно і його легко відрізнити від людського.

Останній підхід заснований на методах нейронних мереж, який дуже часто використовують сучасні системи синтезу мовлення. Це дає змогу створювати звук, якиц дуже важко відрізнити від людського, тому що він звучить дуже природньо. Цей підхід враховує паузи, ритм, та навіть особливості вимови певної людини, за допомогою такого підходу можна згенерувати голос реальної людини і він буде звучати не гірше. Такі системи навчаються на величезних наборах даних, на текстах і аудіо реальних людей. Для сучасних вебсистем цей підхід є найкращим і саме так у більшості випадків створюється автоматичне озвучування тексту [3].

Для розроблюваного вебдодаткусинтез мовлення є дуже важливим етапом, тому що створення аудіо на основі тексту, який вводить користувач, є його основною функцією. Користувач повинен мати можливість ввести текст, або завантажити його з файлу, після чого обрати параметри голосу які йому найбільше підходять. Після цього відбувається автоматична генерація голосу, а згенеровані файли можуть використовуватися в подальшому як основа для створення відео.

Генерація відео у застосунку відбувається як окремий процес шляхом з’єднання зображень із аудіо в один мультимедійний файл. У цьому випадку відео не створюється повністю, як у порівнянні з складними системами штучної

генерації, де відео генерується за описом. Користувач самостійно завантажує зображення, а система з’єднує їх із аудіо автоматично. Такий підхід зрозуміліше для користувача та краще підходить для створення навчальних відео, презентаційних матеріалів та іншого типу контенту.

Процес генерації відео також можна розділити на кілька етапів. На першому кроці, користувач вставляє власні зображення, які і будуть основою майбутнього відео. Далі система використовує уже згенеровані аудіо, на основі тексту, який також ввів користувач. Після чого визначається тривалість показу кожного зображення. На останньому кроці аудіо та зображення об’єднуються в один відеофайл.

Важливе завдання під час створення відео, це правильна синхронізація аудіо та зображень, тому що саме це впливає на сприйняття відео. Також від цього залежить його монетизація. Для прикладу, на платформі YouTube відео із хорошим утриманням є одним із найважливіших показників для алгоритмів платформи. Алгоритми YouTube частіше рекомендують відео, які утримують увагу глядачів довше. Висока середня тривалість перегляду (утримання аудиторії) сигналізує, що контент цікавий, тому платформа просуває його вище в пошуку та рекомендаціях. Це дозволяє розмістити більше рекламних пауз, що збільшує загальний дохід від реклами (RPM).

Під час створення відео враховується багато технічних налаштувань, а саме: формат відео, роздільна здатність, співвідношення сторін, частота кадрів та якість кодування. Ці налаштування на пряму впливають на розмір відео, швидкість генерації. Для вебдодатку доцільно обирати ті формати, які використовують та підтримують більшість пристроїв та браузерів.

Поєднання цих підходів дає єдиний процес обробки інформації: від вводу тексту до отримання готового відеофайлу. Такий метод дуже зручний, адже позбавляє користувача від роботи в різних середовищах та програмах, а дозволяє зробити всю роботу в одному місці, без спеціальних технічних навичок та знання програмного забезпечення. Крім того автоматична обробка зменшує час монтажу та мінімізує шанс виникнення помилок.

Отже, головними способами створення звуків є конкатенативний, параметричний та метод нейронних мереж. Нейромережевий підхід для сучасних вебсистем – є найбільш ефективним та популярним, оскільки він дає більш природнє звучання голосу, яке важко відрізнити від людського. Для створення відео в даній роботі використовують поєднання зображень та аудіо, що дозволяє автоматично генерувати прості відео матеріали на основі тексту. Збіг цих двох напрямів є основою роботи розроблюваного вебдодатку.

1.3 Огляд сучасних вебтехнологій для створення мультимедійних застосунків

Вебдодаток це дуже популярне рішення для створення сучасних мультимедійних програм. Це дозволяє користувачам легко користуватися програмою без необхідності встановлення необхідних компонентів на свій пристрій. Крім того це дозволяє запустити і миттєво скористатися додатком з будь-якого місця та через будь-який пристрій. А саме для таких систем, якими користуються багато користувачів це дуже важливо.

Сучасні технології дозволяють створювати дуже великі та інтерактивні системи, у яких користувач сам вводить інформацію, сам завантажує файли, обирає необхідні параметри та бачить готовий результат. Система має працювати разом із користувачем та з файлами які він завантажив і саме ця взаємодія є дуже важливою під час створення вебдодатку. Для забезпечення такого підходу важливо поєднати різні частини програми, а саме клієнтську частину, серверну логіку, базу даних та інші інструменти для обробки мультимедія.

Клієнтська частина відповідає за взаємодію користувача і з вебсайтом. Вона відображає сторінки, форми, кнопки, повідомлення, завантажені файли та результат генерації. Основні технології для створення клієнтської частини, це: HTML, CSS, та JavaScript. HTML – використовується для будування основної структури веб-сторінок, CSS – для їх візуального оформлення, а JavaScript - для реалізації інтерактивності та взаємодії з елементами інтерфейсу [7, 8].

Крім того, під час розроблення додатку важливо звертати увагу не лише на візуальні елементи, а й на зручність користування. Інтерфейс має бути простим та інтуїтивно зрозумілим. Користувач повинен швидко адаптуватися та зрозуміти, як користуватися вебдодатком. Зазвичай, якщо користувач не може самостійно розібратися із сайтом, він не є достатньо ефективним і не приносить бажаного результату. Наявність адаптивного дизайну дозволяє коректно користуватися вебсайтом на різних пристроях, що може значно збільшити кількість користувачів. Саме це є однією з вимог до сучасних систем [9].

Головну логіку роботи з даними виконує серверна частина вебдодатку. На сервері приймаються та обробляються запити від користувача, перевіряється валідність введених даних, відбувається взаємодія з базою даних, працюють модулі, які відповідають за озвучування тексту та генерацію відео, а також повертається кінцевий результат. Для виконання таких завдань добре підходить Python, оскільки за його допомогою можна взаємодіяти з API, файлами, асинхронними запитами та зовнішніми сервісами.

Важливе місце займає архітектурний підхід, за допомогою якого частини вебдодатку можуть взаємодіяти між собою через програмний інтерфейс API. У такій моделі клієнтська частина надсилає запит до сервера, а сервер повертає відповідь. Це дає можливість розділити інтерфейс користувача та бізнес-логіку системи, що дозволяє спростити підтримку програмного продукту та забезпечити можливість його подальшого розширення і масштабування [6].

Для правильної роботи з файлами та мультимедіа важливо забезпечити можливість завантаження і зберігання файлів. Користувач повинен мати змогу самостійно завантажувати зображення, працювати з текстом, отримувати та зберігати готові файли. У таких програмах дуже важливо враховувати розміри файлів та інші обмеження, зокрема формати, швидкість процесу обробки та безпеку зберігання даних. Якщо робота з файлами організована неправильно, це дуже часто призводить до серйозних помилок під час створення контенту і може знизити ефективність роботи системи в цілому.

Дуже важливою для такої вебсистеми є база даних, тому що саме вона використовується для зберігання інформації про користувачів, їхні проєкти, параметри генерації, історію завантажень та результати обробки. База даних є ключовою частиною системи, адже вона зберігає не лише технічні дані, а й забезпечує можливість роботи з особистим кабінетом користувача, де міститься список проєктів та історія його активності. Проєктування бази даних включає визначення структури даних, створення зв'язків між об'єктами та вибір способу їх обробки [6].

Для роботи з мультимедійними даними можуть використовуватися спеціальні програми. Під час створення відео потрібно з'єднувати зображення та синхронізувати аудіо. Такі операції дуже часто виконуються на сервері, оскільки вони потребують значної обчислювальної потужності та доступу до програм для редагування файлів. Це може суттєво зменшити навантаження на пристрій користувача і забезпечити якість відео незалежно від пристрою або браузера, яким він користується.

Не менш важливою є можливість масштабування вебдодатку. Якщо кількість користувачів зростає, система повинна витримувати більше навантаження, обробляти більшу кількість запитів, зберігати більше файлів і виконувати більше операцій. Для сучасних застосунків такий підхід є дуже важливим, особливо для систем, які працюють з мультимедійними файлами, адже це може створювати значне навантаження на сервер.

Не менш важливою є безпека додатку. Система має перевіряти дані, які вводить користувач, контролювати тип завантажених файлів, обмежувати доступ до особистих проєктів та захищати інформацію кожного користувача. Для мультимедійних сервісів це дуже актуально.

Таким чином, сучасні технології дозволяють створювати функціональні та адаптивні вебдодатки, які можуть поєднувати користувацький інтерфейс, серверну частину, модулі для роботи з аудіо та відео, а також базу даних. Для створення такої системи потрібно мати адаптивний інтерфейс, серверну логіку,

підтримку завантаження файлів та можливість подальшого розширення. Саме комбінація всіх цих елементів забезпечує ефективну роботу додатку.

1.4 Аналіз існуючих сервісів та аналогів

На сьогоднішній день існує багато сервісів та існуючих рішень для генерації мультимедійного контенту. Ознайомлення з існуючими рішеннями, які виконують подібні функції, може дати уявлення про роботу таких вебдодатків та допоможе визначити його доцільність. Проте, основна відмінність таких сервісів від розроблюваного вебдодатку – це те що вони пропонують єдиний напрямок роботи, тобто роблять окремо або синтез мовлення та генерація голосу, або генерація відео.

Одним із найбільш популярних та якісних сервісів у сфері синтезу мовлення є ElevenLabs. Його основний функціонал – створення якісного голосу на основі тексту. Сервіс дозволяє генерувати голос багатьма мовами світу, встановлювати параметри голосу, обирати мовну модедь. Також можна створити власний голос, навіть на основі свого. Для цього в модель треба завантажити фрагмент власного голосу і після обробки – можна отримати результат.

Приклад інтерфейсу сервісу ElevenLabs наведено на рис. 1.1.

Рисунок 1.1 - Інтерфейс сервісу ElevenLabs

Цей сервіс – хороший приклад синтезу голосу для розроблюваного вебдодатку. Проте цей сервіс не може створити повне відео [10].

Ще один приклад – це Synthesia. Synthesia – це платформа для створення відео за допомогою штучного інтелекту. Synthesia створює відео за допомогою, віртуальних аватарів і озвучення. Користувач вводить текст або сценарій, а система сама робить відео, де аватар читає цей матеріал. Інтерфейс Synthesia показаний на рис. 1.2.

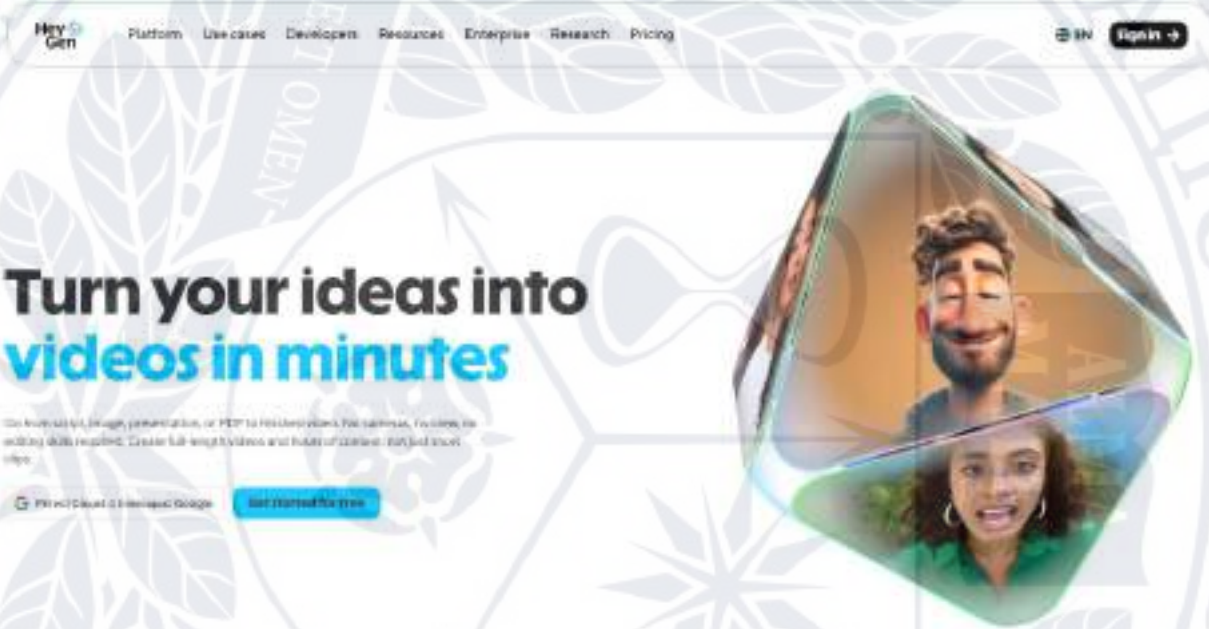

Рисунок 1.2 - Інтерфейс платформи Synthesia

Цей метод допомагає створювати навчальні, корпоративні і презентаційні відео. Сервіс в основному створює відео з аватарами, а не збирає ролики з зображень та автоматично створеного аудіо [11].

Canva - відомий онлайн-інструмент для створення відео. Canva дозволяє створювати дизайни, презентації, рекламні матеріали і відео. Canva має шаблони, дозволяє додавати зображення, анімації, аудіо. Приклад інтерфейсу Canva показаний на рис. 1.3.

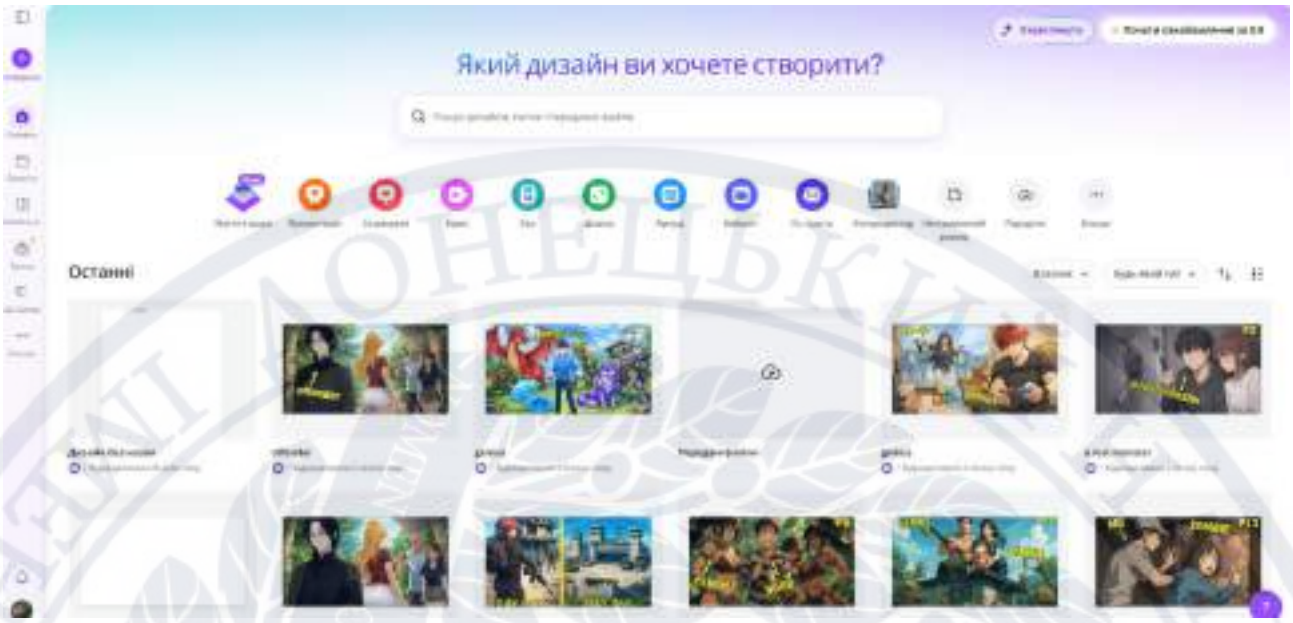

Рисунок 1.3 - Інтерфейс сервісу Canva

Canva має простий інтерфейс та велику кількість готових зразків, проте для автоматичного поєднання тексту, озвучення та зображень у певному сценарії користувачеві все одно необхідно виконувати частину кроків вручну [12].

Іншим схожим сервісом є Kapwing – онлайн-платформа, яка дозволяє творити та редагувати відео. На цій платформі є інструменти для перетворення тексту у відео, додавання субтитрів, озвучення, зображень, музики та інших складників. Приклад інтерфейсу Kapwing показано на рис. 1.4.

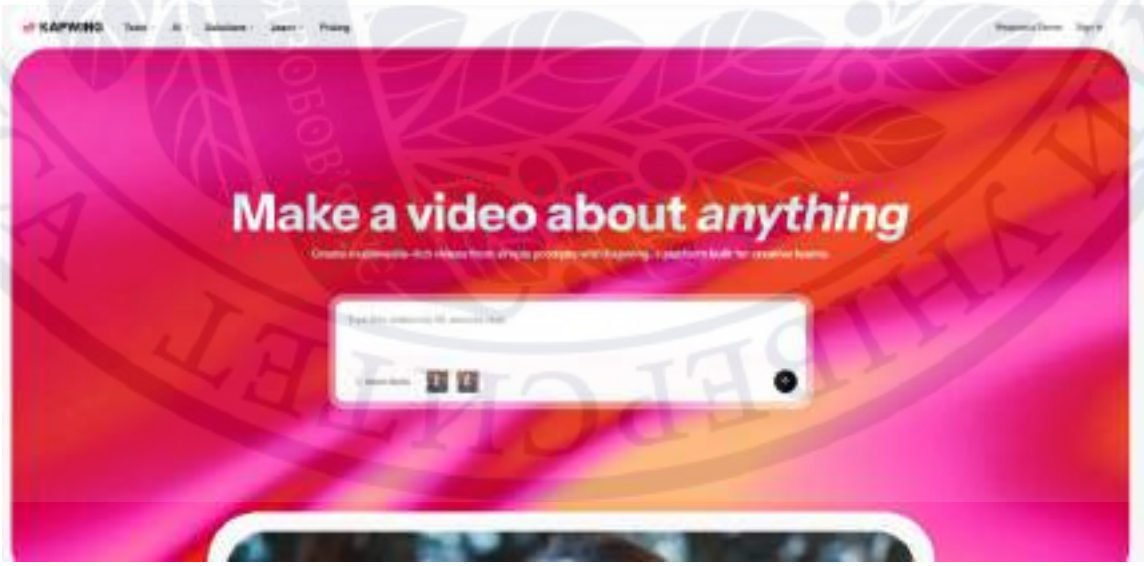

Рисунок 1.4 - Інтерфейс сервісу Kapwing

Kapwing спрямований на швидке створення контенту для соціальних мереж, реклами та коротких відео форматів. Водночас функціонал такого сервісу є досить об'ємним, тому користувачеві потрібно самостійно опрацьовувати багато параметрів редактора, щоб досягти потрібного результату.

Для більш наочного порівняння головні наявні сервіси та їхні особливості наведено у табл. 1.1.

Таблиця 1.1 - Порівняння існуючих сервісів для створення відео контенту та розроблюваного вебдодатку

Сервіс	Основне призначення	Переваги	Обмеження
ElevenLabs	Синтез мовлення з тексту	Якісне звучання, вибір голосів, API	Не формує повноцінне відео з фото
Synthesia	Генерація відео з AI-аватарами	Аватари, озвучення, шаблони	Орієнтація на аватарні відео
Canva	Створення дизайнів і відео	Простий інтерфейс, шаблони	Частина дій виконується вручну
Kapwing	Онлайн-редагування та генерація відео	Багато інструментів для контенту	Широкий функціонал може ускладнювати простий сценарій
Розроблюваний вебдодаток	Озвучування тексту та генерація відео з фото	Об’єднання озвучення і відео в одному середовищі	Залежність від зовнішнього сервісу синтезу мовлення

Таблиця1.1 показує, що кожен розглянутий сервіс виконує свою задачу і має свій набір можливостей. ElevenLabs – інструмент для синтезу мови. Synthesia – інструмент для створення відео з аватарами. Canva – інструмент для дизайну і відео за шаблоном. Kapwing – інструмент для онлайн-редагування відео. Але жоден інструмент не дозволяє просто генерувати відео з тексту, голосу і власних зображень.

Розроблюваний вебдодаток зібрав усі потрібні функції в одному вікні. Його перевага полягає у простоті робочого сценарію: користувач вводить текст, обирає параметри озвучення, завантажує зображення, після чого система автоматично формує готовий відеофайл. Програма сама створює готовий відеофайл. Все робиться автоматично, без зайвих кроків.

Отже, аналіз поточних сервісів показав, що на ринку вже є ефективні інструменти для створення звукового синтезу і відео. Проте більшість інструментів працює лише на окремих етапах або має надмірно широкий функціонал. Тому доцільно розробити спеціальну програму, яка автоматично озвучує текст і створює відео з зображень, які завантажують користувачі.

1.5 Постановка задачі

По завершенню аналізу актуальності теми та основних методів для творення мовлення, генерації відеоряду та сучасних вебтехнологій постає практичне завдання - розробити вебдодаток, що поєднує процес автоматичного озвучення тексту та створення відеофайлу на підставі зображень, завантажених користувачем. Потреба розробки такого програмного продукту спричинена тим, що більшість наявних сервісів вирішують лише окремі аспекти цієї задачі: деякі концентруються на синтезі мовлення, інші - на відеомонтажі чи створенні відео за шаблонами.

Основна концепція проєкту полягає у розробці єдиної вебплатформи, де користувач матиме змогу виконати увесь ланцюжок роботи з мультимедійними матеріалами: ввести текст, згенерувати голосове повідомлення, завантажити світлини й отримати готовий відеофайл. Такий підхід прискорить процес

створення відеоматеріалів та зменшить потребу застосовувати декілька окремих програм чи онлайн-сервісів.

Завдання проєкту - створити вебзастосунок, який автоматично озвучує текст та генерує відео. Він дає змогу користувачеві легко взаємодіяти із системою, самостійно створювати аудіо на основі введеної ним текстової інформації, а потім поєднувати це аудіо з фотографіями, щоб отримати фінальний відеофайл.

Для досягнення поставленої мети варто виконати такі завдання:

1. проаналізувати предметну сферу створення мультимедійного контенту;
2. дослідити головні методи створення мовлення та відеоряду.
3. розглянути наявні сервіси й аналоги;
4. обґрунтувати вибір технологій для впровадження вебдодатку;
5. розробити архітектуру системи;
6. реалізувати серверний компонент, що обробляє запити, генерує аудіо та відео.
7. реалізувати клієнтський компонент, що забезпечує взаємодію користувача із системою, без штучного стилю:
8. реалізувати клієнтський компонент, який дозволяє користувачеві працювати з системою.
9. створити базу даних для зберігання користувачів, проєктів та підсумків генерації.
10. забезпечити можливість завантаження світлин і отримання фінального відеофайлу.
11. провести випробування функціональних потужностей системи.

Функціональні вимоги до розроблюваного вебдодатку наведено в табл. 1.2.

Таблиця 1.2 - Функціональні вимоги до вебдодатку

№	Функція	Опис
1	Введення тексту	Користувач має можливість ввести текст для подальшого озвучування
2	Генерація відео	Система автоматично перетворює введений текст у звуковий файл
3	Завантаження зображень	Користувач може додати зображення, які будуть використані у відео
4	Генерація відео	Система об’єднує зображення та аудіофайл у готовий відеофайл
5	Завантаження результату	Користувач може отримати готове відео після завершення обробки
6	Збереження проєктів	Система може зберігати інформацію про створені користувачем матеріали
7	Особистий кабінет	Користувач може переглядати власні проєкти та результати генерації

Таблиця 1.2 показує, що система автоматизує створення відео-змісту. Користувач має робити всі кроки сам, без знань монтажу чи програмування, тому дуже важливо забезпечити простоту інтерфейсу.

Окрім функціональних вимог, потрібні і нефункціональні вимоги. Нефункціональні вимоги показують, як система працює. Нефункціональні вимоги включають зручність, стабільність, швидкість, адаптивність інтерфейсу, безпечне збереження даних і можливість розширювати функції. Таблиця 1.3 містить основні нефункціональні вимоги. Нефункціональні вимоги визначають якість роботи вебдодатку та забезпечують стабільність, швидкодію, зручність використання, адаптивність інтерфейсу й безпечне збереження даних під час роботи системи.

Таблиця 1.3 - Нефункціональні вимоги до вебдодатку

№	Вимога	Опис
1	Зручність використання	Інтерфейс має бути зрозумілим для користувача
2	Адаптивність	Вебдодаток має коректно працювати на різних пристроях
3	Швидкодія	Система повинна забезпечувати прийнятний час генерації аудіо та відео
4	Надійність	Помилки під час обробки файлів мають оброблятися коректно
5	Маштабованість	Архітектура має дозволяти розширення функціоналу
6	Безпека	Користувацькі дані та файли мають бути захищені від несанкціонованого доступу

З таблиця 1.3 видно, що вебдодаток має бути корисним, зручним і надійним. Крім того, дуже важливо забезпечити роботу вебдодатку без збоїв. Оскільки система працює з текстом, аудіо, зображеннями і відео, важливо правильно перевіряти дані, контролювати формати файлів і обробляти помилки.

Вебдодаток має кілька частин: клієнтську частину, серверну частину, базу даних, модуль синтезу мовлення і модуль генерації відео. Клієнтська частина дозволяє користувачу працювати з системою. Серверна частина приймає запити, керує файлами, підключається до зовнішнього сервісу синтезу мовлення і запускає процес створення відео. База даних зберігає дані про користувачів, проєкти та результати генерації. Модуль генерації відео збирає зображення і аудіо в один відеофайл.

Користувач відкриває вебдодаток, вводить текст, вибирає параметри озвучення, завантажує зображення. Після цього система робить аудіофайл,

поєднує його з зображеннями і збирає відео. Користувач може переглянути результат або завантажити його.

Реалізація такого програмного продукту дасть змогу поєднати синтез мовлення, роботу із зображеннями, обробку відео та збереження результату. Це створить основу для подальшої практичної реалізації системи, яка буде розглянута у другому розділі роботи.

РОЗДІЛ 2.РОЗРОБКА ВЕБДОДАТКУ ДЛЯ ОЗВУЧУВАННЯ ТЕКСТУ ТА ГЕНЕРАЦІЇ ВІДЕО

2.1 Загальна архітектура системи

Для розроблюваного вебдодатку для автоматичного озвучування тексту та створення відео, потрібно створити чітку архітектуру, яка дозволяє взаємодіяти різним елементам, таким як: інтерфейс користувача, серверна частина, база даних, зовнішній сервіс для перетворення в тексту в голос та модулем для обробки відео. Вебдодаток працює із різними типами даних, такими як текст, аудіо, зображення, тому архітектура має бути простою, зрозумілою і в той самий час ефективною. Крім того, побудова такої архітектури дозволить легко додавати нові зміни та маштабувати вебдодаток в майбутньому.

Для розробки вебдодатку було використано клієнт-серверну архітектуру. Цей підхід передбачає розділення додатку на дві основні частини: клієнтську частину для взаємодії користувача з інтерфейсом додатку, та серверну частину, на стороні якої виконується обчислення та логіка роботи програми. Через клієнта запити відправляються на сервер, де відбувається обробка, аналогічним чином відбувається взаємодія із базою даних та іншими зовнішніми сервісами. Після завершення обробки на стороні сервера – користувачеві повертається результат. Такий підхід дає змогу окремо розглядати інтерфейс користувача від внутрішньої логіки програми і широко застосовується в сучасній розробці [6]. Така архітектура забезпечує гнучкість розробки та дозволяє незалежно вдосконалювати клієнтську і серверну частини системи. Крім цього, використання клієнт-серверного підходу спрощує масштабування вебдодатку та інтеграцію із зовнішніми API і сервісами.

Архітектура розроблюваної системи зображена на рис. 2.1.

