МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему:

«РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ ПЕРЕТВОРЕННЯ ТЕКСТУ В АУДІОМОВЛЕННЯ»

Виконав: студент групи ІТ-62

спеціальності 126 «Інформаційні системи та технології»

Юрчишин Н. Ю.
(Прізвище та ініціали)

Керівник: Смолінський В.Б.
(Прізвище та ініціали)

ДУБЛЯНИ 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Освітній ступінь «Магістр»
Спеціальність 126 «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____
(підпис)
д.т.н., професор, Тригуба А. М.
(вч. звання, прізвище, ініціали)
"___" _____202__ року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Юрчишина Назара Юрійовича
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розробка веб-застосунку для перетворення тексту в аудіомовлення»*

керівник роботи *к. е н., доцент., Смолінський В.Б.*
(наук.ступінь, вч. звання, прізвище, ініціали)

Затверджені наказом по університету 12 вересня 2024 року № 616/к-с.

2. Строк подання студентом роботи 06.12.2024 р.
3. Вихідні дані: *характеристика предметної сфери; вихідні дані та вимоги до роботи програмного застосунку, опис використаних технологій та мов програмування, науково-технічна і довідкова література.*

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)
Вступ
1. Постановка задачі дослідження
2. Теоретичні основи та аналіз існуючих рішень
2. Розробка веб-застосунку .....
3. Програмна реалізація проекту .....
4. Охорона праці та безпека в надзвичайних ситуаціях
Висновки
Список використаних джерел
5. Перелік графічного матеріалу
Рисунки, таблиці

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		Відмітка про виконання
		завдання видав	завдання прийняв	
1, 2, 3, 4	*Смолінський В.Б.*			
5				

7. Дата видачі завдання *12.09.2024 р.*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Відмітка про виконання
1	*Отримання завдання. Вивчення рекомендованої літератури по темі роботи. Написання аналітичного огляду предметної області.*	*12.09.2024 – 03.10.2024*	
2	*Проектування та опис технічного завдання, функціональних вимог та технічної сторони реалізації проекту (написання проектної частини).*	*04.10.2024 – 24.10.2024*	
3	*Програмна реалізація проекту (вибір мови програмування, розробка застосунку, тестування його роботи)*	*25.10.2024 – 20.11.2024*	
4	*Розгляд питань з охорони праці та безпеки у надзвичайних ситуаціях*	*21.11.2024 – 25.11.2024*	
5	*Завершення оформлення основної частини, написання висновків та підготовка презентаційного матеріалу*	*26.11.2024 – 01.12.2024*	
6	*Завершення роботи в цілому. Підготовка до захисту кваліфікаційної роботи*	*02.12.2024 – 06.12.2024*	

Студент _____ *Юрчишин Н.Ю.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Смолінський В.Б.*
(підпис) (прізвище та ініціали)

УДК 004.4, 004.5

Розробка веб-застосунку для перетворення тексту в аудіомовлення. Юрчишин Н.Ю. Кваліфікаційна (магістерська) робота: Кафедра інформаційних технологій. – Дубляни, Львівський НУП, 2024 р.

Кваліфікаційна робота викладена на 60 сторінках, містить 5 розділів, 3 таблиці, 29 рисунків, 17 літературних джерел

У кваліфікаційній роботі успішно розроблено веб-застосунок для ефективного засвоєння навчального матеріалу через озвучування текстів. Застосунок інтегрує сучасні технології синтезу мовлення, забезпечуючи користувачам можливість перетворення текстової інформації в аудіоформат.

У першому розділі обґрунтовано мету створення веб-застосунку для перетворення тексту в аудіомовлення. Основні завдання проекту включають розробку зручного інтерфейсу, інтеграцію TTS-технологій, забезпечення зберігання аудіофайлів, автентифікацію користувачів, а також дотримання безпеки даних. Розробка спрямована на покращення навчального процесу для учнів і викладачів.

Розділ другий охоплює огляд сучасних технологій синтезу мовлення, включаючи формантний, конкатенативний та нейронний підходи. Особлива увага приділена перспективам розвитку TTS, таким як інтеграція з AI, адаптація до різних мов і покращення емоційного синтезу. Проведено аналіз схожих програм (Google Cloud TTS, Amazon Polly тощо) з акцентом на їх переваги та недоліки.

