

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Додаток для створення аудіоісторій із синтезом мовлення»

Виконав студент групи ІПЗ-21-1
Ліневич Данііл Тимофійович

Керівник роботи:

Власенко Олег Васильович

Рецензент:

Вакалюк Тетяна Анатоліївна

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна Вакалюк

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: Ліневич Данііл Тимофійович

Керівник роботи: Власенко Олег Васильович

Тема роботи: «Додаток для створення аудіоісторій із синтезом мовлення»

затверджена Наказом закладу вищої освіти від «14» лютого 2025 р., №61/с

Вихідні дані для роботи:

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Власенко О.В.	14.02.2025	14.02.2025
2	Власенко О.В.	14.02.2025	14.02.2025
3	Власенко О.В.	14.02.2025	14.02.2025

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з додатку для створення аудіоісторій з синтезом мовлення та пояснювальної записки. Пояснювальна записка до випускної роботи містить 61 сторінки, 32 ілюстрацій та 25 таблиць

Метою роботи є проектування та розробка вебзастосунку для створення персоналізованих аудіоісторій із використанням синтезу мовлення.

Застосунок дозволяє створювати текстові історії, обирати персонажів, налаштовувати параметри голосу та генерувати відповідні аудіофайли. Реалізовано базу даних для зберігання інформації про користувачів, аудіоісторії та пов'язані об'єкти.

Результатом роботи є функціональний вебзастосунок, що забезпечує інтуїтивну взаємодію з користувачем та застосовує сучасні методи синтезу мовлення для створення аудіоконтенту.

Ключові слова: вебсервіс, синтез мовлення, TTS, аудіоісторії, Laravel, FastAPI, Livewire

					ІПЗ.КР.Б-121-25-ІІЗ		
Змн.	Арк.	№ докум.	Підпис	Дата	Додаток для створення аудіоісторій із синтезом мовлення		
Розроб.		Ліневич Д. Т.					
Перевір.		Власенко О. В.					
Керівник.		Власенко О. В.					
Н. Контр.		Тичина О. П.					
Зав.каф.		Вакалюк Т. А.					
					Літ.	Арк.	Аркушів
						2	26
					ФІКТ, гр. ІПЗ-21-1		

ABSTRACT

The bachelor's thesis consists of an application for creating audio stories with speech synthesis and an explanatory note. The explanatory note to the final thesis contains 61 pages, 32 illustrations, and 25 tables

The purpose of the thesis is to design and develop a web application for creating personalized audio stories using speech synthesis.

The application allows you to create text stories, select characters, configure voice settings, and generate corresponding audio files. A database has been implemented to store information about users, audio stories, and related objects.

The result of the work is a functional web application that provides intuitive user interaction and uses modern speech synthesis methods to create audio content.

Keywords: web application, speech synthesis, TTS, audio stories, Laravel, FastAPI, Livewire.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ СТВОРЕННЯ АУДІО-ІСТОРІЙ	9
1.1 Постановка задачі.....	9
1.2 Аналіз аналогів програмного продукту	12
1.3. Вибір інструментальних засобів та вимоги до апаратного забезпечення.....	17
Висновок до першого розділу	23
РОЗДІЛ 2. ПРОЄКТУВАННЯ ДОДАТКУ ДЛЯ СТВОРЕННЯ АУДІОІСОРІЙ ІЗ СИНТЕЗОМ МОВЛЕННЯ	25
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	25
2.2 Розробка бази даних системи.....	33
2.3 Проектування та реалізація алгоритмів роботи системи	36
2.4. Порядок встановлення та налаштування параметрів додатку	39
2.5. Реалізація функціоналу застосунку	41
Висновок до другого розділу	44
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ДОДАТКОМ ДЛЯ СТВОРЕННЯ АУДІОІСТОРІЙ ІЗ СИНТЕЗОМ МОВЛЕННЯ	45
3.1 Опис інтерфейсу додатку	45
3.2 Тестування платформи	54
Висновок до третього розділу	58
ВИСНОВКИ.....	59
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	60
ДОДАТКИ.....	61

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – прикладний програмний інтерфейс

БД – база даних

ПЗ – програмне забезпечення

MVC – схема розділення даних додатків і керуючої логіки на три окремі компоненти Model-View-Controller

CRUD – основні операції, що використовуються для управління даними (Create , Read , Update , Delete)

TTS – технології для озвучення тексту Text-To-Speech

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Сучасні технології штучного інтелекту та синтезу мови відкривають нові можливості для створення інтерактивних мультимедійних додатків. Одним із перспективних напрямків є розробка веб-додатків для генерації аудіо-історій, які можуть використовуватись у сферах освіти, розваг, маркетингу та доступності контенту для людей з обмеженими можливостями.

Актуальність теми полягає у зростаючому попиті на персоналізований аудіоконтент, який можна швидко створювати без необхідності професійного озвучення. Використання технологій із синтезом мовлення дозволяє автоматизувати процес генерації голосових повідомлень, аудіокниг, навчальних матеріалів та інших форматів аудіоісторій з високою якістю та природністю звучання.

Метою даної роботи є створення веб-додатку, який дозволяє генерувати аудіоісторії за допомогою сучасних технологій синтезу мовлення. Основна ідея полягає в тому, щоб надати користувачам простий та інтуїтивно зрозумілий інструмент для перетворення текстової інформації на якісний аудіоконтент. Для досягнення цієї мети передбачається здійснити глибокий аналіз існуючих TTS-рішень і вебдодатків, що спеціалізуються на генерації голосового контенту, з метою виявлення їхніх переваг, недоліків та потенційних напрямків удосконалення.

Особливу увагу планується приділити створення API для інтеграції TTS у веб-середовище, що дозволить забезпечити сумісність, масштабованість та якісне звучання згенерованих голосів. Крім того, важливо розробити зручний і функціональний інтерфейс, який дозволить користувачам легко вводити текст, обирати голоси, мову, інтонацію та інші параметри звучання відповідно до своїх потреб. Такий підхід забезпечить не лише технічну реалізацію, але й зручність використання розробленого додатку.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Об’єктом дослідження є процес створення аудіо-історій за допомогою технологій синтезу мовлення у веб-середовищі.

Предметом дослідження є функціональні можливості сучасних TTS-систем та способів їх впровадження у веб-додатки з метою автоматизації генерації аудіоконтенту. Дослідження також охоплює аналіз переваг технологій синтезу мовлення порівняно з традиційними методами озвучення, такі як швидкість створення контенту, масштабованість та адаптація під різні мови й голоси.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ВЕБ-ДОДАТКУ ДЛЯ СТВОРЕННЯ АУДІО-ІСТОРІЙ

1.1 Постановка задачі

Метою даної кваліфікаційної роботи є розробка сучасного веб-додатку для генерації аудіо-історій із синтезом мовлення. Сьогодні існує значний попит на автоматизовані інструменти створення аудіо-контенту, оскільки вони дозволяють швидко перетворювати текст у природну мову без необхідності професійного озвучення. Це відкриває нові можливості для освітніх платформ, маркетингових агенцій, розробників мультимедійних продуктів та користувачів, які потребують доступного контенту.

Першим етапом дослідження є аналіз існуючих TTS-рішень та веб-додатків, що пропонують схожий функціонал. На ринку доступні такі популярні TTS API, як Google Cloud Text-to-Speech, Amazon Polly, OpenAI TTS, IBM Watson Text to Speech та інші. Кожен з них має свої переваги, такі як якість генерації, підтримка різних мов і голосів, вартість використання та інтеграційні можливості. Важливо провести порівняльний аналіз цих технологій, щоб визначити оптимальний варіант для реалізації проекту.

Окрім аналізу технологій синтезу мови, необхідно дослідити вже існуючі веб-додатки для створення аудіо-історій. Це дозволить виявити їх сильні та слабкі сторони, зрозуміти, які функції найбільш затребувані серед користувачів, і визначити, які аспекти можна покращити в рамках даного проекту. Наприклад, деякі додатки пропонують лише базовий функціонал перетворення тексту в мову, тоді як інші надають додаткові інструменти для редагування аудіо, збереження історій або інтеграції з хмарними сховищами.

Також важливим етапом є вибір технологічного стеку для розробки. Враховуючи, що додаток повинен бути швидким, зручним і масштабованим, необхідно підібрати сучасні інструменти для фронтенду та бекенду. Для клієнтської частини можна розглядати такі фреймворки, як React або Livewire на

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

основі Laravel, які забезпечують гнучкість та інтерактивність інтерфейсу. Для серверної частини підходять FastAPI або Django, оскільки вони дозволяють ефективно обробляти запити та інтегруватися з TTS.

Крім того, важливим аспектом є проектування бази даних, де будуть зберігатися аудіофайли, історії користувачів та їхні налаштування. Можна використовувати як традиційні реляційні бази даних (PostgreSQL, MySQL), так і NoSQL-рішення (MongoDB), залежно від вимог до структури даних та швидкості їх обробки.

Розробка веб-додатку для створення аудіо-історій має проходити через низку важливих етапів, кожен з яких спрямований на забезпечення функціональності, зручності користування та стабільної роботи системи.

Першим кроком є створення дизайну веб-додатку. На цьому етапі формулюється загальна концепція інтерфейсу, яка враховує як естетичний вигляд, так і зручність користування. Дизайн розробляється відповідно до сучасних принципів UX/UI. Особлива увага приділяється формуванню UI-компонентів: інтерактивних елементів для введення тексту, вибору мовних параметрів (голос, мова), а також елементів керування відтворенням аудіо. Важливим аспектом також є розробка основних сторінок додатку — головної, редактора аудіо, профілю користувача та бібліотеки, де зберігатимуться створені історії.

Далі відбувається побудова архітектури клієнтської частини. Розробляється структура компонентів, визначаються маршрути, формується логіка управління станом додатку. Проводиться оптимізація коду для забезпечення швидкого завантаження контенту та ефективної роботи з аудіофайлами в реальному часі.

На наступному етапі проектується серверна частина додатку. Визначаються кінцеві точки API, що відповідатимуть за обробку тексту, взаємодію з моделями синтезу мовлення, збереження аудіофайлів, а також управління даними

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

користувачів. Паралельно впроваджуються засоби безпечної передачі даних між клієнтом і сервером.

Окремим етапом є розробка системи авторизації та користувацьких профілів. Імплементується механізм реєстрації, входу та автентифікації користувачів. Визначаються ролі користувачів, зокрема звичайний користувач та авторизований. Створюється інтерфейс користувацького профілю, де зберігатимуться персональні налаштування та історії, створені за допомогою додатку.

Ключовим функціональним блоком є інтеграція моделей синтезу мовлення. Вибрана TTS-технологія підключається до системи, налаштовуються її параметри для забезпечення варіативності голосів і стилів озвучування. Реалізується можливість обрати мову, голос, а також функції попереднього прослуховування та редагування згенерованих історій перед збереженням.

Далі впроваджується система збереження та експорту аудіофайлів. Для надійного зберігання застосовується хмарне сховище, наприклад, MinIO з підтримкою S3-сумісності. Передбачено можливість експорту файлів у популярних форматах — MP3, WAV, FLAC.

На завершальному етапі проводиться повноцінне тестування додатку. Перевіряється коректність роботи кожного модуля, стабільність взаємодії з TTS API під час пікових навантажень, сумісність інтерфейсу з різними пристроями та браузерами. У процесі тестування виявляються та усуваються помилки, а також виконується оптимізація продуктивності системи для забезпечення швидкої та безперебійної роботи.

Таким чином, перший етап роботи передбачає комплексний аналіз технологій, існуючих аналогів та архітектурних рішень, що стане основою для подальшої розробки ефективного та зручного веб-додатку для створення аудіо-історій.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

1.2 Аналіз аналогів програмного продукту

Розробка веб-додатку для створення аудіо-історій із синтезом мовлення технологій попереднього аналізу існуючих рішень на ринку. Оскільки синтез мови активно розвивається, розглянемо популярні платформи, які пропонують схожий функціонал.

1. Платформа Speechify

Speechify — це інструмент синтезу мовлення, орієнтований на максимально простий та зручний користувацький досвід (рис. 1.1). Він створений для масового використання — студентами, людьми з дислексією, професіоналами, які слухають книги або документи «на ходу». Його головна сила — доступність, швидкість і простота.

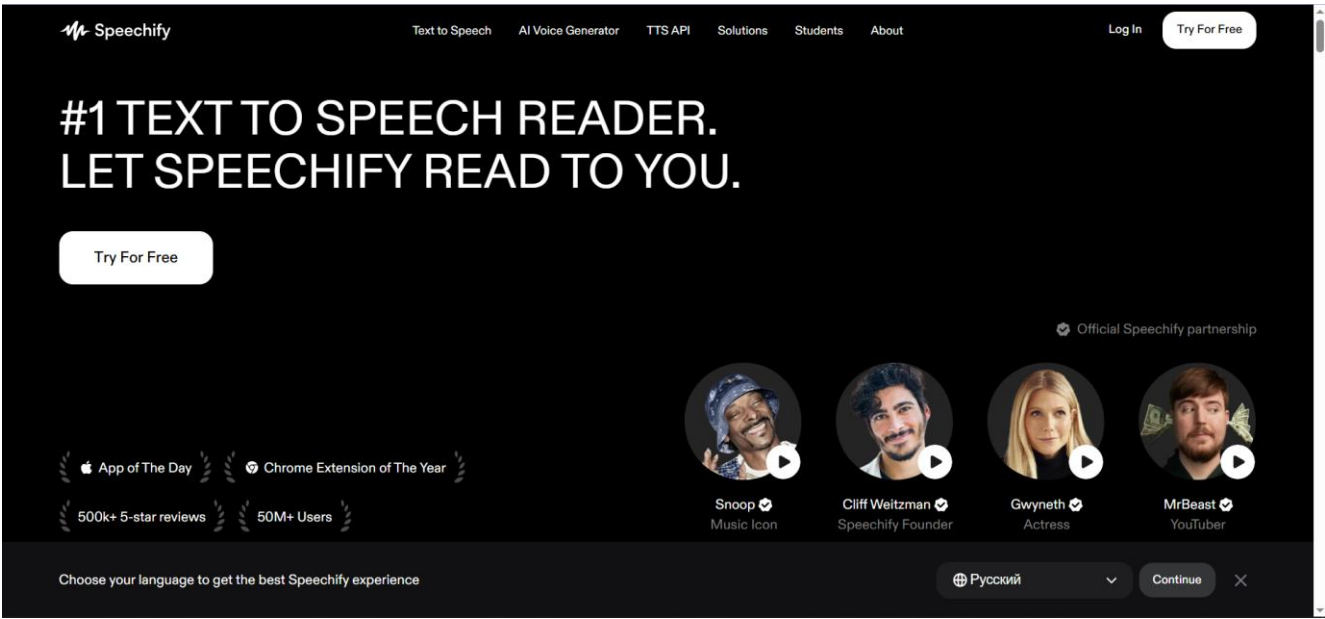


Рис. 1.1. Веб-додаток Speechify

Додаток Speechify дає змогу перетворювати текст на аудіо за допомогою штучного інтелекту. Ви можете вставляти текст вручну або завантажити документи у форматах PDF, ePub, Word, TXT або навіть зчитати вебсторінки. Програма автоматично перетворює їх на мовлення, при цьому дозволяє обирати з кількох десятків голосів. У платній версії додаються преміум-голоси, які звучать реалістичніше — деякі навіть імітують відомих людей або мають інтонації, що наближаються до природної мови. Доступна також регуляція швидкості мовлення

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

— від повільного до пришвидшеного (до 3х). У мобільному додатку можна завантажити озвучені файли й слухати офлайн. Текст, який озвучується, виділяється на екрані, що корисно для людей з порушеннями читання. Speechify доступний як вебзастосунок, а також як мобільні додатки для iOS та Android. Також є розширення для браузера Chrome, яке дає змогу озвучувати текст прямо зі сторінок інтернету.

Speechify вирізняється дуже простою навігацією та інтуїтивним інтерфейсом, з яким легко розібратися. Перетворення тексту в мовлення відбувається миттєво, навіть для довгих текстів, затримка мінімальна. Важливою перевагою є синхронізація між пристроями, завдяки якій тексти і прогрес зберігаються в хмарі. Функція підсвічування тексту робить Speechify ефективним інструментом для вивчення мов та тренування читання. Також є підтримка української мови, хоча вибір голосів поки що обмежений і звучання досить штучне.

Серед недоліків варто відзначити обмеження безкоштовного плану. Базові голоси не дуже виразні, а преміум-можливості досить дорогі. Speechify не пропонує глибокої кастомізації: не можна вставляти паузи, емоції чи змінювати інтонацію вручну. Також відсутня підтримка фонові музики та звукових ефектів, що обмежує можливості створення повноцінних аудіоісторій. Хоча українська мова підтримується, вибір голосів мінімальний, а їх звучання залишає бажати кращого.

Speechify пропонує безкоштовний план з базовими можливостями. Преміум-підписка коштує від 11 доларів на місяць і надає доступ до якісніших голосів, офлайн-функцій та швидшої обробки.

Speechify чудово підійде тим, кому потрібен простий та швидкий інструмент для конвертації тексту в мовлення без складного редагування. Він добре підходить для навчання, прослуховування книг, нотаток та статей, тобто в тих випадках, коли важлива швидкість, а не художня виразність.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

2. Платформа Murf.ai

Murf.ai — це один із найпотужніших і найфункціональніших TTS-сервісів на ринку (рис. 1.2). Його використовують маркетологи, відеопродакшн-команди, викладачі, подкастери та навіть компанії, які створюють озвучення для анімації або корпоративного відео. Він надає повний набір інструментів для кастомізації озвучення — практично як звукова студія, тільки на базі штучного інтелекту.

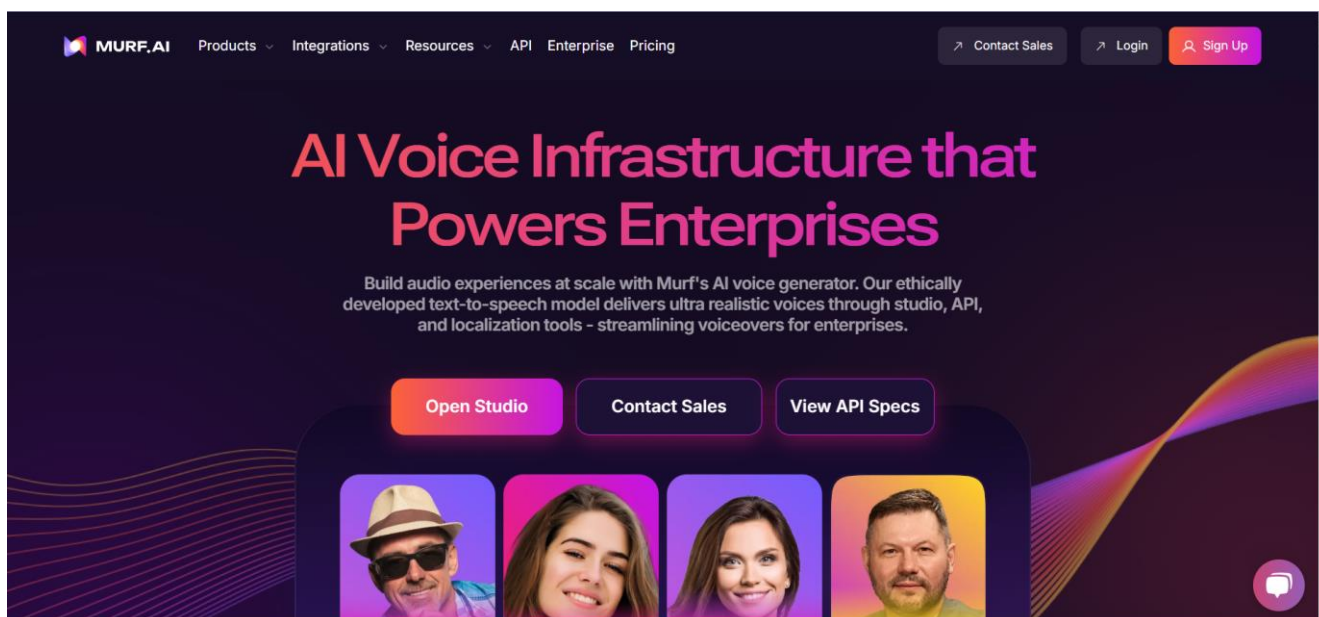


Рис. 1.2. Веб-додаток Murf.ai

Додаток Murf.ai пропонує понад 120 голосів у 20+ мовах, включаючи українську. Ви можете регулювати швидкість, інтонацію, гучність, ставити паузи, додавати наголоси, змінювати вимову окремих слів. Підтримується багатоголосове озвучення — кожен частину тексту можна озвучити іншим голосом. Для ще більшого ефекту ви можете додати фонову музику прямо у вбудованому редакторі. У платній версії відкриваються преміум-голоси з позначкою “Near Human” — це голоси нового покоління, що практично не відрізняються від живих людей. Murf також дозволяє співпрацювати над проєктами в команді, що робить його зручним для креативних агентств. Murf працює як вебзастосунок. Мобільної версії наразі немає, але інтерфейс адаптований для планшетів і ноутбуків.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Сервіс надає повний контроль над озвученням: ви можете самостійно визначати інтонацію, емоції, темп та паузи. Доступна також можливість додавання фонові музики безпосередньо в проєкті, що дозволяє міксувати голос та музику. Murf.ai підтримує багатоголосе озвучення, що дає змогу створювати сценки та діалоги. Сервіс пропонує експорт в різних форматах, включаючи MP3, WAV та FLAC. Також доступні функції для командної роботи, такі як спільний доступ, ролі та контроль версій.