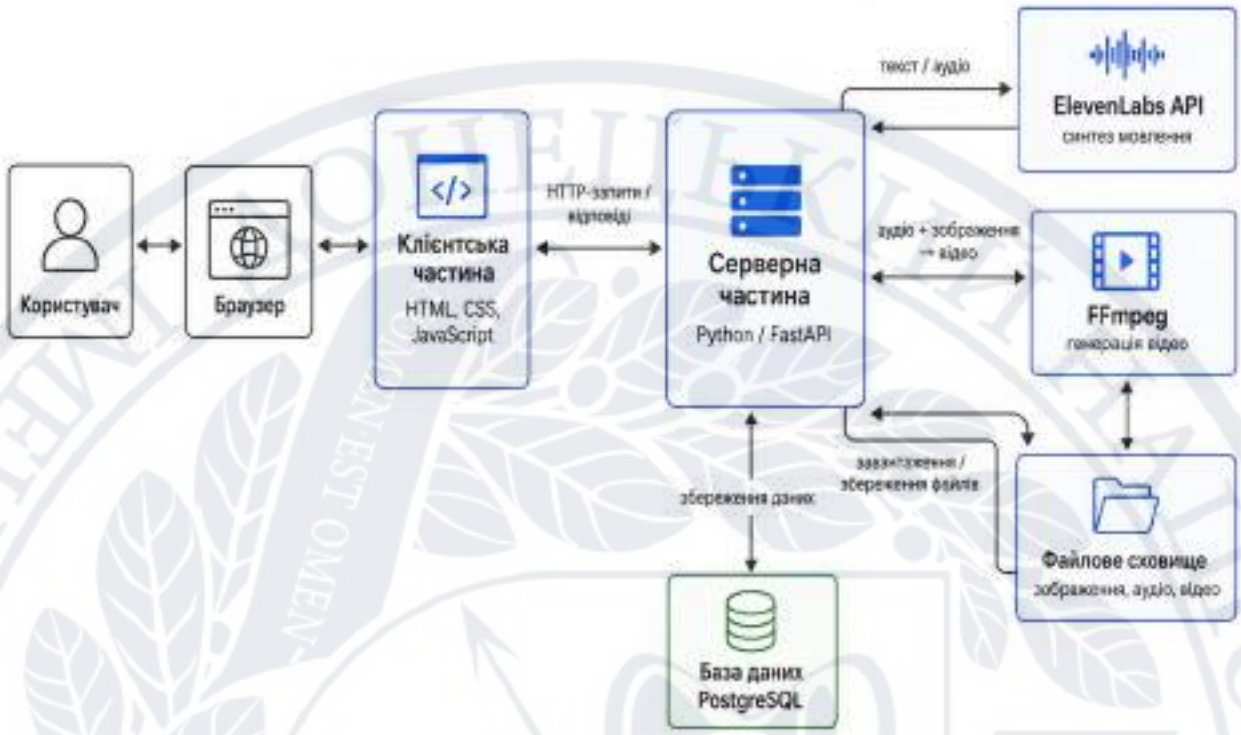

Рисунок 2.1 - загальна структура вебдодатку для озвучування тексту та створення відео

Структура вебдодатку складається з низки ключових елементів: фронтенд (клієнтська частина), бекенд (серверна частина), база даних на базі PostgreSQL, сервіс для генерування голосу від ElevenLabs, компонент для створення відео із застосуванням FFmpeg та сховище для збереження як завантажених, так і згенерованих файлів. Кожна із цих складових виконує унікальну роль у загальному функціонуванні системи.

Клієнтська частина вебзастосунку відповідає за візуалізацію користувацького інтерфейсу. Через цей інтерфейс користувач має змогу ввести текст для озвучення, обрати бажані голосові налаштування, завантажити візуальні матеріали (зображення), ініціювати процес створення аудіо чи відео, а також переглянути кінцевий результат. Ключовою місією фронтенду є забезпечення інтуїтивно зрозумілої та наочної взаємодії користувача з функціоналом системи. Для досягнення цього використовуються стандартні

вебтехнології - HTML для структури, CSS для оформлення та JavaScript для інтерактивності елементів [7, 8].

Серверна частина виступає ядром системи. Вона приймає запити, що надходять від клієнта, валідує отримані дані, здійснює обробку тексту, керує завантаженням візуальних даних, взаємодіє із зовнішнім сервісом для синтезу мовлення та ініціює процес рендерингу відео. У розробленій програмі бекенд реалізовано на мові Python із застосуванням фреймворку FastAPI. Такий підхід дає змогу створити чіткий інтерфейс для обміну даними між клієнтом та сервером [14, 15].

СУБД PostgreSQL задіяна для персистентного зберігання структурованих відомостей. В контексті цього вебзастосунку вона фіксує дані про облікові записи, створені користувачем проєкти, параметри озвучення, метадані про імпортовані зображення, а також інформацію про готові аудіо- та відеофайли. Наявність бази даних уможливлює функціонування персонального кабінету користувача, збереження історії всіх робіт та забезпечення можливості повторного доступу до згенерованих матеріалів [17].

Одним із важливих зовнішніх елементів системи є сервіс для синтезу мовлення, яким у нашому випадку стала платформа ElevenLabs [16]. Серверна логіка надсилає текстові дані разом із обраними голосовими параметрами до цього сервісу, отримуючи натомість готовий аудіофайл. Цей аудіофайл згодом інтегрується як звукова доріжка у фінальне відео. Такий підхід дозволяє уникнути необхідності розробки власної складної системи для синтезу мовлення, замість цього використовуючи інтеграцію перевіреного стороннього рішення.

Модуль, відповідальний за відеогенерацію, займається створенням кінцевого медіафайлу. Його провідне завдання полягає в компіляції завантажених користувачем зображень та згенерованого аудіо в єдиний відеоконтейнер. Для цього використовується інструмент FFmpeg, який дає можливість встановити тривалість аудіодоріжки, розрахувати послідовність відображення зображень та сформувати відеоряд у необхідному контейнері.

Файлове сховище слугує для розміщення як тимчасових, так і постійних файлів, які були завантажені або створені в процесі роботи. Серед таких файлів - зображення, надані користувачем, аудіофайли, отримані після озвучення тексту, та фінальні відеоролики. Згодом посилання та метадані про ці файли можуть бути збережені в базі даних, тоді як самі дані фізично зберігаються у відповідних директоріях на сервері або у віддаленому сховищі.

Основні компоненти архітектури системи наведено у табл. 2.1.

Таблиця 2.1 - Основні компоненти архітектури вебдодатку

Компонент системи	Призначення
Клієнтська частина	Забезпечує взаємодію користувача ізвеб додатком
Серверна частина	Обробляє запити, керує логікою генерації аудіо та відео
База даних PostgreSQL	Зберігає інформацію про користувачів, проєкти та результати генерації
ElevenLabs	Виконує синтез мовлення на основі введеного тексту
FFmpeg	Формує відеофайл на основі аудіо та зображень
Файлове сховище	Зберігає завантажені зображення, аудіо та готові відеофайли

Згідно з даними у таблиці 2.1, кожен компонент функціонує незалежно, проте всі вони функціонально об'єднані у загальному ланцюжку створення мультимедійного продукту. Такий поділ праці спрощує процес проєктування системи, оскільки дає змогу ізольовано розробляти, тестувати та вдосконалювати окремі блоки.

Робота користувача розпочинається із запуску веб-інтерфейсу. Від нього вимагається ввести текстовий вміст, визначити налаштування для голосового супроводу та додати візуальні елементи. Далі, клієнтський програмний модуль передає ці відомості на обчислювальний вузол (сервер). Сервер, у свою чергу,

здійснює валідацію вхідних даних, скеровує текст до служби синтезу мовлення, приймає згенерований аудіофайл і розпочинає процес формування відеоряду. По завершенні усіх стадій обробки кінцевий відеоматеріал стає доступним для користувача, котрий має можливість його переглянути або зберегти на свій пристрій.

Сценарій взаємодії користувача з системою краще представити у вигляді схеми, яка зображена на рис. 2.2.

Рисунок 2.2 - сценарій взаємодії користувача з вебдодатком

Подібна архітектурна схема дає змогу логічно розмежувати відповідальність між різними елементами системи. Клієнтська складова займається виключно відображенням інформації та надсиланням запитів, серверна ж частина зосереджена на опрацюванні отриманих даних і комунікації з модулями, що продукують ці дані. База даних забезпечує зберігання

впорядкованої інформації, тоді як зовнішні утиліти виконують вузькоспеціалізовані завдання.

Ключовою перевагою обраного архітектурного рішення є можливість його поетапного масштабування. Скажімо, згодом стане можливою інтеграція підтримки додаткових мов озвучення, розширення вибору голосів, впровадження нових відеошаблонів, функціоналу субтитрів, персонального кабінету для користувачів, системи тарифікації або збереження історії попередніх генерацій. Через те, що архітектура розділена на незалежні блоки, оновлення можна впроваджувати ітеративно, без необхідності повного реструктурування програми.

Отже, загальна структура вебзастосунку базується на принципі "клієнт-сервер", включаючи такі складові: інтерфейс взаємодії з користувачем, логіку сервера, сховище даних, сервіс синтезу мовлення, компонент для створення відео та сховище файлів. Така конфігурація уможливлює реалізацію цілісного циклу створення мультимедійного контенту - від введення текстової інформації аж до отримання фінального відеофайлу.

2.2 Вибір та обґрунтування технологій

Для створеннння вебдодатку, котрий би автоматично озвучував поданий текст та створював відеоряд, було підібрано певний набір технологічних рішень. Вони дають змогу розробляти інтерфейс для користувача, вибудовувати логіку роботи серверної частини, оперувати з даними у сховищі, взаємодіяти із зовнішнім сервісом для перетворення тексту на мовлення, а також займатися опрацюванням мультимедійних об'єктів. При доборі цих інструментів увага приділялася легкості самого процесу розробки, зручності подальшого обслуговування, можливості оперувати файлами, наявності підтримки програмних інтерфейсів (API), стійкості роботи системи, здатності до нарощування потужності та придатності для майбутнього збагачення функціональністю.

Проєктування будь-якої інформаційної системи передбачає чітке окреслення її ключових компонентів, механізмів їхньої взаємодії, способу впорядкування даних, а також програмних засобів, необхідних для реалізації поставлених завдань [6]. У процесі створення вебдодатку необхідно гарантувати зв'язок кінцевого користувача з візуальною оболонкою, трансфер даних на сервер, генерацію аудіоряду, формування фінального відеофайлу та архівацію отриманих результатів. З огляду на це, обраний технологічний комплект мусить охоплювати як розробку клієнтських інтерфейсів, так і весь цикл обробки мультимедіа.

Перелік ключових технологій, застосованих при розробці вебзастосунку, представлено у табл. 2.2.

Таблиця 2.2 - Основні технології, використані у вебдодатку

Технологія	Опис
HTML	Формування структури сторінок вебдодатку
CSS	Візуальне оформлення інтерфейсу користувача
JavaScript	Реалізація інтерактивності клієнтської частини
Python	Основна мова програмування серверної частини
FastAPI	Створення API та обробка запитів користувача
ElevenLabs API	Автоматичне озвучування тексту
FFmpeg	Генерація відео на основі аудіо та зображень
PostgreSQL	Збереження користувачів, проєктів і результатів генерації

Як видно із таблиці 2.2, обрані технології охоплюють усі ключові складові вебзастосунку. Клієнтський бік (frontend) відповідає за взаємодію з користувачем, тоді як серверний бік (backend) займається обробкою запитів та виконанням процесів генерації. База даних призначена для зберігання структурованої інформації, а зовнішні сервіси забезпечують синтез мовлення та формування відеофайлів.

Для реалізації клієнтської складової було обрано HTML, CSS та JavaScript. HTML слугує для побудови структури вебсторінок, розміщення інтерфейсних елементів, як-от форми, кнопки, поля для вводу даних, блоки для завантаження медіа та відображення кінцевого результату. CSS регулює візуальне оформлення, керує розташуванням елементів, кольоровою гамою та адаптивністю сайту до різних пристроїв. JavaScript ж відповідає за динамічну поведінку сторінок, обробку дій користувача, валідацію введеної інформації та надсилання запитів до серверної частини [7, 8, 9, 19].

Застосування HTML, CSS та JavaScript є доцільним, оскільки це фундаментальні технології для конструювання сучасних веб-інтерфейсів, які підтримуються всіма основними браузерами. Для розроблюваної системи це критично важливо, адже користувач матиме можливість працювати з вебзастосунком без необхідності інсталяції додаткового програмного забезпечення. Через браузер він зможе вводити текст, задавати параметри озвучення, завантажувати зображення, ініціювати процес генерації та отримувати готовий результат.

Для функціонування серверної логіки була обрана мова програмування Python. Вона широко застосовується для розробки вебсервісів, автоматизації роботи з даними, оперування файлами та інтеграції із зовнішніми програмними інтерфейсами (API). У межах цього проєкту Python використовується для прийому запитів від клієнта, обробки текстового контенту, роботи з завантаженими файлами, взаємодії зі службою синтезу голосу, запуску відеопродукції та з'єднання з базою даних [14].

Як вебфреймворк для серверної частини задіяно FastAPI. Цей інструмент дає змогу швидко створювати API, обробляти HTTP-запити, отримувати дані від клієнта та повертати результат у зручному форматі. Для розробки вебдодатку це є надзвичайно важливим, оскільки комунікація між інтерфейсом користувача та серверною логікою здійснюється через запити. Додатково, FastAPI чудово підходить для побудови серверної частини, що вимагає роботи із зовнішніми сервісами, файловою системою та базою даних [15].

Для автоматичного дублювання тексту залучається API сервісу ElevenLabs. Цей інструмент трансформує текстові дані в аудіофайл, який згодом слугує звуковим наповненням для відеоряду. Усередині вебзастосунку сервер передає текст та обрані голосові параметри на сервери ElevenLabs, а потім отримує готовий аудіофайл. Такий підхід є раціональним, бо відпадає потреба у розробці власної складної системи синтезу мовлення; достатньо інтегрувати готовий спеціалізований сервіс [10].

Для формування фінальних відеофайлів використовується утиліта FFmpeg. Це потужний програмний комплекс для маніпуляцій з відео, аудіоданими та зображеннями. У розроблюваному вебзастосунку FFmpeg застосовується для комбінування згенерованого аудіоряду та зображень, наданих користувачем, в єдиний відеопотік. Це дозволяє автоматизувати процес, який зазвичай вимагав би ручного редагування у спеціалізованому програмному забезпеченні [16].

Для персистентного зберігання даних у системі обрано базу даних PostgreSQL. Вона необхідна для фіксації інформації про користувачів, створені проєкти, параметри генерації, а також шляхи до завантажених зображень, аудіо- та відеофайлів. Використання бази даних дає змогу реалізувати функціонал особистого кабінету, зберігати історію згенерованих матеріалів та надавати можливість повторного доступу до результатів. PostgreSQL є відповідним вибором для подібної системи, оскільки він відмінно справляється з роботою над структурованими даними та зв'язками між таблицями [17].

Перевага обраного технологічного стеку полягає у взаємодоповнюваності його складових. Клієнтська частина бере на себе введення даних та візуалізацію результату, серверна логіка обробляє запити, API ElevenLabs формує голосовий супровід, FFmpeg створює відео, PostgreSQL відповідає за збереження інформації. Така архітектура дозволяє незалежно розробляти, тестувати та вдосконалювати кожен компонент вебзастосунку.

Таким чином, вибір технологій для створення вебзастосунку чітко відображає функціональні вимоги системи. HTML, CSS та JavaScript формують інтуїтивно зрозумілий інтерфейс, Python у зв'язці з FastAPI виконує обчислення

на сервері, коннектор ElevenLabs забезпечує перетворення тексту у мовлення, FFmpeg відповідає за відеопродукцію, PostgreSQL зберігає увесь масив даних. В результаті такого набору, можна розробити вебзастосунок, що автоматизує процес конвертації тексту у візуалізований контент.

2.3 Реалізація клієнтської частини вебдодатку

2.3.1 Структура клієнтської частини проєкту

Клієнтська частина вебдодатку слугує ключовим інтерфейсом для взаємодії користувача із системою. Через цей компонент особа користувача відкриває головну сторінку, вводить текст для озвучення, здійснює переклад, обирає тембр голосу, завантажує візуальний контент, ініціює генерацію відео та отримує доступ до особистого кабінету. З огляду на це, при проєктуванні клієнтської частини було принципово важливим забезпечити логічну впорядкованість файлової структури, чітке розмежування функціональних сторінок та зручність подальшої експлуатації інтерфейсу.

У межах реалізованого проєкту клієнтська складова розміщена у теці `frontend`. Тут скупчено HTML-документи вебдодатку, а також виділено окрему директорію `assets`, яка містить файли оформлення (стилі) та скрипти мовою JavaScript. Такий архітектурний підхід сприяє сегментації структури проєкту, що, своєю чергою, полегшує роботу з програмним кодом та підтримує організованість процесу розробки. HTML керує розміткою, CSS відповідає за візуальне представлення, тоді як JavaScript забезпечує динамічні можливості та обробку дій користувача [7-9].

Основним вхідним пунктом для відвідувачів є файл `index.html`. Ця сторінка є "серцем" програми VidVoice і вміщує головні структурні блоки: навігаційну панель, основну робочу зону, блок із переліком можливостей, модуль для введення та трансляції тексту, секцію для завантаження графічних даних, модуль генерації відеоряду, блок із покроковим описом функціоналу, закличну секцію та футер.

Окрім центральної сторінки, клієнтська частина містить окремі розділи, призначені для роботи з персональною інформацією. Сторінка `profile.html` використовується для перегляду налаштувань облікового запису, `projects.html` – для керування створеними проєктами, `credits.html` – для моніторингу витрат кредитів чи лімітів символів під час озвучення, а `settings.html` – для коригування параметрів роботи застосунку. Наявність таких спеціалізованих сторінок дозволяє трансформувати систему з простого інструменту для створення аудіо/відео у повноцінний вебсервіс з функціоналом особистого кабінету.

Структуру клієнтської частини проєкту подано в табл. 2.3.

Таблиця 2.3 - Структура клієнтської частини вебдодатку

Файл або директорія	Призначення
frontend/index.html	Головна сторінка вебдодатку та основні функціональні блоки
frontend/profile.html	Сторінка профілю користувача
frontend/projects.html	Сторінка перегляду та створення проєктів
frontend/credits.html	Сторінка контролю використаних кредитів або символів
frontend/settings.html	Сторінка налаштувань користувача
frontend/assets/css/styles.css	Файл стилів для оформлення інтерфейсу
frontend/assets/js/api.js	Функції для взаємодії з API серверної частини
frontend/assets/js/common.js	Спільна логіка для сторінок вебдодатку
frontend/assets/js/home.js	Логіка головної сторінки
frontend/assets/js/profile.js	Логіка сторінки профілю
frontend/assets/js/projects.js	Логіка сторінки проєктів
frontend/assets/js/credits.js	Логіка сторінки кредитів
frontend/assets/js/settings.js	Логіка сторінки налаштувань

Згідно з даними, представленими у таблиці 2.3, клієнтська складова побудована на модульному принципі. Кожна вебсторінка (HTML) має своє чітко

визначене призначення, а програмний код на JavaScript рознесено відповідно до функціональності цих сторінок. До прикладу, логіка, що відповідає за головну сторінку, міститься у файлі `home.js`, тоді як обробка відображення та взаємодії на сторінці проєктів реалізована у `projects.js`. Окремо виділено модуль `api.js`, який цілком присвячений комунікації із сервером – надсиланню запитів. Такий підхід сприяє уникненню дублювання коду та спрощує процес взаємодії з бекендом.

Візуальне оформлення інтерфейсу користувача втілено у файлі `styles.css`. Саме тут визначено ключові параметри стилізації: палітру кольорів, гарнітури шрифтів, інтервали між елементами, розташування структурних блоків, а також зовнішній вигляд інтерактивних елементів, таких як кнопки, поля для вводу даних, блоки-картки з описом функціоналу тощо. Загальна естетика вебдодатку розроблена у стилістиці, що гармоніює з концепцією мультимедійного сервісу, при цьому акцентуючи увагу на ключових діях користувача, зокрема на ініціації процесів синтезу голосу та формування відеоконтенту. Крім того, у таблиці стилів передбачено адаптивну верстку, що забезпечує коректне відображення інтерфейсу на різних типах пристроїв, включно з мобільними телефонами та планшетами. Особливу увагу приділено візуальній ієрархії елементів, що дозволяє користувачеві інтуїтивно орієнтуватися в функціоналі системи та швидко знаходити необхідні інструменти.

Для більш наочного пояснення архітектури головної сторінки було розроблено відповідну структурну діаграму, яку можна побачити на рис. 2.3. Ця схема детально ілюструє ключові компоненти інтерфейсу та послідовність їх розміщення у межах цього екрану.

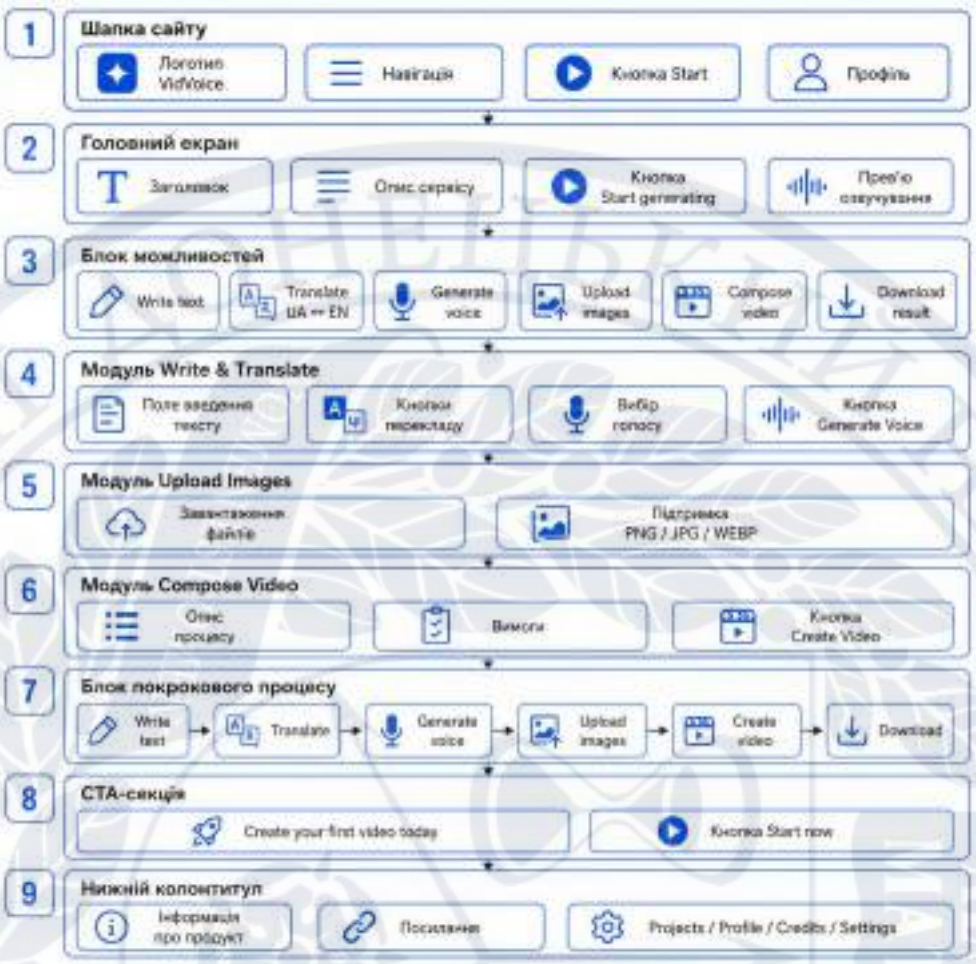

Рисунок 2.3 - будова головної сторінки веб-додатку VidVoice

На рисунку 2.3 видно, що головна сторінка організована логічно: спершу йде верхній елемент сайту (шапка), далі – основний видимий блок (перший екран), слідує секція, що описує функціонал, потім інтегровані модулі для роботи з текстовими, графічними та відеоматеріалами, після цього – блок, що візуалізує покроковий алгоритм, потім заклик до певної дії та завершує все нижній блок (футер). Таке розташування елементів сприяє планомірному засвоєнню користувачем функціоналу вебзастосунку та спрямовує його до виконання ключових операцій.

Отже, клієнтська складова розробки вирізняється акуратною файловою ієрархією та розподілом на ізольовані сторінки і логічні блоки функцій. Центральний файл, index.html, слугує точкою входу до найважливіших опцій вебзастосунку, тоді як допоміжні сторінки утворюють базу для персоналізованого користування. Застосування окремих файлів для

стилістичного оформлення та виконання скриптів значно полегшує супровід візуальної частини, забезпечує гнучкість у трансформації вигляду та масштабуванні можливостей системи.

2.3.2 Розробка головної сторінки вебдодатку

Стартова вітальна сторінка застосунку VidVoice слугує точкою входу для користувачів системи. Цей інтерфейс виконує подвійне завдання: спершу він інформує про призначення веб-сервісу, а потім демонструє ключові функції, що доступні в ньому. Звідси здійснюється перехід до різних робочих блоків, а саме: модулів для синтезу мовлення з тексту, завантаження візуальних матеріалів або генерації відеороликів.

При проектуванні цієї стартової сторінки було застосовано метод поетапного подання відомостей. На початку користувач бачить верхній елемент, який містить навігаційне меню, далі слідує головний екран із лаконічним представленням суті сервісу, потім – блок із переліком функціоналу, робочі блоки для аудіо- та відео-генерації, секція, що описує процес покроково, блок із прямим закликом до дії, і, нарешті, нижній колонтитул. Така архітектура сприяє поступовому засвоєнню охопленого функціоналу програми та орієнтує користувача на виконання першочергових завдань.

У верхній частині сторінки розташовується загальний хедер сайту. У цьому блоці розмістили фірмовий знак VidVoice Studio, навігаційні посилання на основні рубрики, кнопку ініціації роботи та індикатор облікового запису користувача. Навігаційний набір складається з пунктів Home, Audio, Images та Video, кожен з яких відповідає певній функціональній складовій вебдодатка. Це надає користувачеві можливість миттєво перейти до потрібного розділу, уникнувши необхідності тривалого прокручування контенту.

Відразу за шапкою сайту йде секція вітального екрану. Її головна мета — стисло викласти основну концепцію застосунку. Цей розділ містить заголовок, що акцентує увагу на можливості трансформації тексту у голосовий супровід та відеоряд, короткий опис доступних опцій програми, а також кнопку для негайного старту. Вітальний екран доповнено ілюстративним елементом, який

візуалізує процес озвучування: тут відображається вхідний текст, вибір мовних та голосових параметрів, а також фінальний аудіо-вихід. Це дає змогу користувачеві швидко зрозуміти кінцевий результат роботи з інструментом.

Зовнішній вигляд головного екрану вебдодатку зображено на рис. 2.4.

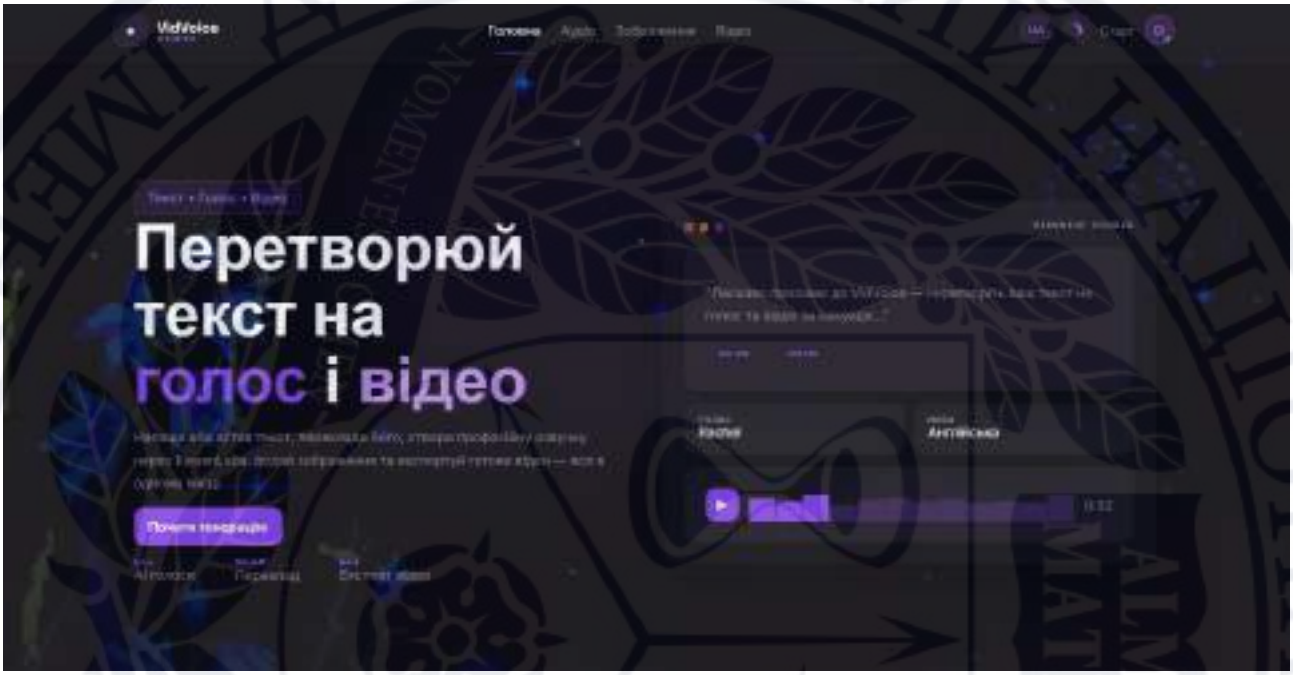

Рисунок 2.4 - Головний екран вебдодатку VidVoice

Ключовим компонентом стартової сторінки слугує секція, присвячена функціональним можливостям. Вона складається із шести окремих карток, де кожна окрема картка детально описує один із кроків, які може виконати користувач: від створення текстового вмісту та його подальшого перекладу, до синтезу мовлення, додавання графічних матеріалів, формування відеоряду, і завершуючи вивантаженням кінцевого продукту. Така візуалізація суттєво полегшує розуміння логіки роботи онлайн-сервісу, подаючи весь цикл взаємодії з платформою у максимально доступному форматі. Візуальне представлення цієї секції функцій відображено на рис. 2.5.