У третьому розділі розглянуто вибір технологій для створення застосунку: React для інтерфейсу, Flask для серверної частини, AWS S3 для зберігання даних та gTTS для синтезу мовлення.

У розділі четвертому описано ключові функції, такі як перетворення тексту в аудіофайли, управління створеними файлами, забезпечення реєстрації, логіну та безпеки користувачів. Представлено механізми роботи із завантаженими файлами, процес тестування системи та усунення помилок для забезпечення стабільності та точності роботи.

У пятому розділі подана охорона праці та безпека у надзвичайних ситуаціях.

Використання веб-застосунку для синтезу мовлення має велику доцільність у навчанні учнів та студентів, оскільки сприяє покращенню засвоєння матеріалу через слухове сприйняття. Аудіоматеріали можна використовувати під час виконання інших завдань, що дозволяє ефективніше використовувати час.

Ключові слова: TTS-технології, React, Flask, AWS S3, синтез мовлення, перетворення тексту, PostgreSQL, gTTS, аудіомовлення, веб-застосунок

ЗМІСТ

ВСТУП

Сучасне суспільство все більше залежить від технологій, які сприяють отриманню та засвоєнню інформації. З розвитком інформаційних технологій, особливо в освітній сфері, виникає потреба у нових методах та інструментах, які допомагають покращити навчальний процес та зробити його більш ефективним. Одним з таких інструментів є веб-застосунки, що використовують технології синтезу мовлення (TTS – Text To Speech) для конвертації текстової інформації у звукову форму. Технології синтезу мовлення мають широке застосування в різних галузях, таких як освіта, медіа, розваги, а також у підтримці людей з обмеженими можливостями.

Актуальність використання TTS у навчальному процесі обумовлена доступністю інформації, покращенням засвоєння матеріалу, можливістю багатозадачності, інтерактивного навчання та підтримкою багатьох мов. TTS технології роблять інформацію доступною для людей з вадами зору, а також для тих, хто має труднощі з читанням. Аудіо матеріали дозволяють студентам краще засвоювати інформацію, оскільки використовується слуховий канал сприйняття, що зменшує навантаження на зоровий канал. Можливість слухати навчальні матеріали під час виконання інших завдань дозволяє ефективніше використовувати час. Веб-застосунки з підтримкою TTS дозволяють створювати інтерактивні навчальні курси, де студенти можуть взаємодіяти з матеріалом у зручний для них спосіб. Сучасні TTS системи підтримують велику кількість мов, що дозволяє використовувати їх у глобальному контексті та для навчання іноземних мов.

Метою даної роботи є створення веб-застосунку, який використовує технології синтезу мовлення (TTS) для покращення ефективності засвоєння навчального матеріалу. Основна ідея полягає в тому, щоб надати студентам можливість прослуховувати навчальні матеріали, що сприятиме кращому сприйняттю інформації, її запам'ятовуванню, а також зробить навчальний процес доступним для осіб із вадами зору чи труднощами у читанні. Для

досягнення цієї мети необхідно провести аналіз сучасних технологій TTS, їхніх можливостей і обмежень, визначити найбільш підходящі рішення для інтеграції у веб-застосунок, а також спроектувати та реалізувати систему, яка зможе забезпечити якісне перетворення тексту на мовлення. Застосунок має бути зручним для використання, підтримувати різні мови та голоси, а також надавати користувачам можливість зберігати результати синтезу для подальшого використання.

Об’єктом дослідження є технології синтезу мовлення, зокрема їхні технічні характеристики, функціональні можливості та обмеження. Увага приділяється вивченню способів їх застосування у різних галузях, зокрема в освітньому процесі, а також аналізу того, як технології TTS можуть сприяти покращенню доступності навчальних матеріалів. Крім того, досліджується процес зберігання та доступу до результатів синтезу мовлення у контексті цифрових платформ, що дозволяє забезпечити швидкий і зручний доступ до згенерованих аудіофайлів.