Серед недоліків Murf.ai варто відзначити високу ціну. Стартовий тариф починається від 29 доларів на місяць, а повний доступ ще дорожчий. Безкоштовний план має суттєві обмеження: лише 10 хвилин озвучення та обмежений набір голосів.

3. Платформа Play.ht

Play.ht — платформа, яка зосереджується на багатомовності та публікації аудіо у форматі подкастів (рис. 1.3). Вона дозволяє швидко генерувати мовлення з тексту, зберігати, поширювати й навіть автоматично транслювати через RSS-канали.

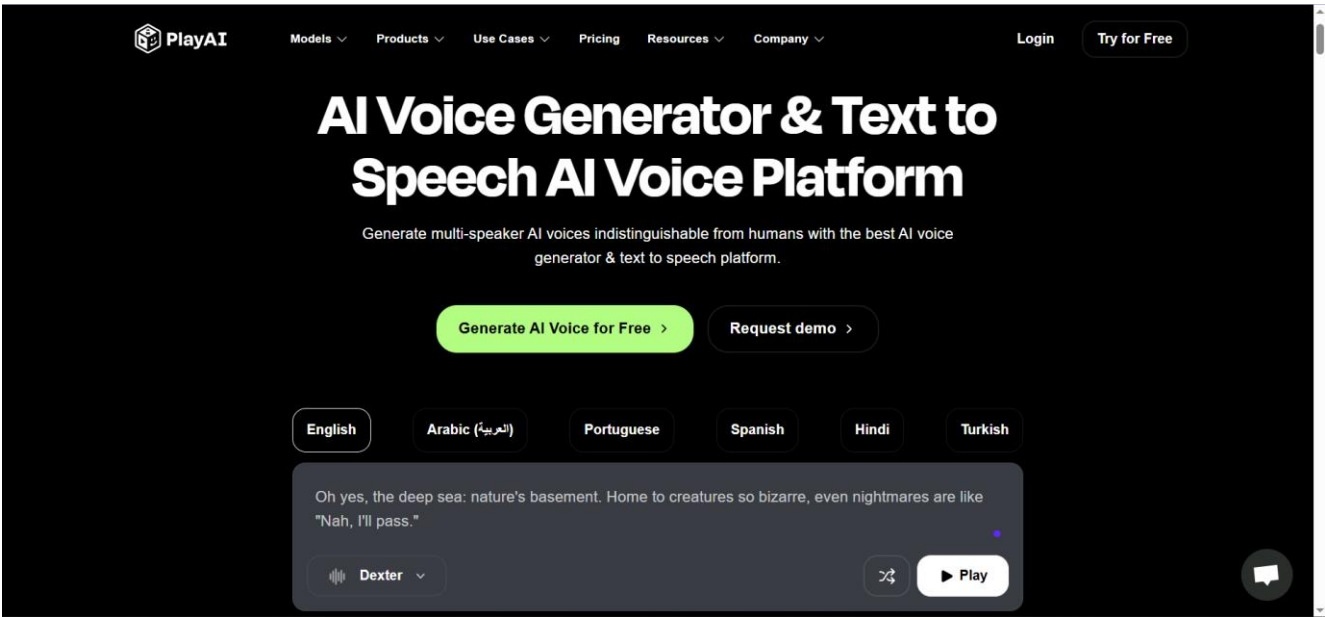


Рис. 1.3. Веб-додаток Play.ht

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

У Play.ht представлено понад 800 голосів у 130+ мовах. Ви можете редагувати інтонацію, вставляти паузи, ставити наголоси. Інтерфейс простий: вставляєш текст, обираєш мову та голос, при потребі редагуєш, натискаєш «генерувати». Унікальна можливість — створення подкаст-стрічки: кожен згенерований файл можна одразу публікувати як епізод, або вивантажувати на інші платформи. Крім того, розробники мають доступ до API для інтеграції TTS у власні застосунки. Повністю браузерний сервіс. Підтримує експорт у соцмережі, поштові розсилки та подкаст-сервіси. Мобільних застосунків немає.

Однією з головних переваг сервісу є найбільша підтримка мов у порівнянні з Murf чи Speechify. Він також пропонує зручну інтеграцію з подкаст-сервісами, що дозволяє одразу публікувати створений контент. Експорт готового матеріалу дуже простий та зручний для поширення. Сервіс дозволяє редагувати інтонації, що робить звучання більш живим та природним. Також доступний API для інтеграції з власними сайтами та системами.

Серед недоліків варто зазначити дуже обмежену безкоштовну версію, яка дозволяє генерувати текст обсягом лише до 2500 символів. Якість голосів не завжди однакова, деякі звучать досить штучно. Сервіс не підтримує додавання фонові музики, тому для цього доведеться використовувати сторонні редактори.

Основні характеристики порівняльного аналізу аналогічних платформ, що включають технології синтезу мовлення наведено в таблиці 1.1.

Таблиця 1.1

Таблиця порівняння аналогів

Критерій	Speechify	Murf.ai	Play.ph	Розробляємий додаток
Якість голосів	Висока	Дуже висока	Середня	Висока
Підтримка мов	Обмежена	20+ мов	130+ мов	10 мов (з можливістю розширення)

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

Критерій	Speechify	Murf.ai	Play.ph	Розробляємий додаток
Формати експорту	MP3	MP3, WAV	MP3, RSS	MP3,WAV,FLAC
Фонові музика	Немає	Так	Немає	Так
Редагування аудіо	Немає	Так	Так	Так
Зручний інтерфейс	Так	Так	Немає	Так

Проведений аналіз показав, що сучасні платформи із синтезом мовлення мають як сильні, так і слабкі сторони. Найкращі рішення пропонують високу якість звучання, але вимагають технічної підготовки для інтеграції. Інші сервіси більш доступні для звичайних користувачів, але мають обмеження у функціоналі.

1.3. Вибір інструментальних засобів та вимоги до апаратного забезпечення

Вибір інструментальних засобів для створення аудіо-історій із синтезом мовлення є критичним етапом, що визначає якість фінального продукту, зручність роботи з матеріалом, а також можливості масштабування проєкту в майбутньому. У цьому пункті розглядаються ключові критерії відбору програмного забезпечення, що базуються на технічних вимогах, функціональних можливостях сервісів, підтримці української мови, гнучкості редагування аудіо та рівні користувацького досвіду.

Додаток для створення аудіо-історій із синтезом мовлення потребує ретельно підбраної архітектури та інструментарію для забезпечення стабільної та ефективної роботи. В якості основного фреймворку для бекенду обрано Laravel, що обумовлено його зручністю, швидкістю розробки та можливістю масштабування. Laravel надає вбудовані функції для автентифікації, маршрутизації та роботи з базою даних, що значно прискорює процес розробки.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

Порівняння фреймворків PHP з їхніми аналогами наведено в таблиці 1.2.

Таблиця 1.2

Таблиця порівняння фреймворків PHP

Критерій	Laravel	Symfony	Yii
Складність освоєння	Низька (орієнтований на початківців)	Середня/висока (потрібно знання паттернів)	Низька/середня (простий для старту)
Швидкість розробки	Дуже висока (Artisan, шаблони)	Середня (потрібна конфігурація)	Висока (Gii-генератор)
Продуктивність	Середня (оптимізований для розробки)	Висока (гібкі налаштування)	Дуже висока (оптимізований для навантажень)
ORM	Eloquent (простий та потужний)	Doctrine (гнучкий, але складніший)	ActiveRecord (простий, схожий на Laravel)
API-розробка	Добре підтримується (Lumen, Sanctum)	Відмінна (API Platform, REST)	Гарна (вбудовані інструменти)
Масштабованість	Середня (підходить для середніх проєктів)	Висока (ідеальний для великих систем)	Висока (оптимізований для високих навантажень)

Додаток також потребує зручного, гнучкого та інтерактивного інтерфейсу, що дозволяє користувачам легко взаємодіяти з функціоналом системи. В якості основного фреймворку для побудови інтерфейсу обрано Livewire, який забезпечує тісну інтеграцію з Laravel та дозволяє створювати динамічні компоненти без необхідності писати складний JavaScript-код. Завдяки цьому розробка значно спрощується та прискорюється, особливо у випадках, коли логіка компонента тісно пов'язана з бекендом. Livewire чудово підходить для малих і середніх проєктів, де важливо забезпечити реактивну поведінку інтерфейсу при мінімальних витратах на конфігурацію та підтримку.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Порівняння фронтенд фреймворків для Laravel з їхніми аналогами наведено в таблиці 1.3.

Таблиця 1.3

Таблиця порівняння фронтенд фреймворків для Laravel

Критерій	Livewire	React	Vue
Складність освоєння	Низька (знайомий Laravel-розробникам)	Середня/висока (JS/JSX, екосистема)	Середня (простий старт, поступове ускладнення)
Інтеграція з Laravel	Глибока, повна інтеграція	Потребує API, додаткове налаштування	Потребує API або Inertia.js
Продуктивність	Середня (через запити до сервера)	Висока (на клієнті, без оновлення сторінки)	Висока (реактивна система, легкість рендерингу)
Реактивність	Обробляється на сервері (через AJAX)	Повна клієнтська реактивність	Вбудована реактивність
Зручність розробки	Дуже висока для Laravel-проектів	Висока, але потребує конфігурації та Webpack/Vite	Висока, особливо з CLI
Підтримка SPA	Обмежена (неповноцінна SPA, можна поєднувати з Volt)	Повна (Next.js, React Router)	Повна (Vue Router, Vuex/Pinia)
Масштабованість	Середня (добре підходить для малих і середніх проєктів)	Висока (використовується у великих системах)	Висока (підтримує великі SPA)

Для генерації аудіо через технології синтезу мовлення використовується мікросервіс на базі FastAPI, який забезпечує високу продуктивність завдяки асинхронній обробці запитів. FastAPI також надає автоматичну документацію через Swagger/OpenAPI, що спрощує процес інтеграції з іншими компонентами системи.

Порівняння фреймворків Python з їхніми аналогами наведено в таблиці 1.4

Таблиця 1.4

Таблиця порівняння Python фреймворків

Фреймворк	Переваги	Недоліки
Django	Повноцінний MVC-фреймворк з адмін-панеллю	Важчий за інші Python-фреймворки
	Вбудована ORM (робота з БД без SQL)	Менш гнучкий для мікросервісів
	Автоматична адміністративна панель	Вимагає більше ресурсів
	Висока безпека (CSRF, XSS захист)	
	Велика спільнота та документація	
FastAPI	Найвища продуктивність серед Python-фреймворків	Молодий фреймворк (менше готових рішень)
	Автоматична документація API (Swagger/OpenAPI)	Немає повноцінної адмін-панелі
	Асинхронна підтримка	Вимагає знання Pydantic для схем даних
	Ідеальний для мікросервісів та API	
Flask	Найлегший для старту	Мінімалістичний (багато треба додавати самостійно)
	Максимальна гнучкість	Немає вбудованої ORM
	Ідеальний для невеликих проєктів	Менш безпечний за Django (треба додавати захист)
	Велика кількість розширень	
	Мінімальні вимоги до ресурсів	

В якості основного сховища аудіо-даних обрано MongoDB - NoSQL базу даних, яка забезпечує гнучкість у роботі з різними типами даних та дозволяє ефективно зберігати метаінформацію про аудіо-файли. MongoDB добре масштабується та підтримує швидке виконання складних запитів завдяки індексації.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

В якості основного сховища даних обрано PostgreSQL - реляційну базу даних, яка забезпечує роботу з різними типами даних та дозволяє ефективно зберігати інформацію про користувачів, діалоги, тощо.

Порівняння баз даних з їхніми аналогами наведено в таблиці 1.5

Таблиця 1.5

Таблиця порівняння баз даних

БД	Тип	Переваги	Недоліки
PostgreSQL	Реляційна (SQL)	Повноцінна SQL-сумісність	Вимагає більше ресурсів
		Розширені типи даних (JSON, геодані)	Складніше масштабування
		Потужна система індексів	
		Висока надійність ACID	
MySQL	Реляційна (SQL)	Висока продуктивність	Обмежена підтримка NoSQL
		Проста масштабованість реплікацією	Менш гнучка схема даних
		Широке хмарне підтримка	
MongoDB	Документоорієнтована	Гнучка схема (JSON-документи)	Немає JOIN-операцій
		Горизонтальне масштабування	Менш надійні транзакції
		Висока швидкість записів	
Redis	Ключ-значення	Найвища швидкість (оперативна пам'ять)	Обмежений обсяг даних
		Підтримка різних структур даних	Немає повноцінної мови запитів
		Публікація/підписка	

Для зберігання самих аудіофайлів використовується MinIO - об'єктне сховище, сумісне з S3 API. Воно дозволяє організувати надійне та масштабоване зберігання даних, підтримує реплікацію та може бути легко інтегроване з хмарними сервісами. MinIO особливо добре підходить для роботи з великими об'ємами аудіоданих.

Порівняння об'єктних сховищ з їхніми аналогами наведено в таблиці 1.6

Таблиця 1.6

Таблиця порівняння об'єктних сховищ

Сховище	Тип архітектури	Переваги	Недоліки	Оптимальні сценарії використання
MinIO	Об'єктне сховище	Висока продуктивність (до 1.32 Тб/сек)	Обмежена підтримка керування даними	Хмарні додатки з високими вимогами до IOPS
		Повна S3-сумісність	Відсутність вбудованого CDN	AI/ML робочих навантажень
		Проста установка та керування	Менша зрілість у корпоративному середовищі	Kubernetes storage
		Оптимізований для Kubernetes		
Amazon S3	Об'єктне сховище	Найбільша екосистема інструментів	Vendor lock-in	Корпоративні сховища даних
		Глобальна мережа доставки контенту (CloudFront)	Висока вартість при великих обсягах	Веб-сайти з медіаконтентом
		99.999999999% надійність		Резервне копіювання
		Розширені функції керування даними		

Сховище	Тип архітектури	Переваги	Недоліки	Оптимальні сценарії використання
Ceph	Об'єктне/блокове/файлове	Висока доступність (без єдиної точки відмови)	Складність налаштування	Великі приватні хмари
		Лінійне масштабування	Вимагає значних апаратних ресурсів	Наукові обчислення
		Відкрите ПЗ (Apache 2.0)		

Щодо апаратних вимог, серверна інфраструктура повинна забезпечувати достатні ресурси для стабільної роботи всіх компонентів системи. Для бекенду на Laravel рекомендовано використовувати сервер з 4 vCPU та 8 GB RAM. Мікросервіс генерації аудіо вимагає більш потужних ресурсів - оптимально 8 vCPU, 16 GB RAM, а для досягнення найкращої продуктивності бажано використовувати GPU. тоді як для MongoDB рекомендовано 8 vCPU, 16 GB RAM та SSD диски для швидкого доступу до даних. MinIO як сховище даних потребує 4 vCPU, 8 GB RAM та SSD диски великого об'єму.

Обрана архітектура та інструментарій дозволяють створити масштабовану, надійну та продуктивну систему для генерації аудіоісторій. Використання сучасних технологій та фреймворків забезпечує гнучкість розробки, можливість подальшого розширення функціоналу та легку інтеграцію з новими TTS-сервісами в майбутньому. Такий підхід дозволяє забезпечити високу якість послуг для кінцевих користувачів при оптимальних витратах на інфраструктуру.

Висновок до першого розділу

У першому розділі було визначено основну мету проєкту — створення веб-додатку для генерації аудіо-історій за допомогою сучасних TTS-технологій. Також

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

розглянуто, чому ця тема є актуальною сьогодні, зокрема для освітнього контенту, подкастів та медіа. Було проаналізовано популярні сервіси, як-от Play.ht, Speechify і Murf.ai. З одного боку, вони мають зручний інтерфейс і хорошу якість озвучення, а з іншого — їм часто бракує гнучкості, є обмеження у використанні, або складно інтегрувати в інші системи. Це дало розуміння, що саме потрібно реалізувати у власному рішенні. Окрім цього, визначено технології, на яких буде базуватись проєкт. Для клієнтської частини обрано Laravel з Livewire, для серверної — FastAPI, а для зберігання аудіо — MinIO. Такий вибір дозволяє створити зручний, швидкий у роботі та легко масштабований додаток.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

РОЗДІЛ 2. ПРОЄКТУВАННЯ ДОДАТКУ ДЛЯ СТВОРЕННЯ АУДІОІСТОРІЙ ІЗ СИНТЕЗОМ МОВЛЕННЯ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Додаток для створення аудіоісторій із синтезом мовлення надає користувачам зручний веб-інтерфейс, через який вони можуть генерувати озвучений контент, редагувати сценарії. Функціональність системи базується на активній участі користувачів, які в основному належать до однієї категорії: автори історій, які є також авторизованими користувачами, що створюють та редагують текстові сценарії для подальшого озвучення. Автори можуть створювати текстові історії безпосередньо у редакторі додатку. Після генерації, історії можуть бути збережені та експортовані у форматі аудіофайлу. Для зручності взаємодії між користувачами та системними компонентами було створено діаграму варіантів використання, яка демонструє ключові сценарії взаємодії з додатком (рис. 2.1).



Рис 2.1 Діаграма варіантів використання

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

Опис діаграми варіантів використання наведено в таблицях 2.1 – 2.9

Таблиця 2.1

Опис прецеденту «Реєстрація / Авторизація»

Назва варіанту використання	Реєстрація / Авторизація
Основні актори	Користувач
Короткий опис	Користувач може зареєструватися в додатку або авторизуватися, якщо вже має обліковий запис.

Таблиця 2.2

Опис прецеденту «Зміна мови додатку»

Назва варіанту використання	Вибір мови для додатку
Основні актори	Користувач, Авторизований користувач
Короткий опис	Користувач може обрати мову, якою буде налаштований інтерфейс додатку

Таблиця 2.3

Опис прецеденту «Генерація аудіофайлу з тексту»

Назва варіанту використання	Генерація аудіофайлу з тексту
Основні актори	Користувач, Авторизований користувач
Короткий опис	Користувач може ввести текст, який буде перетворено на аудіофайл з використанням синтезу мовлення.

Таблиця 2.4

Опис прецеденту «Створення аудіоісторії»

Назва варіанту використання	Створення аудіоісторії
Основні актори	Авторизований користувач

Короткий опис	Авторизований користувач може об'єднувати згенеровані аудіофайли та інші елементи для створення аудіоісторії.
---------------	---

Таблиця 2.5

Опис прецеденту «Вибір мови та персонажів озвучення»

Назва варіанту використання	Вибір мови та персонажів озвучення
Основні актори	Авторизований користувач
Короткий опис	Авторизований користувач може обирати різні мови та відповідно різних персонажів для синтезу мовлення.

Таблиця 2.6

Опис прецеденту «Перегляд персонажів озвучення»

Назва варіанту використання	Вибір моделі синтезу мовлення
Основні актори	Користувач, Авторизований користувач
Короткий опис	Авторизований користувач може переглядати всіх наявних персонажів та інформацію про них

Таблиця 2.7

Опис прецеденту «Експорт згенерованих аудіофайлів»

Назва варіанту використання	Експорт згенерованих аудіофайлів
Основні актори	Авторизований користувач
Короткий опис	Авторизований користувач може експортувати створені аудіофайли в потрібному форматі.

Таблиця 2.8

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

Опис прецеденту «Зміна теми додатку»

Назва варіанту використання	Опис прецеденту «Зміна теми додатку»
Основні актори	Користувач, Авторизований користувач
Короткий опис	Користувач може обирати вигляд інтерфейсу додатку, темний або світлий

Таблиця 2.9

Опис прецеденту «Оновлення та видалення згенерованих аудіо»

Назва варіанту використання	Перегляд звітності про порушення
Основні актори	Авторизований користувач
Короткий опис	Користувач може видаляти, або оновлювати репліки, щоб отримати бажаний результат

Основна ціль розробки додатку для створення аудіоісторій із синтезом мовлення полягає у створенні інтерактивного онлайн-середовища, яке дозволяє користувачам легко генерувати якісний аудіоконтент на основі текстових сценаріїв із використанням сучасних технологій штучного інтелекту. Цей додаток спрямований на забезпечення зручного інтерфейсу для написання, редагування, озвучення та публікації аудіоісторій різного типу — від художніх оповідей до освітніх або розважальних матеріалів.