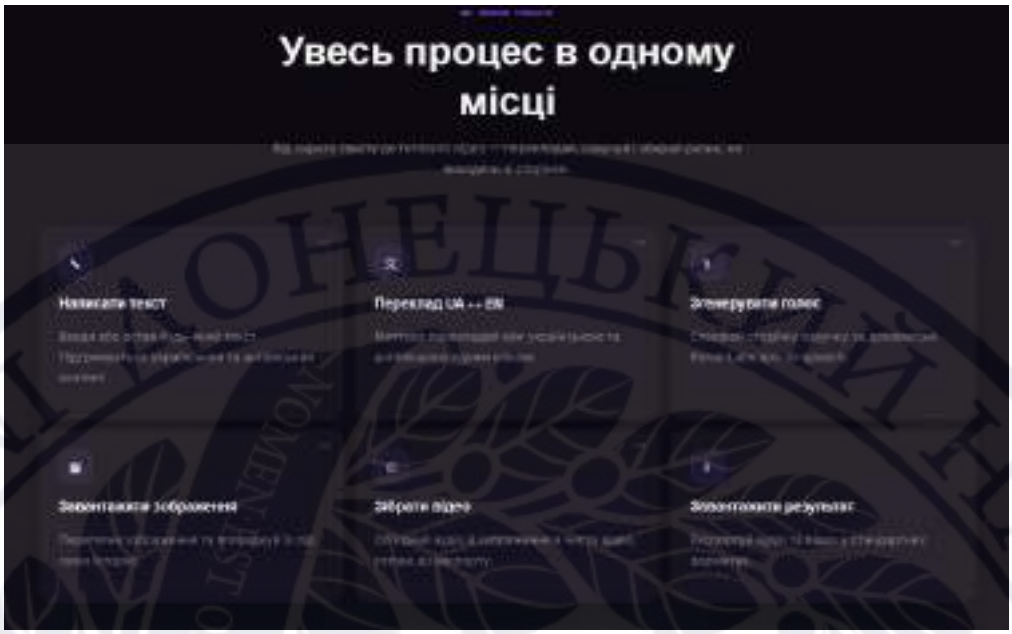

Рисунок 2.5 - Блок можливостей вебдодатку VidVoice

Слідом за інформаційними секціями на цій сторінці розташувалася функціональна одиниця "Write & Translate". Її метою є прийом тексту, його мовний трансфер та можливість обрання аудіовиконавця для озвучення. Користувачеві надається можливість ввести текст або українською, або англійською, за необхідності здійснити переклад, вибрати бажаний тембр голосу зі сформованого переліку та активувати кнопку генерування. Крім генерації, він має можливість завантажити уже згенеровані раніше аудіофайли, аби не генерувати їх повторно. Ця складова частина слугує початковою практичною ланкою у процесі формування мультимедійного продукту в межах вебзастосунку. Візуальне представлення інтерфейсу модуля "Write & Translate" відображено на рис. 2.6. Також інтерфейс модуля містить зрозумілу структуру елементів керування, що мінімізує час взаємодії користувача з системою та зменшує ймовірність помилок під час введення даних. Особлива увага приділена зручності перемикання між режимами введення тексту, перекладу та вибору голосу, що забезпечує логічну послідовність виконання дій. Завдяки цьому процес створення аудіоконтенту стає більш інтуїтивним та ефективним.

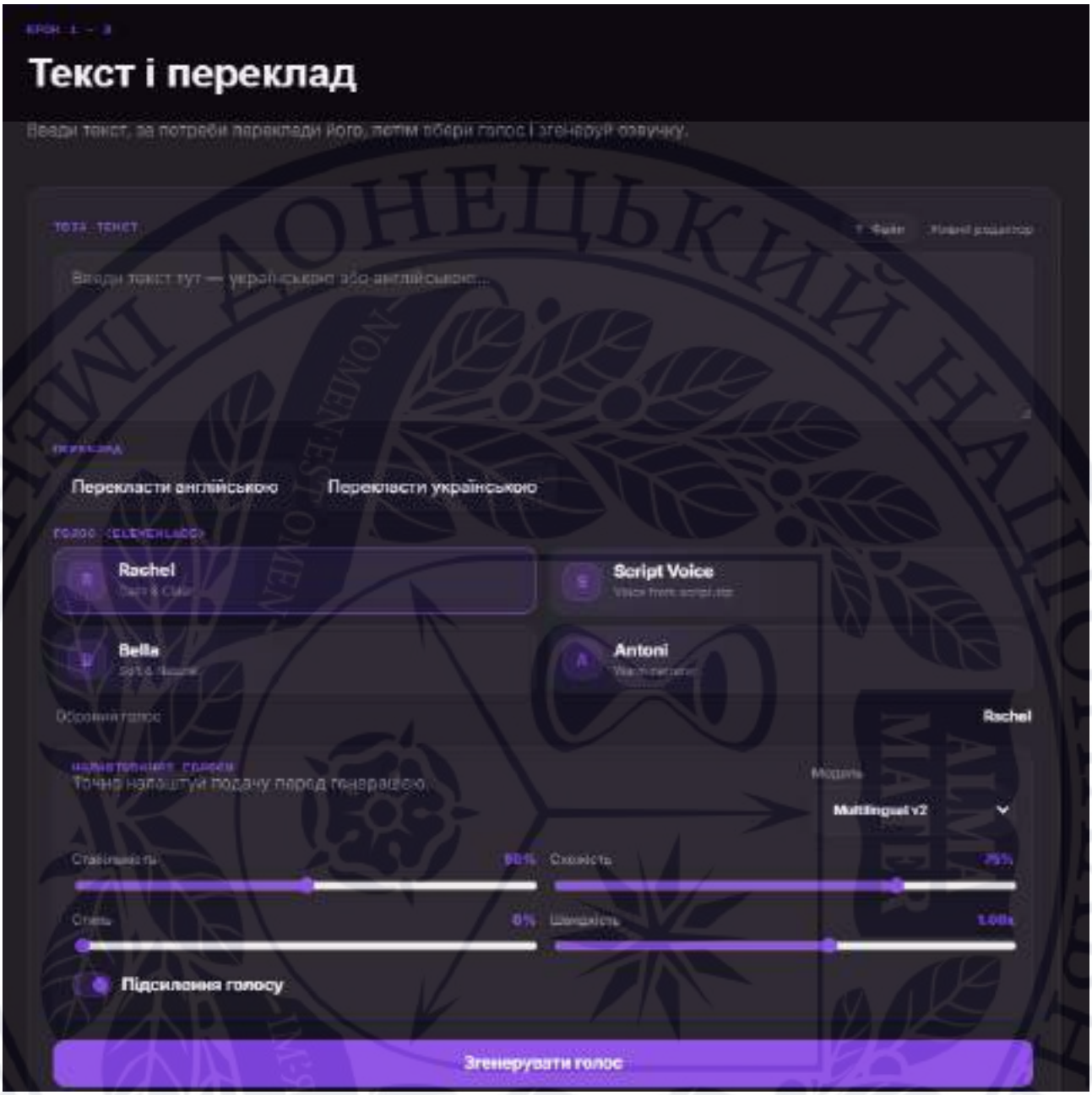

Рисунок 2.6 - Інтерфейс модуля Write & Translate

Як це можна спостерігати на рисунку 2.6, блок "Write & Translate" укомплектований текстовим полем для вводу, елементами керування для ініціювання перекладу, опціями вибору та налаштування голосового супроводу, а також кнопкою для аудіогенерації. Така архітектура є вельми практичною, адже вся необхідна функціональність для підготовки текстового матеріалу й формування аудіоряду зосереджена в межах одного логічного вузла. Це, у свою чергу, мінімізує потребу в навігації між різними екранами, спрощуючи загальний процес отримання озвучки.

Далі на сторінці розташовано секцію під назвою "Upload Images". Цей компонент призначений для імпорту графічних елементів, які планується задіяти у процесі відеопродукції. Вказаний модуль пропонує як механізм перетягування файлів (drag-and-drop), так і можливість стандартного вибору файлів із локального сховища. Крім того, тут чітко зазначено перелік прийнятних розширень для зображень: зокрема, PNG, JPG та WEBP. Візуальне представлення модуля завантаження зображень демонструє рис. 2.7.

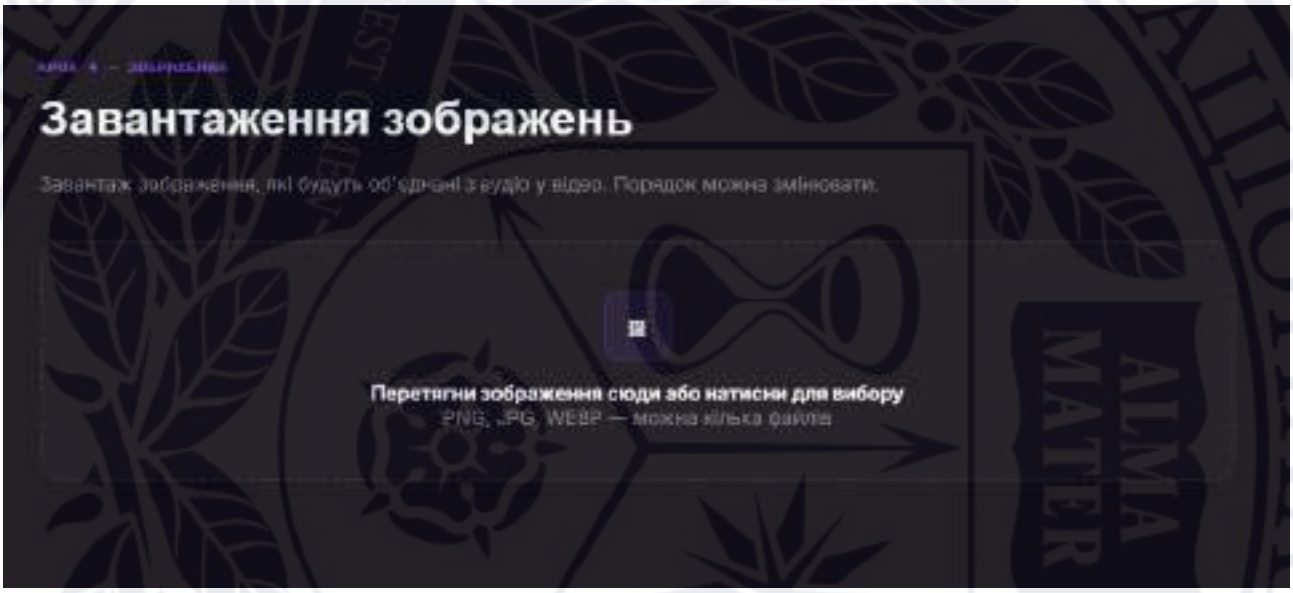

Рисунок 2.7 - Інтерфейс модуля Upload Images

На рисунку 2.7 можна побачити, що компонент завантаження світлин має доволі спрощений вигляд і скеррований на оперативне внесення файлів. Людина має змогу або перетягнути потрібні світлини у виділену зону, або ж обрати їх через стандартне діалогове вікно. Інформація про те, які формати приймаються, мінімізує ризики помилок, скеровуючи користувача до належної підготовки потрібних файлів.

Коли світлини завантажено, користувач переходить до розділу "Зібрати Відео" (Compose Video). Цей фрагмент інтерфейсу лаконічно описує логіку створення ролика: апаратура з'єднує згенерований звуковий супровід із завантаженими зображеннями, формуючи цілісний відеофайл. Також тут наведено умови, котрі мають бути дотримані для активації відеопродукції, а саме:

мусить бути і згенерований аудіоряд, і завантажені світлини. Щойно ці вимоги виконані, користувачеві стає доступною опція натиснути "Створити Відео" (Create Video) і розпочати процес формування фінального відеофайлу. Екранний вигляд модуля "Складання Відео" представлено на рис.2.8.

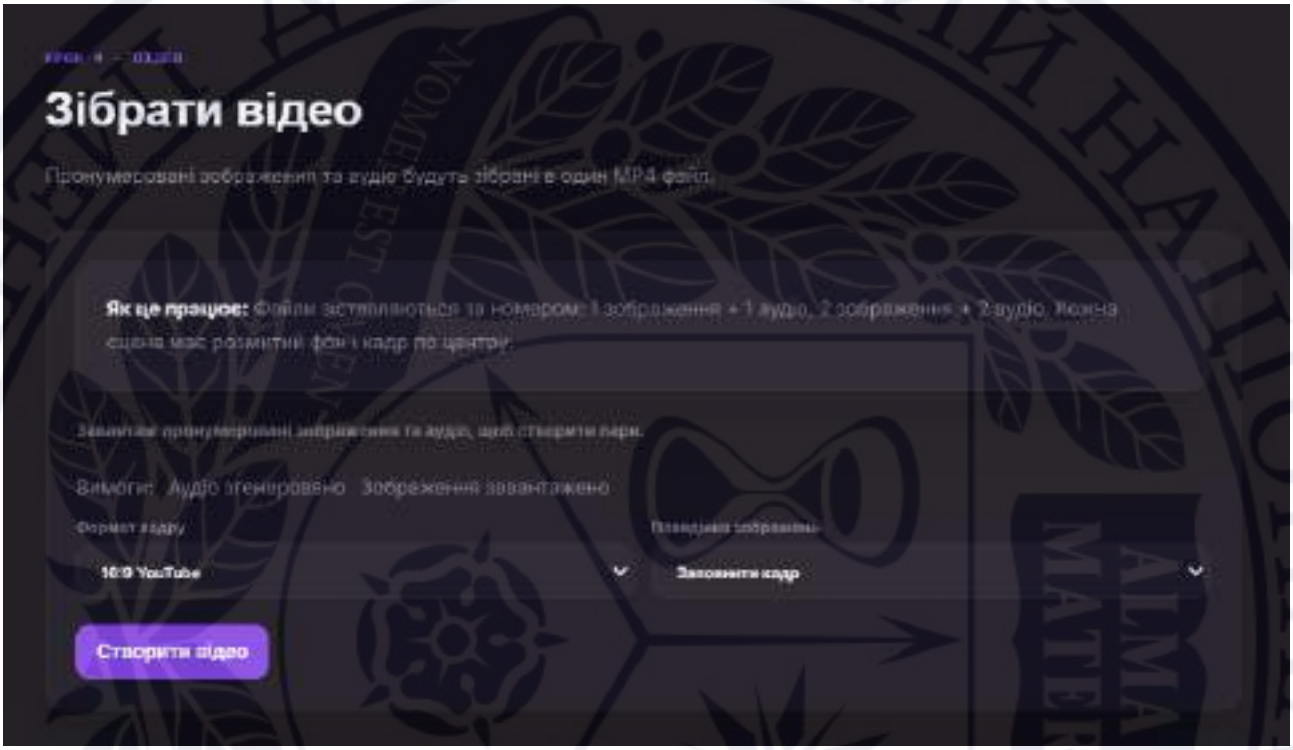

Рисунок 2.8 - Інтерфейс модуля Compose Video

Згідно з рисунку 2.8, інтерфейс модуля "Створення Відео" є досить лаконічним. Він вміщує стислий опис процедури, список необхідних умов та елемент для ініціації генерації відео. Такий підхід є обґрунтованим, адже значна частина обчислень відбувається на сервері, а від користувача вимагається лише активувати процес після того, як аудіо та візуальні матеріали будуть готові.

Крім того, на головній панелі розташовано розділ, що деталізує послідовність дій. Він повторює ключові етапи функціонування вебзастосунку, розбиті на шість послідовних фаз: формування текстового вмісту, здійснення перекладу, синтез мовлення, додавання візуальних файлів, компіляція відеоряду та отримання готового файлу. Як виглядає цей блок з етапами, показано на рис. 2.9.

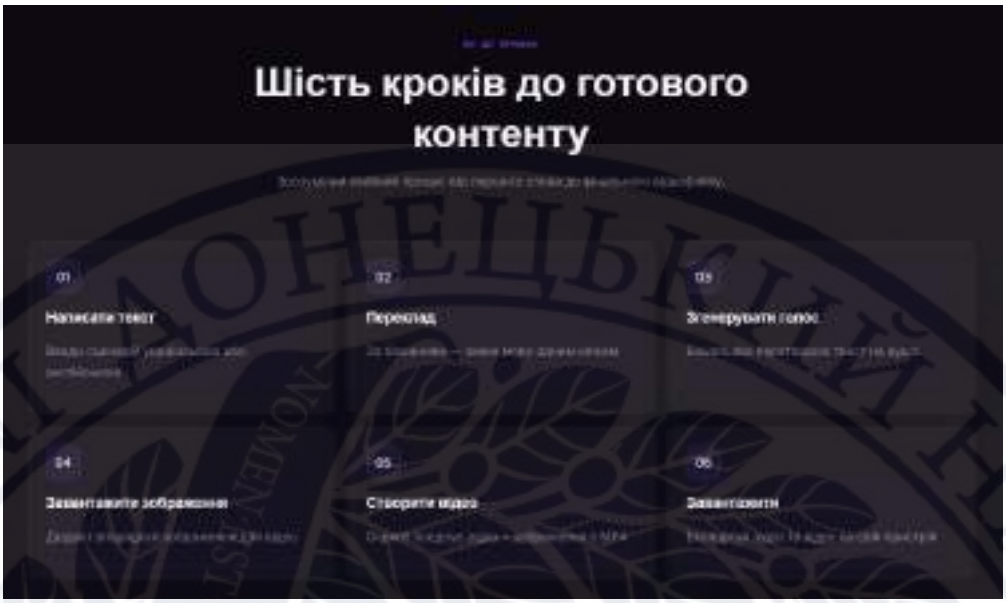

Рисунок 2.9 - Блок покрокового процесу створення контенту

Рисунок 2.9 ілюструє послідовність кроків, які долає користувач під час взаємодії з системою. Цей графічний елемент слугує навігаційно-інформаційній меті, адже дає змогу миттєво осягнути, з яких стадій складається процес формування фінального відеоряду. Наявність цієї схеми є безперечною перевагою для тих, хто вперше користується цим вебзастосунком.

У нижній частині головної сторінки розташовані блок із закликом до дії (CTA) та футер. CTA-блок містить спонукання розпочати створення свого першого відео та кнопку для оперативного старту роботи. Футер же вміщує стислі відомості про розробку, посилання на ключові розділи сайту та довідкову інформацію. Як виглядають CTA-блок та футер, продемонстровано на рис.2.10. CTA-блок також виконує важливу навігаційну функцію, оскільки спрямовує користувача до ключової дії без необхідності додаткового пошуку відповідних елементів на сторінці. Його візуальне оформлення виділяється на фоні основного контенту, що підсилює увагу та стимулює взаємодію з інтерфейсом. Футер, у свою чергу, забезпечує завершеність структури сторінки та створює відчуття цілісності вебзастосунку.

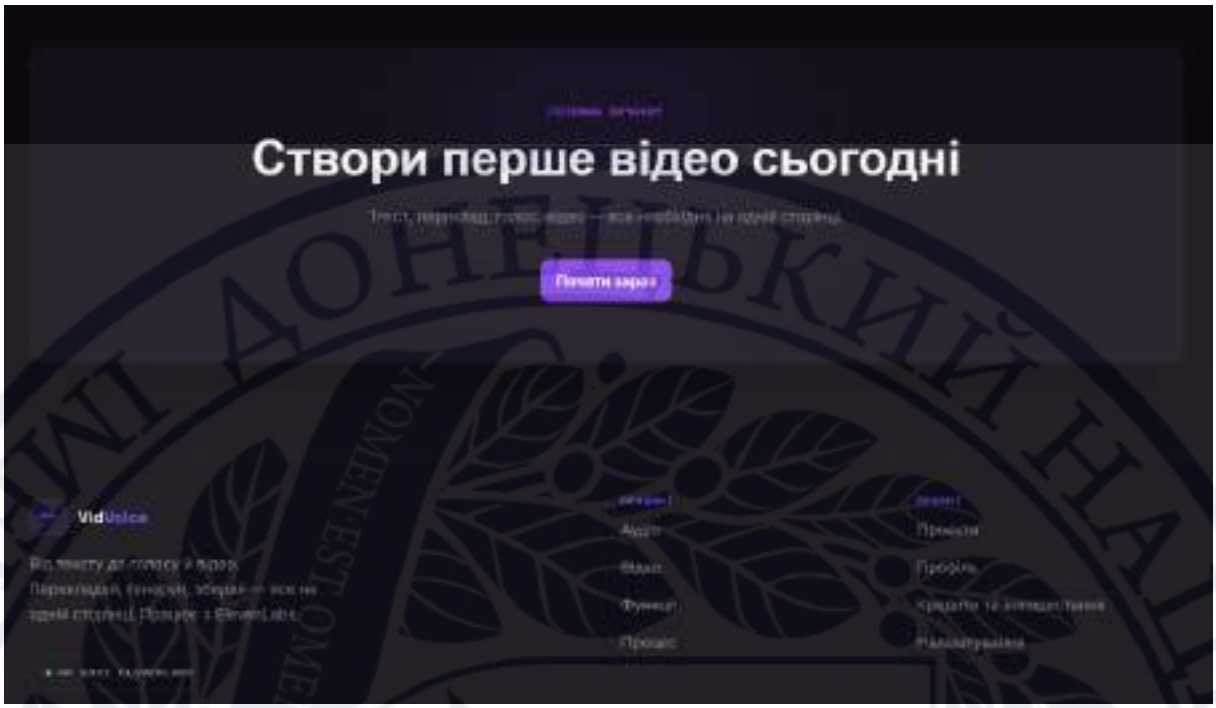

Рисунок 2.10 - CTA-секція та нижній колонтитул вебдодатку VidVoice

Згідно з рисунком 2.10, нижній фрагмент сторінки витриманий у єдиній стилістиці всього вебзастосунку та вміщує другорядну інформацію. Блок із закликом до дії (CTA) стимулює користувача розпочати створення матеріалів, тоді як футер надає можливість скористатися навігацією до службових ресурсів та інших сторінок проєкту.

Загалом, зазначена частина інтерфейсу завершує логічну структуру головної сторінки та формує завершене сприйняття користувацького сценарію взаємодії з вебзастосунком. Вона поєднує в собі як маркетингові елементи, спрямовані на залучення користувача до активної дії, так і інформаційно-довідкові компоненти, що підвищують рівень довіри до платформи. Завдяки цьому досягається баланс між функціональністю та візуальною завершеністю інтерфейсу.

Окремо варто відзначити, що футер виконує роль стабільного навігаційного орієнтира, який залишається доступним незалежно від глибини перегляду сторінок. Це підвищує зручність користування системою та забезпечує швидкий доступ до ключових розділів і службової інформації. Таким чином,

CTA-блок і футер у комплексі формують важливий завершальний елемент архітектури користувацького інтерфейсу.

Табл. 2.4 демонструє ключові складові інтерфейсу домашньої сторінки та їхні функції.

Таблиця 2.4 - Основні блоки головної сторінки вебдодатку

Блок сторінки	Призначення
Шапка сайту	Містить логотип, навігацію, кнопку старту та профіль користувача
Головний екран	Пояснює основну ідею вебдодатку та містить кнопку початку роботи
Блок можливостей	Демонструє основні функції системи у вигляді карток
Модуль Write & Translate	Забезпечує введення тексту, переклад і вибір голосу
Модуль Upload Images	Дозволяє завантажити зображення для майбутнього відео
Модуль Compose Video	Запускає процес формування відеофайлу
Блок покрокового процесу	Пояснює послідовність роботи користувача із системою
CTA-секція	Містить заклик до дії та кнопку початку роботи
Нижній колонтитул	Містить інформацію про продукт і допоміжні посилання

Як можна помітити, розглядаючи таблицю 2.4, стартовий екран об'єднує як інформаційні складові, так і елементи, призначені для дії. Він не просто демонструє, які функції пропонує вебдодаток, але й дає змогу користувачу безпосередньо здійснювати найважливіші операції: набирати текст, синтезувати голосовий контент, імпортувати графіки та ініціювати процес створення відеороликів.

Оформлення цього головного екрана базується на використанні темної палітри з вкрапленнями фіолетового. Подібна стилістика акцентує увагу на мультимедійній суті застосунку та виділяє критично важливі компоненти інтерфейсу. Центральні керуючі елементи, активні зони, заголовки секцій та візуальні маркери оформлені у фіолетових тонах, тоді як фонова площина утримується у темній гамі. Це сприяє єдності візуального сприйняття та надає проєкту актуального вигляду.

Отже, головна сторінка вебдодатку VidVoice спроектована як логічний ланцюжок кроків, які користувач виконує при взаємодії з платформою. Вона вміщує всі необхідні модулі для знайомства з можливостями, внесення тексту, формування озвучення, завантаження візуального контенту та активації відеосинтезу. Такий підхід гарантує, що інтерфейс буде інтуїтивно зрозумілим, послідовним та комфортним для роботи.

2.3.3 Реалізація інтерфейсу озвучування тексту

Функціонал інтерфейсу синтезу мовлення є однією з ключових складових клієнтської частини вебдодатку VidVoice. Його головне завдання - забезпечити введення тексту, його переклад, вибір вокального тембру, регулювання параметрів озвучення та запуск процесу створення аудіофайлу. На головній сторінці цей механізм розміщений в модулі "Write & Translate", та розташований у файлі frontend/index.html.

Візуальне представлення інтерфейсу модуля озвучення показано на рис. 2.6.

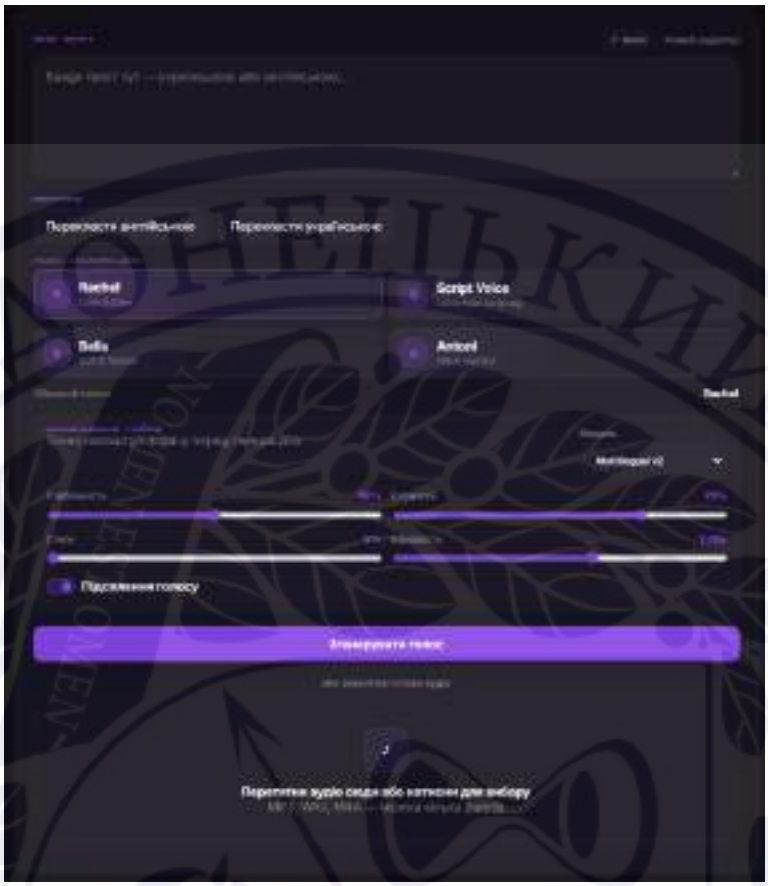

Рисунок 2.6 - Вигляд модуля "Write & Translate"

Як показано на рисунку 2.6, блок "Write & Translate" складається з кількох основних зон: поля для введення тексту, кнопок для виконання перекладу, секції вибору голосу, панелі налаштування голосових характеристик, кнопки ініціювання генерації озвучки та області для завантаження готового аудіо. Така архітектура дозволяє користувачу керувати усім циклом роботи з озвученням тексту в межах одного єдиного функціонального вузла.

Центральним елементом цього модуля слугує текстове поле. Воно призначене для набору або вставки тексту як українською, так і англійською мовами. Крім того, модуль підтримує опцію імпорту тексту із файлу за допомогою елемента text-file-input. Після вибору файлу скрипт на JavaScript зчитує його вміст та автоматично поміщає текст у поле введення. Це значно спрощує роботу, якщо текстовий матеріал вже підготовлений у файловому форматі.