Предметом дослідження є розробка веб-застосунку, який використовує технології TTS для покращення ефективності навчального процесу. У цьому контексті розглядаються різні системи синтезу мовлення, аналізуються їхні технічні можливості та обмеження, а також вивчаються методи їх інтеграції у веб-застосунки. Особливий акцент робиться на створенні інтуїтивно зрозумілого інтерфейсу, який дозволить користувачам легко взаємодіяти із системою, а також на забезпеченні надійного зберігання і доступу до створених аудіоматеріалів.

РОЗДІЛ 1.
ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

Розробка веб-застосунку для ефективного засвоєння матеріалу передбачає створення системи, яка дозволяє користувачам перетворювати текстову інформацію на мовлення. Це забезпечує зручний спосіб навчання для людей з різними стилями сприйняття інформації та допомагає особам з обмеженими можливостями. Сучасні технології надають можливість інтеграції тексту в мовлення (TTS) у веб-застосунки, що значно розширює їх функціональні можливості.

У процесі роботи над проектом були поставлені наступні завдання:

1) Розробка інтерфейсу користувача. Створити зручний та інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам легко взаємодіяти з функціоналом застосунку.

2) Реалізація функціоналу TTS. Інтегрувати технології синтезу мовлення для перетворення тексту на мовлення.

3) Забезпечення зберігання аудіофайлів. Реалізувати можливість зберігання створених аудіофайлів, забезпечуючи їх легкий доступ для користувачів.

4) Реєстрація та авторизація користувачів. Впровадити систему реєстрації та авторизації користувачів для персоналізації досвіду використання.

5) Забезпечення безпеки даних. Гарантувати безпеку даних користувачів та захист від несанкціонованого доступу.

Розроблений веб-застосунок розрахований на дві основні категорії користувачів:

– Студенти та учні – вони зможуть використовувати застосунок для перетворення текстових матеріалів на аудіо, що сприятиме кращому засвоєнню матеріалу.

– Викладачі - можуть використовувати застосунок для створення аудіоматеріалів для своїх студентів, що зробить процес навчання більш інтерактивним та доступним.

Очікувані результати проєкту полягають у створенні веб-застосунку, що сприятиме підвищенню ефективності навчального процесу завдяки впровадженню технологій синтезу мовлення (TTS). Основним результатом стане можливість студентів та учнів слухати навчальні матеріали, що дозволить їм краще запам’ятовувати інформацію через активне залучення слухового сприйняття. Це особливо важливо для тих, хто віддає перевагу аудіальним методам навчання, а також для людей із вадами зору або труднощами у читанні, для яких застосунок стане необхідним інструментом. Крім того, реалізація такого рішення забезпечить гнучкість у навчанні: користувачі зможуть отримувати доступ до матеріалів у будь-який час, навіть під час виконання інших справ, таких як подорожі чи заняття спортом, що підвищує зручність і ефективність використання ресурсу.

Разом з тим, під час реалізації проєкту очікуються певні обмеження. Технічні обмеження стосуватимуться інтеграції зовнішніх API для синтезу мовлення, які можуть мати встановлені ліміти на кількість запитів або обсяг даних, що створює виклик у забезпеченні стабільної роботи системи при великій кількості користувачів. Фінансові обмеження пов’язані з тим, що використання якісних TTS-сервісів чи інструментів може вимагати значних витрат, зокрема на підписки, ліцензії або серверні ресурси для обробки запитів. Часові обмеження виникають через необхідність розробки, тестування та впровадження всіх функцій веб-застосунку, що потребує ретельного планування та значних ресурсів часу, особливо при створенні якісного інтерфейсу та перевірці стабільності роботи застосунку.

Таким чином, очікується створення універсального інструменту, який поєднуватиме зручність використання, інтеграцію сучасних технологій TTS, а також можливість зберігання й подальшого використання аудіофайлів. Це забезпечить ефективність засвоєння навчального матеріалу, зробить навчальний процес доступнішим і дозволить значно розширити аудиторію користувачів.