Вимоги користувачів

Внутрішній користувач — зареєстрований користувач:

- Створення та редагування текстових сценаріїв.
- Генерація аудіо на основі тексту.
- Вибір мов, голосів, параметрів синтезу.
- Збереження та експорт готових аудіофайлів (MP3, WAV).
- Вибір мови додатку, та інтерфейс.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Внутрішній користувач — гість:

- Авторизація / реєстрація в системі.
- Перегляд наявних персонажів.
- Вибір мови додатку, та інтерфейс.

Функціональні вимоги

Розроблюваний вебдодаток має забезпечувати повноцінний цикл створення аудіоісторій за допомогою технологій синтезу мовлення. Основною функцією системи є перетворення текстових сценаріїв у готові аудіофайли з можливістю їх редагування, налаштування параметрів голосу, додавання фонових звуків, а також подальшого експорту.

Ключовий функціонал синтезу мовлення має включати вибір моделі (нейромережі), мови, персонажа озвучення, швидкості, тощо. Важливою опцією є підтримка кількох персонажів в одній історії.

Обов'язковою є наявність редактора сценаріїв — інструмента, в якому можна формувати структуру історії, діалоги, змінювати репліки, призначати голоси різним героям.

Після створення, користувач може експортувати аудіофайл у форматі MP3 WAV або FLAC.

Нефункціональні вимоги

З погляду користувача, взаємодія з додатком має бути максимально простою та інтуїтивно зрозумілою. Новий користувач повинен опанувати основні функції за 10–15 хвилин. Усі дії в інтерфейсі мають супроводжуватися швидкою реакцією системи — не більше 3 секунд для звичайних кліків і не більше 90 секунд для складної генерації великих історій. Система повинна бути стабільною та доступною 24/7 без значних простоїв. Оновлення варто

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

реалізувати так, щоб вони не порушували доступ до основного функціоналу або відбувалися в фоновому режимі. З технічного боку, вебзастосунок має витримувати навантаження великої кількості одночасних користувачів без втрати продуктивності. Це критично важливо у випадках масового запуску кампаній або активного використання генерації.

Системні вимоги

Для користування додатком достатньо стандартного обладнання. Система має бути адаптована для роботи на пристроях з мінімальною конфігурацією: процесором не менше 1.4 ГГц, 2 ГБ оперативної пам'яті та стабільним підключенням до Інтернету. Таким чином, враховуючи всі вищезазначені вимоги, система здатна забезпечити гнучкий, безпечний, сучасний інструмент для креативної генерації аудіоконтенту на базі синтезу мовлення.

Аналіз функціональних вимог дозволив виділити наступні сутності, що забезпечать програмну реалізацію додатку. На рис. 2.2. наведено діаграму класів, яка використовується для його роботи.

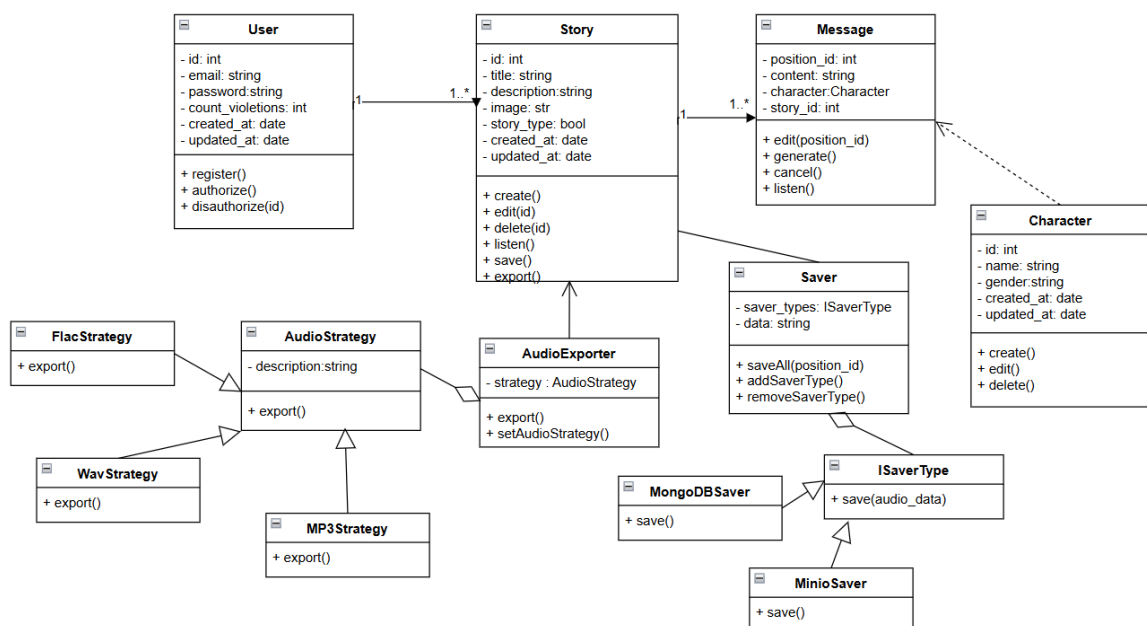


Рис. 2.2. Діаграма класів

Опис діаграми класів представлено в таблицях 2.10 – 2.18.

Таблиця 2.10

Клас User

Метод	Опис
register()	Реєструє нового користувача в системі
authorize()	Авторизує користувача
disauthorize(id)	Деавторизує користувача за ідентифікатором

Таблиця 2.11

Клас Story

Метод	Опис
create()	Створює нову історію
edit(id)	Редагує історію за ідентифікатором
delete(id)	Видаляє історію
listen()	Відтворює аудіоісторію
save()	Зберігає історію
export()	Експортує аудіоісторію

Таблиця 2.12

Клас Audio

Метод	Опис
edit(position_id)	Редагує фрагмент за позицією
generate()	Генерує аудіо на основі тексту
cancel()	Скасовує генерацію
listen()	Відтворює аудіофрагмент

Таблиця 2.13

Клас AudioExporter

Метод	Опис
export()	Експортує історію з використанням обраної стратегії
setAudioStrategy(strategy)	Встановлює нову стратегію експорту

Таблиця 2.14

Абстрактний клас AudioStrategy

Метод	Опис
export()	Абстрактний метод експорту

Таблиця 2.15

Класи FlacStrategy, WavStrategy, Mp3Strategy

Метод	Опис
export()	Реалізує експорт у відповідний формат

Таблиця 2.16

Клас Saver

Метод	Опис
saveAll()	Викликає save у всіх зберезжуваних об'єктах
addSaverType()	Додає новий об'єкт збереження
removeSaverType()	Видаляє об'єкт збереження

Таблиця 2.17

Абстрактний клас ISaverType

Метод	Опис
save(audio_data)	Зберігає дані в конкретному місці

Таблиця 2.18

Класи MongoDBSaver, MinioSaver

Метод	Опис
save(audio_data)	Зберігає у MongoDB або Minio

2.2 Розробка бази даних системи

Для збереження користувацьких даних, аудіоісторій та технічної інформації у додатку використовуються дві бази даних: PostgreSQL та MongoDB. Такий підхід дозволяє ефективно розділити структуровані й неструктуровані типи даних, оптимізуючи швидкодію та масштабованість системи.

PostgreSQL — це реляційна система управління базами даних, яка використовується для збереження строго структурованої інформації, зокрема:

- інформацію про користувачів,
- аудіоісторії користувачів
- персонажі,
- доступні мови.
- моделі синтезу мовлення.

Описи таблиць бази даних PostgreSQL наведено в таблицях 2.18-2.22

Таблиця 2.18

Таблиця Users

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
email	varchar	Адреса електронної пошти
password	varchar	Хеш паролю
firstName	varchar	Ім'я користувача
lastLoginAt	date	Дата останнього входу

Таблиця 2.19

Таблиця Stories

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор

Поле	Тип	Призначення
userId	int	Ідентифікатор автора
name	varchar	Назва історії
path	varchar	Причина скарги
createdAt	date	Дата створення
updatedAt	enum	Дата оновлення

Таблиця 2.20

Таблиця Languages

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
name	varchar	Ідентифікатор автора
icon	varchar	Зображення прапора мови

Таблиця 2.21

Таблиця Characters

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
name	varchar	Ім'я персонажа
gender	varchar	Стать персонажа
quality	varchar	Якість озвучення
language_id	int	Ідентифікатор мови
model_id	int	Ідентифікатор моделі синтезу мовлення
description	text	Короткий опис персонажу

Таблиця Models

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
name	varchar	Назва моделі
slug	varchar	Скорочена назва моделі

Діаграму бази даних PostgreSQL зображено на рисунку 2.3

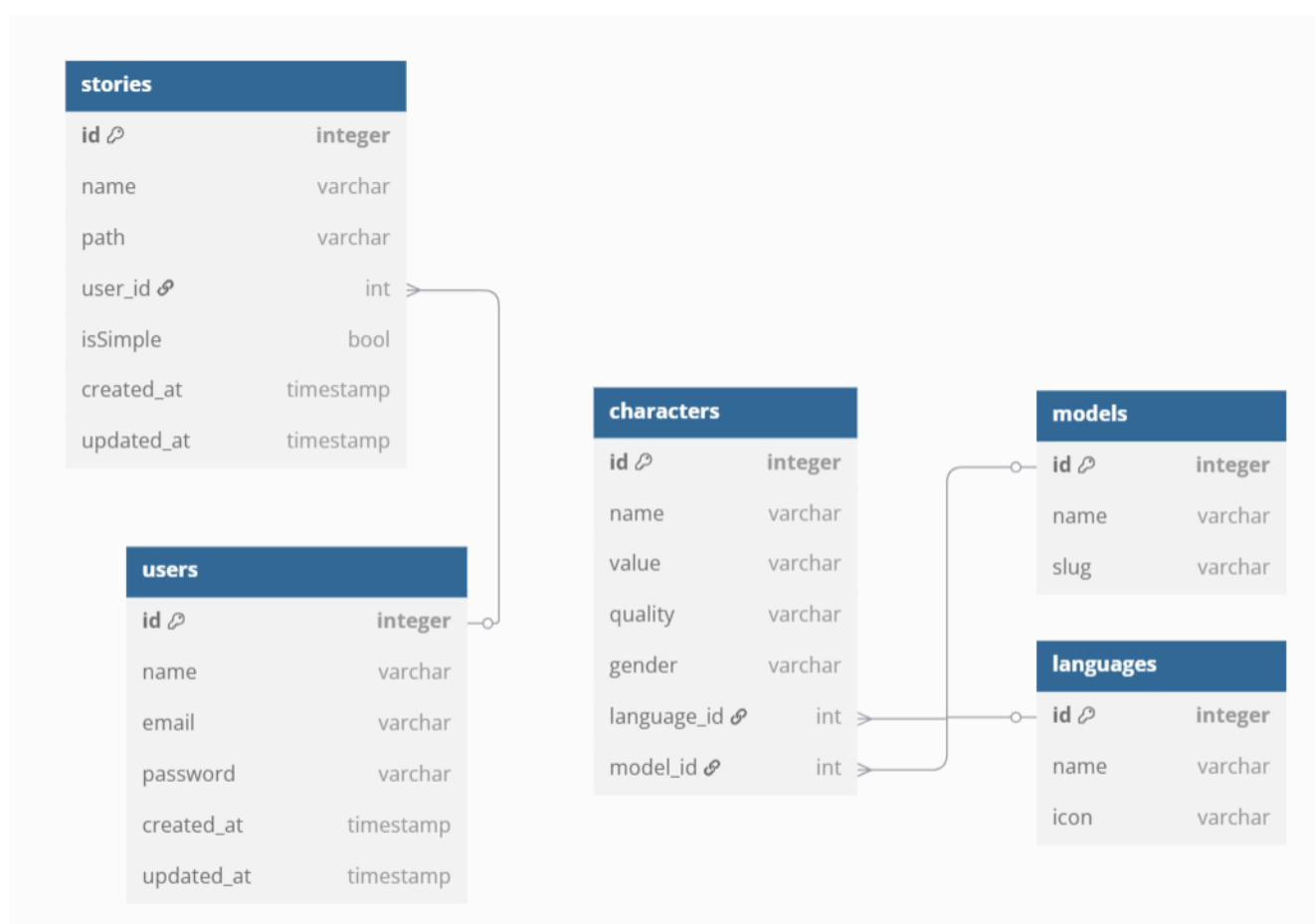


Рис. 2.3 Діаграма бази даних PostgreSQL

MongoDB — документо-орієнтована нереляційна база даних, що ідеально підходить для зберігання великих обсягів гнучко структурованих даних. У цій системі зберігаються створені аудіофайли, репліки аудіоісторій.

Колекція Audios

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
storyId	varchar	Назва моделі
fileUrl	varchar	Скорочена назва моделі
size	int	Розмір файлу
text	text	Текст озвучення
character	varchar	Персонаж озвучення
voiceModel	varchar	Модель, яка використовувалась для генерації цього аудіо

2.3 Проектування та реалізація алгоритмів роботи системи

Ефективні та чітко визначені алгоритми є ключовим елементом для забезпечення зручної та інтуїтивно зрозумілої взаємодії користувача з платформою, а також для коректної обробки даних та виконання основних функцій додатку. Нижче наведено покроковий опис алгоритму роботи користувача з системою, починаючи від входу та закінчуючи збереженням створеної аудіоісторії, а також огляд ролей різних категорій користувачів та їхніх відповідних можливостей. Алгоритм роботи з додатком зображено на рисунку 2.4.

Робота з додатком розпочинається з початкової точки входу, де після запуску програми користувач потрапляє на екран привітання або одразу на екран реєстрації чи авторизації. Користувачу пропонується створити обліковий запис, якщо він є новим, натиснувши відповідну опцію, що переводить його на екран реєстрації. Якщо ж користувач вже має обліковий запис, він обирає опцію авторизації, після чого переходить на екран введення своїх облікових даних.

На екрані реєстрації користувачу необхідно заповнити форму, вказавши свою електронну адресу, пароль (з підтвердженням) та, за бажанням, ім'я користувача. Після заповнення всіх полів користувач натискає кнопку "Зареєструватися". У випадку, якщо введені дані відповідають усім вимогам, таким як валідна електронна адреса та відповідність пароля встановленим критеріям, обліковий запис успішно створюється, і користувач автоматично авторизується або йому пропонується увійти до системи.

Для авторизації користувачу надається форма, де потрібно ввести свою електронну адресу (логін) та пароль. Після введення цих даних користувач натискає кнопку "Увійти" або "Авторизуватися". Система перевіряє введені облікові дані, і в разі їхньої вірності користувач отримує доступ до всіх основних функцій додатку та переходить до етапу вибору дії. Якщо ж введені облікові дані є невірними, користувачу відображається повідомлення про помилку, наприклад, про невірний логін або пароль, і він повертається на екран авторизації для повторної спроби.

Після успішної авторизації або реєстрації користувач отримує можливість обрати одну з основних дій. Опція "Створити нову історію" відкриває інтерфейс для створення аудіоісторії, де користувач може ввести необхідний текст для озвучування, використовуючи клавіатуру або вставляючи скопійований текст. Також користувачу може бути надана можливість завантажити текстовий файл. На наступному етапі користувач обирає голос (персонажа) з доступного списку для озвучування введеного тексту, при цьому може бути доступна функція попереднього прослуховування обраного голосу та генерує остаточний аудіо файл.

Опція "Перегляд створених історій" відображає список усіх історій, створених поточним користувачем. Для кожної історії в цьому списку користувач може мати можливість редагувати її, переходячи до відповідного інтерфейсу для внесення змін у текст, голос та інші параметри, після чого зміни зберігаються.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Також користувач може мати можливість видалити обрану історію, підтвердивши своє рішення.

Після введення тексту та вибору голосу або після редагування існуючої історії користувач переходить до етапу збереження. На цьому етапі йому може бути запропоновано ввести заголовок для своєї історії. Після натискання кнопки "Зберегти" історія зберігається в базі даних. Після успішного збереження користувач повертається до екрану з основними діями, де він може продовжити роботу з додатком, створюючи нові історії, здійснюючи пошук або переглядаючи вже створені. Таким чином, додаток надає користувачу повний цикл роботи зі створенням та керуванням аудіоісторіями.

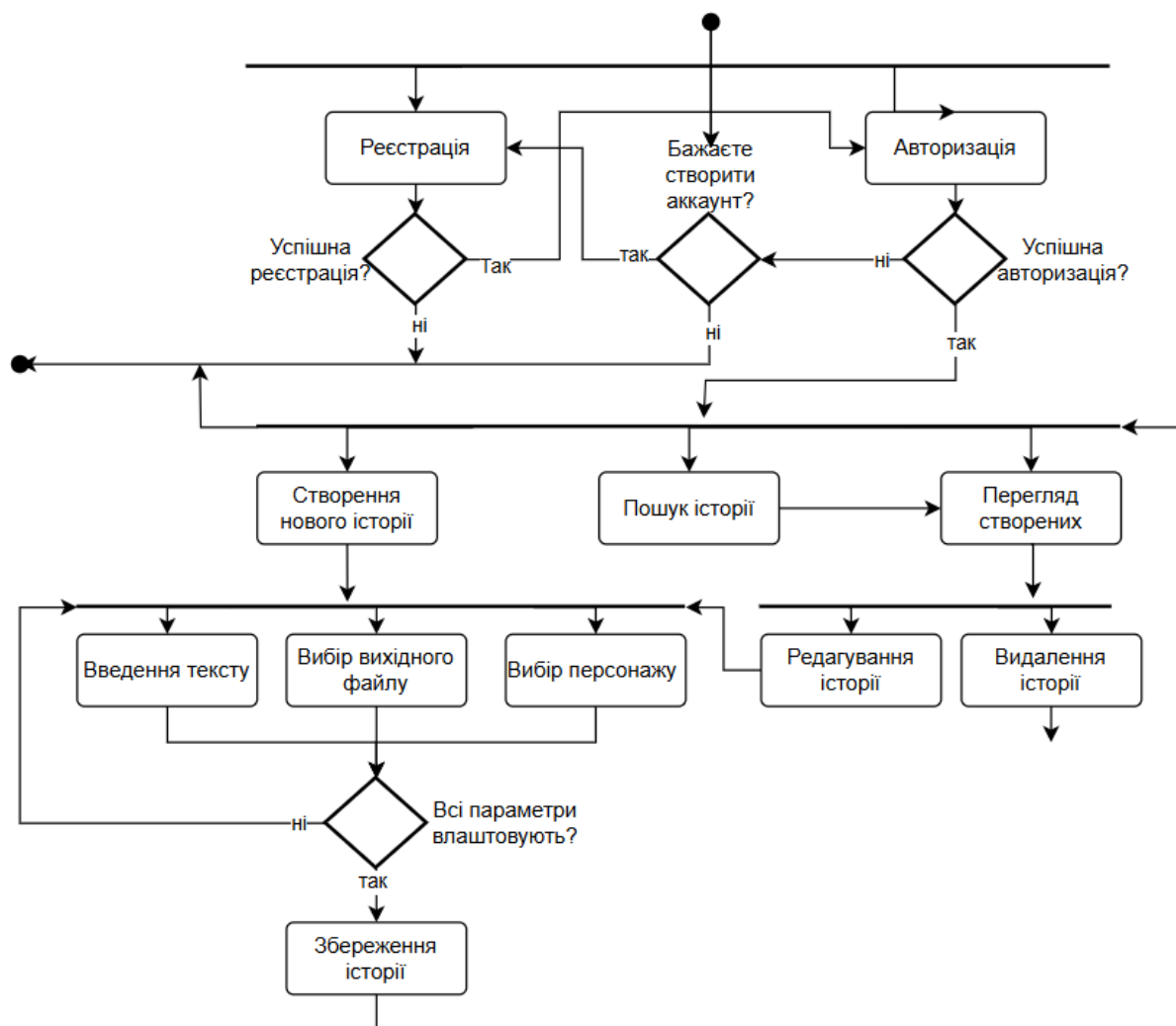


Рис 2.4 Діаграма активності

2.4. Порядок встановлення та налаштування параметрів додатку

Встановлення та налаштування додатку для створення аудіоісторій із синтезом мовлення передбачає розгортання всіх її компонентів та налаштування взаємодії між ними. Кожен компонент виконує окрему роль у функціонуванні системи, а їхня взаємодія забезпечує повний цикл генерації та відтворення аудіоісторій. Для пояснення архітектури платформи створено діаграму компонентів (рис. 3.1) та діаграму розгортання (рис. 3.2).