Для перекладу тексту в інтерфейсі передбачено дві керуючі кнопки: translate-en та translate-ua. Перша відповідає за переклад на англійську мову, а друга - на українську. При натисканні на відповідну кнопку, клієнтська логіка робить виклик до функції api.translate() з файлу api.js, яка відправляє POST-запит на кінцеву точку /api/translate. У цьому запиті передаються як сам текст, так і бажаний напрямок перекладу. Після отримання відповіді із сервера, результат операції повертається і заноситься назад у поле script-text [6].

Вибір бажаного голосу реалізовано через блок voice-options. При завантаженні сторінки активується функція loadVoices(), яка звертається до API для отримання списку доступних тембрів. Для цього використовується метод api.getVoices(), що ініціює запит до маршруту /api/voices. Якщо відповідь сервера містить перелік голосів, вони візуалізуються у вигляді окремих карток. У випадку недоступності списку, система використовує резервний набір голосів, визначений у файлі home.js.

Кожен доступний голос у модулі відображається як самостійна картка, що містить назву та короткий опис особливостей його звучання. Коли користувач клацає на картку, обраний голос зберігається у внутрішньому стані клієнтської частини як selectedVoice. Після цього інтерфейс оновлює текстовий індикатор обраного голосу за допомогою функції syncSelectedVoice(). Такий підхід забезпечує моментальний зворотний зв'язок щодо того, який саме тембр буде задіяний для синтезу аудіо.

Окрім вибору тембру, в модулі є панель для тонкого налаштування параметрів озвучення. Вона містить вибір моделі для синтезу мовлення, а також регулятори для значень stability, similarity_boost, style та speed. Зчитування поточних значень цих налаштувань виконує функція getVoiceSettings(). При взаємодії з регуляторами (слайдерами) на сторінці миттєво оновлюються відсоткові чи числові позначки, що дозволяє користувачу візуально контролювати характеристики майбутнього аудіо.

Ключові компоненти інтерфейсу озвучення тексту згруповані у табл. 2.5.

Таблиця 2.5 - Складові інтерфейсу модуля синтезу мовлення

Елемент інтерфейсу	Ідентифікатор/файл	Призначення
Поле введення тексту	script-text	Введення або вставлення тексту для озвучування
Завантаження текстового файлу	text-file-input	Імпорт тексту з локального файлу
Кнопка перекладу англійською	translate-en	Надсилання запиту на переклад тексту англійською мовою
Кнопка перекладу українською	translate-ua	Надсилання запиту на переклад тексту українською мовою
Блок вибору голосу	voice-options	Відображення доступних голосів ElevenLabs
Вибір моделі	voice-model	Вибір моделі синтезу мовлення
Параметри голосу	stability, similarity-boost, style, speed	Налаштування особливостей звучання
Кнопка генерації голосу	generate-audio	Запуск процесу створення аудіофайлу
Завантаження готового аудіо	audio-input	Додавання вже готових аудіофайлів користувача
Список аудіофайлів	audio-file-list	Відображення згенерованих або завантажених аудіофайлів

З таблиці 2.5 видно, що модуль озвучення тексту сформовано не як проста форма для вводу тексту, а як багатофункціональний блок. Він підтримує ручне введення тексту, завантаження з файлу, послуги перекладу, селекцію голосу, калібрування параметрів синтезу, генерування аудіо та можливість імпорту вже готових звукових ресурсів.

Після активації кнопки generate-audio у файлі home.js відбувається первинна перевірка введеного тексту. Якщо поле залишається порожнім, процес

генерації не розпочинається. Якщо текст присутній, кнопка тимчасово блокується, а її мітка змінюється на індикатор активного процесу генерації. Це зроблено для того, щоб користувач бачив, що запит обробляється, і запобігти повторному натисканню [20, 21].

Далі клієнтська частина формує об'єкт з усіма необхідними даними для генерації. Цей об'єкт містить сам текст, ідентифікатор обраного голосу, його метадані та налаштування синтезу. Після цього відбувається виклик функції api.generateAudio(), яка формує POST-запит до серверного маршруту /api/generate-audio. На сервері цей маршрут делегує отримані параметри відповідному сервісу синтезу мовлення, після чого повертає клієнту інформацію про створений звуковий файл.

Після успішного отримання відповіді від сервера, згенерований аудіофайл зберігається у внутрішньому стані клієнта як generatedAudio. Крім того, він додається до масиву audioFiles, після чого викликається функція renderAudioFiles(). Вона оновлює блок audio-file-list, де користувач може переглянути перелік наявних файлів, відтворити їх, обрати потрібний для подальшого створення відео або завантажити на свій пристрій.

Окремо реалізовано механізм завантаження користувачем власних аудіофайлів. Для цього задіяно елемент audio-input, що приймає файли у форматах MP3, WAV, M4A, AAC, та OGG. Після вибору файлів або їх перетягування у відповідну область, активується функція importAudioFiles(), яка передає ці дані на сервер через api.importGeneratedFiles(). Отримані аудіозаписи також додаються до списку доступних файлів і можуть бути використані для створення відеоконтенту.

У модулі також присутня опція вибору конкретного аудіо для подальших кроків. Якщо користувач натискає кнопку "використати аудіо", відповідний запис закріплюється як активний у змінній generatedAudio. Це критично важливо для наступного етапу роботи системи, оскільки саме цей обраний аудіофайл буде задіяний у процесі формування відео.

Загальний алгоритм роботи інтерфейсу синтезу тексту виглядає наступним чином:

1. користувач вводить текст або ініціює завантаження файлу з текстом;
2. за потреби виконується переклад (на українську чи англійську);
3. обирається потрібний тембр із доступного переліку;
4. коригуються параметри звучання;
5. ініціюється натискання кнопки генерації аудіо;
6. клієнтська частина надсилає запит на серверну частину;
7. сервер надає дані про згенерований аудіофайл;
8. звуковий файл інтегрується у перелік доступних аудіоресурсів;
9. користувач може прослухати, експортувати або використати аудіо для відеомонтажу.

У разі виникнення збою під час генерації аудіо, система сповіщає користувача через системне діалогове вікно alert(). Додатково, інформація про помилку фіксується у журналі консолі браузера за допомогою console.error(). Це значно полегшує процес діагностики під час розробки та допомагає оперативно з'ясувати першопричину проблеми [20, 22].

Підсумовуючи, інтерфейс синтезу мовлення у VidVoice представлений як ефективний, практичний модуль, що об'єднує роботу з текстом, перекладом, апаратом вибору голосу, налаштуваннями синтезу та списком готових аудіофайлів. Його функціональність базується на взаємодії HTML-структури (у файлі index.html), JavaScript-логіки (у home.js) та визначених API-викликів (у api.js). Такий архітектурний підхід забезпечує зручну взаємодію користувача з системою та готує необхідні аудіодані для наступного етапу - створення відеоряду.

2.3.4 Реалізація інтерфейсу генерації відео

Інтерфейс для створення відеоряду являє собою наступний етап функціональності, що слідує за згенерованими чи завантаженими аудіозаписами. Його головне завдання - надати користувачеві змогу додати візуальні матеріали, пересвідчитися у наявності необхідних компонентів та ініціювати процес

формування відеофайлу. У вебдодатку VidVoice ця можливість інтегрована на головній сторінці у двох взаємопов'язаних секціях: "Завантаження зображень" (Upload Images) та "Складання відео" (Compose Video).

Створення інтерфейсу для відеогенерації базується на використанні технологій HTML, CSS та JavaScript. HTML відповідає за розміщення зони для завантаження, кнопок управління та інформаційних блоків. CSS забезпечує зовнішнє оформлення, а JavaScript реалізує обробку взаємодій користувача, завантаження файлів, валідацію умов та надсилання запитів до серверної частини [7-9, 19]. Такий підхід дає змогу сконструювати інтерактивне середовище, де користувач може підготувати весь необхідний візуальний пакет для відео без необхідності навігації по інших сторінках.

Початковим кроком роботи з відео є внесення зображень. Саме для цього у вебдодатку передбачено компонент "Завантаження зображень", чий вигляд демонструє рис. 2.7.

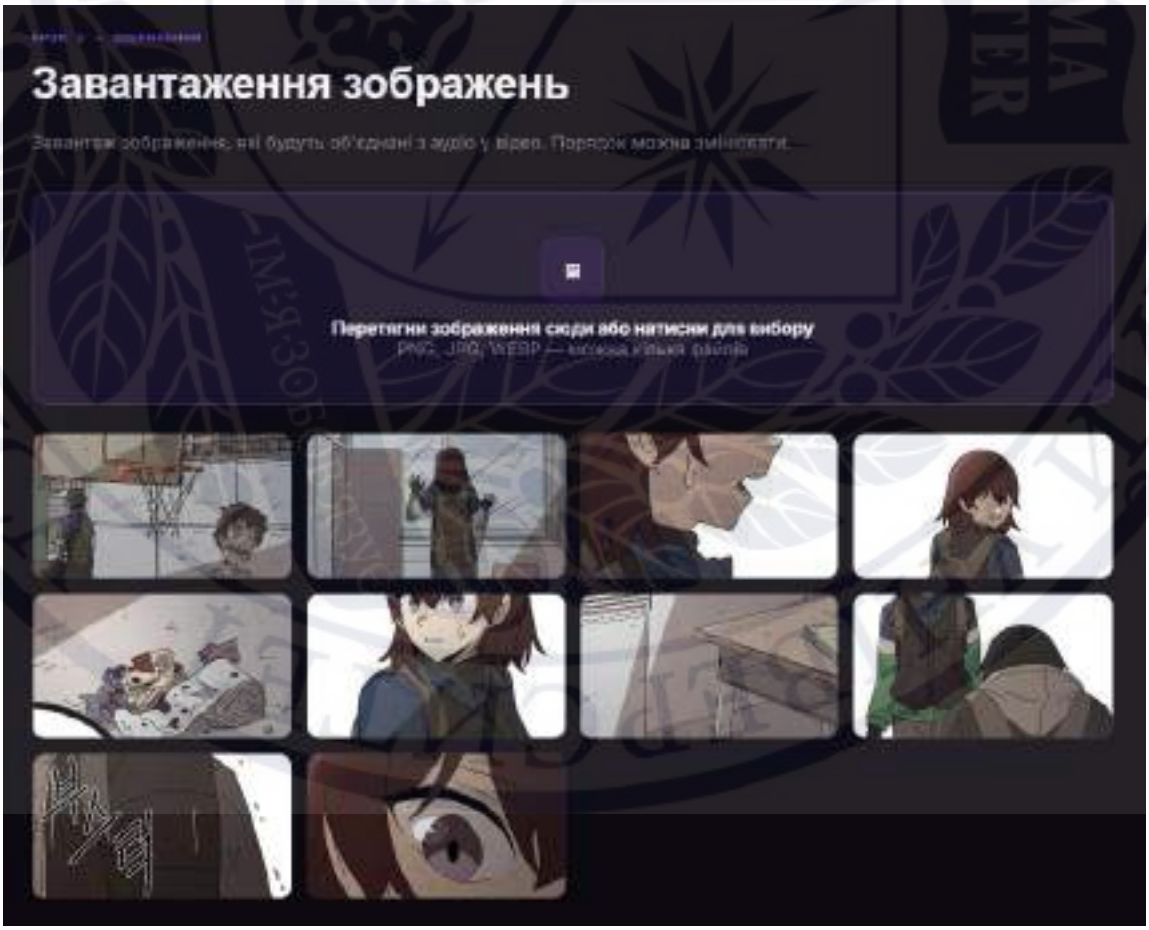

Рисунок 2.7 - Оформлення компонента "Завантаження зображень"

Як видно із рисунка 2.7, секція для внесення візуальних даних містить виділену ділянку, призначену для вибору або перетягування файлів (drag-and-drop). Цей метод взаємодії є інтуїтивно зрозумілим: можна клікнути на цю зону для вибору файлів через системний файловий менеджер, або ж просто перенести потрібні зображення у цей віконний блок. В інтерфейсі також чітко зазначені формати, які підтримуються, а саме PNG, JPG та WEBP, що допомагає користувачу одразу визначити придатні для використання візуальні матеріали.

У структурі клієнтської частини блок для завантаження зображень описаний у файлі `frontend/index.html`. Для селекції файлів задіяно відповідний елемент вводу зображень, тоді як основна логіка опрацювання цих файлів зосереджена у скрипті `frontend/assets/js/home.js`. Після внесення або перетягування файлів, логіка на JavaScript здійснює перевірку їхнього типу, формує перелік завантажених візуалів та готує їх до надсилання на сервер.

Для оперування завантаженими файлами у клієнтській частині використовується спеціальний масив, де зберігається інформація про вибрані зображення. Це забезпечує можливість подальшого виведення списку файлів, контролю їхньої кількості та передачі їх на сервер у момент формування відео. Цей механізм є ключовим, оскільки відеоконтент створюється не лише на основі аудіо, але й завдяки набору візуальних елементів, які самостійно додає користувач.

Після успішного внесення зображень користувач переходить до блоку "Складання відео" (Compose Video). Саме цей модуль ініціює запуск процесу створення фінального відеофайлу. Його вигляд представлено на рис.2.8.

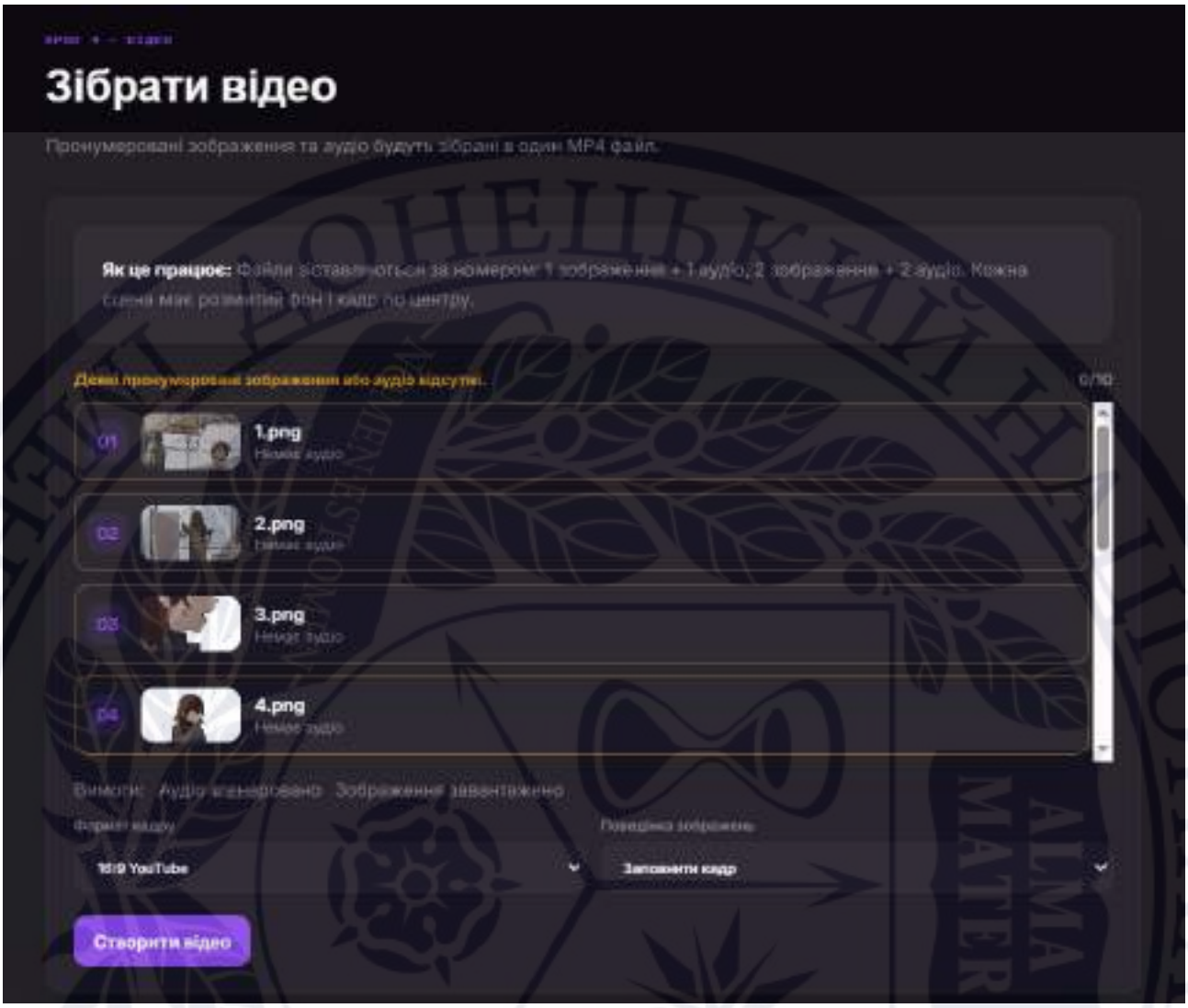

Рисунок 2.8 - Інтерфейс компонента "Складання відео"

На рисунку 2.8 демонструється, що блок "Складання відео" містить стислий опис логіки роботи, перелік необхідних умов для старту генерації та кнопку початку створення. Додано спеціальний текстовий супровід, який пояснює, що відбувається. Завдяки такому підходу користувач чітко бачить те, що відбувається і одразу може реагувати на роботу програми.

Перед запуском виконується перевірка, де користувач може впевнитися в тому, що всі аудіо та фото було додано. Відсутність перевірки може призвести до помилок, тому що під час роботи із великою кількістю файлів – дуже легко щось загубити та не побачити. Система не зможе почати генерацію відео, якщо хоча б один із двох елементів буде відсутнім [20, 22].

У табл. 2.6 внесені основні елементи, які задіяні під час генерації відео.

Таблиця 2.6 - Компоненти інтерфейсу відеогенерації

Елемент інтерфейсу	Ідентифікатор/файл	Призначення
Область завантаження зображення	image-drop-zone	Приймає зображення через вибір або перетягування файлів
Поле вибору зображень	image-input	Дозволяє вибрати файли з локального пристрою
Список зображень	image-file-list	Відображає додані зображення
Блок створення відео	Compose Video	Пояснює процес генерації відео
Кнопка створення відео	create-video	Запускає процес формування відеофайлу
Активний аудіофайл	generatedAudio	Використовується як аудіосупровід для відео
Масив зображень	imageFiles	Зберігає список завантажених зображень
API-запит створення відео	api.createVideo()	Передає аудіо та зображення на серверну частину

Як показує таблиця 2.6, інтерфейс відеогенерації складається не лише з однієї пускової кнопки, а з низки взаємопов'язаних елементів. Спочатку користувач імпортує зображення, потім система обов'язково верифікує наявність аудіо, і лише після цього стає доступним створення відео. Така архітектура відповідає загальній логіці функціонування програми і дозволяє користувачеві послідовно пройти етапи підготовки матеріалів.

При натисканні кнопки створення відео у файлі `home.js` ініціюється перевірка поточного стану клієнтської частини. Насамперед, верифікується, чи існує вже активний аудіофайл, який міг бути згенерований чи попередньо завантажений. Далі перевіряється, чи було додано хоча б одне зображення. Лише за умови виконання цих критеріїв формується запит до серверу. У разі

відсутності необхідних даних, система або сповіщає користувача про помилку, або вказує на необхідність виконати попередній крок.

Для передачі актуальних даних на сервер використовується функція, визначена у файлі `api.js`, яка надсилає HTTP-запит до відповідної кінцевої точки серверної логіки. У цьому запиті передається інформація про обраний аудіофайл та перелік візуальних файлів, що мають бути задіяні у відео. На сервері ці дані проходять обробку, після чого запускається процес збирання відео, який виконується за допомогою утиліти FFmpeg.

Загальний алгоритм функціонування інтерфейсу відеогенерації виглядає наступним чином:

1. Користувач створює або імпортує звукову доріжку;
2. Користувач завантажує візуальні матеріали через компонент “Завантаження зображень";
3. Клієнтська частина формує перелік візуальних файлів;
4. Користувач переходить до компонента “Складання відео";
5. Система проводить валідацію наявності аудіо та зображень;
6. Після натискання кнопки “Створити відео” ініціюється запит до сервера;
7. Серверна частина розпочинає етап збирання відео;
8. По завершенні обробки користувач отримує готовий відеофайл.

Під час взаємодії з інтерфейсом генерації відео критично важливим є забезпечення зворотного зв'язку для користувача. Якщо створення відео не відбувається миттєво, кнопка генерації повинна бути тимчасово деактивована, а користувач має бачити індикатор або повідомлення про те, що обробка триває. Це запобігає дублюванню запитів і робить роботу системи більш передбачуваною.

Особливої уваги заслуговує механізм обробки помилок. Збої можуть статися, якщо не було створено аудіо, не завантажено жодного зображення, було обрано некоректний формат файлу або виникла помилка на серверному етапі. У таких випадках клієнтська частина повинна вивести зрозуміле повідомлення для

користувача, а технічні деталі відобразити у консолі браузера для подальшого діагностування.

Специфічною рисою реалізації інтерфейсу відеогенерації є його пряма залежність від попереднього модуля озвучування тексту. Якщо користувач не створив або не вніс звуковий файл, компонент “Складання відео” не зможе сформувати фінальний відеопродукт. Таким чином, інтерфейс побудований як лінійний сценарій: спершу створюється чи імпортується аудіо, потім вносяться зображення, і лише після цього активується формування відео.

Для максимальної зручності користувача, компоненти “Завантаження зображень” та “Складання відео” розміщені в межах однієї сторінки, поруч один з одним. Це дозволяє оперативно переходити від імпорту візуалів до ініціації відеостворення, без потреби відкривати нові вкладки чи сторінки.

З технічної точки зору, клієнтський рівень не виконує безпосереднього опрацювання відеоданих. Він лише приймає інформацію від користувача, здійснює її валідацію та передає на сервер. Основний обчислювальний тягар, пов'язаний з обробкою аудіо та графічних даних, лягає на серверну частину.

Підсумовуючи, інтерфейс відеогенерації у програмі реалізований як послідовний модуль, що інтегрує завантаження візуальних матеріалів, перевірку наявності аудіо та запуск процесу формування відео. Його функціональність спирається на взаємодію HTML-розмітки, клієнтської JavaScript-логіки та API-викликів. Такий підхід дозволяє користувачам комфортно підготувати необхідні візуальні елементи, зв'язати їх із звуковою доріжкою та отримати кінцевий відеопродукт.

2.3.5 Особистий кабінет користувача та сторінка проєктів

Окрім ключових функціональних блоків, а саме модуля синтезу мовлення та генерації відеоряду, клієнтська складова вебзастосунку VidVoice містить секції, відведені під персональний простір користувача. Сюди входять: меню доступу до профілю, власне сторінка профілю, розділ управління проєктами, секція відображення використаних лімітів (кредитів) та блок конфігурації. Наявність цих складових перетворює вебзастосунок із простого інструменту для

разового створення аудіо чи відео на більш повноцінну інформаційну екосистему, де користувач може оглядати свої особисті дані, архів створених робіт та стан використаних ресурсів [6].

Доступ до персонального профілю користувача організовано через клікабельну іконку, що асоціюється з профілем; ця іконка розташована у правому сегменті верхньої панелі (хедеру) сайту. Після активації цієї іконки розгортається випадаючий перелік, де відображається ім'я, адреса електронної пошти користувача та прямі посилання на основні секції його кабінету. Візуалізація меню профілю користувача представлена на рис. 2.11.

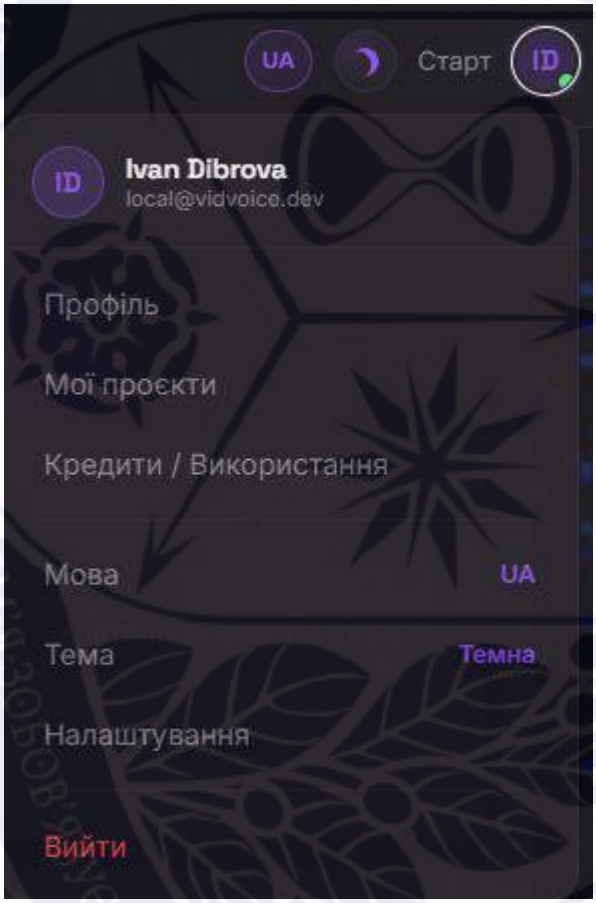

Рисунок 2.11 - Меню доступу до особистого профілю у вебзастосунку VidVoice

Як демонструє рисунок 2.11, меню профілю забезпечує швидке переміщення на сторінки «Профіль», «Мої проєкти», «Кредити / Використання» та «Налаштування». Крім того, у цьому ж меню надається інформація про поточну мовну локалізацію інтерфейсу та обрану візуальну тему оформлення.

Окремим елементом виділено кнопку для завершення активної сесії. Така архітектура є ергономічною, оскільки акумулює всі дії, специфічні для користувача, в одній локації, запобігаючи при цьому перевантаженню основного екрану вебзастосунку.

Сторінка, відведена під профіль, служить для перегляду й часткової модифікації персональних даних. Тут відображається аватарка з ініціалами, повне ім'я користувача, його електронна пошта, дата реєстрації, поле для оновлення імені та кнопка для збереження змін. Також на цій сторінці презентується стисла аналітика активності: сумарна кількість проєктів, обсяг згенерованого аудіо, кількість готових відеороликів та загальний облік використаних лімітів. Візуальне представлення інтерфейсу сторінки профілю наведено на рис.2.12.

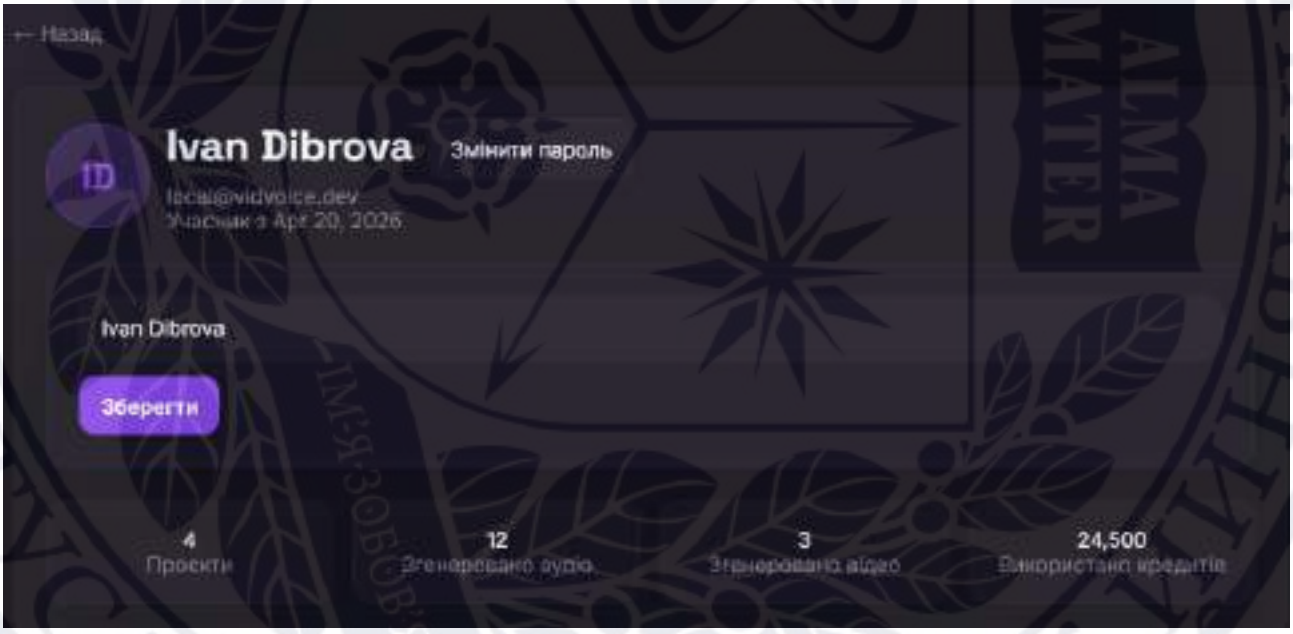

Рисунок 2.12 - Інтерфейс сторінки профілю користувача

Рисунок 2.12 засвідчує, що сторінка профілю має раціональну й легкосприйнятну структуру. У верхньому блоці міститься ключова ідентифікаційна інформація, а нижче розташовані блоки, що агрегують статистичні метрики. Такий підхід дає змогу користувачеві оперативно оцінити рівень своєї взаємодії із системою та ознайомитися з фундаментальними даними

свого облікового запису. Наявність функціоналу для зміни пароля та збереження внесених змін закладає основу для майбутнього розширення можливостей персонального простору.