РОЗДІЛ 2.
ТЕОРЕТИЧНІ ОСНОВИ ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

2.1 Огляд технологій синтезу мовлення

Технології синтезу мовлення (Text-to-Speech, TTS) дозволяють перетворювати текстові дані на звучне мовлення, що знаходить широке застосування у різних галузях: від освітніх програм і інтерактивних помічників до медіа та розважальних додатків. TTS системи забезпечують зручність доступу до інформації, особливо для людей з обмеженими можливостями, та дозволяють ефективніше використовувати час, поєднуючи навчання або роботу з іншими справами.

Технологія перетворення тексту в мовлення (Text-to-Speech, TTS) є однією з ключових у сфері взаємодії людини з комп'ютером. Вона дозволяє синтезувати мовлення з тексту і знаходить широке застосування у різних галузях: від підтримки користувачів з обмеженими можливостями до інтерактивних помічників і автоматизації роботи з інформацією. Розвиток TTS технологій має довгу історію, яка почалася задовго до появи комп'ютерів, і продовжується досі, інтегруючись із сучасними штучними інтелектами та нейронними мережами.

Перша ідея синтезу людського мовлення виникла ще в XVII столітті. У 1770 році австрійський інженер Вольфганг фон Кемпелен створив “мовний автомат”, який міг імітувати звуки людського мовлення за допомогою механічних компонентів. Цей пристрій складався з повітряного насоса, резонаторів і клапанів, які моделювали голосові зв'язки та рот.

У XIX столітті була створена ще одна знакова конструкція – фонограф Томаса Едісона (1877 рік). Хоча фонограф не був TTS-пристроєм у сучасному розумінні, він став основою для запису й відтворення звуків, що надалі вплинуло на розвиток мовних технологій.

До початку XX століття проводилися численні експерименти з аналізу та синтезу звуків мовлення. Наприклад, в 1930-х роках Гомер Дадлі розробив Voder (Voice Operating Demonstrator), перший електронний пристрій, здатний синтезувати мовлення. Користувач міг управляти пристроєм, створюючи звуки, які нагадували людське мовлення.

У середині XX століття розробка комп'ютерів відкрила нові можливості для автоматизації процесу синтезу мовлення. Перші дослідження у сфері TTS були зосереджені на розробці програм, які могли аналізувати текст і перетворювати його в послідовність звуків. Одним із перших значущих досягнень став DAISY Bell, комп'ютерний голос, синтезований в 1961 році комп'ютером IBM 7094.

Цей експеримент продемонстрував, що комп'ютер може генерувати зрозуміле мовлення. Хоча якість звуку залишалася низькою, цей прорив заклав основу для подальших досліджень.

У 1960-х і 1970-х роках почали розвиватися алгоритми для синтезу мовлення, зокрема, формантний синтез на основі аналізу основних формант (резонансів) звуків мовлення. У той період використовували спеціальний пристрій вокодер, який був призначений для розподілу мовлення на окремі частотні компоненти й реконструюювання їх.

Серед перших застосувань TTS були системи для телефонного обслуговування, які могли озвучувати номери або повідомлення.

У 1980-х роках з’явилися перші комерційні TTS-продукти. Вони базувалися на двох основних підходах: правило-орієнтований синтез (програми аналізували текст, перетворюючи його на фонеми, які потім озвучувалися синтезатором) та записані сегменти (ці системи використовували базу записаних голосів, які комбінувалися для формування слів і речень). Прикладом успішного продукту того часу є DECtalk, розроблений компанією Digital Equipment Corporation у 1984 році. Голоси DECtalk використовувалися навіть для створення музичних композицій і стали голосом Стівена Гокінга.

З розвитком ПК у 1990-х роках TTS отримала значне поширення. Платформи на базі Windows та Mac почали підтримувати базові функції озвучування тексту. Наприклад, Microsoft Speech API (SAPI) забезпечував розробників інструментами для інтеграції TTS у їхні програми.

У 2000-х роках почали використовувати статистичні моделі для покращення якості синтезу мовлення. Одним із найбільш поширених підходів став HMM (Hidden Markov Models). Цей метод дозволяв будувати мовлення на основі аналізу великих обсягів даних, що значно підвищило його природність.

Нейронні мережі. Справжню революцію в TTS спричинило використання нейронних мереж. У 2016 році Google представила WaveNet — глибоку нейронну мережу, яка створює мовлення шляхом генерації звукових хвиль. WaveNet змогла значно перевершити попередні методи за якістю мовлення, наблизивши синтез до природного людського голосу.