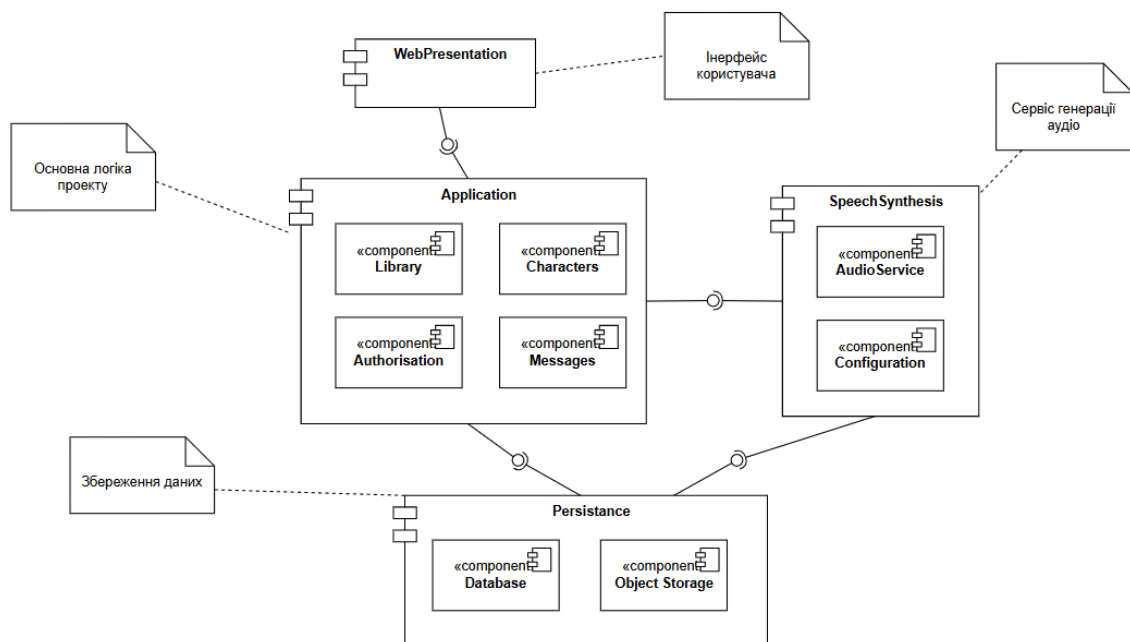


Рис. 3.1. Діаграма компонентів

Application – це центральний компонент, що об'єднує в собі декілька підкомпонентів, таких як "Library" (Бібліотека) для керування аудіоісторіями, "Characters" (Персонажі), "Authorisation" (Авторизація) для управління доступом користувачів, та "Messages" (Повідомлення або Репліки) для обробки комунікацій. Цей компонент надає інтерфейс для "WebPresentation" (Веб-презентація) та взаємодіє з компонентом "Persistence".

Speech-Synthesis – відповідає за генерацію мовлення. Він включає підкомпоненти "AudioService" (Аудіосервіс), що надає базові можливості для

створення звуку, та "Configuration" (Конфігурація) для налаштувань. Цей компонент також взаємодіє з компонентом "Persistence".

Persistence – керує зберіганням даних. Він містить підкомпоненти "Database" (База даних) для традиційного збереження інформації та "Object Storage" (Об'єктне сховище) для збереження бінарних об'єктів. Цей компонент надає інтерфейси, які необхідні як для "Application", так і для "Speech-Synthesis", що підкреслює його роль як центрального сховища даних.

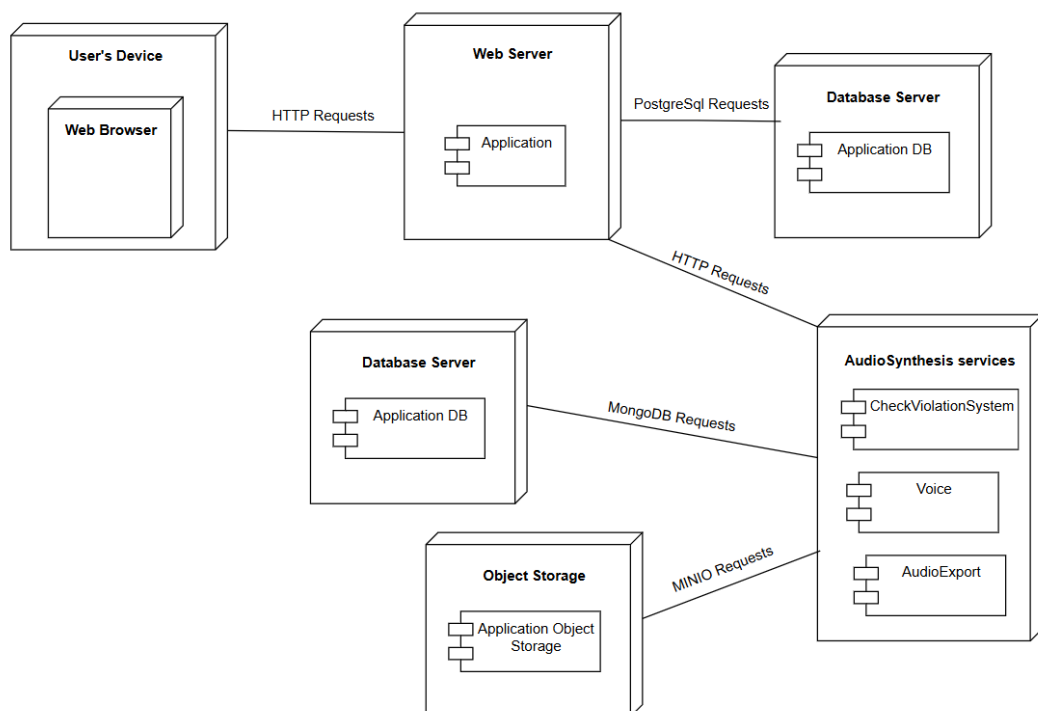


Рис. 3.2. Діаграма розгортання

Web Server – клієнтська та серверна частина застосунку, що відповідає за інтерфейс користувача, дозволяє вводити текст, переглядати історії, обирати голоси та керувати налаштуваннями генерації а також зберігання користувацьких даних. Виконує запити до аудіосервісу через REST API.

Audio Synthesis Service – окремий мікросервіс, відповідальний за синтез мовлення. Реалізований на базі спеціалізованих TTS (Text-to-Speech) бібліотек.

Отримує запити з текстом від основного веб-сервера, генерує аудіофайл і повертає його.

Object Storage Service – служба зберігання згенерованих аудіофайлів
Взаємодіє з аудіосервісом та бекендом для збереження і видачі аудіо.

Database Server – служби зберігання даних як для веб сервера так і для аудіосервісу.

2.5. Реалізація функціоналу застосунку

Перш за все, користувач виконує вхід у систему. Реалізовано функцію авторизації, яка надсилає введені облікові дані на сервер для перевірки:

```
public function login(): void
{
    $this->validate();
    $this->ensureIsNotRateLimited();
    if (! Auth::attempt(['email' => $this->email, 'password' => $this->password], $this->remember)) {
        RateLimiter::hit($this->throttleKey());
        throw ValidationException::withMessages([
            'email' => __('auth.failed'),
        ]);
    }
}
```

Після успішного входу користувач перенаправляється на головну сторінку, де може переглядати створені історії або створювати нові. Для отримання даних надсилається запит на сервер:

```
$url_info = str_replace('id',
    $library->id,
    config('services.audios-info.endpoint'));
$response = Http::timeout(30)->retry(2, 60)->get($url_info);
```

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

На сервері запит обробляється відповідним маршрутом, який вибирає історії з бази даних:

```
@app.get("/audios/{audiolog_id}/info")
def get_audios_info_by_audiolog_id(audiolog_id):
    audialogs_service = AudioDialogService()
    return audialogs_service.get_audios(audiolog_id)
```

Коли користувач натискає кнопку «Створити історію», запускається запит, що ініціалізує нову історію і сервер створює запис у базі даних і повертає створену історію:

```
public function save(){
    $user_id = auth()->id();
    $this->validate([
        'title' => 'required|string|max:255',
    ]);
    $audiolog = Audiolog::create([
        'name' => $this->title,
        'path' => "",
        'user_id' => $user_id,
        'isSimple' => $this->isSimple
    ]);
    $audiolog->path = "user-{$user_id}/audiolog-{$audiolog->id}";
    $audiolog->save();
    return redirect()->route('library-show', $audiolog->id);
}
```

Усередині редактора історій користувач додає текстові репліки персонажів. Для кожної репліки можна викликати генерацію аудіо за допомогою синтезу мовлення:

```
$payload = [
    'character' => $this->character,
```

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

```

    'text' => $this->text,
    'model' => $model,
    'audiolog' => [
        'id' => (string)$audiolog->id,
        'name' => $audiolog->name,
        'user_id' => (string)$audiolog->user_id,
        'path' => $audiolog->path,
    ],
];

$response = Http::timeout(120)->post(config('services.generate.endpoint'),
    $payload);

```

Генерація обробляється на сервері із використанням сервісу синтезу мовлення:

```
@app.post("/add")
```

```
def create_audio(audio:AudioRequest):
```

```
    audialogs_service = AudioDialogService()
```

```
    file, filename, err = audialogs_service.add_audio(audio)
```

```
    if err != None:
```

```
        raise HTTPException(status_code=400, detail=err["error"])

```

Історію можна експортувати в один із підтримуваних форматів (наприклад, MP3, WAV, FLAC) за допомогою вибору відповідної стратегії експорту. Для експорту реалізовано функцію, яка надсилає запит до сервера:

```
public function download($format){
```

```
    $generate_url = str_replace(':path', $this->audioPath,
    config('services.export.endpoint'));

```

```
    $response = Http::timeout(60)->post($generate_url, ["format"=>$format]);
```

```
    if ($response->successful()){
```

```
        $stream = $response->getBody();
```

```
        $filename = 'result_audio.' . $format;
```

```
        return response()->streamDownload(function () use ($stream) {
```

```
            while (!$stream->eof()) {
```

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

```

        echo $stream->read(1024);
    }
    }, $filename, [
        'Content-Type' => "audio/{ $format}",
        'Content-Disposition' => "attachment; filename={ $filename}",
        'Access-Control-Expose-Headers' => 'Content-Disposition',
    ]);
}
return back()->with('error', 'Audio export failed.');
```

Висновок до другого розділу

У процесі проектування додатку для створення аудіоісторій із синтезом мовлення було детально проаналізовано усі можливі сценарії використання програмного продукту, відповідно до визначених вимог. На основі цього аналізу була розроблена структура додатку, що включала моделювання взаємодії між його основними компонентами та функціональними можливостями. Особливу увагу було приділено проектуванню структури зберігання даних. Для забезпечення ефективної роботи додатку та збереження створених аудіоісторій, налаштувань користувачів та іншої необхідної інформації було спроектовано відповідну базу даних. У розділі було детально описано алгоритм роботи користувача з додатком, починаючи від етапів реєстрації та авторизації, проходячи через створення, пошук та перегляд аудіоісторій, і завершуючи процесом їх збереження. Опис кожного кроку дозволяє чітко зрозуміти логіку взаємодії користувача з програмним забезпеченням. Крім того детально розглянуто процес розгортання платформи для створення аудіоісторій із синтезом мовлення. Було описано кожен структурний компонент системи, а також проаналізовано його взаємодію з іншими частинами платформи. Це дозволило сформувати цілісне уявлення про архітектуру системи та зв'язки між її модулями.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ДОДАТКОМ ДЛЯ
СТВОРЕННЯ АУДІОІСТОРІЙ ІЗ СИНТЕЗОМ МОВЛЕННЯ

3.1 Опис інтерфейсу додатку

Взаємодія користувачів відбувається через веб-додаток. На головній сторінці (рис. 3.1) користувач може перейти до входу в систему, натиснувши на кнопку “Ввійти”. (рис 3.2).

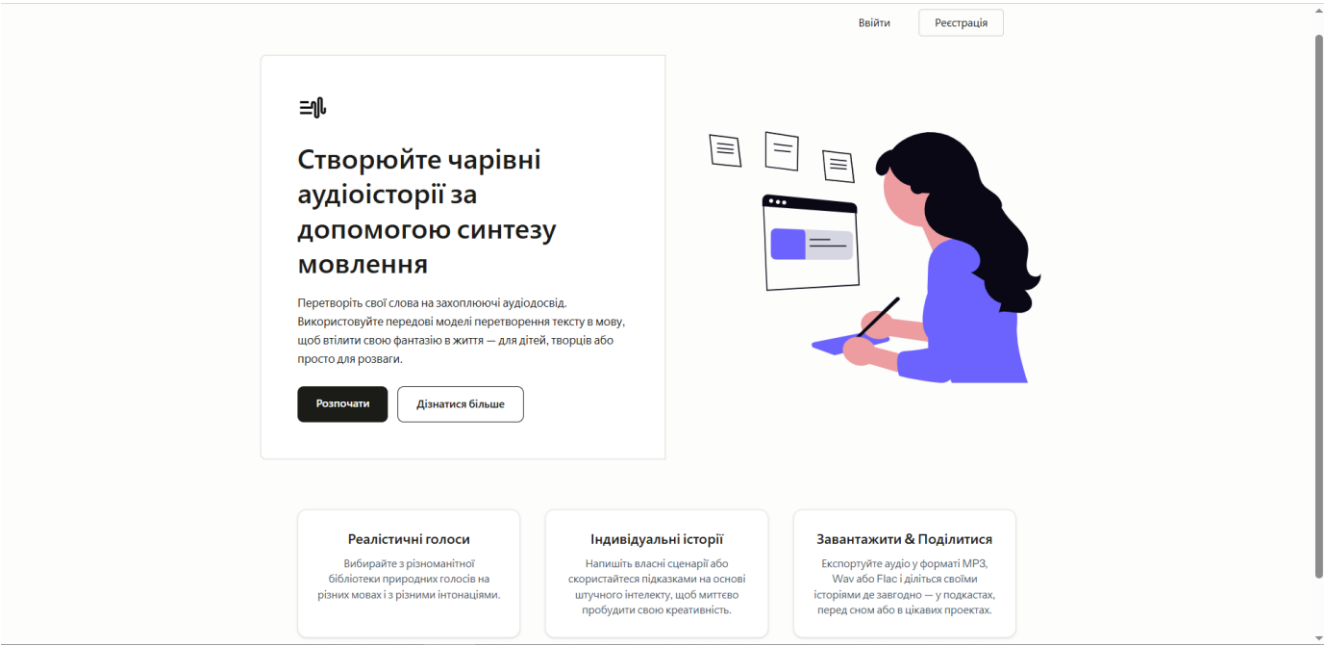


Рис 3.1 Початкова сторінка додатку

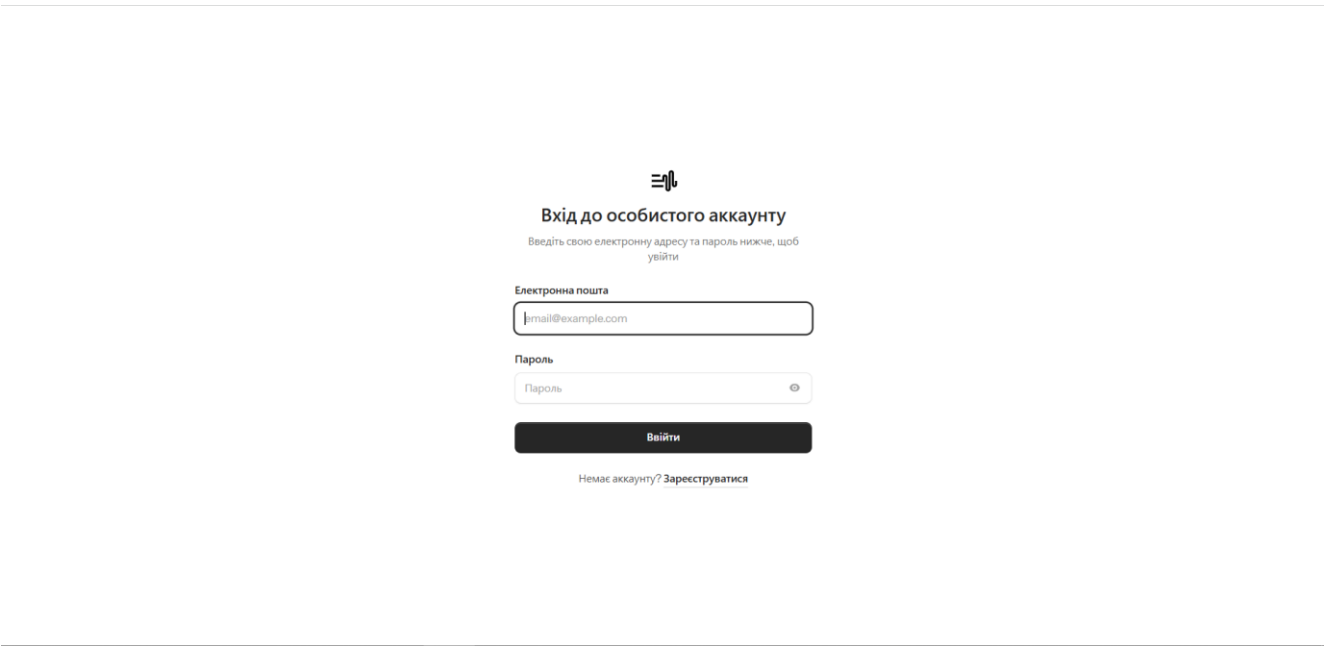


Рис 3.2 Вхід до особистого аккаунту

Якщо користувач ще не має зареєстрованого облікового запису, система надає можливість пройти процедуру реєстрації для створення персонального акаунта, за допомогою якого він зможе отримати повний доступ до функціоналу веб-додатку (рис. 3.3).

Рис 3.3 Створення особистого акаунту

Після успішної автентифікації користувач перенаправляється на панель приладів, яка виконує функцію стартової сторінки особистого кабінету. На цій сторінці відображається загальна статистична інформація щодо активності у додатку, а також перелік нещодавно створених або збережених аудіоісторій, що забезпечує зручний доступ до останніх дій користувача (рис. 3.4).

Рис 3.4 Панель приладів

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

Ліва навігаційна панель інтерфейсу надає користувачам прямий доступ до розділу бібліотеки аудіоісторій, що є ключовим елементом організації робочого простору. Цей структурний елемент системи виконує важливу функціональну роль, забезпечуючи швидкий та інтуїтивно зрозумілий перехід між основними розділами додатку.

Бібліотека аудіоісторій реалізована як централізоване сховище всіх створених користувачем матеріалів, що забезпечує ефективний доступ до контенту (рис. 3.5). Інтерфейс розділу розроблений з урахуванням принципів зручності користування. Кожен елемент бібліотеки містить набір інструментів для управління відповідною історією, включаючи функції редагування та видалення.