Окрема функціональна роль у клієнтській частині відводиться сторінці «Мої роботи». Її функція - репрезентація переліку створених користувачем медіа-артефактів. На цій сторінці доступний огляд назв проєктів, їхньої мовної основи, кількості сцен, дат створення та останнього оновлення, а також поточного робочого статусу. Крім того, передбачено поле для пошукового запиту та кнопка для ініціювання створення нового проєкту. Зовнішній вигляд сторінки проєктів представлений на рис.2.13.

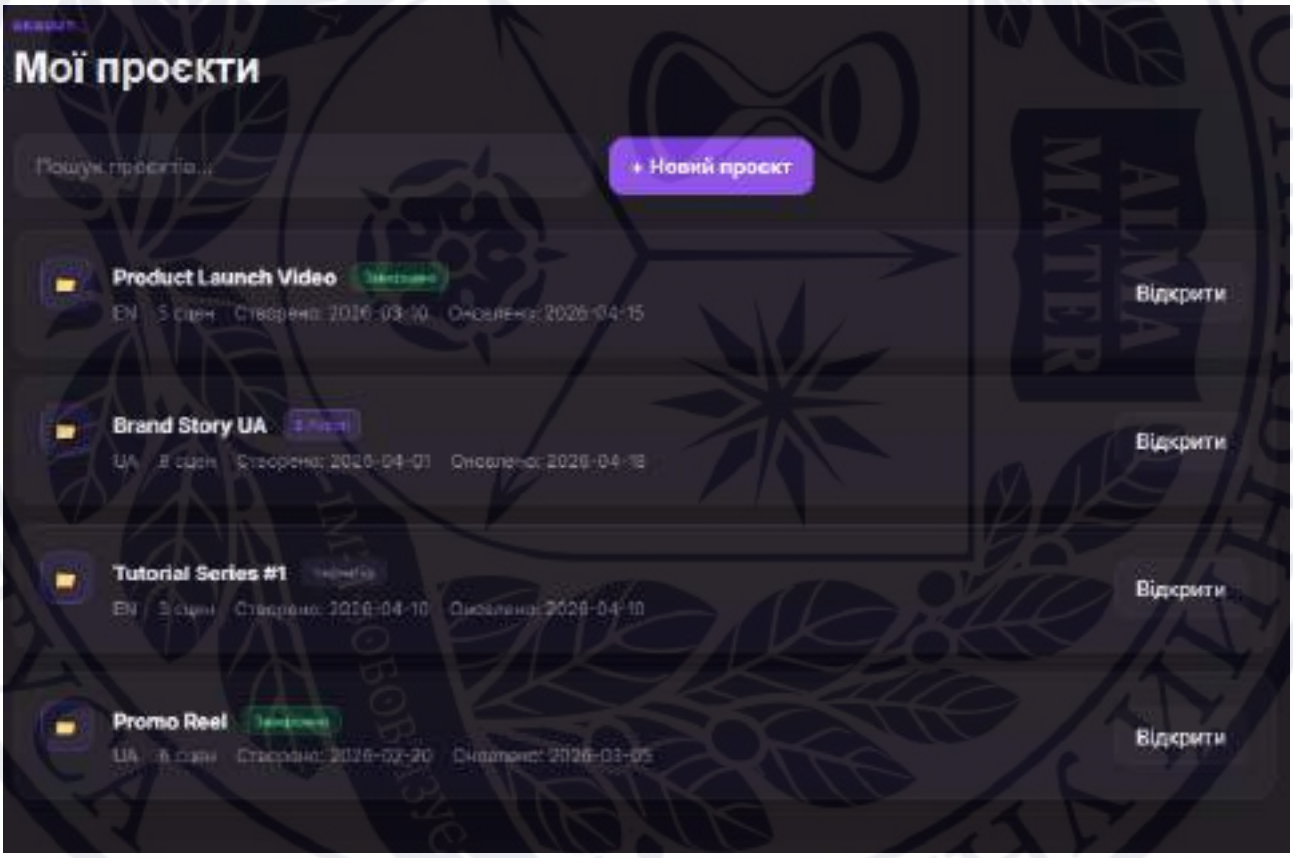

Рисунок 2.13 - Сторінка управління проєктами користувача

Як показано на рисунку 2.13, кожен проєкт відображається у форматі індивідуальної картки. Картка містить такі атрибути, як назва роботи, її статус, мова, кількість сцен, дата фіксації та дата останнього доопрацювання. Для кожного запису передбачена активна кнопка «Відкрити», яка слугує для переходу

до режиму перегляду або продовження редагування контенту. Така організаційна модель є зручною для кінцевого споживача, адже сприяє швидкому пошуку потрібного матеріалу та ефективному моніторингу прогресу виконання завдань.

Секція «Ліміти» використовується для прозорого контролю споживання ресурсів, які надають сервіси синтезу мовлення. Оскільки процес озвучування, що базується на зовнішніх рішеннях, може мати обмеження щодо обсягу символів чи кількості запитів, для користувача критично важливо мати можливість відстежувати використані кредити. Це забезпечує кращий контроль над процесом аудіопродукції та допомагає уникнути несподіваних перешкод під час роботи із системою [10].

Розділ «Параметри» призначений для кастомізації робочих характеристик вебзастосунку. До таких налаштовних параметрів можуть входити: мовний вибір інтерфейсу, візуальна схема оформлення, пресети для синтезу мовлення, опції експорту відеоряду та інші індивідуальні конфігурації. Наявність цього розділу підвищує адаптивність системи та зручність її експлуатації.

Отже, персональний простір у вебзастосунку VidVoice слугує доповненням до основного функціоналу системи, забезпечуючи юзеру доступ до управління профілем, проєктами, лімітами та конфігураціями. Впровадження меню профілю, сторінки з особистими даними та розділу проєктів робить вебзастосунок цілісним, придатним для тривалого використання та готовим до подальшої імплементації розширених можливостей.

2.4 Реалізація серверної частини вебдодатку

2.4.1 Структура серверної частини проєкту

Бекенд мережевої програми VidVoice реалізовано на Python із використанням FastAPI та відповідає за обробку запитів клієнтської частини, роботу з файлами, інтеграцію з сервісом ElevenLabs, генерацію аудіодоріжок і монтаж відео за допомогою FFmpeg. Він має модульну структуру, розміщену в каталозі backend, де окремі компоненти відповідають за запуск застосунку,

конфігурацію, API-ендпоінти та сервіси для обробки мультимедійних даних, що забезпечує зручність супроводу та масштабування системи.

Ключові складові структури бекенду проєкту показано в табл. 2.8.

Таблиця 2.8 - Організація серверної частини вебдодатка

Файл або директорія	Призначення
backend/app/main.py	Основний файл запуску FastAPI-застосунку
backend/app/api/routes.py	Опис API-маршрутів для взаємодії з клієнтською частиною
backend/app/core/config.py	Налаштування шляхів до директорій і параметрів середовища
backend/app/services/elevenlabs_service.py	Сервіс для роботи з ElevenLabs API та генерації аудіо
backend/app/services/video_service.py	Сервіс для створення відео за допомогою FFmpeg
backend/app/services/generated_files_service.py	Сервіс для роботи зі згенерованими та імпортованими файлами
backend/app/services/store.py	Тимчасове збереження даних профілю, проєктів і кредитів
backend/storage/uploads	Директорія для завантажених зображень
backend/storage/audio	Директорія для згенерованих або імпортованих аудіофайлів
backend/storage/videos	Директорія для готових відеофайлів

Згідно з таблицею 2.8, бекенд розбитий на декілька функціональних блоків. `main.py` забезпечує запуск програми, підключення маршрутів та рендеринг сторінок клієнтської частини. У файлі `routes.py` визначено API-запити, через які клієнтська сторона надсилає дані на сервер. Файл `config.py` містить налаштування, зокрема директорії та критичні параметри, як-от ключ доступу до ElevenLabs API.

Окремі сервісні функції винесені в директорію `services`. Це унеможливлює перевантаження файлу маршрутизації та дозволяє логічно розділити бізнес-правила. Наприклад, `elevenlabs_service.py` відповідає за синтез аудіо з тексту, тоді як `video_service.py` займається компонуванням відеофайлу з аудіодоріжки та зображень. Для обробки медіаконтенту використовується FFmpeg, який підтримує роботу з широким спектром аудіо- та відеоформатів [16].

Для фіксації файлових об'єктів у системі виділена директорія `storage`, що містить підкаталоги для зображень, аудіо та відео. Така структура забезпечує порядок завантажених та згенерованих файлів та спрощує їх подальше асоціювання з проєктами користувачів. У перспективі, відомості про ці файли можуть бути перманентно збережені у базі даних PostgreSQL [17].

Отже, архітектура бекенду вебдодатка VidVoice є чіткою та логічно сегментованою на окремі складові. Вона гарантує належну обробку запитів, синтез аудіо, формування відео, файлові операції та готова до масштабування функціонала у майбутньому.

2.4.2 Реалізація API-запитів для роботи з користувачем

API-запити у вебдодатку VidVoice забезпечують обмін даними між клієнтською та серверною частинами відповідно до клієнт-серверної архітектури, де користувацькі дії ініціюють запити, що обробляються сервером із подальшим поверненням результату у форматі JSON. Реалізацію API-маршрутів виконано за допомогою FastAPI у файлі backend/app/api/routes.py, а звернення до них з фронтенду здійснюються через функції, визначені у frontend/assets/js/api.js, що забезпечує чіткий розподіл відповідальності між частинами системи.Основні API-запити серверної частини наведено в табл. 2.9.

Таблиця 2.9 - Основні API-маршрути серверної частини

API-маршрут	Метод	Призначення
/api/health	GET	Перевірка працездатності серверної частини
/api/profile	GET, PUT	Отримання та оновлення даних профілю
/api/projects	GET, POST	Отримання списку проєктів і створення нового проєкту
/api/credits	GET	Отримання інформації про використані кредити
/api/translate	POST	Переклад тексту у заданому напрямі
/api/voices	GET	Отримання списку доступних голосів
/api/generate-audio	POST	Генерація аудіофайлу на основі тексту
/api/generated-files	GET	Отримання списку згенерованих файлів
/api/generated-files/import	POST	Імпорт готових аудіо- або відеофайлів
/api/upload-images	POST	Завантаження зображень користувача
/api/compose-video	POST	Формування відеофайлу на основі аудіо та зображень

Як видно з таблиці 2.9, API охоплює як службові функції, так і основні дії користувача. Маршрути /api/profile, /api/projects і /api/credits використовуються для роботи з особистим кабінетом. Вони дозволяють отримувати інформацію про користувача, переглядати створені проєкти та контролювати використання кредитів для озвучування.

Основні функціональні маршрути пов’язані з генерацією аудіо та відео. Запит /api/translate використовується для перекладу тексту, /api/voices - для отримання доступних голосів ElevenLabs, а /api/generate-audio — для створення аудіофайлу на основі введеного тексту [10].

Для створення відео використовуються маршрути /api/upload-images та /api/compose-video. Перший маршрут приймає зображення від користувача та зберігає їх на сервері, а другий запускає процес формування відеофайлу на основі аудіо та завантажених зображень. Безпосередня обробка мультимедійних файлів виконується за допомогою FFmpeg [16].

Таким чином, API-запити забезпечують зв’язок між інтерфейсом користувача та серверною логікою вебдодатку. Вони дають змогу працювати з профілем, проєктами, перекладом, голосами, аудіо, зображеннями та відеофайлами. Така структура робить серверну частину зрозумілою та зручною для подальшого розширення.

2.4.3 Інтеграція сервісу синтезу мовлення ElevenLabs

Впровадження сервісу ElevenLabs у вебдодаток VidVoice слугує для автоматичного перетворення тексту, введеного користувачем, у звуковий файл. Це усуває потребу у розробці власної моделі синтезу мовлення, натомість використовуючи готовий зовнішній засіб, що підтримує генерацію голосу через програмний інтерфейс (API) [10].

На стороні сервера логіка взаємодії з ElevenLabs винесена у відокремлений модуль backend/app/services/elevenlabs_service.py. Таке архітектурне рішення сприяє відділенню операцій із зовнішнім засобом від головних маршрутів API, роблячи кодову структуру більш зрозумілою. Ключ доступу до ElevenLabs

зчитується з конфігурації середовища, що запобігає його прямому зберіганню у клієнтській частині або у відкритому коді.

Процес створення звуку ініціюється, коли користувач вписує текст у блоці "Write & Translate", вибирає бажаний голос та активує кнопку генерації. Після цього клієнтська сторона надсилає запит на серверний шлях /api/generate-audio. У цьому запиті передаються текст, обрана модель голосу та параметри озвучення. Сервер приймає ці дані і направляє їх до сервісу, визначеного у elevenlabs_service.py.

Ключові етапи інтеграції можливостей ElevenLabs зображено у Табл. 2.10.

Таблиця 2.10 - Етапи підключення ElevenLabs API у вебдодатку

Етап	Опис
Отримання тексту	Сервер приймає текст від клієнтської частини
Вибір голосу	Передається ідентифікатор голосу, обраного користувачем
Налаштування параметрів	Передаються параметри стабільності, схожості, стилю та швидкості
Запит до ElevenLabs	Сервер надсилає дані до зовнішнього API
Отримання аудіо	ElevenLabs повертає згенерований аудіофайл
Збереження результату	Аудіофайл зберігається в директорії storage/audio
Повернення відповіді	Сервер повертає клієнтській частині інформацію про створений файл

Як це видно з таблиці 2.10, сервер виступає як проміжний вузол між інтерфейсом користувача та API ElevenLabs. Це є важливим вибором дизайну, оскільки клієнт не здійснює прямих звернень до ElevenLabs. Завдяки цьому конфіденційний API-ключ залишається в безпеці на сервері, а взаємодія користувача обмежується лише інтерфейсом програми.

У ході створення звуку сервер формує запит до ElevenLabs, включаючи текст та визначені характеристики голосу. По успішній обробці, сервіс видає

звукові дані, які фіксуються як окремий файл. Далі сервер готує відповідь для клієнта, яка містить назву файлу, шлях до нього та іншу необхідну технічну інформацію. Після цього користувач отримує можливість прослухати отриманий аудіозапис, завантажити його або використати для подальшого створення відео.

Для доступу до доступних голосових моделей реалізовано окремий шлях /api/voices. Він надає можливість отримати перелік голосів, придатних для тексту-у-мову. На клієнтському рівні ці голоси відображаються у вигляді візуальних карток, з яких користувач може зробити свій вибір. У випадку, коли отримання списку голосів із сервера неможливе, клієнтська частина може підстрахуватися, застосовуючи попередньо визначений резервний перелік, що забезпечує безперебійну роботу інтерфейсу.

Ключовим аспектом цієї інтеграції є належна обробка можливих помилок. Збій може статися через відсутність дійсного API-ключа, некоректний ідентифікатор голосу, передачу порожнього тексту або недоступність самого зовнішнього сервісу. У таких випадках сервер повинен повернути чітке повідомлення про помилку, яке клієнтська частина відобразить користувачеві. Такий регламент значно підвищує загальну стійкість системи та спрощує процес верифікації [20, 22].

Отже, інтеграція через API ElevenLabs надає основну функціональність вебдодатку - автоматичне озвучування текстового вмісту. В реалізації VidVoice ця логіка ізольована у вільному серверному сервісі, що забезпечує безпечне поводження з API-ключем, коректну передачу параметрів озвучення, надійне збереження згенерованих звукових файлів та передачу кінцевого продукту клієнтській частині для його використання у створенні відеоматеріалів.

2.4.4 Обробка завантажених файлів та мультимедійних даних

Опрацювання файлів, що були завантажені, є невід'ємним компонентом серверної логіки для вебдодатку VidVoice, адже ця платформа оперує не лише текстовою інформацією, а й графічними, звуковими та відеоресурсами. Користувач має змогу завантажувати потрібні світлини для формування відеоряду, додавати вже готові звукові доріжки або ж використовувати аудіо,

синтезоване за допомогою ElevenLabs. Увесь цей масив даних мусить бути прийнятий сервером, збережений у відповідних сховищах та підготовлений до наступних етапів опрацювання.

На серверному рівні для зберігання файлового контенту задіяно директорію `storage`, яка логічно сегментована на декілька підрозділів. Графічні файли, завантажені користувачами, розміщуються у `storage/uploads`, звукові файли - у `storage/audio`, а фіналізовані відеоролики - у `storage/videos`. Такий поділ спрощує каталогізацію мультимедійних активів та полегшує подальшу роботу з ними.

Основні розташування для зберігання файлів деталізовано у табл. 2.11.

Таблиця 2.11 - Місця для збереження мультимедійних ресурсів

Директорія	Призначення
storage/uploads	Збереження зображень, завантажених користувачем
storage/audio	Збереження згенерованих або імпортованих аудіофайлів
storage/videos	Збереження готових відеофайлів
frontend/assets	Збереження статичних файлів клієнтської частини

Для процесу завантаження зображень задіяно API-точку доступу `/api/upload-images`. Клієнтська частина надсилає файли на сервер, який згодом розміщує їх у підкаталозі `storage/uploads` та надсилає назад підтвердження успішної операції. У цьому процесі критично важливим є моніторинг розширення файлів, адже для відеокомпіляції припустимими є лише певні формати зображень, наприклад PNG, JPG або WEBP.

Окремо врегульовано роботу зі звуковими файлами. Саундтрек може бути або згенерований через інтерфейс ElevenLabs, або імпортований безпосередньо користувачем. Для маніпуляції цими файлами залучається сервіс

`generated_files_service.py`, який відповідає за збереження, ведення обліку та надання списку доступних звукових і відеофайлів. Це дає змогу користувачу повторно експлуатувати вже створені елементи у майбутніх завданнях генерації відео.

Під час обробки мультимедійних даних серверна інфраструктура повинна обов'язково верифікувати наявність усіх необхідних компонентів перед активацією процесу рендерингу відео. Якщо відсутнє зображення чи не створено звуковий супровід, система не має ініціювати обробку, оскільки FFmpeg не отримає повного набору вхідних даних, потрібних для формування фінального відеофайлу [16].

Не менш значущим аспектом є генерація унікальних імен для файлів. Це запобігає ситуаціям перезапису даних, якщо декілька запитів або користувачів намагаються зберегти файли з ідентичними назвами. Унікальні ідентифікатори спрощують подальше знаходження збережених результатів та звернення до них з боку клієнтського інтерфейсу.

Отже, механізм опрацювання завантажених файлів у VidVoice реалізовано через спеціалізовані серверні ендпоінти та визначені сховища для різних типів мультимедіа. Така архітектурна побудова забезпечує впорядковану взаємодію із зображеннями, аудіо та відео, готуючи їх до фінального етапу збирання готового відеопродукту.

2.4.5 Реалізація генерації відео за допомогою FFmpeg

Створення відеороликів у вебдодатку VidVoice є фінальним етапом роботи з мультимедійним контентом. Після того, як користувач або сконструював, або завантажив аудіозапис разом із зображеннями, серверна логіка ініціює формування відеофайлу. Для цього залучається FFmpeg — потужний інструмент для опрацювання аудіо- та відеоданих, котрий дає змогу інтегрувати зображення, аудіо та відеопотоки в єдиний готовий файл [16].

У серверній архітектурі логіка, що відповідає за створення відео, винесена в окремий модуль за шляхом `backend/app/services/video_service.py`. Таке рознесення забезпечує відокремлення процедури генерування відео від API-

маршрутів, що робить структуру серверної частини значно чистішою. API-маршрут з адресою `/api/compose-video` отримує дані від клієнтського інтерфейсу, валідує наявність аудіофайлу та необхідних зображень, після чого передає ці дані до сервісу відеогенерації.

Основний принцип роботи полягає у тому, що завантажені користувачем зображення трансформуються у візуальний ряд, а згенерований чи наданий аудіофайл інтегрується як звуковий супровід. З цією метою сервер спершу з'ясовує загальну тривалість аудіо, а потім пропорційно розподіляє час показу між усіма завантаженими зображеннями. Наприклад, якщо аудіо триває шістдесят секунд, а користувач додав десять світлин, кожна світлина буде експонуватися близько шести секунд.

Для точного визначення загальної довжини аудіозапису застосовується допоміжний утиліт ffprobe, який є складовою FFmpeg. Цей інструмент дає змогу видобути технічну інформацію про медіафайл, включаючи його тривалість. Далі сервер формує командний рядок для FFmpeg, що запускає створення відеофайлу на основі наданих аудіо та зображень.

Під час самого процесу генерації життєво важливо врахувати формат кінцевого файлу, роздільну здатність відео, послідовність відображення кадрів та тривалість кожного окремого кадру. У межах цього вебзастосунку результат зберігається у форматі MP4, оскільки він має широку сумісність із браузерами, медіаплеєрами та мобільними пристроями. Отриманий відеофайл розміщується у папці `storage/videos`, після чого сервер надсилає клієнтській частині відомості про успішне створення результату.

Із точки зору кінцевого користувача, процес виглядає максимально спрощено: достатньо натиснути кнопку для створення відео, а система автоматично виконує всі необхідні обчислення та обробку на сервері. Це є суттєвою перевагою, адже користувачеві не потрібно вдаватися до використання відеоредакторів чи самостійно вручну компонувати аудіо та візуальні елементи.

При розробці механізму генерації відео було також інтегровано перевірку вхідних даних. Якщо аудіофайл не був наданий або користувач не завантажив

жодного зображення, сервер не розпочинає обробку і повертає повідомлення про помилку. Подібна валідація запобігає некоректному запуску FFmpeg і підвищує загальну стійкість системи [20, 22].

Отже, створення відео у вебдодатку VidVoice реалізовано завдяки серверному сервісу `video_service.py` та інструментарію FFmpeg. Серверна частина приймає аудіо та зображення, визначає тривалість звукової доріжки, конструює візуальний ряд, приєднує звук і зберігає готовий MP4-файл. Цей підхід дозволяє повністю автоматизувати процес створення відео та надати користувачеві фінальний продукт без необхідності використання стороннього програмного забезпечення для монтажу.

2.5 Проєктування бази даних

2.5.1 Загальна структура бази даних

Збереження інформації для вебзастосунку VidVoice реалізовано за допомогою реляційної бази даних PostgreSQL. Її головне завдання полягає у фіксації даних про користувачів, їхні профілі, робочі проєкти, окремі сцени, медіафайли, конфігурації озвучування, результати створення аудіо та відео, а також події, пов'язані з використанням кредитних одиниць. Застосування реляційної БД є логічним, адже система оперує великою кількістю взаємопов'язаних сутностей, які вимагають структурованого та надійного зберігання [6, 17].

Загальний проєкт структури бази даних було розроблено у програмному забезпеченні DBeaver. На рис. 2.14 показано ER-діаграму бази даних, яка відображає ключові таблиці системи та взаємозв'язки між ними.

Рисунок 2.14 - ER-схема бази даних вебдодатку VidVoice, створена у DBeaver

Як можна побачити з рисунка 2.14, база даних поділена на декілька логічних блоків таблиць. Перший блок відповідає за облікові записи користувачів та їхні налаштування. До нього входять таблиці `app_user`, `user_profile` та `credit_account`. У таблиці `app_user` зберігаються базові дані акаунту, зокрема електронна пошта, зашифрований пароль, ім'я користувача, його роль та дата реєстрації. Таблиця `user_profile` слугує для збереження додаткових уподобань користувача, таких як бажана мова інтерфейсу, тема оформлення, формат виведення даних, налаштування автозбереження та автоперекладу. Таблиця `credit_account` призначена для моніторингу наявного залишку кредитів у користувача.

Другий набір таблиць пов'язаний із проєктами, створеними користувачами. Ядром є таблиця `project`, де фіксуються назва проєкту, його опис, власник, основна мова, поточний стан, а також дати створення та останнього оновлення. Для деталізації внутрішньої структури проєкту задіяна таблиця `project_scene`, яка містить дані про окремі сцени, їхній текстовий вміст, переклад, мову сцени,

пов'язані графічні елементи, аудіо та часову тривалість. Такий підхід дає змогу розділяти відеоматеріал на логічні сегменти, які згодом використовуються під час рендерингу фінального відеофайлу.

Третій блок таблиць стосується мультимедійних активів. Таблиця `media_file` накопичує відомості про файли, як завантажені, так і згенеровані: інформацію про власника, тип медіа, оригінальну назву, фізичне розташування файлу, публічне посилання, MIME-тип, обсяг даних та контрольну суму. Для категоризації медіафайлів використовується довідкова таблиця `media_type`, яка допомагає розрізняти зображення, аудіо та відеоматеріали.

Окрема частина бази даних виділена для фіксації процесів генерації контенту. Таблиця `tts_generation` реєструє результати синтезу мовлення: прив'язку до проєкту, сцену, обраного провайдера, параметри голосу, використану модель, статус операції, вхідний текст, налаштування голосу, кількість оброблених символів, отриманий аудіофайл та можливі повідомлення про помилки. Для створення відео використовується таблиця `video_generation`, яка містить дані про проєкт, статус, формат кадру, шлях до фінального відеофайлу, його тривалість та інформацію про помилки. Зв'язок між процесом створення відео, сценами та застосованими анімаціями забезпечується через таблицю `video_generation_scene`.

У структурі бази також передбачено таблиці для керування перекладами та моніторингу активності користувачів. Таблиця `translation_job` зберігає дані про завдання перекладу: вихідну та цільову мови, оригінальний текст, перекладений варіант, статус завдання та можливі помилки. Таблиця `usage_event` застосовується для реєстрації всіх дій у системі: будь-якої генерації аудіо, перекладу, створення відео або інших операцій, які можуть впливати на списання кредитів. Класифікація цих подій здійснюється за допомогою таблиці `usage_event_type`.

Крім того, у схемі бази даних присутні довідкові таблиці, такі як `language`, `project_status`, `job_status`, `export_format`, `ui_theme`, `aspect_ratio`, `animation_type`, `provider`, `voice` та `voice_model`. Вони призначені для

зберігання константних (незмінних) значень: переліку мов, статусів виконання завдань, форматів експорту, тем оформлення інтерфейсу, співвідношень сторін, типів анімацій, провайдерів синтезу мовлення, доступних голосів та моделей озвучування. Використання таких довідників унеможливлює дублювання одних і тих самих даних і робить архітектуру бази більш організованою.

Отже, сукупна архітектура бази даних вебдодатку VidVoice охоплює увесь цикл роботи системи: від реєстрації користувача та збереження його налаштувань профілю, через створення проєктів і маніпуляції зі сценами, роботу з медіаконтентом, до генерації аудіо, механізмів перекладу тексту, фінального створення відео та обліку використання ресурсів (кредитів). Така конструкція є практично обґрунтованою для майбутнього масштабування вебдодатку, оскільки дозволяє надійно зберігати всі результати роботи користувача, забезпечуючи логічний зв'язок між усіма етапами виробництва мультимедійного продукту.

2.5.2 Таблиця користувачів

В інформаційній базі вебзастосунку VidVoice спеціальний модуль таблиць відведено для фіксації відомостей про осіб-користувачів, їхні профілі, конфігурації та наявні кредитні одиниці. Це є необхідним для втілення особистого простору користувача, сторінки проєктів, моніторингу споживання ресурсів озвучення та фіксації індивідуальних параметрів роботи системи. Подібна архітектура узгоджується з загальноприйнятими постулатами розробки інформаційних систем, де відомості про користувачів відокремлюють від функціональних даних, що стосуються проєктів та результатів обробки [6, 17].

Ключовою таблицею в цьому блоці є `app_user`. Вона акумулює фундаментальні відомості про обліковий запис користувача. До її складу входять такі атрибути, як `id`, `email`, `password_hash`, `display_name`, `role`, `created_at` та `updated_at`. Атрибут `id` слугує первинним ключем таблиці та використовується для формування зв'язків з іншими таблицями бази даних. Поле `email` призначене для зберігання електронної адреси користувача, а `password_hash` - для пароля у захищеному вигляді (через хешування). Це

забезпечує відмову від зберігання відкритого тексту пароля, що є критично важливим для безпеки системи.