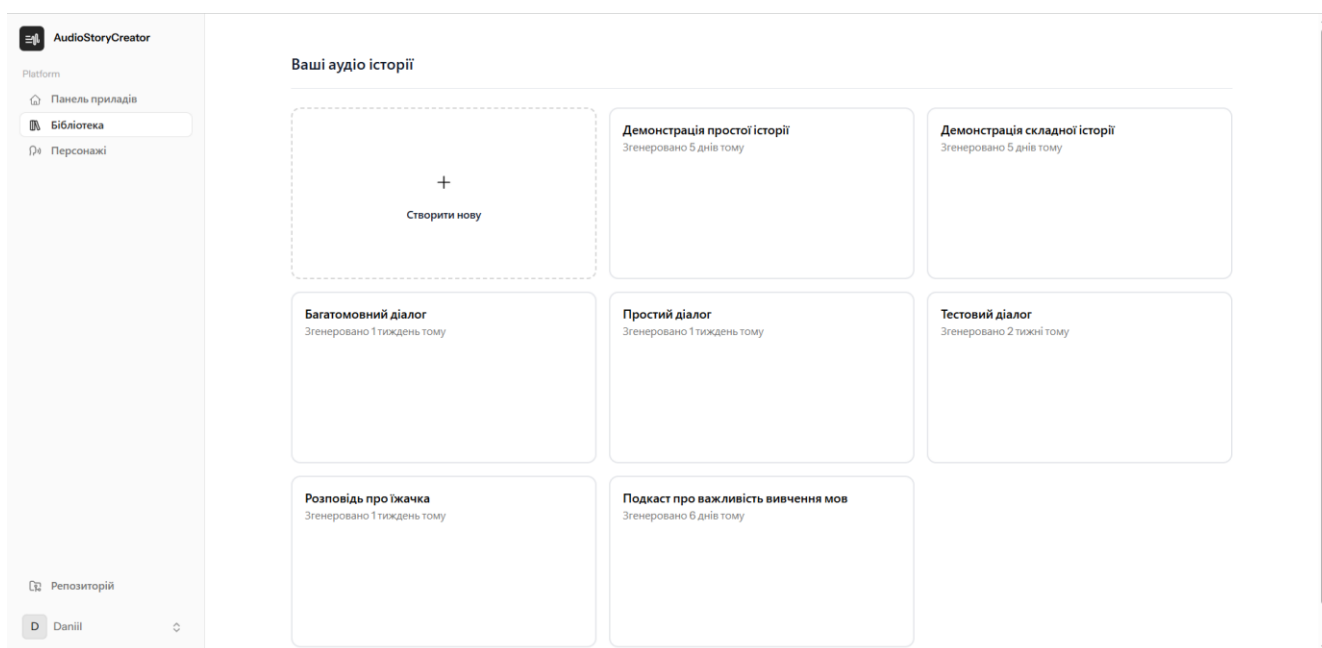


Рис 3.5 Бібліотека історій

Після натискання користувачем на прямокутну кнопку з написом «Створити нову», цей елемент інтерфейсу змінює зовнішній вигляд і перетворюється на компактну форму для введення даних. Форма містить поле, у якому можна вказати назву нової історії, а також перемикач, що дозволяє обрати тип історії – простий (з одним персонажем) або складний (з можливістю додавання кількох персонажів). Відповідний інтерфейсний перехід ілюстровано на рис. 3.6.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

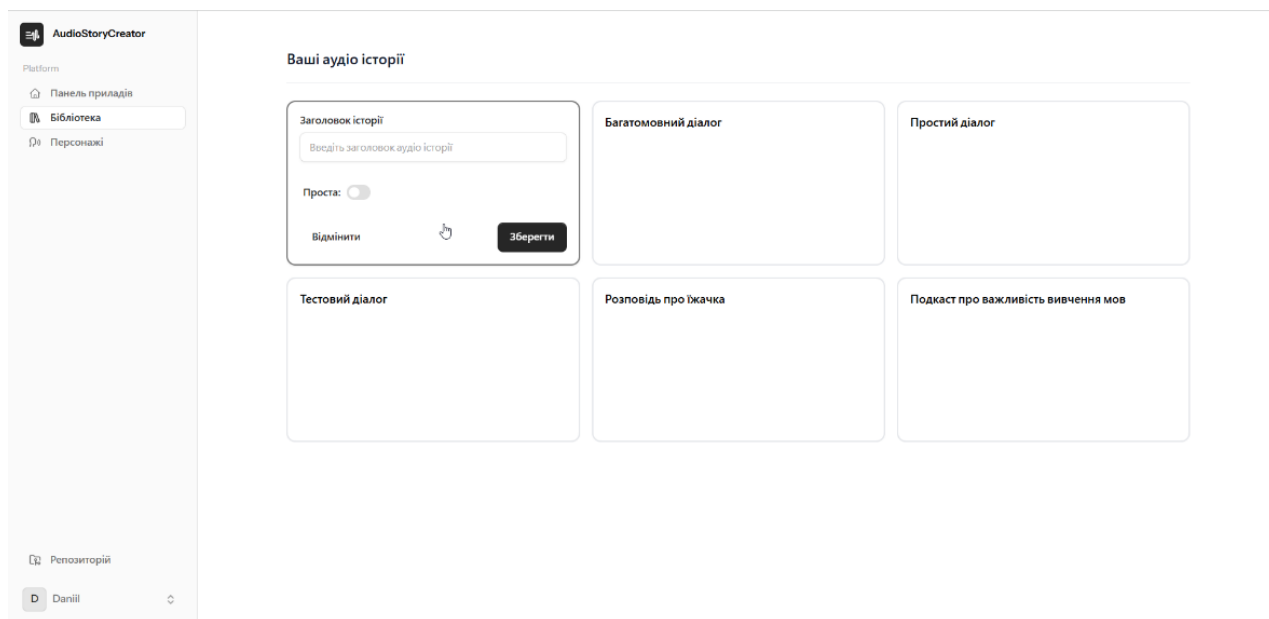


Рис 3.6 Створення нової історії

Після вибору режиму «Проста історія» система автоматично перенаправляє користувача на спеціалізовану веб-сторінку, призначену для створення та редагування історій з одним персонажем. На цій сторінці передбачено інтерфейс для введення текстового вмісту, вибір персонажа та збереження готової історії. Відповідний інтерфейсний перехід та структура сторінки наведені на рис. 3.7.

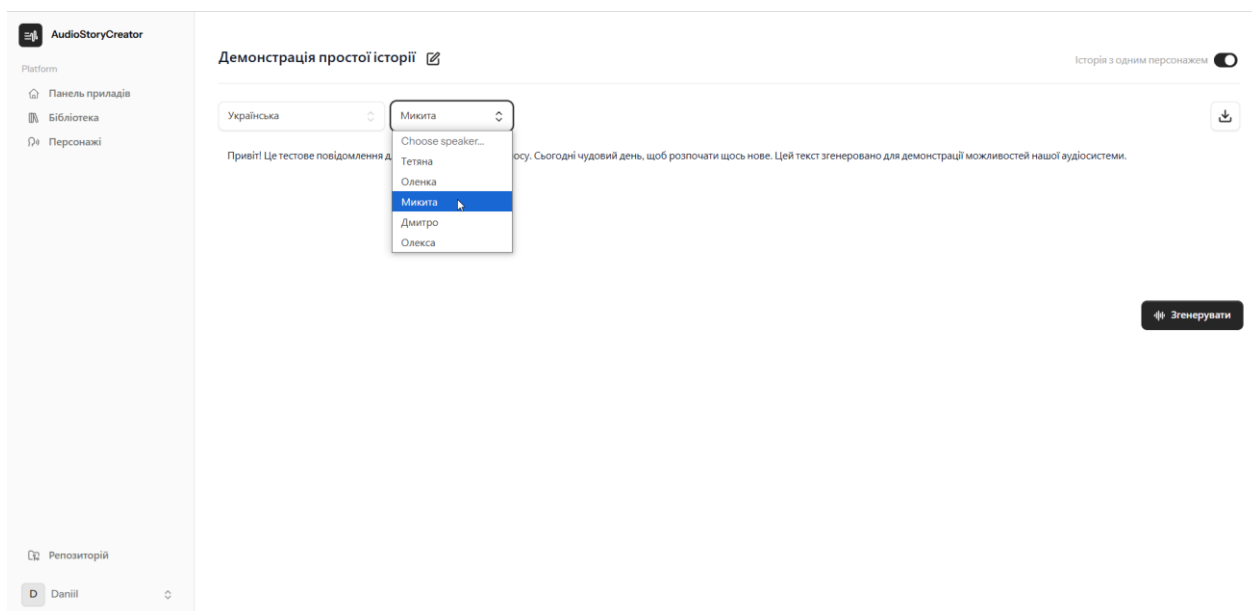


Рис 3.7 Веб-сторінка простої історії

Після завершення введення текстового вмісту та вибору відповідного персонажа, користувач має можливість активувати функцію генерації аудіозапису

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

шляхом натискання кнопки "Згенерувати". Після ініціювання цього процесу система автоматично обробляє введені дані, створює аудіофайл у відповідності до заданих параметрів та завантажує його до інтерфейсу браузера. Протягом обробки кнопка відображає індикатор виконання операції. Результат у вигляді готового аудіозапису стає доступним для прослуховування. Візуалізація даного процесу представлена на рис. 3.8.

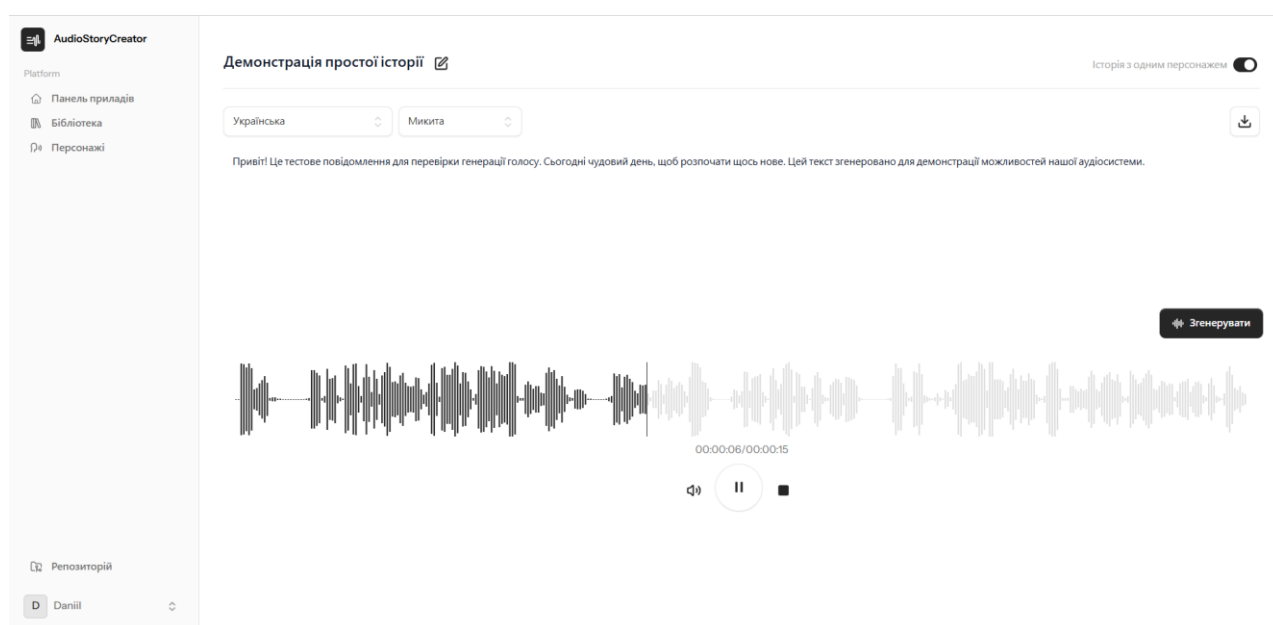


Рис 3.8 Програвання згенерованого аудіо

Після успішного створення історії з підтримкою кількох персонажів система автоматично перенаправляє користувача на спеціалізовану веб-сторінку (рис. 3.9). Ця сторінка призначена для подальшого наповнення та редагування багатоперсонажної історії. Користувач має змогу додавати нові репліки, обирати персонажа до репліки (рис. 3.10), генерувати аудіо та прослуховувати його після успішної генерації. Інтерфейс дозволяє керувати послідовністю діалогів та здійснювати загальний контроль над структурою історії.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

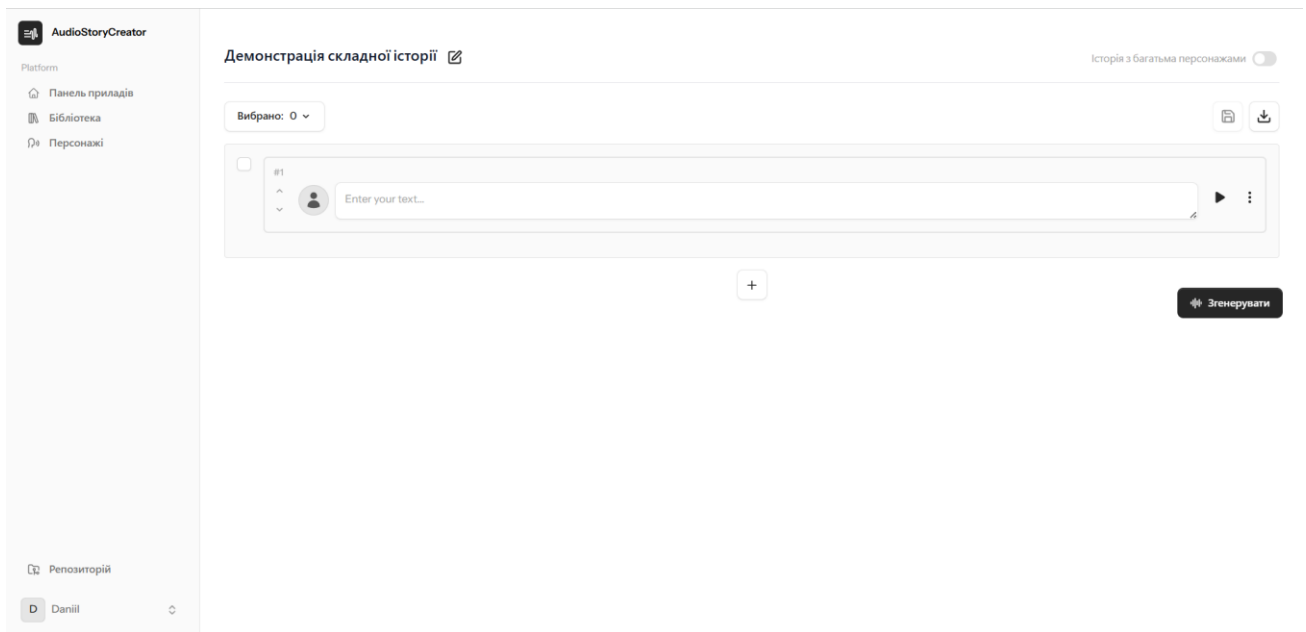


Рис 3.9 Веб-сторінка складної історії

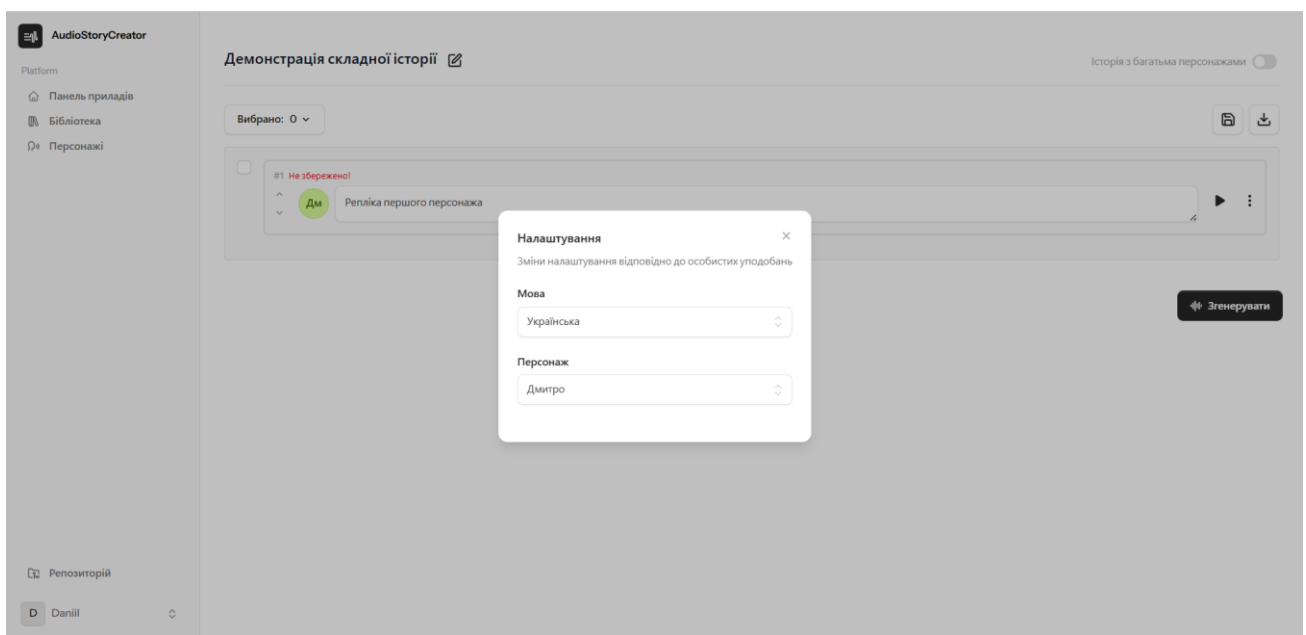


Рис 3.10 Вибір персонажа озвучення

Після додавання нових реплік до історії користувач може одночасно оновити відразу кілька елементів. Для цього потрібно відзначити потрібні репліки, поставивши галочки у спеціальних полях зліва від кожної з них (рис. 3.11). Після цього система генерує всі оновлені аудіозаписи, які одразу стають доступними для прослуховування. При успішному завершенні процесу зникає позначка "Не збережено", що підтверджує збереження всіх змін.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		50

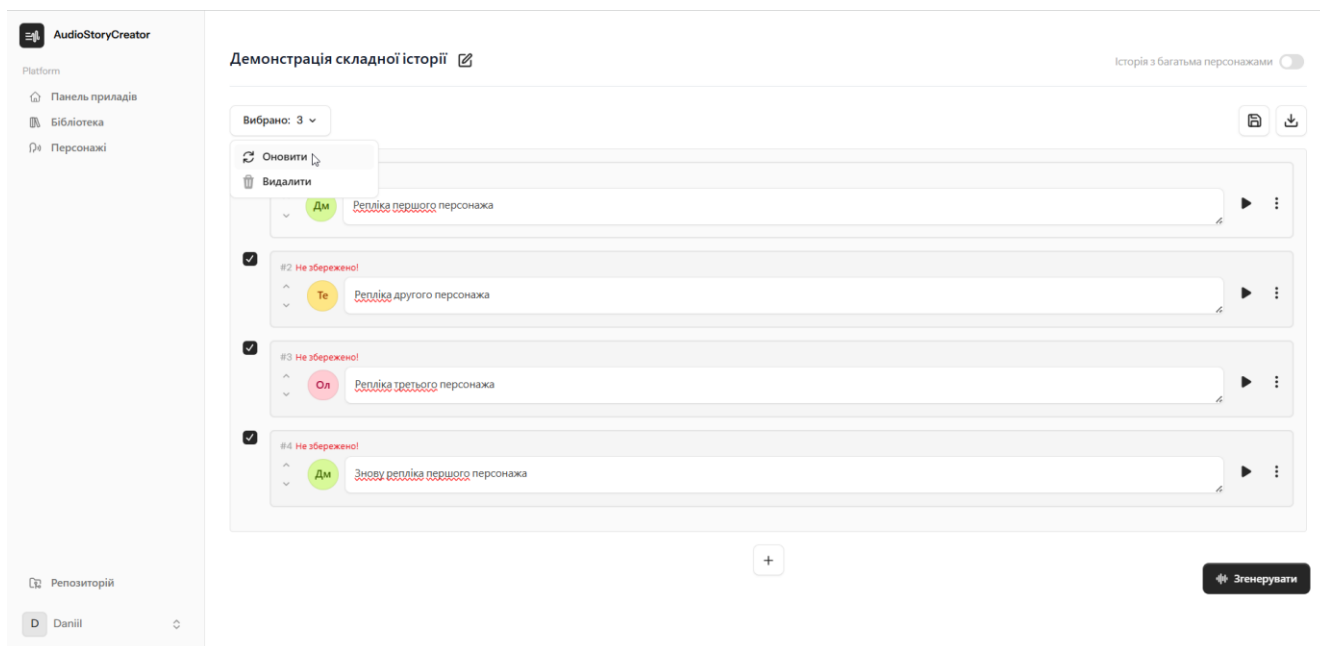


Рис 3.11 Оновлення багатьох реплік

Після перевірки коректності введених даних користувач може перейти до завершального етапу - одночасної генерації всієї історії. Для цього необхідно скористатись кнопкою "Згенерувати" в інтерфейсі програми (рис. 3.12). Система автоматично виконує комплексну обробку всіх компонентів створює об'єднує аудіоверсії кожної репліки та формує кінцевий результат у єдиному форматі. Після успішного завершення процесу користувач отримує готове аудіо з усіма звуковими репліками.

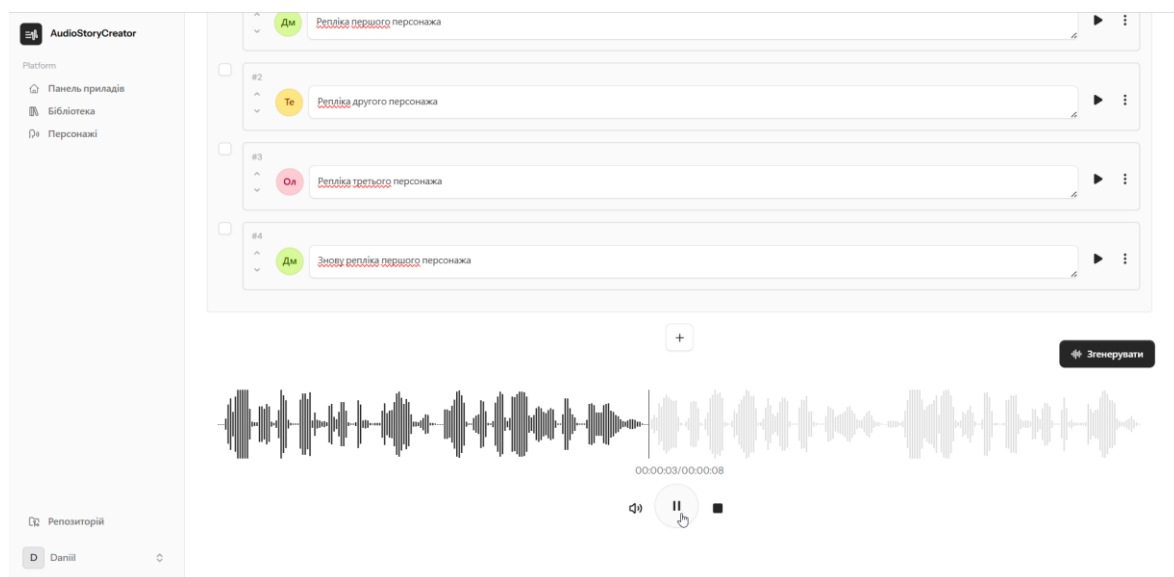


Рис 3.12 Результат генерації об'єднаного аудіо

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Для детального ознайомлення з параметрами та характеристиками кожного персонажа, користувач може скористатися спеціальним розділом "Список персонажів" (рис. 3.13). У цьому розділі представлений повний перелік усіх доступних персонажів з можливістю перегляду їхніх детальних карток.

Кожна картка персонажа містить набір заданих параметрів, включаючи стать, мову озвучення та характеристики якості. Інтерфейс розділу дозволяє зручно орієнтуватись серед великої кількості персонажів завдяки можливостям фільтрації (рис. 3.14). При необхідності, користувач може внести зміни до імені персонажа безпосередньо в його картці.

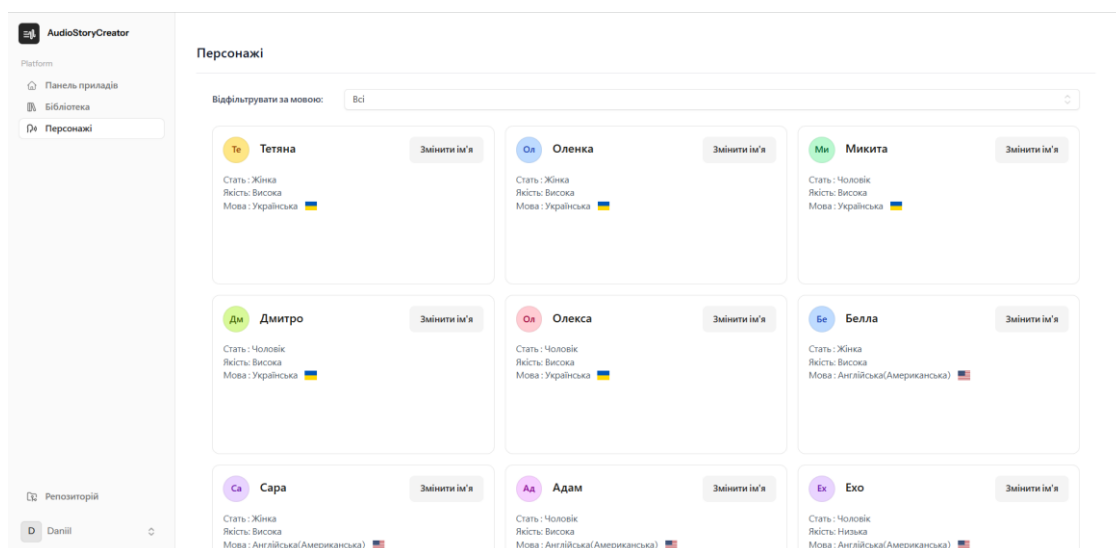


Рис 3.13 Відображення всіх персонажів

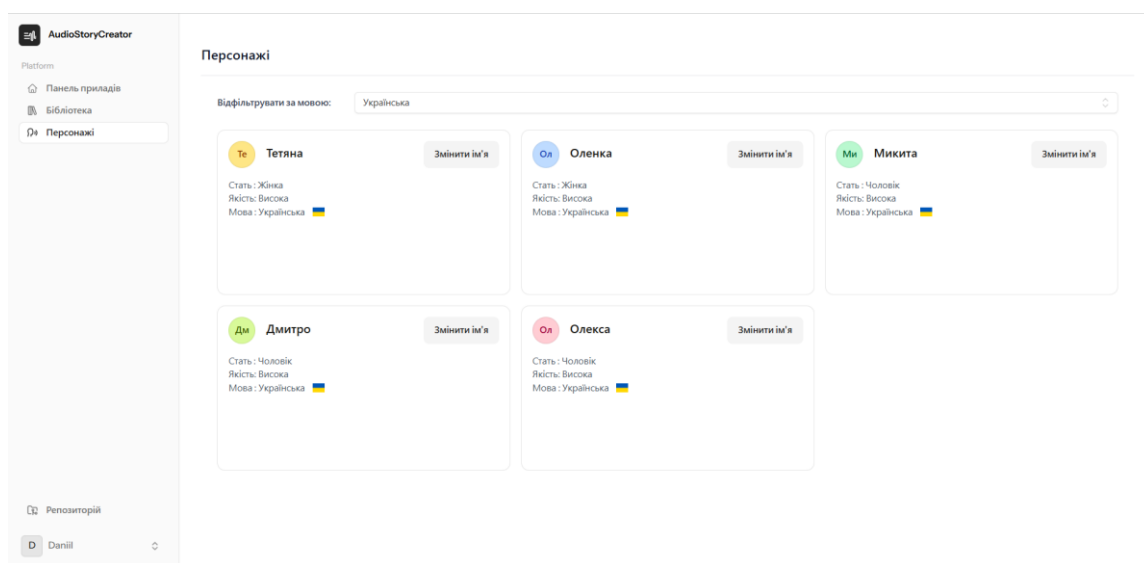


Рис 3.14 Відфільтровані україномовні персонажі

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				52
Змн.	Арк.	№ докум.	Підпис	Дата		

Додаток забезпечує гнучкі налаштування візуального оформлення, підтримуючи два варіанти інтерфейсу – світлу та темну теми (рис. 3.15). Користувач може самостійно обрати оптимальний режим відображення в залежності від особистих уподобань або зовнішніх умов. Світла тема, яка є типовим варіантом оформлення, забезпечує чіткість та контрастність елементів інтерфейсу, що сприяє комфортній роботі при яскравому освітленні. Темна тема, розроблена з урахуванням сучасних тенденцій, зменшує навантаження на зір та енергоспоживання пристрою, що особливо актуально під час тривалої роботи в умовах недостатнього освітлення. Перемикання між темами здійснюється миттєво через відповідний пункт меню налаштувань. Система автоматично адаптує всі елементи інтерфейсу – кольорову гаму, шрифти та графічні компоненти – відповідно до обраного режиму.

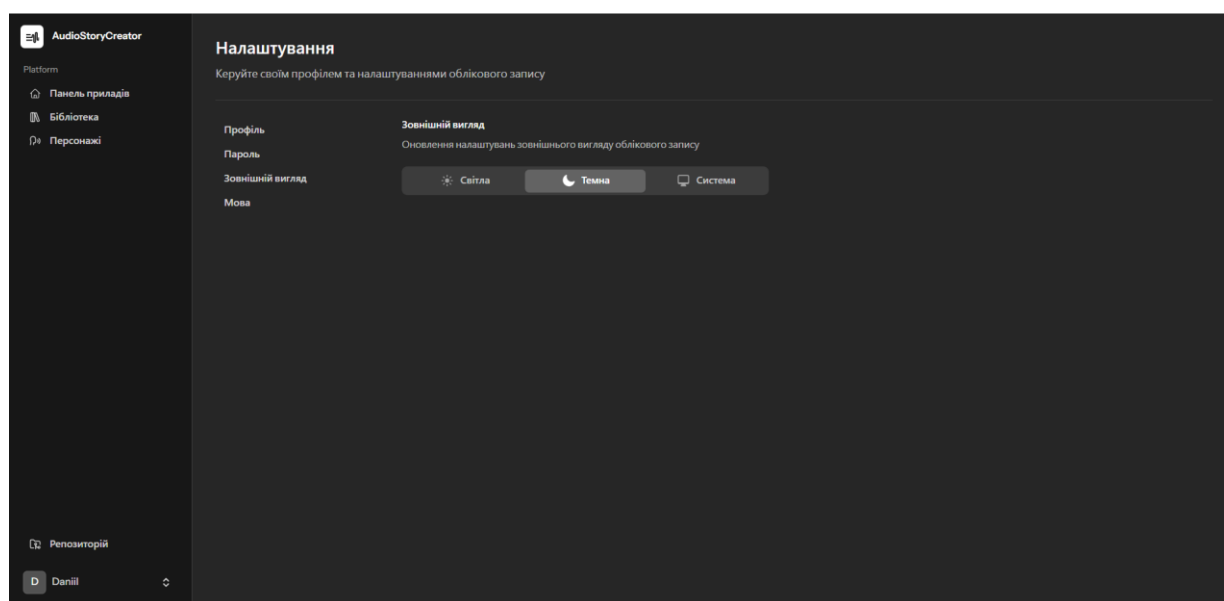


Рис 3.15 Відображення темної теми

Додаток має вбудовану систему підтримки різних мов, що дозволяє користувачам обирати зручну для них мову інтерфейсу. На даний момент доступні українська та англійська версії. При зміні мови автоматично оновлюються всі елементи інтерфейсу: меню, кнопки, повідомлення та підказки. Перехід на іншу мову відбувається миттєво без необхідності перезапуску програми. Мовні

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

налаштування легко знайти в розділі параметрів додатку. Візуальний приклад роботи системи локалізації можна побачити на рис. 3.16.

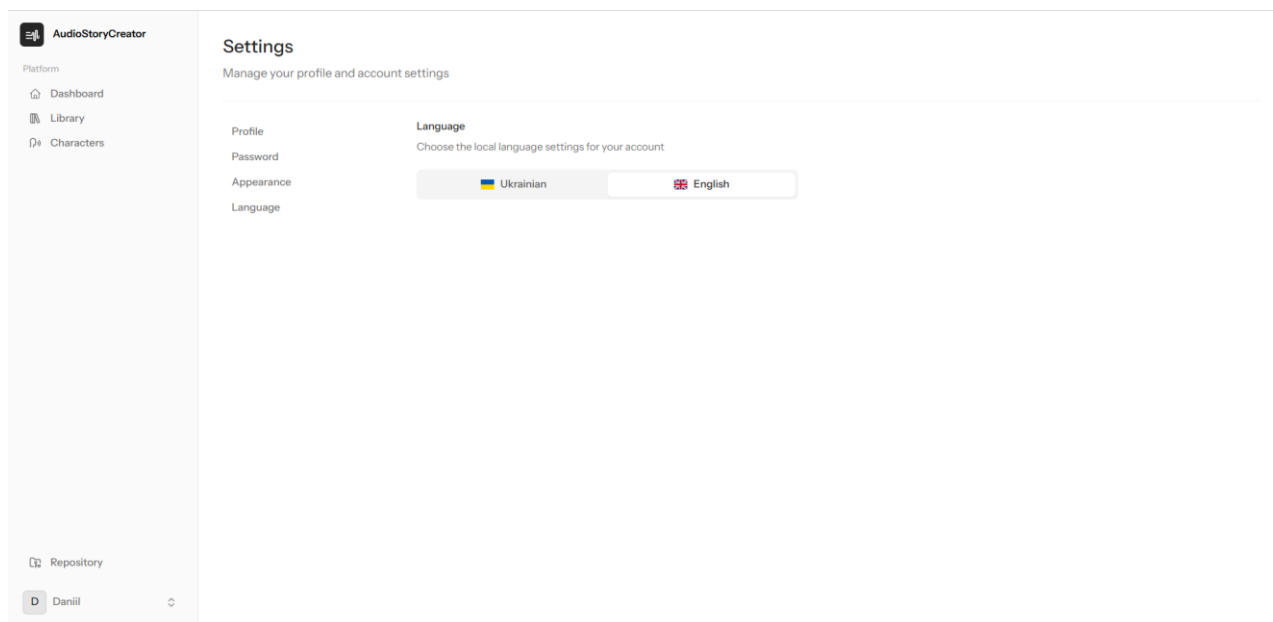


Рис 3.16 Відображення інтерфейсу англійською мовою

3.2 Тестування платформи

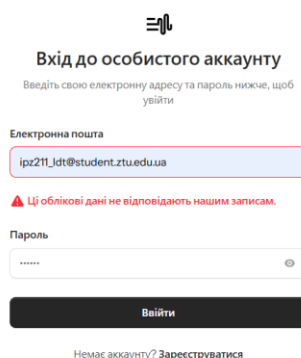
Для перевірки працездатності основних функцій платформи було проведено ручне тестування ключових сценаріїв взаємодії користувача із застосунком. Метою тестування було виявлення можливих помилок під час авторизації, створення, генерації, збереження та редагування аудіоісторій, а також перевірка механізмів синтезу мовлення та роботи з репліками персонажів.

Під час тестування роботи системи було перевірено сценарій введення некоректних облікових даних, зокрема неправильного логіну або пароля. У разі виявлення таких помилкових вхідних даних, система автоматично ініціює процедуру відмови в авторизації та інформує користувача про неможливість входу (рис. 3.17).

У разі введення некоректних даних під час реєстрації (наприклад, некоректної електронної пошти або слабкого пароля) система також виводить відповідні повідомлення про помилку з підказками щодо вимог до вхідних даних

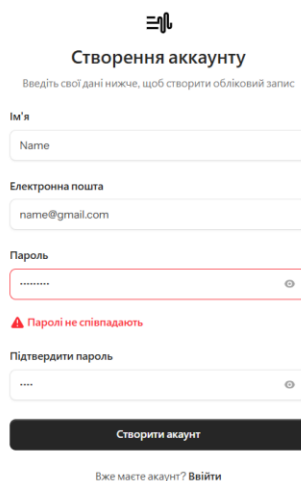
		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				54
Змн.	Арк.	№ докум.	Підпис	Дата		

(рис. 3.18). Це дозволяє користувачам оперативно виправити помилки та успішно завершити процес реєстрації.



The screenshot shows a login form titled "Вхід до особистого акаунту" (Login to personal account). Below the title is a subtitle: "Введіть свою електронну адресу та пароль нижче, щоб увійти" (Enter your email address and password below to log in). The form contains two input fields: "Електронна пошта" (Email) with the value "ipz211_jdt@student.ztu.edu.ua" and "Пароль" (Password) with masked characters. A red error message is displayed below the email field: "Ці облікові дані не відповідають нашим записам." (These login details do not match our records). Below the password field is a "Війти" (Log in) button. At the bottom, there is a link: "Немає акаунту? Зареєструватися" (Don't have an account? Register).

Рис 3.17 Відображення помилки у разі введення некоректних даних при вході в систему



The screenshot shows a registration form titled "Створення акаунту" (Account creation). Below the title is a subtitle: "Введіть свої дані нижче, щоб створити обліковий запис" (Enter your details below to create an account). The form contains four input fields: "Ім'я" (Name) with the value "Name", "Електронна пошта" (Email) with the value "name@gmail.com", "Пароль" (Password) with masked characters, and "Підтвердити пароль" (Confirm password) with masked characters. A red error message is displayed below the password field: "Паролі не співпадають" (Passwords do not match). Below the confirm password field is a "Створити акаунт" (Create account) button. At the bottom, there is a link: "Вже маєте акаунт? Війти" (Already have an account? Log in).

Рис 3.18 Відображення помилки у разі введення некоректних даних під час реєстрації

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

Також перевірено можливість створення нової історії. Якщо користувач не вводить назву або залишає поле порожнім, система не дозволяє зберегти історію та виводить попередження про обов'язковість заповнення (рис. 3.19).

Рис 3.19 Відображення помилки у разі створення нової історії без назви

Під час тестування функціоналу додавання реплік було ретельно перевірено роботу системи у різних сценаріях. Для успішної генерації аудіо користувач обов'язково має ввести текст репліки та обрати персонажа. У разі, якщо користувач забув ввести текст або не обрав персонажа, система реагує на цю ситуацію. Відповідні поля виділяються візуально, а на екрані з'являється чітке повідомлення про конкретну помилку з поясненням, які саме дані потрібно заповнити (рис. 3.20).

Рис 3.20 Відображення помилки спроби згенерування аудіо без обраного персонажа

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

Під час процесу генерації аудіо відбувається автоматична трансформація елементів інтерфейсу для надання зрозумілого зворотнього зв'язку. Кнопка відтворення, яка зазвичай містить іконку пуску, тимчасово замінюється на візуальний індикатор процесу завантаження (рис. 3.21). Цей індикатор виконаний у формі стилізованої анімації, яка наочно відображає хід виконання операції. Він виконує кілька ключових функцій: візуально підтверджує початок обробки, інформує про стан виконання завдання та запобігає передчасним спробам взаємодії з елементом до завершення процесу.

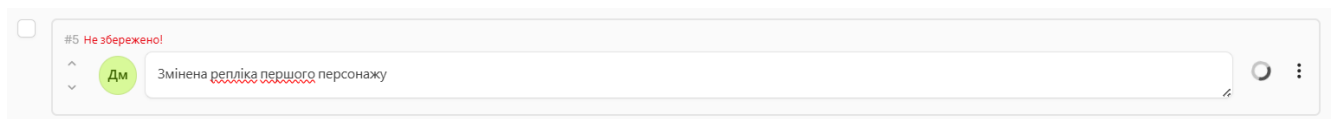


Рис 3.21 Відображення індикатора завантаження під час генерації аудіо

Під час виконання операції “збереження” система надає візуальний зворотній зв'язок через спеціальний індикатор завантаження, розташований поряд з кнопкою виконання дії. Цей елемент інтерфейсу активізується автоматично при ініціюванні процесу збереження (рис. 3.22).

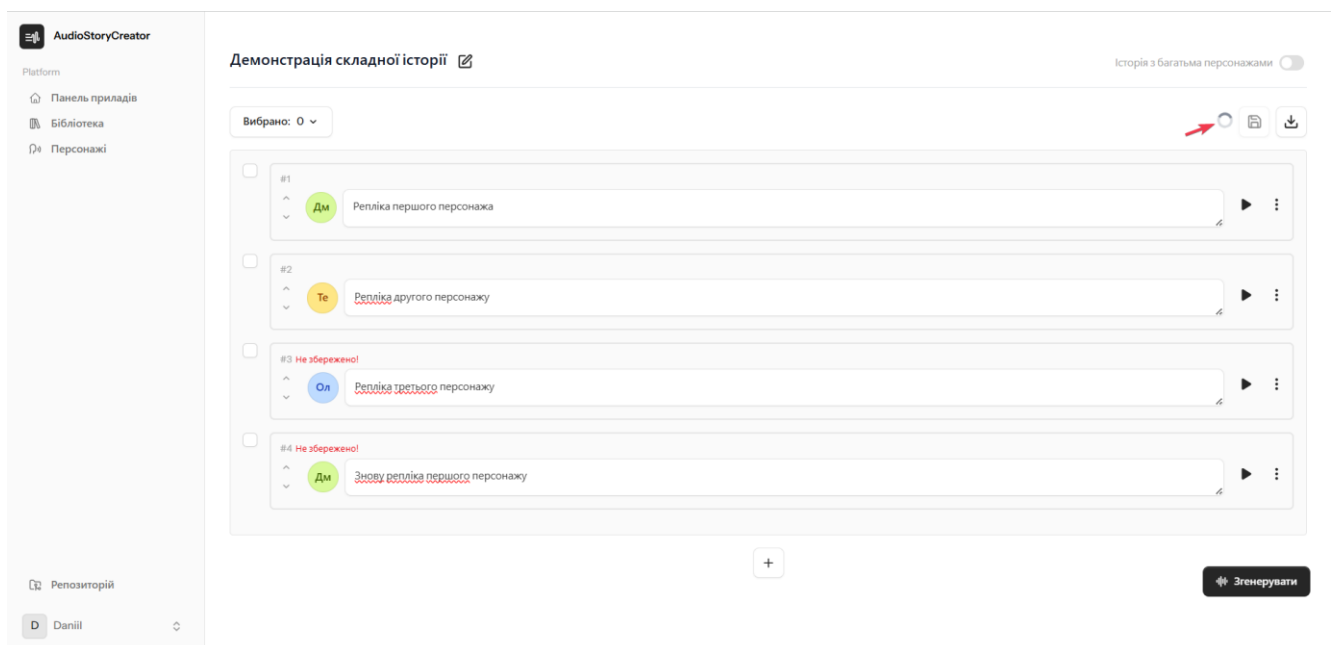
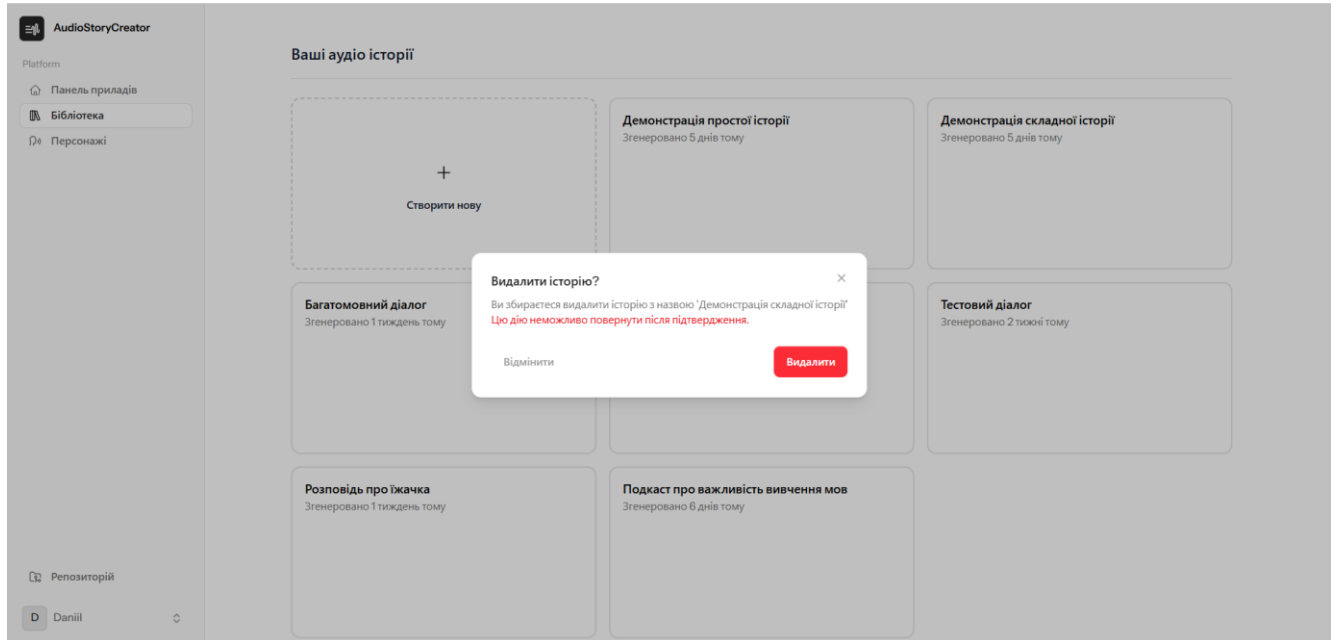


Рис 3.22 Відображення індикатора завантаження під час збереження

Функція видалення історії реалізована з додатковим захистом від випадкових дій. При спробі видалення система вимагає явного підтвердження наміру

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

користувача. Коли користувач натискає кнопку "Видалити", з'являється діалогове вікно з підтвердженням (рис. 3.23). Воно містить назву обраної історії, попередження про незворотність дії та два варіанти відповіді - підтвердити або скасувати операцію.