Атрибут `display_name` залучається для показу імені користувача в інтерфейсі, приміром, на сторінці його профілю чи у меню особистого кабінету. Поле `role` може бути задіяне для розмежування користувачів за рівнями доступу, наприклад, звичайний користувач чи адміністратор. Атрибути `created_at` та `updated_at` фіксують дати створення та останньої модифікації запису, що дає можливість простежувати життєвий цикл облікового запису.

Для зведення додаткових параметрів користувача задіяна таблиця `user_profile`. Вона корелює з таблицею `app_user` через поле `user_id`. У цій таблиці зберігаються конфігурації, які не є безпосередньо пов'язаними з даними авторизації. Наприклад, `language_id` визначає мову інтерфейсу, `theme_id` - обрану візуальну тему, `default_export_format_id` - стандартний формат вивантаження, тоді як поля `auto_save` та `auto_translate` регулюють функції автоматичного збереження та автоматичного перекладу. Такий поділ є доцільним, оскільки базова інформація про користувача та його індивідуальні налаштування зберігаються окремо.

Окремо в базі даних передбачено сутність `credit_account`. Вона також має зв'язок із користувачем через `user_id` і використовується для обліку наявних у розпорядженні кредитів. Таблиця містить поле `total_credits`, яке відображає поточний обсяг доступних ресурсів користувача, та поле `updated_at`, що фіксує час останньої зміни цього балансу. Наявність такої таблиці є значущою, оскільки генерування аудіо через зовнішній сервіс, такий як ElevenLabs, може мати обмеження, пов'язані з обсягом символів чи використаною кількістю одиниць кредиту [10].

Конфігурація таблиць, що стосуються користувачів, представлена у табл. 2.10.

Таблиця 2.10 - Склад таблиць користувачів та їхнє функціональне призначення

Таблиця	Основні поля	Призначення
app_user	id, email, password_hash, display_name, role, created_at, updated_at	Збереження основних даних облікового запису користувача
user_profile	user_id, language_id, theme_id, default_export_format_id, auto_save, auto_translate	Збереження персональних налаштувань користувача
credit_account	user_id, total_credits, updated_at	Облік доступних кредитів користувача

Як демонструє таблиця 2.10, відомості про користувача розчленовані на три логічні сегменти: сам обліковий запис, налаштування його профілю та кредитний залишок. Це надає структурі бази даних більшої прозорості та спрощує її подальше розширення. Наприклад, у майбутньому до профілю можна буде додати нові параметри, не вносячи змін до таблиці даних авторизації, а до кредитного рахунку - лог поповнень чи витрат ресурсів.

Зв'язки між таблицями цього сегмента реалізовані за принципом «один до одного». Кожен користувач із таблиці `app_user` асоціюється з одним записом у `user_profile` та одним записом у `credit_account`. Такий метод організації дозволяє тримати персональні дані в структурованому вигляді та уникнути дублювання даних.

Таким чином, таблиці, призначені для користувачів у базі даних VidVoice, забезпечують зберігання ключової інформації про обліковий запис, індивідуальні налаштування та ліміти кредитів. Вони є базовим елементом системи, оскільки саме через сутність користувача здійснюється зв’язування проєктів, медіафайлів, результатів генерації аудіо та відео, а також фіксація історії використання ресурсів, що забезпечує цілісність і узгодженість даних у межах застосунку.

2.5.3 Таблиця проєктів користувача

Записи, створені користувачами у системі VidVoice, зберігаються у спеціалізованому модулі таблиць, який тісно інтегрований з даними про проекти та окремі сцени. Це розроблено для того, аби користувачі мали змогу працювати над кількома проектами одночасно, повертатися до них пізніше, вносити зміни у певні сцени та фіксувати результати генерації аудіо- чи відеоматеріалів. Такий архітектурний підхід унеможливлює обмеження вебдодатку лише одноразовим створенням файлу, натомість дозволяючи повноцінну роботу з управлінням проектами [6, 17].

Ключовим елементом цього блоку є таблиця `project`. Вона служить для агрегації загальної інформації, притаманної користувацькому проекту. Серед її основних стовпців виокремлюються: `id`, `owner_id`, `status_id`, `primary_language_id`, `title`, `description`, `created_at` та `updated_at`. Поле `id` виступає первинним ідентифікатором таблиці, тоді як `owner_id` забезпечує зв'язок між проектом та конкретним обліковим записом користувача у таблиці `app_user`, гарантуючи таким чином приналежність кожного проекту певному власнику.

Поле `title` відображає найменування проекту на сторінці, присвяченій роботам користувача («Мої проєкти»). Стовпець `description` призначений для короткого резюме вмісту проекту. Зв'язок `status_id` веде до таблиці `project_status`, де зберігається перелік можливих станів проекту, як от «завершено», «в стадії опрацювання» або «чернетка». Атрибут `primary_language_id` констатує основну мову проекту, посилаючись на таблицю `language`.

Для деталізації структури кожного проекту використовується таблиця `project_scene`. Вона фіксує кожну окрему сцену, з яких складається кінцевий відеоматеріал. Основні поля цієї таблиці включають: `id`, `project_id`, `scene_number`, `source_text`, `translated_text`, `source_language_id`, `target_language_id`, `image_media_id`, `audio_media_id`, `duration_seconds`,

`created_at` та `updated_at`. Поле `project_id` необхідне для прив'язки сцени до відповідного проекту. Поле `scene_number` диктує послідовність сцени у загальному проекті, що є критичним для коректного формування відеоряду.

Поля `source_text` та `translated_text` вміщують початковий текст та його переклад відповідно. Це дозволяє підтримувати роботу з двома мовами - українською та англійською - у межах одного проекту. Поля `source_language_id` та `target_language_id` також посилаються на таблицю `language`, визначаючи мови вихідного та цільового тексту. Поля `image_media_id` та `audio_media_id` створюють зв'язки з потрібними файлами у таблиці `media_file`, завдяки чому кожна сцена може мати індивідуальні візуальні та аудіокомпоненти.

Схема організації таблиць, що оперують даними користувацьких проектів, представлена у табл. 2.11.

Таблиця 2.11 - Таблиці, що описують проекти користувача

Таблиця	Основні поля	Призначення
project	id, owner_id, status_id, primary_language_id, title, description, created_at, updated_at	Збереження загальної інформації про проєкти користувача
project_scene	id, project_id, scene_number, source_text, translated_text, image_media_id, audio_media_id, duration_seconds	Збереження окремих сцен проєкту
project_status	id, code, name	Збереження статусів проєктів
language	id, code, name	Збереження мов, які використовуються в проєктах і сценах

Як демонструє таблиця 2.11, логіка побудови проектних таблиць сфокусована на принципі "один до багатьох": один користувач може мати численні проекти, а кожен проект, у свою чергу, може містити безліч сцен. Це відповідає схемі взаємозв'язків: один рядок у `app_user` може бути пов'язаний з багатьма записами у `project`, і, відповідно, один запис у `project` - з великою кількістю записів у `project_scene`.

Практична цінність цієї архітектури полягає у можливості користувача відокремлювати різні відеоматеріали у різні проекти. Наприклад, один проект може бути виділений під навчальний матеріал, інший - для ділової презентації, а третій - для короткого рекламного ролика. Кожен із цих проектів буде незалежно зберігати свої сцени, наповнення тексту, переклади, візуальні та аудіодані.

Підсумовуючи, таблиці `project` та `project_scene` формують цілісну основу для фіксації результатів роботи користувачів у VidVoice. Вони забезпечують структурування процесу створення відео, дозволяючи поетапно зберігати текстові та мультимедійні компоненти, а також чітко прив'язувати проекти до конкретного користувача, мовних налаштувань, поточного статусу та медіаресурсів.

2.5.4 Таблиці для збереження аудіо, зображень та відео

Щодо опрацювання мультимедійних даних у сховищі VidVoice, виділено окремий масив таблиць. Цей набір слугує для фіксації відомостей про завантажені візуальні файли, створені чи імпортовані звукові доріжки, фінальні відеоматеріали, а також результати різних процесів створення контенту. Такий розподіл необхідний, аби кожна одиниця контенту мала чітке прив’язування до користувача, проєкту чи конкретної сцени. Використання ізольованих таблиць для медіа-елементів є логічним кроком, адже система оперує різнорідними даними, що вимагають структурованого зберігання [6, 17].

Фундаментальною таблицею для фіксації мультимедійних елементів є `media_file`. Вона акумулює інформацію про усі файли, задіяні у вебзастосунку. Серед ключових атрибутів цієї таблиці варто відзначити: ідентифікатор (`id`), ідентифікатор власника (`owner_id`), ідентифікатор типу медіа (`media_type_id`),

первинну назву файлу (`original_filename`), шлях до зберігання (`storage_path`), публічну адресу (`public_url`), тип вмісту (`mime_type`), розмір у байтах (`file_size_bytes`), контрольну суму SHA256 (`checksum_sha256`) та дату створення (`created_at`). Поле `owner_id` забезпечує зв'язок файлу з особою-користувачем, тоді як `media_type_id` визначає сутність файлу - чи це візуальний контент, аудіо, чи відео. Фізичне розміщення файлів відбувається у файловому сховищі, а база даних утримує метадані, зокрема шлях до них.

Для категоризації мультимедійного контенту застосовується таблиця `media_type`. Вона містить такі поля: ідентифікатор (`id`), унікальний код (`code`) та назву (`name`). Завдяки цій таблиці система здатна визначити призначення файлу в певному робочому циклі: чи це зображення для сцени, аудіокомпонент для озвучення, чи фінальний відеовихід в результаті генерації.

Для документування етапів створення звукових доріжок використовується таблиця `tts_generation`. Вона реєструє дані щодо генерації аудіо: прив'язку до проєкту, сцени, обраного провайдера, голосу, моделі, поточний статус, вихідний текст, налаштування голосових параметрів, кількість оброблених символів, посилання на згенерований аудіофайл та можливі помилки. Поле `output_audio_media_id` встановлює зв'язок результату генерації з таблицею `media_file`, де зберігаються повні відомості про звуковий файл. Цей механізм дає змогу фіксувати не лише кінцевий продукт, а й параметричні умови його створення.

Процедура формування відеодоріжок відображається у таблиці `video_generation`. Вона фіксує дані про створення відеофайлу: прив'язку до проєкту, стан процесу, ідентифікатор співвідношення сторін, посилання на результат відео, тривалість, час початку та завершення, а також будь-яке повідомлення про збій. Атрибут `output_video_media_id` корелює фінальне відео з відповідним записом у `media_file`. Сам процес рендерингу відео здійснюється на серверній інфраструктурі із залученням інструментарію FFmpeg, що забезпечує злиття аудіо та зображень в уніфікований відеопотік [16].

Оскільки відео може складатися з послідовності окремих кадрів (сцен), передбачено таблицю `video_generation_scene`. Вона слугує для асоціації конкретного процесу відеогенерації зі сценами проєкту, визначаючи при цьому їхню черговість відтворення. Ця таблиця включає поля: ідентифікатор відеогенерації (`video_generation_id`), ідентифікатор сцени (`scene_id`), порядковий номер (`position`) та ідентифікатор типу анімаційного ефекту (`animation_type_id`). Це дає змогу системі чітко розуміти, які сцени входять до фінального відео, як вони мають бути розташовані, та які візуальні переходи застосовувати.

Крім того, у базі присутні довідкові таблиці `aspect_ratio` та `animation_type`. Таблиця `aspect_ratio` використовується для зберігання конфігурацій співвідношень сторін екрану, наприклад, зазначення ширини та висоти в пікселях. Таблиця `animation_type` містить перелік можливих стилів анімації, які можуть бути застосовані до сцен під час компонування відео. Наявність таких класифікаторів підвищує адаптивність системи до майбутніх розширень, наприклад, для інтеграції нових відеоформатів чи впровадження свіжих ефектів переходу.

Також така структура бази даних забезпечує гнучкість при масштабуванні функціоналу генерації відео, оскільки дозволяє легко додавати нові типи сцен, змінювати логіку їхнього відтворення або розширювати набір доступних анімацій без необхідності суттєвих змін у загальній архітектурі системи. Завдяки цьому досягається висока модульність і підтримуваність проєкту, що є важливим для подальшого розвитку вебзастосунку VidVoice.

Окрім цього, використання окремих довідникових таблиць дозволяє уникнути дублювання даних і забезпечує їхню цілісність у межах всієї системи. Такий підхід також спрощує адміністрування бази даних та підвищує зручність внесення змін у конфігураційні параметри без впливу на основні бізнес-дані.

Схема структури таблиць, призначених для маніпуляцій з аудіо, зображеннями та відеоматеріалами, представлена у табл. 2.12.

Таблиця 2.12 - Схематизація таблиць для зберігання мультимедійних даних

Таблиця	Основні поля	Призначення
media_file	id, owner_id, media_type_id, original_filename, storage_path, public_url, mime_type, file_size_bytes, created_at	Збереження інформації про завантажені та згенеровані файли
media_type	id, code, name	Класифікація медіафайлів за типами
tts_generation	id, project_id, scene_id, voice_id, voice_model_id, source_text, output_audio_media_id, character_count, created_at, completed_at	Збереження інформації про генерацію аудіо
video_generation	id, project_id, status_id, aspect_ratio_id, output_video_media_id, duration_seconds, created_at, completed_at	Збереження інформації про генерацію відео
video_generation_scene	video_generation_id, scene_id, position, animation_type_id	Зв’язок між відеогенерацією та сценами
aspect_ratio	id, code, width, height, name	Збереження форматів кадру відео
animation_type	id, code, name	Збереження типів анімацій для сцен

Як можна побачити з таблиці 2.12, самі файли (аудіо, зображення, відео) не розміщуються безпосередньо у сховищі бази даних. Табличні записи містять лише службову інформацію: метадані про файли, їхню класифікацію, місця розташування, автентифікацію власника та референси до інших об'єктів. Такий підхід є прагматичним, оскільки завантаження великих обсягів бінарних даних у БД є неефективним; доцільніше тримати їх у зовнішньому сховищі, а в реляційній моделі залишати лише описові дані.

Отже, набір таблиць, призначений для обліку аудіо, візуальних та відеоматеріалів, забезпечує повне управління мультимедійними ресурсами у застосунку VidVoice. Вони забезпечують коректне зв’язування файлів із користувачами, робочими проєктами, окремими сценами, а також процесами синтезу звуку та створення відео. Це формує необхідну основу для трасування історії роботи користувача, сприяє повторному використанню активів та дозволяє масштабувати функціональність системи у майбутньому.

2.5.5 Зв’язки між таблицями бази даних

Зв'язки у базі даних VidVoice організовані з метою гарантувати повну та несуперечливу збереженість даних, що стосуються користувачів, їхніх розробок, окремих епізодів, файлів медіа, процесів створення аудіо та відео, перекладів та обліку використаних кредитів. Ключовим принципом цієї архітектури є сегментація даних на осмислені блоки, де кожна база даних відповідає за певну функцію, а взаємозв'язки між ними реалізуються через зовнішні ключі. Цей метод відповідає усталеним нормам проєктування реляційних сховищ даних та ефективно запобігає надмірному дублюванню інформації [6, 17].

Ядром для зберігання інформації про користувачів є таблиця `app_user`. Вона має зв'язки з таблицями `user_profile`, `credit_account`, `project` та `media_file`. Це означає, що один обліковий запис користувача може мати відповідний профіль, баланс кредитів, значну кількість проєктів та медіафайлів. Зв'язок між `app_user` та `project` реалізовано за моделлю «один-до-багатьох», оскільки один користувач може ініціювати створення багатьох проєктів.

Таблиця `project` інтегрована з таблицею `project_scene`, адже кожен проєкт складається з одного або декількох епізодів. Епізод (сцена) містить текстовий вміст, його переклад, порядок відображення, а також посилання на графічні та аудіоресурси. Для прив'язки візуального та звукового супроводу у `project_scene` задіяні поля `image_media_id` та `audio_media_id`, які вказують на відповідні записи у `media_file`. Таким чином, кожен епізод може мати унікальний мультимедійний супровід.

Таблиця `media_file` слугує універсальним сховищем для всіх типів медіа-вмісту: зображень, звуку та відео. Візуалізація типу контенту відбувається через асоціацію з таблицею `media_type`. Такий підхід усуває потребу у створенні окремих таблиць для кожного формату, дозволяючи агрегувати всі медіадані в єдиній, централізованій таблиці.

Окрема група зв'язків стосується процесу синтезу мовлення. У таблиці `tts_generation` фіксуються дані про екзекуцію озвучення тексту. Вона взаємодіє з таблицями `project`, `project_scene`, `provider`, `voice`, `voice_model`, `job_status` та `media_file`. Таблиці `provider`, `voice` та `voice_model` використовуються для каталогізації сервісу озвучування, доступних голосових профілів та використовуваних моделей синтезу. У контексті нашої розробки таким постачальником є платформа ElevenLabs [10]. Поле `output_audio_media_id` забезпечує зв'язок результату синтезу із записом у `media_file`, де зберігається готовий аудіофайл.

Для задач перекладу задіяно таблицю `translation_job`. Вона пов'язана з користувачем, його проєктом, сценою, а також мовними параметрами та статусом виконання завдання. Зв'язки з таблицею `language` дозволяють чітко ідентифікувати мову оригіналу та мову перекладу. Таблиця `job_status` необхідна для фіксації поточного стану роботи, як-от очікування, успішне виконання чи виникнення помилки.

Створення відеоматеріалу деталізовано у таблицях `video_generation` та `video_generation_scene`. Таблиця `video_generation` акумулює загальні параметри відеопроцесу: прив'язку до проєкту, статус, співвідношення сторін,

кінцевий результат та загальну протяжність. Вона пов'язана з `project`, `job_status`, `aspect_ratio` та `media_file`. Таблиця `video_generation_scene` слугує для асоціації конкретного відеоряду з окремими сценами проєкту. Вона також містить ідентифікатор `animation_type_id`, який вказує на тип застосованої анімації або перехідного ефекту для сцени. Безпосереднє формування фінального відеофайлу в системі забезпечується інструментарієм FFmpeg [16].

Облік ресурсного споживання реалізовано через таблицю `usage_event`. Вона пов'язана з користувачем, його проєктом, типом виконаної операції та відповідними процесами генерації чи перекладу. Таблиця `usage_event_type` класифікує цю подію, наприклад як генерація аудіо, створення ролика чи переклад. Така структура дає можливість точно зафіксувати обсяг використаних ресурсів у ході роботи користувача із системою, з подальшою можливістю зіставлення цих даних з таблицею `credit_account`.

У структурі також присутні довідкові таблиці `export_format` та `ui_theme`. Вони корелюють із таблицею `user_profile` і використовуються для стандартизації формату експорту та налаштувань візуального оформлення інтерфейсу.

Також важливу роль у структурі бази даних відіграє таблиця user_profile, яка містить додаткову інформацію про користувача та його персональні налаштування. Через зв’язки з довідковими таблицями система зберігає обраний стиль оформлення інтерфейсу, параметри експорту та інші індивідуальні налаштування. Такий підхід дозволяє забезпечити персоналізацію роботи вебдодатку та створює основу для подальшого розширення функціональних можливостей системи без зміни основної структури бази даних.

Для підсумування ключових взаємозалежностей між елементами бази даних наведено табл. 2.13.

Таблиця 2.13 - Основні взаємозв’язки між складовими бази даних

Основна таблиця	Прив’язана таблиця	Тип зв’язку	Призначення
app_user	user_profile	один до одного	Збереження налаштувань користувача
app_user	credit_account	один до одного	Облік кредитів користувача
app_user	project	один до багатьох	Збереження проєктів користувача
project	project_scene	один до багатьох	Поділ проєкту на окремі сцени
media_type	media_file	один до багатьох	Визначення типу медіафайлу
media_type	project_scene	один до багатьох	Прив’язка зображень та аудіо до сцен
project	tts_generation	один до багатьох	Збереження генерацій аудіо в межах проєкту
voice	tts_generation	один до багатьох	Визначення голосу для озвучування
voice_model	tts_generation	один до багатьох	Визначення моделі синтезу мовлення
project	video_generation	один до багатьох	Збереження генерацій відео в межах проєкту
video_generation	video_generation_scene	один до багатьох	Зв’язок відео з окремими сценами
project_scene	video_generation_scene	один до багатьох	Використання сцен у відеогенерації
app_user	usage_event	один до багатьох	Облік дій користувача
usage_event_type	usage_event	один до багатьох	Класифікація подій використання

Як можна констатувати з огляду таблиці 2.13, архітектура бази даних центрована навколо трьох магістральних сутностей: користувача, проєкту та медіаконтенту. Користувач ініціює створення проєктів, які, своєю чергою, складаються зі сцен, сцени використовують медіафайли, а процеси генерації аудіо та відео чітко асоціюються з відповідними робочими потоками. Завдяки такій організації забезпечується повне відстеження ланцюжка створення будь-

якого мультимедійного продукту - від введення текстової основи до випуску фінального відеоматеріалу.

Отже, взаємозв'язки у базі даних VidVoice формують логічно зв'язну та інкапсульовану структуру даних. Усі двадцять три таблиці виконують свою специфічну роль: частина відповідає за зберігання базових даних про суб'єктів, їхні роботи та файли, тоді як інша частина виконує довідкову або сервісну функцію. Це забезпечує високу адаптивність бази даних для майбутнього масштабування вебдодатку, дозволяючи легко інтегрувати нові голосові пакети, відеоформати, типи анімації, тарифні плани чи розширені журнали активності користувачів.

2.6 Алгоритми роботи системи

2.6.1 Алгоритм автоматичного озвучування тексту

Процедура автоматичного перетворення тексту на мовлення є однією з центральних частин функціоналу вебзастосунку VidVoice. Її призначення полягає у трансформуванні тексту, наданого користувачем, у звуковий файл, залучаючи до роботи сервіс синтезу мовлення ElevenLabs. У межах архітектури ця процедура об’єднує роботу фронтенд-частини, бекенд-частини, зовнішнього програмного інтерфейсу (API) та сховища файлів.

Початок роботи над озвучуванням ініціюється діями користувача в інтерфейсі модуля "Write & Translate". Користувач вводить текст або завантажує його вміст, за необхідності здійснює переклад, обирає бажаний тембр голосу, налаштовує нюанси генерації мовлення та активує кнопку створення. Після цього, фронтенд формує відповідний запит до бекенд-частини. Обмін даними між клієнтом і сервером реалізується через API, що відповідає принципам клієнт-серверної побудови вебзастосунків [6, 15].

Загальну схему автоматичного перетворення тексту на мовлення ілюструє рис. 2.15.

Рисунок 2.15 - Схема автоматичного озвучення тексту

На рисунку 2.15 логічно продемонструвати послідовність кроків: починаючи від введення тексту користувачем і закінчуючи отриманням фінального аудіофайлу. Основні етапи включають: введення тексту, валідацію даних, вибір голосового профілю, відправлення запиту на сервер, звернення до API ElevenLabs, збереження згенерованого звуку та повернення результату споживачеві.

На першій стадії користувач вносить текст у поле "script-text" або завантажує відповідний файл. Якщо було обрано текстовий файл, фронтенд зчитує його вміст та автоматично поміщає його у відповідне вікно. Після цього

користувачу надається можливість виконати переклад тексту українською або англійською. Переклад не є обов'язковим кроком, але він розширює можливості системи, дозволяючи підготувати текст для озвучення потрібною мовою.

Далі, користувач обирає голос для синтезу. Перелік доступних голосів отримується завдяки API-запиту до бекенду, після чого він відображається в інтерфейсі у форматі карткових елементів. Обраний голос фіксується у стані фронтенду та використовується під час формування запиту на генерацію звуку. Окрім голосу, користувач має можливість налаштувати параметри синтезу мовлення, зокрема рівень стабільності, ступінь подібності тембру, стилістику та швидкість відтворення.

Перед надсиланням запиту здійснюється перевірка наданих даних. Якщо текстове поле залишається порожнім, процес генерації не розпочинається, а користувач отримує сповіщення про необхідність ввести текст. Така перевірка унеможливлює надмірні звернення до серверної частини та стороннього сервісу. Обробка некоректних дій з боку користувача є важливою складовою забезпечення стабільної роботи програмного забезпечення [20, 22].

Після успішного підтвердження валідації, фронтенд викликає функцію `api.generateAudio()`, яка ініціює POST-запит до серверного маршруту `/api/generate-audio`. У такому запиті передаються сам текст, унікальний ідентифікатор обраного голосу, обрана модель синтезу та конфігурація параметрів голосу. Сервер приймає ці відомості та перенаправляє їх до сервісу `elevenlabs_service.py`, який відповідає за комунікацію з ElevenLabs API.

Бекенд виконує функцію проміжної ланки між клієнтським інтерфейсом і зовнішнім сервісом ElevenLabs. Це критично важливо з міркувань безпеки, адже ключ доступу API зберігається виключно на сервері, а не передається на клієнт. Отримавши запит, сервер формує звернення до ElevenLabs API, надсилає йому текст та налаштування озвучення, і очікує на повернення даних у формі аудіо.

У випадку успішної генерації, сервер фіксує отриманий аудіофайл у каталозі `storage/audio`. Після цього формується відповідь для фронтенду, яка містить інформацію про створений файл: його найменування, шлях або публічне

посилання. Фронтенд інтегрує аудіофайл у перелік доступних файлів та візуалізує його для користувача. Далі користувач може прослухати аудіо, завантажити його або використати для подальшого створення відео.

При виникненні будь-якої помилки система має повернути користувачеві зрозуміле повідомлення. Помилка може виникнути через відсутність тексту, некоректний ідентифікатор голосу, відсутність API-ключа, перевищення встановлених лімітів сервісу або недоступність самого ElevenLabs API. У такій ситуації сервер повертає сповіщення про проблему, яке фронтенд відображає у користувацькому інтерфейсі.

Основний ланцюжок роботи процедури можна описати наступним чином:

1. користувач вносить текст або завантажує його;
2. за потреби ініціюється переклад наданого тексту;
3. користувач обирає тембр голосу та налаштування озвучення;
4. фронтенд виконує перевірку введених даних;
5. генерується запит на маршрут `/api/generate-audio`;
6. сервер перенаправляє дані до ElevenLabs API;
7. ElevenLabs повертає згенерований аудіопотік;
8. сервер зберігає аудіофайл у `storage/audio`;
9. фронтенд візуалізує фінальний результат для користувача.

Таким чином, процедура автоматичного перетворення тексту на мовлення у вебдодатку VidVoice забезпечує повний цикл роботи з текстовим контентом: від введення до отримання готового звукового файлу. Її реалізація ґрунтується на скоординованій роботі фронтенду, API бекенду, сервісу ElevenLabs та системи зберігання файлів. Такий підхід дозволяє автоматизувати процес створення аудіо та підготувати його для наступного етапу - генерації відеоматеріалу.

2.6.2 Алгоритм генерації відео на основі аудіо та зображень

Процедура творення відео у вебзастосунку VidVoice стартує після того, як користувач підготував аудіофайл та завантажив візуальні матеріали. Його головна мета - зведення аудіодоріжки та набору картинок в єдиний відеофайл формату MP4. У цій системі відео не створюється "з нуля" через штучний інтелект, а

формується шляхом компіляції вже підготовлених користувачем елементів. Цей підхід є раціональним, оскільки він уможливлює автоматизацію нескладного процесу зведення без потреби у застосуванні окремих редакторів відео.

У вебдодатку процес творення відео починається з блоку "Завантаження зображень", де користувач додає картинки у форматах PNG, JPG чи WEBP. Далі, у блоці "Компонування відео", користувач ініціює створення відеофайлу. Сторона клієнта перевіряє, чи є згенероване чи завантажене аудіо, а також чи додано хоча б один візуальний матеріал. Якщо необхідні компоненти відсутні, система повинна сповістити користувача про помилку та не розпочинати обробку [20, 22].

Додатково на серверній стороні виконується фінальна валідація отриманих даних, що включає перевірку цілісності аудіо- та зображень, а також відповідність їхніх форматів вимогам системи. Після успішної перевірки запускається процес обробки за допомогою FFmpeg, який здійснює синхронізацію аудіодоріжки з послідовністю зображень та формує готовий відеофайл. Отриманий результат зберігається на сервері та стає доступним для користувача через інтерфейс завантаження або перегляду.

Після завершення обробки система формує посилання на готовий відеофайл та повертає його клієнтській частині вебдодатку. Користувач отримує можливість переглянути результат безпосередньо у браузері або завантажити його на свій пристрій для подальшого використання. Такий підхід забезпечує повний цикл створення відеоконтенту в межах одного вебдодатку та значно спрощує процес підготовки мультимедійних матеріалів для кінцевого користувача.

Загальна послідовність дій щодо творення відео на базі аудіо та зображень відображена на рис. 2.16.

Рисунок 2.16 - Послідовність дій з творення відео на основі аудіо та зображень

На рисунку 2.16 продемонстровано послідовність етапів від завантаження візуальних матеріалів до отримання фінального відеофайлу. Ключовими стадіями є вибір аудіофайлу, завантаження зображень, верифікація наявності необхідних даних, надсилання запиту на сервер, опрацювання файлів за допомогою FFmpeg, фіксація готового відео та передача результату власнику.