Висновок до третього розділу

У цьому розділі було представлено інтерфейс користувача розробленого додатку. Детально розглянуто функціональні можливості системи. Користувач має доступ до індивідуального набору функцій, зокрема: створення текстових історій, вибір голосу для озвучення, попередній перегляд аудіо та керування бібліотекою аудіофайлів. Це забезпечує зручність у користуванні та ефективне використання функціоналу відповідно до потреб кожного користувача.

Крім того було проведено тестування у ході якого було перевірено роботу системи у різних сценаріях використання, зокрема при введенні некоректних даних та нестандартних діях користувача. Результати тестування підтвердили, що система адекватно реагує на подібні випадки, своєчасно виявляючи помилки та надаючи зрозумілий зворотній зв'язок.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано додаток для створення аудіоісторій із використанням синтезу мовлення. Проведено аналіз предметної області, зокрема сучасних рішень у сфері генерації мовлення, а також визначено ключові вимоги до функціональності системи, з огляду на потреби користувачів і технологічні можливості.

У процесі дослідження була розроблена архітектура вебзастосунку, обрано відповідні технології реалізації, серед яких Laravel для бекенду, Livewire для інтерактивного інтерфейсу та FastAPI як мікросервіс для генерації аудіо. Проєктування платформи охоплювало аналіз варіантів використання системи, моделювання структури та взаємодії компонентів, а також створення бази даних для зберігання інформації про аудіоісторії, користувачів та їхні дії.

Розроблено зручний та інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам створювати текстові історії, обирати персонажів, налаштовувати голосові параметри та отримувати згенеровані аудіофайли. Описано послідовність роботи користувача з додатком, включаючи створення проєктів, взаємодію з голосовими персонажами, збереження та експортом аудіо.

Таким чином, розроблений додаток поєднує сучасні технології синтезу мовлення та зручність використання, забезпечуючи ефективний інструмент для створення персоналізованих аудіоісторій відповідно до потреб користувачів.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Laravel Framework – Laravel Documentation [Електронний ресурс]. – Режим доступу: <https://laravel.com/docs>.
2. FastAPI Web Framework – FastAPI Documentation [Електронний ресурс]. – Режим доступу: <https://fastapi.tiangolo.com>.
3. Livewire Documentation – Building Dynamic Interfaces with Laravel Livewire [Електронний ресурс]. – Режим доступу: <https://livewire.laravel.com>.
4. ESPnet-TTS – End-to-End Speech Processing Toolkit [Електронний ресурс]. – Режим доступу: <https://github.com/espnet/espnet>.
5. OpenAI – Speech Synthesis and Audio Models Documentation [Електронний ресурс]. – Режим доступу: <https://platform.openai.com/docs/guides/text-to-speech>.
6. Flux UI – UI Components for Laravel Volt & Livewire [Електронний ресурс]. – Режим доступу: <https://volthq.dev/docs/flux>.
7. Hugging Face – Models for Text-to-Speech & Speech Processing [Електронний ресурс]. – Режим доступу: https://huggingface.co/models?pipeline_tag=text-to-speech.
8. Hugging Face – KokoroTTS [Електронний ресурс]. – Режим доступу: <https://huggingface.co/hexgrad/Kokoro-82M>.
9. Hugging Face – Ukrainian-TTS [Електронний ресурс]. – Режим доступу: <https://huggingface.co/spaces/robinhad/ukrainian-tts>.
10. Alpine.js Documentation – A Minimal Framework for Composable JavaScript Behavior [Електронний ресурс]. – Режим доступу: <https://alpinejs.dev>.
11. Tailwind CSS Documentation – A utility-first CSS framework for rapidly building custom designs [Електронний ресурс]. – Режим доступу: <https://tailwindcss.com/docs>.
12. WaveSurfer.js Documentation – Audio waveform visualization and audio player library [Електронний ресурс]. – Режим доступу: <https://wavesurfer-js.org>.

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Ліневич Д.Т.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Власенко О. В.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

Авторизація та реєстрація:

```
<?php
```

```
use Illuminate\Auth\Events\Logout;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\RateLimiter;
use Illuminate\Support\Facades\Route;
use Illuminate\Support\Facades\Session;
use Illuminate\Support\Str;
use Illuminate\Validation\ValidationException;
use Livewire\Attributes\Layout;
use Livewire\Attributes\Validate;
use Livewire\Volt\Component;

new #[Layout('components.layouts.auth')] class extends Component {
    #[Validate('required|string|email')]
    public string $email = "";

    #[Validate('required|string')]
    public string $password = "";

    public bool $remember = false;

    /**
     * Handle an incoming authentication request.
     */
    public function login(): void
    {
        $this->validate();

        $this->ensureIsNotRateLimited();
    }
}
```

```

        if (! Auth::attempt(['email' => $this->email, 'password' => $this->password], $this->remember)) {
            RateLimiter::hit($this->throttleKey());

            throw ValidationException::withMessages([
                'email' => __('auth.failed'),
            ]);
        }

        RateLimiter::clear($this->throttleKey());
        Session::regenerate();

        $this->redirectIntended(default: route('dashboard', absolute: false), navigate: true);
    }

    /**
     * Ensure the authentication request is not rate limited.
     */
    protected function ensureIsNotRateLimited(): void
    {
        if (! RateLimiter::tooManyAttempts($this->throttleKey(), 5)) {
            return;
        }

        event(new Lockout(request()));

        $seconds = RateLimiter::availableIn($this->throttleKey());

        throw ValidationException::withMessages([
            'email' => __('auth.throttle', [
                'seconds' => $seconds,
                'minutes' => ceil($seconds / 60),
            ]),
        ]);
    }
}

```

```

/**
 * Get the authentication rate limiting throttle key.
 */
protected function throttleKey(): string
{
    return Str::transliterate(Str::lower($this->email).'.'.request()->ip());
}
}; ?>

```

```

<div class="flex flex-col gap-6">

```

```

    <x-auth-header :title="__('messages.LoginTitle')"
:description="__('messages.LoginDescription')" />

```

```

    <x-auth-session-status class="text-center" :status="session('status')" />

```

```

<form wire:submit="login" class="flex flex-col gap-6">

```

```

    <flux:input
        wire:model="email"
        :label="__('form.Email')"
        type="email"
        required
        autofocus
        autocomplete="email"
        placeholder="email@example.com"
    />

```

```

<!-- Password -->

```

```

<div class="relative">

```

```

    <flux:input
        wire:model="password"
        :label="__('form.Password')"
        type="password"
        required
        autocomplete="current-password"
    />

```

```

        :placeholder="__('form.Password')"
        viewable
    />

    {{!--@if (Route::has('password.request'))
        <flux:link class="absolute end-0 top-0 text-sm" :href="route('password.request')"
wire:navigate>
            {{ __('Forgot your password?') }}
        </flux:link>
    @endif--}}
</div>

<!-- Remember Me -->
{{!--<flux:checkbox wire:model="remember" :label="__('Remember me')" />--}}

<div class="flex items-center justify-end">
    <flux:button variant="primary" type="submit" class="w-full">{{ __('messages.LogIn')
}}</flux:button>
</div>
</form>

@if (Route::has('register'))
    <div class="space-x-1 rtl:space-x-reverse text-center text-sm text-zinc-600 dark:text-zinc-
400">
        {{ __('messages.NoAccount') }}
        <flux:link :href="route('register')" wire:navigate>{{ __('messages.SignUp') }}</flux:link>
    </div>
@endif
</div>
<?php

```

```

use App\Models\User;
use Illuminate\Auth\Events\Registered;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;

```

```
use Illuminate\Validation\Rules;
use Livewire\Attributes\Layout;
use Livewire\Volt\Component;
```

```
new #[Layout('components.layouts.auth')] class extends Component {
    public string $name = "";
    public string $email = "";
    public string $password = "";
    public string $password_confirmation = "";

    /**
     * Handle an incoming registration request.
     */
    public function register(): void
    {
        $validated = $this->validate([
            'name' => ['required', 'string', 'max:255'],
            'email' => ['required', 'string', 'lowercase', 'email', 'max:255', 'unique:' . User::class],
            'password' => ['required', 'string', 'confirmed', Rules\Password::defaults()],
        ]);

        $validated['password'] = Hash::make($validated['password']);

        event(new Registered(($user = User::create($validated))));

        Auth::login($user);

        $this->redirectIntended(route('dashboard', absolute: false), navigate: true);
    }
}; ?>
```

```
<div class="flex flex-col gap-6">
    <x-auth-header :title="__('messages.RegisterTitle')"
    :description="__('messages.RegisterDescription')" />
```

```
<!-- Session Status -->
```

```
<x-auth-session-status class="text-center" :status="session('status')" />
```

```
<form wire:submit="register" class="flex flex-col gap-6">
```

```
  <!-- Name -->
```

```
  <flux:input
    wire:model="name"
    :label="__('form.Name')"
    type="text"
    required
    autofocus
    autocomplete="name"
    :placeholder="__('form.NamePlaceholder')"
  />
```

```
  <!-- Email Address -->
```

```
  <flux:input
    wire:model="email"
    :label="__('form.Email')"
    type="email"
    required
    autocomplete="email"
    placeholder="email@example.com"
  />
```

```
  <!-- Password -->
```

```
  <flux:input
    wire:model="password"
    :label="__('form.Password')"
    type="password"
    required
    autocomplete="new-password"
    :placeholder="__('form.Password')"
    viewable
  />
```

```
/>
```

```
<!-- Confirm Password -->
```

```
<flux:input
    wire:model="password_confirmation"
    :label="__('form.ConfirmPassword')"
    type="password"
    required
    autocomplete="new-password"
    :placeholder="__('form.ConfirmPassword')"
    viewable
/>
```

```
<div class="flex items-center justify-end">
    <flux:button type="submit" variant="primary" class="w-full">
        {{ __('messages.CreateAccount') }}
    </flux:button>
</div>
</form>
```

```
<div class="space-x-1 rtl:space-x-reverse text-center text-sm text-zinc-600 dark:text-zinc-400">
    {{ __('messages.HaveAccount') }}
    <flux:link :href="route('login')" wire:navigate>{{ __('messages.LogIn') }}</flux:link>
</div>
</div>
```

Створення простої історії:

```
<?php
```

```
namespace App\Livewire;
```

```
use Livewire\Component;
use App\Models\Language;
use App\Models\Character;
use App\Models\TTSTModel;
```

```
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Storage;
```

```
class CreateSingleAudioForm extends Component
```

```
{
```

```
    public $text = "";
```

```
    public $audiodialog = null;
```

```
    public $audioPath = "";
```

```
    public $character_id = "";
```

```
    public $language_id = "";
```

```
    public $languages = [];
```

```
    public $characters = [];
```

```
    public function mount($audiodialog, $audio)
```

```
    {
```

```
        $this->audiodialog = $audiodialog;
```

```
        if (!empty($audio)){
```

```
            $this->text = $audio["text"];
```

```
            $this->audioPath = $audio["path"];
```

```
            $character = Character::where(["value"=>$audio["character"]])->first();
```

```
            $this->language_id = $character->language_id;
```

```
            $this->setCharacters();
```

```
            $this->character_id = $character->value;
```

```
        }
```

```
        try{
```

```
            /*$response = Http::timeout(10)->retry(2, 30)->get(config('services.languages.endpoint'));
```

```
            if ($response->successful()) {
```

```
                $this->languages = $response->json();
```

```
            } else {
```

```
                $this->languages = [];
```

```
            }*/
```

```
            $this->languages = Language::all();
```

```
        }catch(Exception $e){
```

```

        Log::error('Language fetch error', ['error' => $e->getMessage()]);
        $this->languages = [];
    }

}

public function setCharacters(){

    /*$url = str_replace(
        'id',
        $this->language_id,
        config('services.characters.endpoint')
    );*/

    try{
        /*$response = Http::timeout(10)->retry(2, 30)->get($url);
        if ($response->successful()) {
            $this->characters = $response->json();
        } else {
            $this->characters = [];
        }*/

        $this->characters = Character::where(['language_id'=>$this->language_id])->get();
    }catch(\Exception $e){
        Log::error('Language fetch error', ['error' => $e->getMessage()]);
        $this->characters = [];
    }
}

public function fetchLanguageData($languageId)
{
    $url = str_replace(
        'id',
        $languageId,
        config('services.languages.endpoint')

```

```
);

return Http::retry(2, 100)
    ->timeout(10)
    ->get($url);
}
```

```
protected function rules()
{
    return [
        'text' => 'required|string|min:1',
        'character_id' => 'required',
        'audiolog.id' => 'required',
        'audiolog.name' => 'required|string',
        'audiolog.user_id' => 'required',
        'audiolog.path' => 'required|string',
    ];
}
```

```
public function generate()
{
    $this->validate();

    $character = Character::where(["value"=>$this->character_id])->first();
    $model = TTSMModel::get_slug_by_id($character->model_id);

    $payload = [
        'character' => $this->character_id,
        'text' => $this->text,
        'model'=>$model,
        'audiolog' => [
            'id' => (string)$this->audiolog['id'],
            'name' => $this->audiolog['name'],
            'user_id' => (string)$this->audiolog['user_id'],
        ],
    ];
}
```

```

        'path' => $this->audiolog['path'],
    ],
];

$response = Http::timeout(120)->retry(2, 180)->post(config('services.generate.endpoint'),
$payload);
if ($response->successful()){

    $contentDisposition = $response->header('Content-Disposition');
    $filename = 'default.wav';

    if ($contentDisposition && preg_match('/filename="?([^"]+)"?/', $contentDisposition,
$matches)) {
        $filename = $matches[1];
    }

    $stream = $response->toPsrResponse()->getBody();
    $final_path = "user-{$this->audiolog->user_id}/audiolog-{$this->audiolog-
>id}/".$filename;
    if (Storage::disk('public')->exists($final_path)){
        Storage::disk('public')->delete($final_path);
    }
    Storage::disk('public')->put($final_path, $stream);
    $this->dispatch("updateAudio", path:$final_path);

    return back();
}

return back()->with('error', 'Audio generation failed.');
```

```

}

public function render()
{
    return view('livewire.create-single-audio-form');
}

```

```
}  
}
```

Створення складної історії:

```
<?php
```

```
namespace App\Livewire;
```

```
use Livewire\Component;
```

```
use App\Models\Audiolog;
```

```
use Illuminate\Support\Facades\Http;
```

```
use Illuminate\Support\Facades\Storage;
```

```
use Livewire\Attributes\On;
```

```
class CreateMultipleAudioForm extends Component
```

```
{
```

```
    public $audios = [];
```

```
    public $audiolog_id = 0;
```

```
    public $audioPath = '';
```

```
    public $position = 0;
```

```
    public $isChanged = false;
```

```
    public $changed_audios = [];
```

```
    public $added_audios = [];
```

```
    public $selected_audios = [];
```

```
    public $save_loading = false;
```

```
    public $selected_loading = false;
```

```
    public $listeners = ['updateList', 'getAudio', 'updateChanged', 'updateChecked', 'deleteAudioItem',  
    'setAudioPath'];
```

```
    public function mount($audios, $audiolog_id, $audioPath = '')
```

```
    {
```

```
        $this->audios = $audios;
```

```
        $this->audiolog_id = $audiolog_id;
```

```
        $this->audioPath = $audioPath;
```

```
        if (!empty($this->audios)) {
```

```

        $this->resetPositions();
    } else {
        $this->audios[] = ["text" => "", "character" => "", "position" => $this->position++];
    }
}

```

```

public function resetPositions()
{
    $this->position = 0;
    foreach ($this->audios as &$audio) {
        $audio['position'] = $this->position++;
    }
}

```

```

public function addAudio()
{
    $current_position = $this->position++;
    $this->audios[] = ["text" => "", "character" => "", "position" => $current_position];
    $this->added_audios[] = $current_position;
    $this->isChanged = true;
}

```

```

public function updateList($position)
{
    $this->audios = array_filter($this->audios, function($audio) use ($position) {
        return $audio['position'] != $position;
    });
}

```

```

    $this->changed_audios = array_diff($this->changed_audios, [$position]);
    $this->added_audios = array_diff($this->added_audios, [$position]);
    $this->selected_audios = array_diff($this->selected_audios, [$position]);
}

```

```

public function getAudio($position){

```

```

foreach($this->audios as $audio){
    if ($audio["position"] == $position){
        return $audio;
    }
}
return null;
}

```

```

public function updateChanged($state, $position = null){
    if ($position != null){
        $this->changed_audios[] = $position;
    }
    $this->isChanged = $state;
}

```

```

public function updateChecked($position, $add){
    if ($add == true){
        $this->selected_audios[] = $position;
    } else {
        $this->selected_audios = array_diff($this->selected_audios, [$position]);
    }
}

```

```

public function updateSelected(){
    $this->selected_loading = true;
    if (!empty($this->selected_audios)){
        foreach ($this->selected_audios as $position) {
            $this->dispatch("call-update-{$position}", add:false);
            $this->dispatch("unset-checked-{$position}");
        }
        $this->selected_audios = [];
    }
}

```

```

public function deleteAudioItem($position){
    $this->audios = array_filter($this->audios, function ($audio) use ($position) {
        return $audio['position'] !== $position;
    });

    $this->audios = array_values($this->audios);

    $this->dispatch("call-delete-{$position}", add:false);
}

```

```

public function deleteSelected(){
    if (!empty($this->selected_audios)){
        foreach ($this->selected_audios as $position) {
            $this->deleteAudioItem($position);
        }
        $this->selected_audios = [];
    }
}

```

```

public function setAudioPath($position, $path){
    foreach ($this->audios as &$audio) {
        if ($audio['position'] == $position){
            $audio['audio_path'] = $path;
            break;
        }
    }
}

```

```

#[On('stop-saving-loading')]
public function stopSavingLoader(){
    $this->save_loading = false;
}

```

```

#[On('stop-selected-loading')]
public function stopSelectedLoader(){
    $this->selected_loading = false;
}

public function save(){
    $this->save_loading = true;
    if (!empty($this->added_audios)){
        foreach ($this->added_audios as $position) {
            $this->dispatch("call-update-{$position}", add:true);
        }
        $this->changed_audios = array_diff($this->changed_audios, $this->added_audios);
        $this->added_audios = [];
    }
    if (!empty($this->changed_audios)){
        $unique_changes = array_unique($this->changed_audios);
        foreach ($unique_changes as $position) {
            $this->dispatch("call-update-{$position}", add:false);
        }
        $this->changed_audios = [];
    }
    $this->isChanged = false;
}

public function generate_whole_audio(){

    $audios_name_array = array_column($this->audios, 'audio_path');

    $audiolog = Audiolog::where(["id"=>$this->audiolog_id])->first();

    $generate_url = str_replace(':path', $audiolog->path, config('services.generate-
all.endpoint'));
    $response = Http::timeout(60)->post($generate_url, $audios_name_array);
    if ($response->successful()){

```

```

$contentDisposition = $response->header('Content-Disposition');
$filename = 'default.wav';

if ($contentDisposition && preg_match('/filename="?([^"]+)"?/', $contentDisposition,
$matches)) {
    $filename = $matches[1];
}

$stream = $response->toPsrResponse()->getBody();

$final_path = "user-{$audiodialog->user_id}/audiodialog-{$audiodialog->id}/".now()-
>timestamp.$filename;

if (Storage::disk('public')->exists($final_path)){
    Storage::disk('public')->delete($final_path);
}

Storage::disk('public')->put($final_path, $stream);
$this->audioPath = $final_path;
$this->dispatch('updateAudio', path: $final_path);
$this->dispatch('result-audio-updated');

return back();
}
}

public function render()
{
    return view('livewire.create-multiple-audio-form', [
        'audios' => $this->audios
    ]);
}
}

```

Компонент репліки:

<?php

namespace App\Livewire;

```
use Livewire\Component;
use App\Models\Language;
use App\Models\Character;
use App\Models\Audiolog;
use App\Models\TTSModel;
use Illuminate\Support\Facades\Http;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Storage;
use Livewire\Attributes\On;
```

```
class MessageForm extends Component
```

```
{
    public $languages = [];
    public $language_id = "";
    public $characters = [];
    public $character = "";
    public $isChecked = false;
    public $text = "";
    public $position = "";
    public $audiolog_id = 0;
    public $audio_id = "";
    public $audio_path = "";
    public $isPlaying = false;
    public $currentlyPlayingId = null;
    public $isChanged = false;
    public $message_loading = false;
```

```
public function mount($position, $text, $character, $audiolog_id, $audio_path, $audio_id){
    $this->position = $position;
    $this->character = $character;
    $this->text = $text;
    $this->audiolog_id = $audiolog_id;
    $this->audio_path = $audio_path;
```

```

$this->audio_id = $audio_id;

if (!empty($this->character)) {
    $character = Character::where("value", $this->character)->first();
    if ($character) {
        $this->language_id = $character->language_id;
        $this->setCharacters();
        $this->character = $character->value;
    }
}

$this->languages = Language::all();
}

public function setCharacters(){
    try{
        $this->characters = Character::where(['language_id'=>$this->language_id])->get();
    }catch(Exception $e){
        Log::error('Language fetch error', ['error' => $e->getMessage()]);
        $this->characters = [];
    }
}

public function setChanged($state){
    $this->isChanged = $state;
    if ($state == true){
        $this->dispatch("updateChanged", state:$state, position:$this->position);
    } else {
        $this->dispatch("updateChanged", state:$state);
    }
}

public function setChecked(){

```

```

    if ($this->isChecked == false){
        $this->dispatch('updateChecked', position:$this->position, add:false);
    } else {
        $this->dispatch('updateChecked', position:$this->position, add:true);
    }

}

public function moveUp(){
    //$this->dispatch('updateList', position:$this->audio["position"], direction:1);
}

public function moveDown(){
    //$this->dispatch('updateList', position:$this->audio["position"], direction:-1);
}

protected function rules()
{
    return [
        'text' => 'required|string|min:1',
        'character' => 'required',
        'audiolog_id' => 'required',
    ];
}

public function handlePlayButton($audio_id){
    if ($this->audio_path == ""){
        $this->generate();
    } else {
        if( $this->isChanged){
            $this->update_audio();
        }
        $this->play($audio_id);
    }
}

```

```
}  
}
```

```
public function update_audio(){  
    $this->validate();  
    $audiolog = Audiolog::where(["id"=>$this->audiolog_id])->first();  
    $character = Character::where(["value"=>$this->character])->first();  
    $model = TTSModel::get_slug_by_id($character->model_id);  
  
    $payload = [  
        'character' => $this->character,  
        'text' => $this->text,  
        'model' => $model,  
        'audiolog' => [  
            'id' => (string)$audiolog->id,  
            'name' => $audiolog->name,  
            'user_id' => (string)$audiolog->user_id,  
            'path' => $audiolog->path,  
        ],  
    ];  
    $update_url = str_replace(':id', $this->audio_id, config('services.audio-update.endpoint'));  
    $response = Http::timeout(120)->put($update_url, $payload);  
    if ($response->successful()){  
  
        $contentDisposition = $response->header('Content-Disposition');  
        $filename = 'default.wav';  
  
        if ($contentDisposition && preg_match('/filename="?([^"]+)"?/', $contentDisposition,  
$matches)) {  
            $filename = $matches[1];  
        }  
  
        $stream = $response->toPsrResponse()->getBody();  
        $final_path = "user-{$audiolog->user_id}/audiolog-{$audiolog->id}/".$filename;
```

```

        if (Storage::disk('public')->exists($final_path)){
            Storage::disk('public')->delete($final_path);
        }
        Storage::disk('public')->put($final_path, $stream);
        $this->audio_path = $final_path;
        $this->dispatch('audio-generated', position: $this->position);
        $this->setChanged(false);
        $this->message_loading = false;

        return back();
    }

}

#[On('call-delete-{position}')]
public function process_deletion(){
    if ($this->audio_path != ""){
        $delete_url = str_replace(':path', $this->audio_path, config('services.audio-
delete.endpoint'));
        $response = Http::timeout(120)->delete($delete_url);

        if ($response->successful()){
            if (Storage::disk('public')->exists($this->audio_path)){
                Storage::disk('public')->delete($this->audio_path);
            }
        }
    }
}

public function delete_audio(){
    $this->dispatch('deleteAudioItem', position:$this->position);
}

#[On('call-update-{position}')]
public function updateFromSaving($add){
    if ($add == true){

```

```

        $this->generate();
    } else {
        $this->update_audio();
    }

    $this->dispatch('stop-saving-loading');
    $this->dispatch('stop-selected-loading');

}

#[On('unset-checked-{position}')]
public function unsetSelected(){
    $this->isChecked = false;
}

protected function generate(){

    $this->validate();
    $audiolog = Audiolog::where(["id"=>$this->audiolog_id])->first();
    $character = Character::where(["value"=>$this->character])->first();
    $model = TTSModel::get_slug_by_id($character->model_id);
    $payload = [
        'character' => $this->character,
        'text' => $this->text,
        'model' => $model,
        'audiolog' => [
            'id' => (string)$audiolog->id,
            'name' => $audiolog->name,
            'user_id' => (string)$audiolog->user_id,
            'path' => $audiolog->path,
        ],
    ];

    $response = Http::timeout(120)->post(config('services.generate.endpoint'), $payload);
    if ($response->successful()){
        $contentType = $response->header('Content-Type');
    }
}

```

```

$filename = 'default.wav';

if ($contentDisposition && preg_match('/filename="?([^"]+)"?/', $contentDisposition,
$matches)) {
    $filename = $matches[1];
}

$stream = $response->toPsrResponse()->getBody();
$final_path = "user-{$audiodialog->user_id}/audiodialog-{$audiodialog->id}/".$filename;
if (Storage::disk('public')->exists($final_path)){
    Storage::disk('public')->delete($final_path);
}
Storage::disk('public')->put($final_path, $stream);
$this->dispatch('setAudioPath', position:$this->position, path:$final_path);
$this->audio_path = $final_path;
$this->dispatch('audio-generated', position: $this->position);
$this->isChanged = false;
return back();
}

return back()->with('error', 'Audio generation failed.');
```

```

}

public function play($audioId)
{
    $this->dispatch('play-audio',
        audioId: $audioId
    );

    $this->currentlyPlayingId = $audioId;
    $this->isPlaying = true;
}

public function handleAudioEnded()
{
    $this->isPlaying = false;
    $this->currentlyPlayingId = null;
}

```

```

        $this->dispatch('audio-ended');
    }
    public function save(){
        $this->isChanged = false;
    }

    public function render()
    {
        $character = Character::where(["value"=>$this->character])->first();

        return view('livewire.message-form', ['character_full'=>$character ?? '']);
    }
}

```

Додаток В

Основний маршрутизатор запитів:

```

from fastapi import FastAPI, HTTPException, UploadFile, Body
from fastapi.responses import StreamingResponse, FileResponse
from typing import List, Dict
import uvicorn

import os
from dotenv import load_dotenv

from models import AudioDialogClass, AudioRequest
from services import StorageService, AudioDialogService, AudioService, ConnectionsChecker,
DatabaseService

app = FastAPI()

@app.get("/characters/{lang_id}")
def get_characters(lang_id:str):

```

```
database = DatabaseService("characters")
return database.find_many_from_collection({"language_id":lang_id})
```

```
@app.get("/audios/{audiolog_id}/info")
def get_audios_info_by_audiolog_id(audiolog_id):
    audialogs_service = AudioDialogService()
    return audialogs_service.get_audios(audiolog_id)
```

```
@app.get("/audio/{audio_path:path}/download")
def get_audios_by_audiolog_id(audio_path):
    print(f"Requested audio path: {audio_path}")
    audialogs_service = AudioDialogService()
    file_obj = audialogs_service.get_audio_file(audio_path)
    if file_obj:
        file_name = audio_path.split("/")[-1]
        return StreamingResponse(
            file_obj,
            media_type="audio/wav",
            headers={
                "Content-Disposition": f'inline; filename="{file_name}"'
            }
        )
    raise HTTPException(status_code=404, detail="Audio file not found")
```

```
@app.get("/audios/{prefix_path:path}/download")
def get_multiple_audios(prefix_path: str):
    print(f"Requested prefix: {prefix_path}")
    audialog_name = prefix_path.split('/')[-1]

    audialogs_service = AudioDialogService()
    zip_buffer, err = audialogs_service.get_audio_files(prefix_path)

    if err != None:
        raise HTTPException(status_code=404, detail="No audio files found")
```

```

return StreamingResponse(
    zip_buffer,
    media_type="application/zip",
    headers={
        "Content-Disposition": f"inline; filename={audiodialog_name}-audios.zip"
    }
)

```

```

@app.get("/file/{audio_path:path}")
def get_file(audio_path):
    print(f"Audio path: {audio_path}")
    audiodialogs_service = AudioDialogService()
    file_obj = audiodialogs_service.get_audio_file(audio_path)
    if file_obj:
        return {"message": "file exists"}
    else: return {"message": "file not found"}

```

```

@app.post("/add")
def create_audio(audio:AudioRequest):
    audiodialogs_service = AudioDialogService()
    file, filename, err = audiodialogs_service.add_audio(audio)

    if err != None:
        raise HTTPException(status_code=400, detail=err["error"])

    headers = {
        "Content-Disposition": f"inline; filename={filename}",
        "Content-Type": "audio/wav",
        "Access-Control-Expose-Headers": "Content-Disposition"
    }

```

```
return StreamingResponse( file,
    media_type="audio/wav",
    headers=headers)
```

```
@app.put('/audio/{id}')
def update_audio(id, audio:AudioRequest):
    audiodialogs_service = AudioDialogService()
    file, filename, err = audiodialogs_service.add_audio(audio, id)

    if err != None:
        raise HTTPException(status_code=400, detail=err["error"])

    headers = {
        "Content-Disposition": f"inline; filename={filename}",
        "Content-Type": "audio/wav",
        "Access-Control-Expose-Headers": "Content-Disposition"
    }

    return StreamingResponse( file,
        media_type="audio/wav",
        headers=headers)
```

```
@app.delete('/audio/{audio_path:path}')
def delete_audio(audio_path):
    print("Delete path:", audio_path)
    audiodialogs_service = AudioDialogService()
    audiodialogs_service.delete_audio(audio_path)
```

```
@app.post('/generate/{prefix_path:path}')
def generate_all_audios(prefix_path:str, ref_array:List[str] = Body(...)):
    print(f'generation all audios started with path: {prefix_path}')
    print(ref_array)
```

```
audiodialogs_service = AudioDialogService()
audio_bytes, err = audiodialogs_service.get_audio_files(prefix_path, ref_array)
print('sorted')
audio_service = AudioService()
merged_audio, audio_name, err = audio_service.merge_audios(audio_bytes, prefix_path)
if err != None:
    raise HTTPException(status_code=404, detail=err["err"])
```

```
headers = {
    "Content-Disposition": f"inline; filename={audio_name}",
    "Content-Type": "audio/wav",
    "Access-Control-Expose-Headers": "Content-Disposition"
}
```

```
return StreamingResponse( merged_audio,
    media_type="audio/wav",
    headers=headers)
```

```
return StreamingResponse( audio,
    media_type=f"audio/{audio_format}",
    headers=headers)
```

```
load_dotenv()
```

```
host = os.getenv("HOST")
port = int(os.getenv("PORT"))
```

```
connection_checker = ConnectionsChecker()
```

```
if __name__ == "__main__":
    connection_checker.database_connection()
```

```
connection_checker.minio_connection()
uvicorn.run("main:app", host=host, port=port, reload=True)
```

Сервіс аудіоісторій:

```
from models import AudioRequest, AudioFull
from services import DatabaseService, AudioService, StorageService, Saver
from services.saving_services import MongoDBSaver, FolderSaver, MinioSaver
from typing import Dict
```

```
from utils import get_dict_for_mongo
import uuid
import io
import zipfile
```

```
import traceback
```

```
class AudioDialogService:
```

```
    def __init__(self):
        self.collection = "audiodialogs"
```

```
    def _create_relative_path(self, user_id, name):
        return f"user-{user_id}/{name}"
```

```
    def add_audio(self, audiorequest:AudioRequest, id:str = "")->Dict[str, str]:
        collection = 'audios'
        print("start generating")
        try:
            audio_service = AudioService()
            audioclass = audio_service.create_audio(audiorequest, id)

            audiodialog_path = audiorequest.audiodialog.path
            audio_path = f"{audiodialog_path}/{audioclass.name}"
```

```
data = AudioFull(id=audioclass.id,
                 name=audioclass.name,
                 audiodialog=audiorequest.audiodialog,
                 character=audiorequest.character,
                 text=audiorequest.text,
                 audio_path=audio_path,
                 audio_bytes=audioclass.audio_bytes,
                 size=audioclass.size,
                 model=audiorequest.model,
                 generation_time=audioclass.generation_time)
```

```
saver = Saver()
minio, mongodb = MinioSaver(), MongoDBSaver(collection)
#folder = FolderSaver()
saver.attach(mongodb)
saver.attach(minio)
print("start saving")
error = saver.saveAll(data)
if error != None:
    return None, "", {"error" : str(error)}
saver.detach_all()
```

```
data.audio_bytes.seek(0)
return data.audio_bytes, data.name, None
```

```
except Exception as e:
    print("Error in add_audio:")
    traceback.print_exc()
    return None, None, {"error": str(e)}
```

```
def delete_audio(self, audio_path:str):
    database = DatabaseService("audios")
    id = self._extract_id_from_path(audio_path)
    print("before database works")
```

```

database.delete_one_by_id(id)

storage = StorageService()

print("before minio works")

storage.delete_file(audio_path)


def get_audios(self, audiodialog_id):
    database = DatabaseService("audios")
    return database.find_many_from_collection({"audiodialog_id":audiodialog_id})


def get_audio_file(self, audio_path):
    storage = StorageService()
    return storage.get_file(audio_path)


def get_audio_files(self, prefix_path, ref_array=[]):
    storage = StorageService()

    audio_files = storage.get_files(prefix_path)

    if not audio_files:
        return None, {"error":"No audio files found"}

    if ref_array:
        audio_map = {}
        for file in audio_files:
            filename = file.object_name
            if 'result' in filename.lower():
                continue
            file_data = storage.get_file(filename)
            if file_data:
                content = file_data.read()
                audio_map[filename] = content

        print("audio_map keys:", list(audio_map.keys()))

```

```

audio_bytes = []
for ref_filename in ref_array:
    if ref_filename in audio_map:
        print(f'exists: {ref_filename}')
        audio_bytes.append(audio_map[ref_filename])
    else:
        return None, {"error": f"Referenced file '{ref_filename}' not found in storage"}

return audio_bytes, None

```

```

zip_buffer = io.BytesIO()
with zipfile.ZipFile(zip_buffer, "w", zipfile.ZIP_DEFLATED) as zip_file:
    for file in audio_files:
        file_data = storage.get_file(file.object_name)
        if file_data:
            content = file_data.read()
            filename = file.object_name.split("/")[-1]
            zip_file.writestr(filename, content)

zip_buffer.seek(0)

return zip_buffer, None

```

```

def _extract_id_from_path(self, path):
    filename = path.split("/")[-1]
    filename_without_extension = filename.split(".")[0]
    id = filename_without_extension[6:]
    return id

```

Сервіс аудіо:

```

from io import BytesIO
import os
import mimetypes
import soundfile as sf
from pathlib import Path

```

```

from pydub import AudioSegment
from fastapi import HTTPException
from services import DatabaseService, StorageService, AudioGeneratorService,
AudioExportService
from services.generating_services import KokoroAudioGenerator, UkrainianAudioGenerator
from services.export_services import WavExportStrategy, Mp3ExportStrategy, FlacExportStrategy
from models import AudioClass, AudioRequest
from interfaces import IGeneratorType
import uuid

class AudioService:
    MAX_FILE_SIZE = 50 * 1024 * 1024
    FOLDER_PATH = "outputs"
    FILE_NAME = "output"
    FILE_EXTENSION = ".wav"
    FILE_SAMPLERATE = 24000
    ALLOWED_EXTENSIONS = [".mp3", ".wav"]

    def __init__(self):
        os.makedirs(self.FOLDER_PATH, exist_ok=True)
        self.collection = "audios"

    def save_to_minio(self, filename:str, audiodialog_path:str):

        try:
            filepath = os.path.join(self.FOLDER_PATH, filename)
            self._validate_audio_file(filepath)

            with open(filepath, 'rb') as file:
                storage = StorageService()
                minio_path = f"{audiodialog_path}/{filename}"

                content_type = self._get_content_type(filename)
                file_size = os.path.getsize(filepath)
                file_bytes = BytesIO(file.read())

```

```

        storage.upload_file(
            minio_path,
            file_bytes,
            content_type,
            int(file_size)
        )

    return {
        "url":minio_path,
        "filename":filename,
        "size":str(file_size),
        "content_type":content_type
    }

except HTTPException as e:
    raise e

except Exception as e:
    raise HTTPException(status_code=500, detail=str(e))

def _validate_audio_file(self, filepath:str):
    if not os.path.exists(filepath):
        raise HTTPException(status_code=404, detail="File doesn't exist")

    if os.path.getsize(filepath) > self.MAX_FILE_SIZE:
        raise HTTPException(status_code=404, detail="File size exceeds maximum limit of 50MB!")

    if Path(filepath).suffix.lower() not in self.ALLOWED_EXTENSIONS:
        raise HTTPException(status_code=404, detail="Invalid file type!")

def _get_content_type(self, filename:str):
    return mimetypes.guess_type(filename)[0]

def save_to_database(self, audio:dict[str, str | int], audiodialog_path:str):
    try:

```

```

        database = DatabaseService(self.collection)

        audio["path"] = f"{audiodialog_path}/{audio['name']}"
        database.add_to_collection(audio)

    return audio

except Exception as e:
    raise HTTPException(status_code=400, detail=f"Inserting audio to DB failed! Error:{e}")

def create_audio(self, audio:AudioRequest, id="")->AudioClass:

    audio_id = id
    if audio_id == "":
        audio_id = self._make_audio_id(audio.audiodialog.id)

    generator_type = self._define_model_by_slug(audio.model)
    audio_generator = AudioGeneratorService(generator_type)
    return audio_generator.generate_audio(audio, audio_id)

size=audio_size, generation_time=generation_time)

def _define_model_by_slug(self, slug) -> IGeneratorType:
    kokoro = KokoroAudioGenerator()
    ukrainian = UkrainianAudioGenerator()

    models = {'kokoro':kokoro, 'ukr':ukrainian}

    return models[slug]

def merge_audios(self, audio_bytes, path:str):
    try:
        merged_audio = AudioSegment.empty()

```

```

for audio in audio_bytes:
    segment = AudioSegment.from_file(BytesIO(audio), format="wav")
    merged_audio += segment

buffer = BytesIO()
merged_audio.export(buffer, format="wav")
buffer.seek(0)

audiolog_name = path.split('/')[-1]
audiolog_id = audiolog_name.split('-')[-1]
audio_id = self._make_audio_id(audiolog_id)

filename = f"{audiolog_name}-result.wav"
file_path = f"{path}/{filename}"

content_type = mimetypes.guess_type(filename)[0]

file_size = buffer.getbuffer().nbytes
buffer.seek(0)

database = DatabaseService('audios')
if database.exists_by_name(filename):
    database.delete_many_by_name(filename)
    database.add_to_collection({"_id": audio_id, "audio_path": file_path,
"audiolog_id": audiolog_id, "name": filename, "size": file_size})

storage = StorageService()
storage.upload_file(file_path, buffer, content_type, file_size)

buffer.seek(0)

return buffer, filename, None
except Exception as e:
    return None, None, {'err': str(e)}

```

```
def export_audio(self, audio_path, format):  
    export_strategy = self._define_export_strategy(format)  
    export_service = AudioExportService(export_strategy)  
  
    return export_service.export_file(audio_path)  
  
def _define_export_strategy(self, format:str):  
  
    wavStrategy = WavExportStrategy()  
    mp3Strategy = Mp3ExportStrategy()  
    flacStrategy = FlacExportStrategy()  
  
    export_strategies = {  
        'wav':wavStrategy,  
        'mp3':mp3Strategy,  
        'flac':flacStrategy  
    }  
  
    return export_strategies[format]
```