Після натискання кнопки "Створити відео" клієнтська частина формує запит до серверного шляху /api/compose-video. У цьому запиті передається інформація про аудіофайл та перелік зображень, які слід задіяти для формування відео. Серверна частина приймає ці відомості, звіряє наявність файлів у сховищі та передає їх до сервісу video_service.py.

На сервері для створення відео залучається FFmpeg. Цей інструмент надає можливість оперувати мультимедійними файлами, зокрема аудіо, візуальними матеріалами та відеопотоками [16]. Спершу сервер визначає часовий відрізок

аудіофайлу. Для цього використовується допоміжний засіб ffprobe, який дозволяє отримати технічні параметри медіафайлу. Після визначення тривалості аудіо система розраховує, який час буде демонструватися кожне зображення у відеоряді.

Далі сервер формує директиву для FFmpeg, за допомогою якої візуальні матеріали інтегруються з аудіодоріжкою. Якщо зображень декілька, вони демонструються одне за одним з однаковим або попередньо визначеним інтервалом. Після завершення опрацювання створюється відеофайл у форматі MP4, який розміщується у теці storage/videos. Відомості про фінальний файл можуть бути записані у таблицю media_file, а сам процес творення — у таблицю video_generation [17].

Основний хід роботи алгоритму можна представити наступним чином:

1. користувач обирає або створює аудіофайл;
2. користувач завантажує візуальні матеріали;
3. клієнтська частина здійснює перевірку наявності аудіо та зображень;
4. надсилається запит за маршрутом /api/compose-video;
5. сервер звіряє наявність файлів у сховищі;
6. встановлюється тривалість аудіофайлу;
7. зображення поєднуються з аудіо засобами FFmpeg;
8. фінальний MP4-файл зберігається у storage/videos;
9. результат повертається до клієнтської частини;
10. користувач отримує змогу завантажити готове відео.

У разі виникнення збою сервер надсилає повідомлення з поясненням причини проблеми. Наприклад, збій може статися через відсутність аудіофайлу, відсутність зображень, некоректний формат файлу чи збій у роботі FFmpeg. Обробка подібних ситуацій є важливою, оскільки це запобігає некоректному завершенню процесу та дозволяє надати користувачу зрозуміле повідомлення [20, 22].

Таким чином, послідовність дій з творення відео у вебзастосунку VidVoice забезпечує автоматизоване формування відеофайлу на основі аудіо та візуальних

матеріалів. Клієнтська частина відповідає за підготовку та відправлення даних, тоді як серверна частина здійснює опрацювання файлів, запуск FFmpeg, фіксацію результату та повернення готового відео власнику.

2.6.3 Загальний сценарій взаємодії користувача із системою

Узагальнений сценарій взаємодії користувача з VidVoice охоплює основні етапи роботи системи - від входу та введення тексту до генерації аудіо, завантаження зображень, створення відео та отримання готового результату — і реалізований за клієнт-серверною архітектурою, де обробка даних виконується на сервері. [6, 15].

Загальну послідовність дій користувача із системою ілюструє діаграма на рис. 2.17.

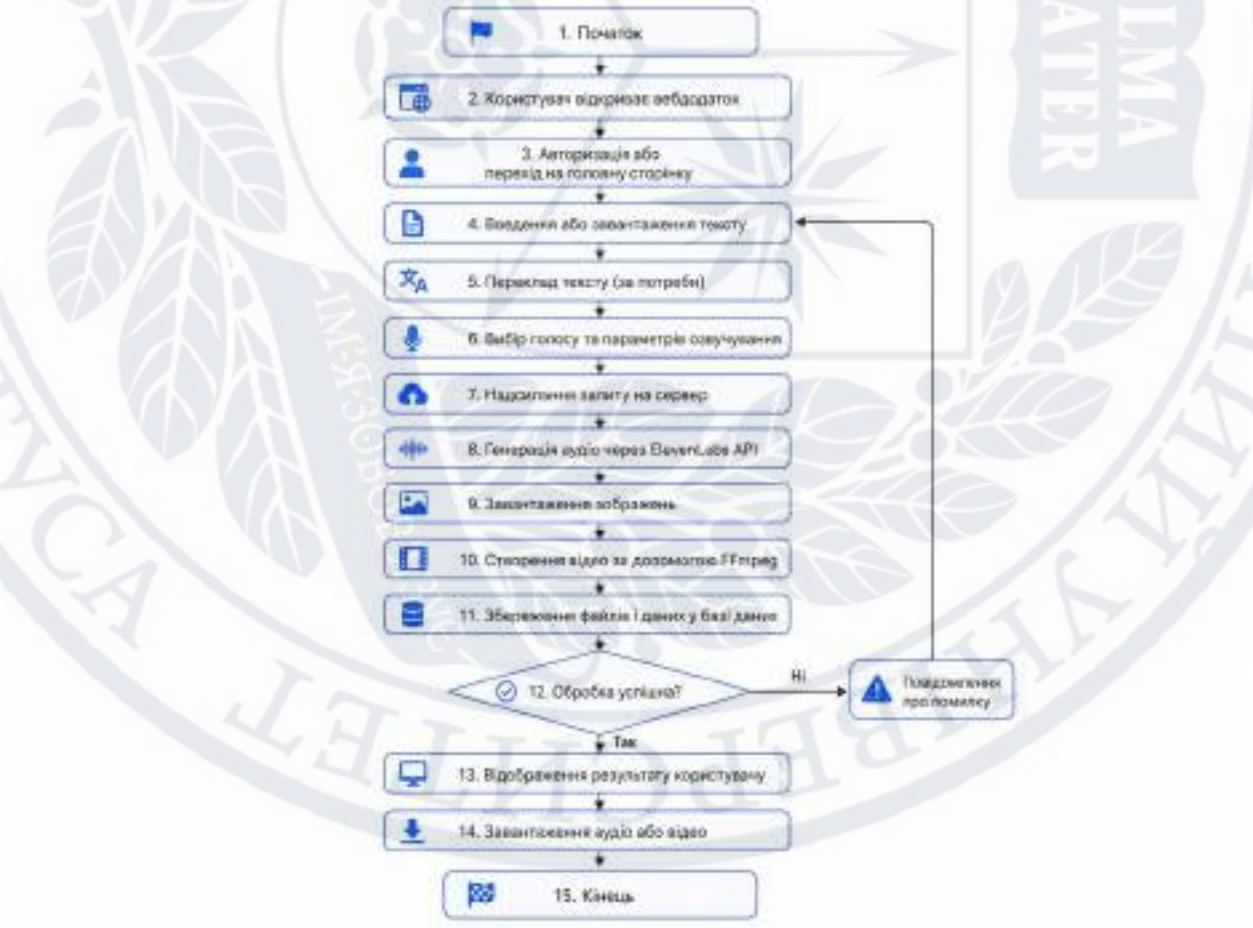

Рисунок 2.17 - Зведена схема взаємодії кінцевого споживача з вебсервісом VidVoice

На рисунку 2.17, має бути відображений увесь шлях користувача: від моменту відкриття програми до фінального завантаження отриманого аудіо чи відео. Цей графік дає змогу інтегрувати роботу клієнтського інтерфейсу, серверного механізму, репозиторію даних, послуги ElevenLabs API, інструменту FFmpeg та системи зберігання файлів.

Насамперед, споживач відкриває вебдодаток у своєму браузері. Після цього він або одразу залучається до основного функціоналу з головної панелі, або ж переходить до особистого розділу для ознайомлення з особистим профілем, своїми проєктами, залишками кредитів та налаштуваннями. Якщо ж ідеться про створення нової роботи, першим робочим кроком є внесення або імпорт текстових даних у секції "Написати та перекласти".

Після введення тексту користувач має можливість здійснити переклад, підібрати бажаний тембр голосу та визначити параметри озвучування. Далі інтерфейс клієнта відправляє запит на серверний шлях /api/generate-audio. Серверна частина приймає цей запит, взаємодіє з ElevenLabs API, отримує згенерований звуковий файл і зберігає його у призначеному сховищі. Дані про цей файл можуть бути прив'язані до профілю користувача, конкретного проєкту або сцени у базі даних [10, 17].

Після успішного формування аудіо відбувається перехід до етапу завантаження візуальних елементів. У блоці "Завантаження зображень" споживач додає файли, які слугуватимуть візуальним супроводом для майбутнього відео. Наступним кроком, у розділі "Складання відео", система перевіряє наявність як звукового файлу, так і доданих зображень. За умови, що всі необхідні компоненти присутні, клієнтська частина надсилає запит за маршрутом /api/compose-video.

На сервері починається процес обробки цих медіа-файлів. За допомогою утиліти FFmpeg відбувається зшивання аудіо та зображень у єдиний відеофайл формату MP4 [16]. Сформований файл розміщується у каталозі storage/videos, а відомості про цей результат можуть бути зафіксовані у таблицях media_file та

video_generation. По завершенні обробки сервер повертає клієнту посилання або повні дані про згенерований відеофайл.

З погляду споживача, весь цей процес виглядає як лінійне оперування єдиним веб-інтерфейсом. Користувач не здійснює безпосередньої роботи з ElevenLabs API, FFmpeg чи базою даних. Усі ці складові функціонують на бекенді, а споживач отримує лише зрозумілий вихідний продукт: або аудіо, або готове відео. Це значно спрощує користування сервісом, роблячи його доступним навіть для тих, хто не володіє навичками відеомонтажу чи програмування.

Основну послідовність операцій можна представити наступним чином:

1. відкриття вебдодатка користувачем;
2. перехід на головну сторінку або до особистого кабінету;
3. введення або завантаження текстової інформації;
4. вибір голосу та налаштувань озвучення;
5. ініціація створення аудіо;
6. сервер генерує аудіофайл через запит до ElevenLabs API;
7. користувач завантажує візуальні матеріали;
8. ініціація формування відео;
9. сервер створює відеофайл, використовуючи FFmpeg;
10. результат зберігається у сховищі файлів та у базі даних;
11. користувач переглядає або завантажує фіналізований файл.

У випадку виникнення будь-якої помилки, система зобов'язана проінформувати користувача про джерело проблеми. Наприклад, збій може статися через порожнє текстове поле, відсутність згенерованого аудіо, незавантажені зображення, недоступність зовнішнього сервісу або невдачу під час компіляції відео. Контроль за подібними ситуаціями є критично важливим для забезпечення стабільної роботи вебдодатка, оскільки це унеможливлює некоректне завершення процесів та надає споживачеві чіткий зворотний зв’язок [20, 22].

Отже, загальний сценарій взаємодії користувача з системою VidVoice покриває повний цикл створення мультимедійного продукту: від початкового

введення тексту до отримання фінального відеофайлу. Клієнтська частина забезпечує зручний візуальний інтерфейс, серверна частина обробляє запити, ElevenLabs API відповідає за синтез вербального контенту, FFmpeg - за компонування відео, а база даних та сховище файлів гарантують збереження всіх отриманих результатів.

РОЗДІЛ 3. РОЗГОРТАННЯ, ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ РОБОТИ ВЕБДОДАТКУ

3.1 Розгортання вебдодатку та підготовка середовища тестування

Для того аби перевірити, чи нормально програма функціонує у реальному середовищі, її було розміщено на сервері. Це було зроблено по завершенню основних етапів розробки. Це дало змогу користуватися вебдодатком реальним користувачам, тестувати його інтерфейс та отримати зворотний зв’язок для розробника. Крім того, користувачі мають змогу звертатися до вебдодатку за доменним іменем, протестувати роботу серверного та клієнтського модулів.

На першому етапі було додане доменне ім'я vidvoice.pp.ua, яке було зареєстровано через сервіс NIC.UA. Доменне ім’я ідентифікує додаток та робить його доступним в мережі Інтернет. Сервіс NIC.UA служить інструментом для пошуку та закріплення доменних адрес у різних доменних зонах [23].

Після конфігурування домену, було проведено верифікацію його DNS-записів. Для цього було використано сервіс whatsmydns.net, що дає змогу моніторити розповсюдження DNS-інформації з різних DNS-серверів по усьому світу. У рамках цього розгортання тестувався A-запис для vidvoice.pp.ua, який має бути налаштований на відповідну IP-адресу сервера [24]. Окрім того, для розуміння механізму мапування домену на IP локально, було проаналізовано довідковий матеріал від HostPro щодо файлу hosts, де пояснюється, що цей файл може слугувати для ручного прив'язування домену до сервера і має вищий пріоритет над стандартними DNS-запитами [25].

Загальну архітектуру виведення вебдодатку VidVoice на сервер проілюстровано на рис. 3.1.

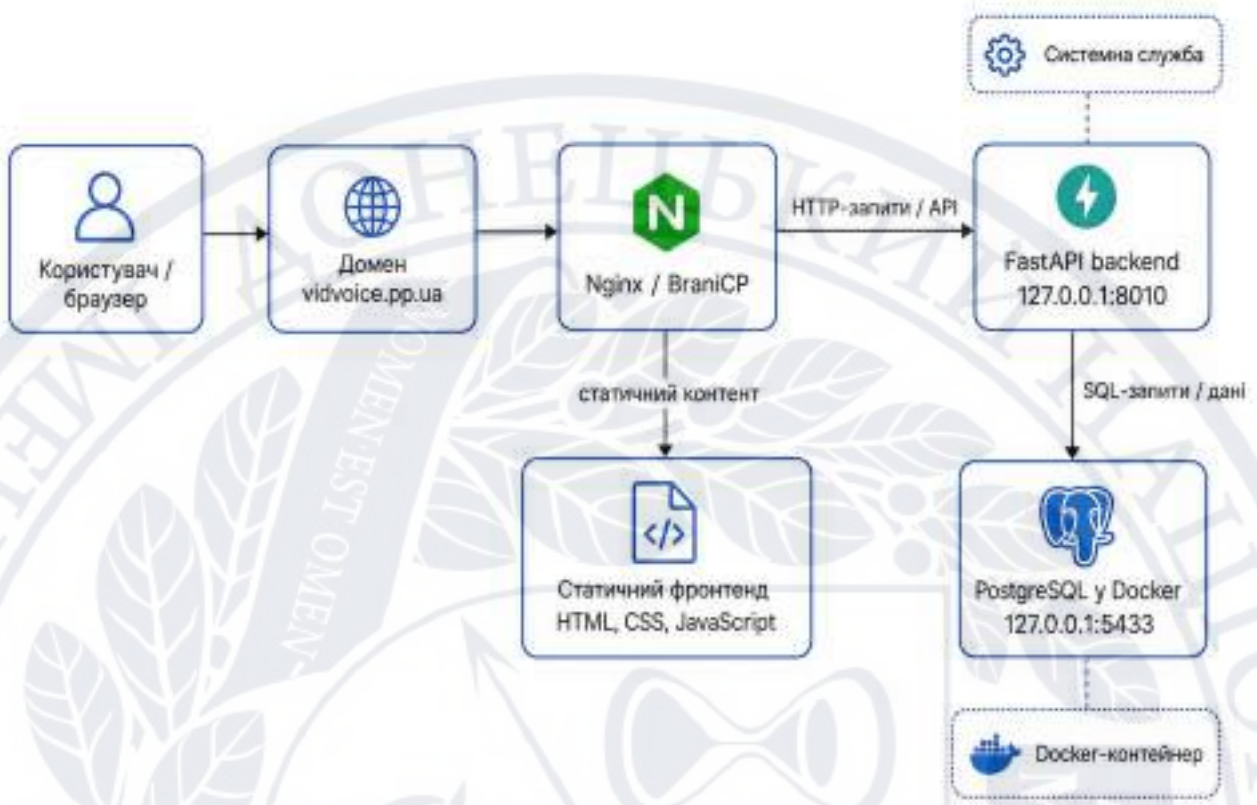

Рисунок 3.1 - Архітектурна будова розгортання вебзастосунку VidVoice на сервері

У ході деплойменту було використано наступну послідовність опрацювання запитів: користувач / Браузер → vidvoice.pp.ua → Nginx / BraniCP → FastAPI бекенд на 127.0.0.1:8010 → PostgreSQL у Docker на 127.0.0.1:5433

Завдяки такі послідовності можна побачити, що саме користувач є ініціатором доступу до вебдодатку за його адресою через власний браузер. Запит обробляється на сервері за допомогою Nginx, який в свою чергу було налаштовано за допомогою панелі BraniCP. Nginx виконує функцію реверс-проксі, перенаправляючи запити до логіки FastAPI, яка функціонує локально за адресою 127.0.0.1:8010. Серверний модуль, у свою чергу, взаємодіє з PostgreSQL, яка працює у контейнері Docker на порту 5433.

Клієнтська частина, яка знаходиться в папці Frontend, була розміщена на сервері як статичний контент. Це означає, що компоненти HTML, CSS та

JavaScript віддаються кінцевому користувачеві без потреби у специфічному фронтенд-сервері і постійному їх завантаженні. Клієнтська частина вимагає складного процесу кампіляції і може обслуговуватися через конфігурацію вебсервера чи самого FastAPI/Nginx. Тому саме такий підхід є зручний для цього проєкту.

Серверна логіка на FastAPI була запущена як незалежна системна служба. Завдяки цьому бекенд-модуль стартує автоматично після перезавантаження сервера і дає змогу контролювати його статус через стандартні інструменти управління Linux. База даних PostgreSQL була інкапсульована у Docker-контейнер і також функціонує як сервіс. Застосування Docker для БД спрощує налаштування оточення, ізолює PostgreSQL від основної системи та полегшує міграцію чи відновлення бази даних.

Така архітектура сприяє підвищенню стабільності роботи всього вебдодатку, тому що кожен компонент працює окремо, не залежно один від одного і може бути змінений чи перезапущений незалежно від інших компонентів. Це особливо важливо під час оновлення або масштабування модулів, коли треба провести перевірку.

Також використання розділення на клієнтську, серверну та контейнеризовану базу даних дозволяє чітко розмежувати відповідальність між рівнями системи. Такий підхід дуже спрощує подальшу підтримку проєкту, підвищує його безпеку та забезпечує можливість легкого розгортання на різних середовищах без суттєвих змін у конфігурації.

Додатковою перевагою такого підходу є можливість незалежного резервного копіювання та відновлення окремих компонентів системи. У разі виникнення помилки в роботі серверної частини або бази даних адміністратор може виконати перезапуск чи оновлення конкретного сервісу без зупинки всього вебдодатку. Це підвищує надійність системи та забезпечує безперервність її роботи під час експлуатації.

Стан активних сервісів на сервері зафіксовано на рис. 3.2.

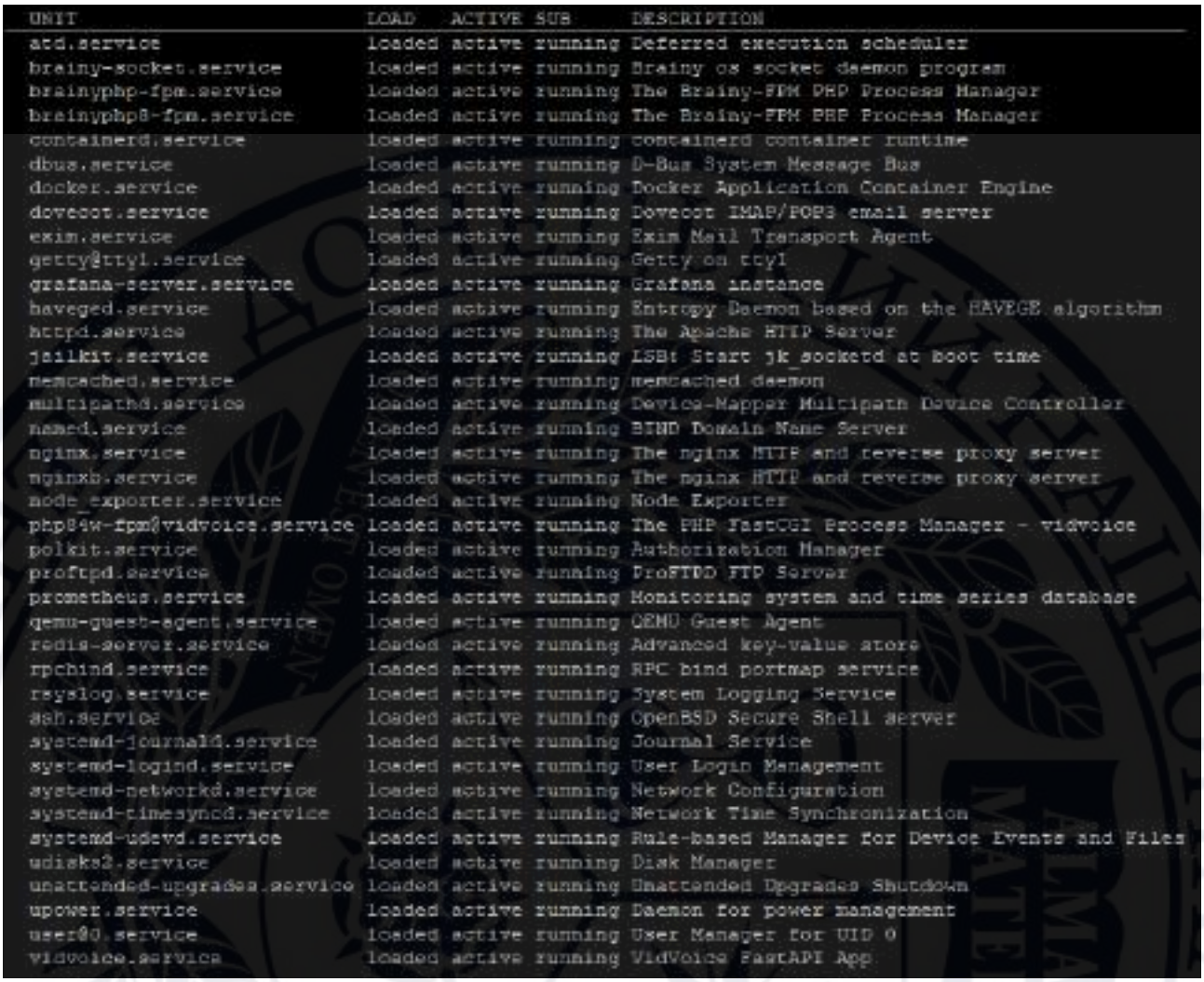

Рисунок 3.2 - Перелік активних сервісів сервера після впровадження вебдодатку

На рисунку 3.2 відображено список сервісів, які працюють на сервері. Серед них видно `docker.service`, відповідальний за функціонування контейнерів Docker, а також `vidvoice.service`, який ініціює запуск FastAPI-додатку. Наявність статусу `loaded active running` підтверджує, що відповідні служби завантажені, активні та виконують свою роботу в системі.

Для підготовки тестового середовища була проведена перевірка доступності усіх головних елементів системи: домену, Nginx, бекенду FastAPI, бази даних PostgreSQL у Docker та статичних файлів клієнтської частини. Далі проводилося тестування ключових функцій вебдодатку: доступ до головної сторінки, процедури реєстрації та авторизації користувача, управління профілем, створення нових проєктів, генерація аудіоконтенту, завантажувальні операції з зображеннями та фінальне формування відеоряду та інших функції вебдодатку.

Отже, вебдодаток VidVoice успішно встановлено на сервер та надано доступ за доменним іменем vidvoice.pp.ua. Для обробки HTTP-запитів залучено Nginx / BraniCP, серверна частина FastAPI функціонує як системний сервіс на локальному порту 8010, а PostgreSQL розміщена у Docker-контейнері на порту 5433. Таке налаштування вебдодатку дозволило перейти від локального тестування, до тестування в умовах реального середовища.

3.2 Методика тестування системи

Для розроблення методології тестування вебдодатку VidVoice було розглянуто ключові функціональні складові системи: клієнтський інтерфейс, бекенд-сервіс, сховище даних, модулі синтезу мовлення, завантаження графічних матеріалів, створення відеоряду та особистий кабінет користувача. Основне завдання тестування – переконатися що програма нормально функціонує та виконує свою роботу після її розгортання на сервері, а також у підтвердженні можливості користувача створювати відео чи аудіо матеріали. Перевірка здійснювалася у середовищі, налаштованому згідно з описом у підпункті 3.1. До вебдодатку можна було потрапити через домен vidvoice.pp.ua; фронтенд працював як статичні ресурси, FastAPI бекенд було запущено як системний сервіс на localhost:8010, а база даних PostgreSQL функціонувала у Docker-контейнері на localhost:5433. Такий підхід дав змогу оцінити взаємодію як окремих елементів, так і системи в цілому в умовах, максимально наближених до робочого серверного розгортання.

У процесі тестування застосовувався ручний метод функціонального тестування. Перевірки проводилися через інтерфейс користувача шляхом послідовного виконання першочергових дій у додатку і генерації аудіо та відео. Цей спосіб виявився найбільш доцільним, оскільки вимагалася оцінка не лише технічної відповідності API, а й зручності взаємодії користувача з інтерфейсом, достовірності кінцевого результату та зручності використання вебдодатку[20, 21].

Модуль компонування відео – один із найважливіших етапів під час тестування. На цьому етапі перевірялась відповідність аудіо до зображень, адже для коректного та правильного створення відео їх кількість має бути однаковою. Якщо одного із компонентів немає – користувач одразу побачить відповідне повідомлення. Це дозволяє користувачеві не витрачати час на ручну перевірку. Ілюстрація перевірки логіки формування відео представлена на рис. 3.3.

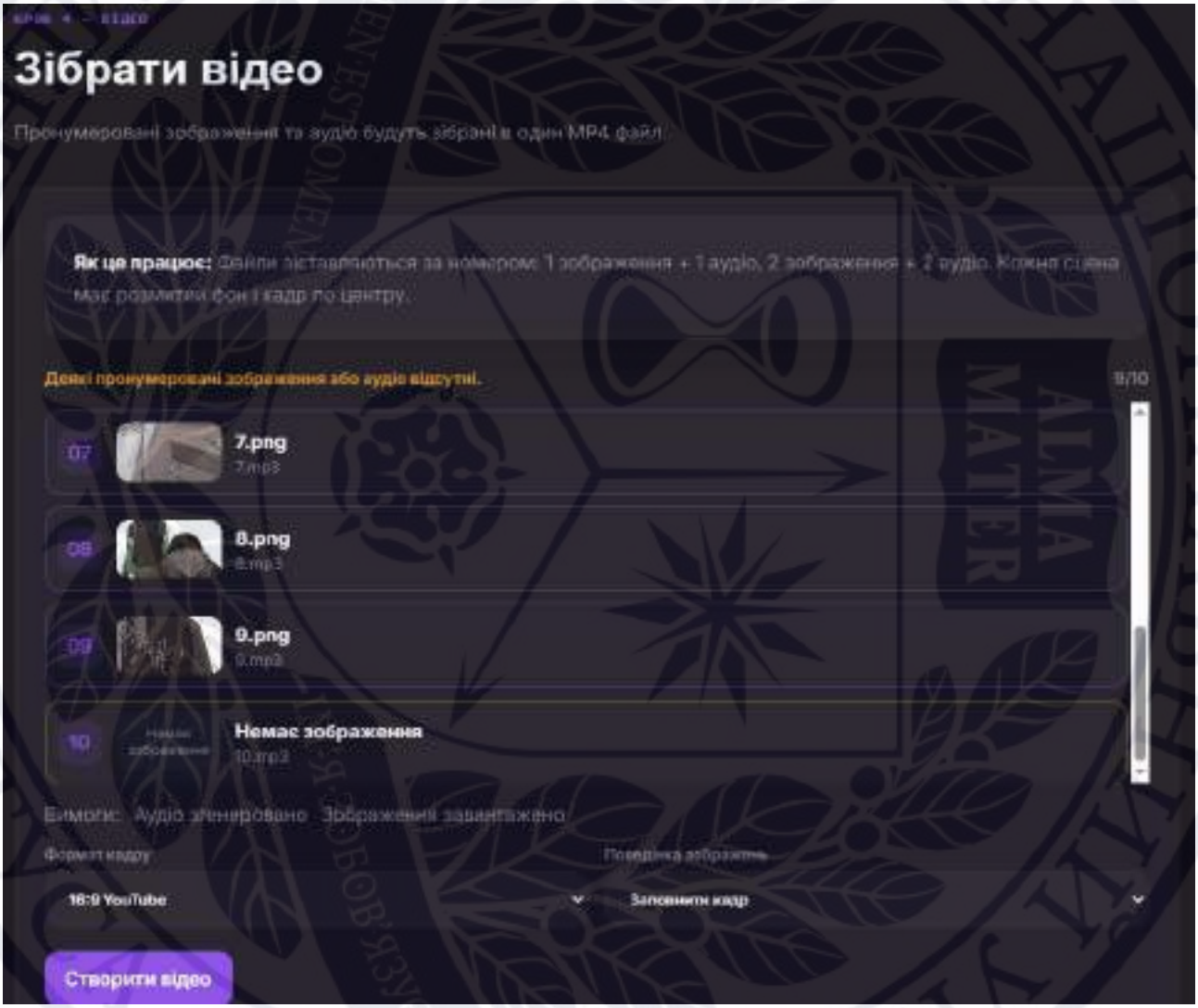

Рисунок 3.3 - Верифікація відеомодуля у вебдодатку VidVoice

На рисунку 3.3 демонструється інтерфейс стадії «Компонування відео», де показано пронумеровані зображення та відповідні їм звукові доріжки. Система також інформує користувача про відсутність певних пронумерованих зображень або аудіофайлів. Це є важливим аспектом перевірки, оскільки вебдодаток

повинен не лише забезпечувати генерацію, а й превентивно оповіщати користувача про неповний набір вхідних даних перед початком обробки.

Для проведення перевірки було визначено кілька ключових напрямків: доступність ресурсу (завантаження сайту, статичного контенту та навігації), коректність серверної логіки (API-запити, робота FastAPI та взаємодія з базою даних) і працездатність мультимедійних функцій (генерація аудіо, завантаження зображень, створення відео та збереження результатів).Основні площини тестування системи перелічено у табл. 3.1.

Таблиця 3.1 - Ключові напрямки тестування вебдодатку VidVoice

Напрям тестування	Що перевірялося
Доступність вебдодатку	Відкриття сайту через домен, робота Nginx та завантаження сторінок
Клієнтська частина	Відображення інтерфейсу, робота кнопок, форм, меню та навігації
Серверна частина	Робота FastAPI, API-запитів і системної служби
База даних	Підключення до PostgreSQL, збереження користувачів, проєктів і результатів
Озвучування тексту	Введення тексту, вибір голосу, генерація та прослуховування аудіо
Генерація відео	Завантаження зображень, перевірка сцен, запуск FFmpeg, створення MP4-файлу
Обробка помилок	Реакція системи на порожні поля, відсутні файли або некоректні дії

Як видно з таблиці 3.1, процес тестування охопив усі головні компоненти вебдодатку. Особливу увагу було приділено валідації зв’язку між клієнтською та серверною частинами, оскільки саме через API-запити передаються текстові дані, параметри голосу, графічні матеріали та команди на рендеринг відео.

Належна робота цих запитів є фундаментальною умовою для забезпечення стабільності функціонування системи [15].

Для кожної функції системи було артикульовано очікуваний результат. Наприклад, після введення тексту та активації кнопки генерації, система мала створити аудіофайл і відобразити його у відповідному розділі інтерфейсу. По завершенню завантаження зображень, вони мали бути прийняті сервером та відображені у списку сцени. При компонуванні відео система повинна була верифікувати наявність аудіо та візуальних елементів для кожної сцени, а після ініціалізації рендерингу - створити MP4-файл, зберегти його у файловому сховищі та надати користувачеві опцію перегляду або даунлоаду фінального продукту.

Окрема увага була спрямована на тестування функціоналу бази даних PostgreSQL. Під час верифікації контролювався процес збереження даних користувачів, їхніх профілів, проєктів, медіафайлів та статусів процесів генерації. Цей аспект є життєво необхідним, оскільки БД є фундаментом для коректної роботи особистого кабінету та збереження історії створеного контенту [17].

Крім того, було заплановано тестування сценаріїв виникнення помилок. До таких ситуацій відносяться спроби генерації аудіо без введеного тексту, запуск створення відео за відсутності необхідного аудіо або зображень, спроби завантаження файлів із забороненими розширеннями, а також потенційні збої при зверненні до зовнішніх сервісів. Перевірка цих граничних випадків дозволяє визначити стійкість системи та якість зворотного зв’язку, що надається кінцевому користувачеві [20, 22].

Таким чином, методологія тестування вебдодатку VidVoice була орієнтована на проведення всебічної перевірки функціональності після його розгортання. Вона включала валідацію доступності сайту, роботи інтерфейсу, API-взаємодій, функціоналу бази даних, генерації аудіо, створення відеоряду та обробки некоректних сценаріїв. Це забезпечило можливість оцінити

працездатність системи в умовах реального серверного середовища та закласти фундамент для подальшого, більш глибокого функціонального тестування.

3.3 Тестування функціональних можливостей

Перевірку функціоналу вебдодатку VidVoice було здійснено з метою підтвердження ключових послідовностей дій користувача після розміщення системи на сервері. Головний акцент було зроблено на тих можливостях, що є суттю програми: маніпуляції з текстом, отримання звукових файлів, імпорт графічних матеріалів, зведення відео, фіксація інформації в сховищі даних та демонстрація підсумків у клієнтському інтерфейсі.

На початковому етапі з'ясовувалася доступність вебдодатку за її доменним ім'ям vidvoice.pp.ua. Під час випробувань головна сторінка відкривалася у браузері, незмінні файли (HTML, CSS та JS) завантажувалися належним чином, а елементи навігації дозволяли переміщуватися між основними секціями сайту. Це свідчить про коректне налаштування домену, апаратного середовища Nginx / BraniCP та клієнтського контенту.

Далі було досліджено функціонування персональної зони користувача. Тестування охоплювало відкриття меню акаунту, перехід до сторінки профілю, огляд розділу "Мої проєкти", сторінок, пов'язаних із кредитами, та налаштувань. У ході цієї перевірки було з'ясовано, що відомості про користувача відображаються в інтерфейсі, а перелік проєктів формується на основі даних, отриманих із серверного компонента та сховища PostgreSQL.

Окремо було випробувано механізм створення та відображення проєктів. На сторінці "Мої проєкти" перевірялося, як відображається список доступних проєктів, їхні найменування, статуси, мова, кількість кадрів, дати ініціювання та оновлення. Також перевірялася працездатність кнопки для започаткування нового проєкту та функції пошуку серед існуючих. Наявність цих елементів підтверджує, що система здатна функціонувати не лише як разовий інструмент для генерації файлів, а як повноцінна програма з підтримкою збереження користувацьких робіт.

Наступним кроком стало випробування модуля озвучення тексту. Користувач вносив текст у призначене поле, обирав бажаний тембр голосу та ініціював процес створення звуку. Після надсилання запиту серверна частина передавала текст до сервісу ElevenLabs API, отримувала звуковий файл та повертала кінцевий результат клієнтській частині [10]. У підсумку тестування підтвердило, що згенерований звук відображається в інтерфейсі, доступний для прослуховування користувачем і може бути використаний на наступній стадії формування відео.

Також було досліджено опцію перекладу тексту. Для цього до поля вводу додавався текст українською чи англійською, після чого активувалася відповідна кнопка перекладу. Очікуваною реакцією системи було оновлення тексту у полі відповідно до обраного напрямку перекладу. Така перевірка необхідна, оскільки переклад є допоміжною функцією, що розширює можливості попередньої підготовки текстового матеріалу перед його озвученням.

Слідом за перевіркою аудіо було протестовано механізм імпорту зображень. Користувач додавав файли у підтримуваних форматах PNG, JPG чи WEBP. Під час тестування з'ясовували, чи приймає система файли, чи відображає їх у переліку сцен, чи коректно нумерує завантажені зображення та чи інформує про відсутні компоненти. Ця перевірка є значущою, бо зображення слугують візуальною основою для майбутнього відеофайлу.

Ключовим етапом функціонального дослідженння стала перевірка процесу створення відео. Для цього використовувалися або згенеровані раніше, або імпортовані звукові файли, а також набір графічних матеріалів. Після активації кнопки створення відео клієнтська частина надсилала запит до серверного маршруту, а серверний компонент запускав обробку за допомогою інструменту FFmpeg. У процесі перевірки з'ясовувалося, чи формується файл формату MP4, чи зберігається кінцевий результат у файловому сховищі та чи отримує користувач можливість переглянути або завантажити готове відео [16].

Підсумки функціонального тестування ключових можливостей вебдодатку викладено у табл. 3.2.

Таблиця 3.2 - Результати дослідження функціоналу вебдодатку VidVoice

№	Функція	Очікуваний результат	Результат виконання
1	Відкриття сайту через домен	Головна сторінка відкривається у браузері	Виконано
2	Завантаження статичного frontend	HTML, CSS та JavaScript відображаються коректно	Виконано
3	Перехід між розділами сторінки	Навігація працює без помилок	Виконано
4	Відкриття меню профілю	Відображаються пункти профілю, проєктів, кредитів і налаштувань	Виконано
5	Перегляд сторінки проєктів	Список проєктів відображається в інтерфейсі	Виконано
6	Введення тексту	Текст додається у поле введення	Виконано
7	Переклад тексту	Текст змінюється відповідно до вибраного напряму переклад	Виконано
8	Вибір голосу	Обраний голос використовується для генерації аудіо	Виконано
9	Генерація аудіо	Створюється аудіофайл і відображається в інтерфейсі	Виконано
10	Завантаження зображень	Файли додаються до списку сцен	Виконано
11	Перевірка неповних даних	Система повідомляє про відсутні аудіо або зображення	Виконано
12	Генерація відео	Створюється MP4-файл на основі аудіо та зображень	Виконано
13	Збереження результатів	Дані про файли та проєкти зберігаються в системі	Виконано

Як видно з таблиці 3.2, основні можливості вебдодатку були перевірені й демонструють роботу згідно з очікуваною логікою. Важливо, що система не лише

успішно виконує штатний сценарій, а й реагує на некоректні дії користувача, наприклад, на відсутність графічного чи звукового файлу перед ініціацією створення відео. Це сприяє підвищенню стійкості роботи програми та знижує вірогідність некоректного завершення процесу.

Під час випробувань також було досліджено взаємодію зі сховищем даних. Після створення користувача, проєкту чи фіксації результату генерації, відповідні дані записувалися у PostgreSQL. Це підтверджує коректне функціонування зв'язку між серверним компонентом FastAPI та базою даних, що працює у контейнері Docker [17].

Отже, функціональне дослідження показало, що вебдодаток VidVoice виконує першочергові завдання, визначені під час проектування. Користувач може отримати доступ до сайту через домен, користуватися особистим кабінетом, створювати проєкти, вносити текст, генерувати аудіо, завантажувати візуальні матеріали, формувати відеоряд та отримувати кінцевий результат. Отримані результати засвідчують працездатність системи після її розгортання на сервері.

3.4 Оцінка якості згенерованого аудіо та відео

Після верифікації ключових функціональних аспектів вебдодатку VidVoice було проведено апробування рівня якості згенерованого аудіо та відеоконтенту. Головна мета цього етапу полягала у з’ясуванні, чи досягнуто поставленого завдання системою: автоматичне озвучення введеного тексту, формування аудіофайлу, його інтеграція із завантаженими візуальними матеріалами та створення фінального відеофайлу.

Спочатку було атестовано якість аудіо, що постало в результаті генерації. Для цього до системи вводилися тестові текстові блоки українською та англійською мовами, після чого відбувалася генерація озвучення через програмний інтерфейс ElevenLabs API. Основними критеріями для цієї перевірки були розбірливість мовлення, відсутність суттєвих дефектів, точна відповідність

аудіо введенню та можливість подальшої експлуатації аудіофайлу у процесі створення відео [10].

Як наслідок проведеного тестування, з'ясувалося, що згенерований аудіоматеріал коректно відтворюється безпосередньо у браузері та придатний для використання як звуковий супровід відеоряду. Користувач має змогу попередньо прослухати отриманий аудіофайл перед фінальною генерацією відео, що є критично важливим кроком для контролю якості. Якщо озвучення не відповідає очікуванням, передбачена опція корекції тексту, вибору іншого голосу чи параметрів озвучення та повторного запуску генерації.

Додатково було встановлено, що інтеграція з ElevenLabs API забезпечує стабільну та передбачувану роботу сервісу навіть при різних типах вхідного тексту, включно зі змішаними мовами та довгими текстовими блоками. Також система демонструє достатню швидкодію генерації, що дозволяє використовувати її у режимі майже реального часу без суттєвих затримок для користувача. Це підтверджує придатність реалізованого рішення для практичного використання у складі вебзастосунку VidVoice.

Також було виконано тестування етапу формування фінального відео на основі згенерованого аудіо та завантажених користувачем візуальних матеріалів. У ході перевірки оцінювалася коректність синхронізації аудіодоріжки з відеорядом, стабільність відтворення готового ролика та узгодженість зміни сцен із темпом озвучення. Отримані результати показали, що система забезпечує коректне об'єднання медіакомпонентів без розсинхронізації, а також стабільну генерацію відеофайлу навіть при роботі з тривалими аудіофрагментами, що підтверджує надійність реалізованого механізму створення фінального контенту.

Апробація аудіо проводилася згідно з декількома критеріями, що відображені у табл. 3.3.

Таблиця 3.3 - Показники оцінки якості згенерованого аудіо

Критерій	Опис оцінювання	Результат
Зрозумілість мовлення	Перевірка, чи чітко відтворюється текст	Задовільно
Відповідність тексту	Порівняння озвучення з введеним текстом	Відповідає
Якість звуку	Перевірка на наявність шумів або критичних спотворень	Критичних помилок не виявлено
Відтворення в браузері	Перевірка можливості прослуховування аудіо в інтерфейсі	Виконується
Подальше використання	Перевірка можливості використання аудіо для відео	Виконується

Як демонструє таблиця 3.3, згенероване аудіо задовольняє базові вимоги до функціонування системи. Воно доступне для прослуховування користувачем, може зберігатися у системі і слугувати основою для формування відеофайлу. Рівень якості озвучення значною мірою детермінований довжиною текстового фрагменту, обраною голосовою моделлю та встановленими генераційними параметрами.

Наступний етап передбачав оцінку якості змонтованого відео. У вебзастосунку VidVoice відео формується шляхом об’єднання аудіодоріжки та візуальних матеріалів, завантажених користувачем. Тобто система не застосовує нейронну генерацію для створення візуального ряду, а автоматизує процес компонування аудіо та зображень у фінальний MP4-файл. Для цієї мети залучається інструмент FFmpeg, який забезпечує обробку мультимедіа та створення кінцевого відеопродукту [16].

Під час апробації відео перевірялася здатність відкритого файлу, коректність відображення кожного зображення, наявність звукової доріжки та можливість завантаження фінального результату. Також було встановлено відповідність формату відео очікуваному виводу. У ході тестування система формувала відео у контейнері MP4, що є зручним для перегляду як у браузері, так і в більшості типових відеоплеєрів.

Показники оцінки якості створеного відеоматеріалу представлено у табл. 3.4.

Таблиця 3.4 - Показники оцінки якості згенерованого відео

Критерій	Опис оцінювання	Результат
Формат файлу	Перевірка створення відео у форматі MP4	Виконується
Відображення зображень	Перевірка наявності завантажених зображень у відеоряді	Виконується
Наявність аудіо	Перевірка звукової доріжки у відеофайлі	Виконується
Відтворення відео	Перевірка відкриття відео після генерації	Виконується
Завантаження результату	Перевірка можливості збереження відеофайлу користувачем	Виконується

Як свідчить таблиця 3.4, змонтоване відео відповідає первинним функціональним вимогам. Воно містить аудіокомпонент, створений на базі тексту, та візуальну частину з використанням завантажених зображень. Можливість перегляду та завантаження фінального файлу підтверджує належне функціонування модуля генерації відео.

Окремої уваги заслуговує той факт, що якість фінального відео прямо залежить від якості візуальних матеріалів, наданих користувачем. Якщо зображення мають низьку роздільну здатність або некоректні співвідношення сторін, кінцевий відеофайл може виглядати менш привабливо. Тому в інтерфейсі передбачено механізми вибору співвідношення сторін кадру та поведінки

зображень, що спрощує адаптацію візуальних ресурсів до вимог майбутнього відео.

Також у процесі апробації було з’ясовано, чи запобігає система генерації відео за умов браку необхідних даних. Якщо користувач не додав аудіо чи не завантажив зображення, застосунок інформує про таку проблему. Цей запобіжний механізм є важливим для забезпечення стабільності роботи вебдодатку, оскільки унеможливлює некоректну обробку даних та збої в роботі FFmpeg [20, 22].

Підсумовуючи, оцінка якості згенерованого аудіо та відео продемонструвала, що вебзастосунок VidVoice успішно виконує свою основну функцію: конвертує текст у звук та створює відеофайл на основі аудіодоріжки та доданих зображень. Сформовані результати придатні для перегляду, прослуховування та збереження. Це засвідчує готовність системи до практичного застосування відповідно до початково визначених завдань.

ВИСНОВКИ

У кваліфікаційній роботі розв’язано актуальне науково-практичне завдання щодо проєктування, розробки, розгортання та тестування вебдодатку «VidVoice» для автоматичного озвучування тексту та генерації відео на основі графічних матеріалів користувача. Реалізація цього проєкту дозволила мінімізувати ручну працю під час створення повторюваного медіаконтенту та об’єднати розрізнені інструменти обробки мультимедіа в єдиному зручному середовищі «в одному вікні».

В рамках теоретичного дослідження було детально проаналізовано предметну область масового створення мультимедійного контенту, вивчено існуючі підходи до синтезу мовлення та обґрунтовано переваги використання методів нейронних мереж для отримання природного звучання голосу в сучасних вебсистемах. Здійснений порівняльний аналіз існуючих аналогів (ElevenLabs, Synthesia, Canva, Kapwing) виявив їхні функціональні обмеження та довів доцільність створення інтегрованого вебзастосунку, який автоматизує лінійний сценарій поєднання тексту, аудіо та зображень без зайвих кроків ручного монтажу.

На основі клієнт-серверної архітектури та реляційної моделі у програмному середовищі DBeaver було розроблено детальну структуру бази даних PostgreSQL, що складається з 23 взаємопов'язаних таблиць. Спроєктована база даних повністю забезпечує бізнес-логіку ведення проєктів, збереження метаданих медіафайлів, авторизацію користувачів, а також прозорий моніторинг використання лімітів списання кредитних одиниць при роботі із зовнішніми сервісами.

Програмна реалізація продукту дозволила успішно розділити інтерфейс користувача та внутрішню бізнес-логіку системи. Клієнтську частину розроблено за модульним принципом за допомогою технологій HTML, CSS та JavaScript, що забезпечило адаптивний, інтуїтивно зрозумілий та ергономічний інтерфейс користувача з функціоналом особистого кабінету. Серверну частину втілено мовою Python на базі високопродуктивного фреймворку FastAPI, що

дозволило побудувати стабільний програмний інтерфейс (API) для асинхронного обміну даними між компонентами.

В межах бекенд-модуля було успішно інтегровано зовнішній сервіс ElevenLabs API для швидкого автоматичного синтезу мовлення українською та англійською мовами, причому конфіденційні ключі доступу безпечно ізольовано на рівні сервера. Крім того, розроблено програмний модуль рендерингу відео на базі інструментарію FFmpeg із залученням утиліти ffprobe для автоматичного аналізу тривалості звукових доріжок, що забезпечило автоматизоване зшивання зображень та аудіо в єдиний відеоконтейнер формату MP4.

Деплоймент вебдодатку у реальне серверне середовище з прив'язкою до зареєстрованого доменного імені vidvoice.pp.ua дозволив перевірити функціональність системи в робочих умовах. Стабільність та безпеку інфраструктури було досягнуто завдяки конфігуруванню вебсервера Nginx як реверс-проксі, запуску FastAPI у вигляді системної служби Linux та контейнеризації СУБД PostgreSQL за допомогою Docker.

Проведене функціональне ручне тестування підтвердило повну відповідність розробленого продукту початковим вимогам та високу якість згенерованого аудіо- та відеоконтенту. Особливу важливість продемонстрували превентивні механізми обробки помилок, які блокують запуск обчислювальних процесів на сервері за умов неповного набору або некоректного формату вхідних даних.

Розроблений вебдодаток «VidVoice» є завершеним практичним інструментом, готовим до експлуатації викладачами, маркетологами та авторами контенту. Побудована модульна архітектура системи закладає міцний фундамент для її подальшого масштабування, зокрема для імплементації черг завдань (Celery/Redis), додавання нових ефектів переходу, автоматичної генерації субтитрів та розширення аналітичного інструментарію.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гуржій А. М., Гуревич Р. С., Коношевський Л. Л., Коношевський О. Л. Мультимедійні технології та засоби навчання: навчальний посібник. Вінниця: Нілан-ЛТД, 2017. 556 с.
2. Задерейко О. В. Мультимедійні системи: конспект лекцій. Одеса, 2023. URL: https://dspace.onua.edu.ua/bitstreams/704ee075-e677-4c94-b143-b031753178e0/download
3. Осадчук О. Р. Розробка системи і інтерфейсу синтезу мовлення українською мовою для сайтів // Мікросистеми, електроніка та акустика. Київ, 2022. Т. 27, № 1. URL: https://elc.kpi.ua/article/view/255961
4. Онисько О. О. Аудіо-модуль веб-сайту для інклюзивної освіти: дипломна робота бакалавра. Київ: КПІ ім. Ігоря Сікорського, 2021. 42 с. URL: https://ela.kpi.ua/bitstreams/0f2b32e6-c8b9-42b8-8acb-5122b4474ecb/download
5. Задерейко О. В., Оріхівська О. Г. Мультимедійні системи: конспект лекцій. Одеса, 2024. URL: https://dspace.onua.edu.ua/bitstreams/f16b26cc-ef9e-4b51-b592-18ee6c6ce5f7/download
6. Коваленко О. С., Добровська Л. М. Проектування інформаційних систем: загальні питання теорії проектування ІС. Конспект лекцій. Київ: КПІ ім. Ігоря Сікорського, 2020. 192 с. URL: https://ela.kpi.ua/items/12865b1b-feef-4fc3-962b-f8caab3a3d4f
7. Мосіюк О. О. WEB-технології. Частина 1. Верстка: навчально-методичний посібник. Житомир: Вид-во ЖДУ ім. Івана Франка, 2020. 56 с.
8. Баран С. В. Основи web-програмування: навчальний посібник. Кривий Ріг: Державний університет економіки і технологій, 2023. 316 с. URL: https://dspace.duet.edu.ua/items/6f97a6c5-0762-41bb-9f9f-8a3129d74ae9
9. Трофименко О. Г., Козін О. Б., Задерейко О. В., Плачінда О. Є. Веб-технології та веб-дизайн: навчальний посібник. Одеса: Фенікс, 2019. 284 с. URL: https://dspace.onua.edu.ua/items/317429a2-55da-426e-a10e-f3cc1343d8db

10. ElevenLabs. Text to Speech API. ElevenLabs. URL: https://elevenlabs.io/text-to-speech-api
11. Synthesia. AI Video Platform for Business. Synthesia. URL: https://www.synthesia.io/
12. Canva. AI Video Generator: Text to Video AI Tool. Canva. URL: https://www.canva.com/features/ai-video-generator/
13. Kapwing. AI Text to Video Generator. Kapwing. URL: https://www.kapwing.com/ai/text-to-video
14. Python Software Foundation. Python Documentation. URL: https://docs.python.org/
15. FastAPI. FastAPI Documentation. URL: https://fastapi.tiangolo.com/
16. FFmpeg Developers. FFmpeg Documentation. URL: https://ffmpeg.org/documentation.html
17. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/
18. Docker Inc. Docker Documentation. URL: https://docs.docker.com/
19. Kaplun V. A., Tsikhotskyi M. S., Lukichov V. V. Основи вебпрограмування. Теорія і практика: навчальний посібник. Вінниця: ВНТУ, 2023. 128 с. URL: https://pdf.lib.vntu.edu.ua/books/2023/Kaplun_2023_128.pdf
20. Єгорова О. В., Бичок В. П. Програмні засоби для тестування програмного забезпечення // Молодий вчений. Київ, 2019. № 11 (75). С. 680–684.
21. Чича В. В. Особливості тестування графічного користувацького інтерфейсу: дис. … канд. техн. наук: 05.13.06 / Тернопільський національний економічний ун-т. Тернопіль, 2012. 165 с.
22. Ремінний О. А. Створення фреймворку автоматизованого тестування графічних користувацьких інтерфейсів // Вісник Хмельницького національного університету. Технічні науки. Хмельницький, 2013. № 3. С. 235–240.
23. NIC.UA. Реєстрація доменних імен. URL: https://nic.ua/
24. WhatsMyDNS. DNS Propagation Checker. URL: https://www.whatsmydns.net/

25. HostPro. Файл hosts, або як направити домен на сервер вже зараз. URL: https://hostpro.ua/wiki/domains/hosts-file/

ДОДАТКИ

ДОДАТОК А

Рисунок А.1 - Приклад успішної публікації та монетизації згенерованого відеоконтенту на платформі YouTube

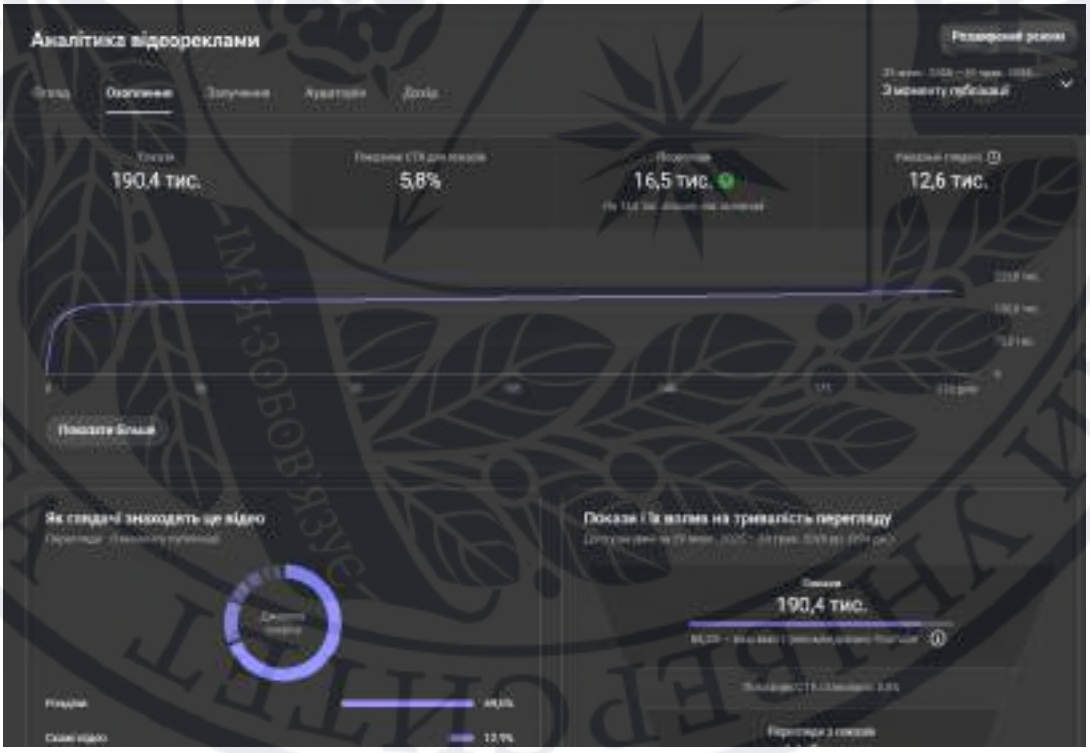

Рисунок А.2 - Статистика охоплення та джерел трафіку згенерованого відеоматеріалу в аналітиці YouTube Studio

ДОДАТОК Б

```
server {
    listen 194.1.182.239:80;
    server_name vidvoice.pp.ua www.vidvoice.pp.ua;
    root /home/vidvoice/apps;

    # Журналювання запитів та помилок сервера
    access_log /etc/nginx/vhost_logs/vidvoice.pp.ua_access;
    error_log /etc/nginx/vhost_logs/vidvoice.pp.ua_error;

    # Дозволити доступ для сертифікатів безпеки
    location ~ /.well-known {
        allow all;
    }

    # Заборона доступу до прихованих файлів конфігурації (.htaccess тощо)
    location ~ /\.ht {
        deny all;
        access_log off;
        log_not_found off;
    }

    # Основна локація: проксірування запитів на FastAPI бекенд
    location / {
        root /home/vidvoice/apps;
```

```
        proxy_pass http://127.0.0.1:8010;
        proxy_redirect off;
        proxy_force_ranges on;

        # Передача реальних IP-адрес користувача у FastAPI
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

        # Конфігурація кешування та обмеження з’єднань
        proxy_cache off;
        proxy_cache_key
"$request_method|$http_if_modified_since|$http_if_none_match|$host|$request_uri";
        proxy_cache_valid 3s;
        proxy_cache_min_uses 2;
        limit_conn lone 100;

        # Налаштування буферів для завантаження великих медіафайлів
        client_body_buffer_size 128k;
        client_max_body_size 1024m;

        # Таймаути для тривалих процесів генерації відео
        proxy_connect_timeout 180;
        proxy_send_timeout 180;
        proxy_read_timeout 180;
```

```
        send_timeout 180;
    }
}
```

Лістинг Б.1 - Конфігураційний файл вебсервера Nginx для забезпечення роботи реверс-проксі та маршрутизації запитів вебдодатку «VidVoice»