Інші компанії, такі як Amazon, Microsoft і IBM, також почали активно використовувати нейронні мережі у своїх TTS-продуктах (наприклад, Alexa, Cortana, Watson).

Інтеграція з AI. Сучасні TTS-системи інтегруються з голосовими асистентами, чат-ботами та іншими технологіями штучного інтелекту. Наприклад, Siri, Google Assistant та Alexa здатні не лише синтезувати мовлення, але й розуміти контекст і адаптувати свій тон до користувача.

Технології синтезу мовлення (TTS) сьогодні знаходять широке застосування у різних сферах. В освіті вони допомагають учням із дислексією чи вадами зору сприймати навчальні матеріали в аудіоформаті, що значно підвищує доступність знань. У мобільних додатках, таких як Google Translate, TTS використовується для озвучування перекладів, забезпечуючи зручність у спілкуванні різними мовами. В автомобілях системи TTS озвучують навігаційні підказки, що покращує безпеку та комфорт водіїв. У сфері розваг ці технології застосовуються для створення голосів у відеоіграх, дубляжу контенту або навіть

у створенні штучних голосів для музичних композицій, що відкриває нові можливості для творчості.

Розвиток технологій TTS – це постійний процес удосконалення, який триває століттями. Від механічних автоматів XVIII століття до сучасних систем на основі штучного інтелекту, TTS залишається важливим інструментом, що змінює спосіб взаємодії людини з машиною.

Технології синтезу мовлення (TTS) знаходять застосування у багатьох галузях, надаючи користувачам зручний доступ до інформації та покращуючи взаємодію з цифровими системами. В освітньому середовищі вони є незамінними для учнів і студентів з порушеннями зору або дислексією, допомагаючи засвоювати навчальні матеріали через аудіоінтерпретацію текстів. Це дозволяє краще запам’ятовувати інформацію завдяки слуховому сприйняттю. У медіа та розважальній сфері TTS широко використовуються для озвучування контенту, створення аудіокниг, інтерактивних ігор та інших видів медіапродукції, значно розширюючи можливості доступу до інформації для різних аудиторій. Для людей із обмеженими можливостями ці технології забезпечують доступ до інформації, дозволяючи їм користуватися комп’ютерами та мобільними пристроями на рівні з іншими, що особливо важливо для осіб із порушеннями зору чи труднощами у читанні. У сфері бізнесу та корпоративних рішень TTS активно застосовується в автоматизованих голосових системах, таких як голосові помічники або контакт-центри, що дозволяє ефективно взаємодіяти з клієнтами та оптимізувати роботу сервісів.

Основними викликами в розробці TTS систем є забезпечення натуральності мовлення, емоційного забарвлення та адаптації до різних мов і діалектів. Сучасні дослідження спрямовані на покращення якості синтезованого мовлення, включаючи: Емоційний синтез: Створення мовлення, яке передає різні емоції, що вимагає більш глибокого розуміння контексту та інтонацій. Адаптація до різних мов: Розробка моделей, які можуть легко адаптуватися до

різних мов і діалектів, забезпечуючи високу якість мовлення незалежно від мови. Покращення алгоритмів нейронних мереж: Використання більш складних моделей та алгоритмів для покращення якості мовлення і зменшення обчислювальних витрат.

З розвитком штучного інтелекту та глибокого навчання TTS технології продовжують еволюціонувати, наближаючись до створення синтетичного мовлення, яке неможливо відрізнити від людського.

2.2 Принципи роботи TTS систем та перспективи їх розвитку

Системи синтезу мовлення (TTS) складаються з кількох ключових етапів, кожен з яких відіграє важливу роль у перетворенні тексту в аудіосигнал. Основні етапи роботи TTS систем включають текстову нормалізацію, токенізацію, визначення фонем, прогнозування просодії та сам синтез мовлення.

Текстова нормалізація є першим кроком у процесі синтезу мовлення. На цьому етапі текст перетворюється з письмової форми у формат, зручний для подальшої обробки.

Цей етап включає наступні роки:

1. Перетворення чисел. Наприклад, "2024" перетворюється на "дві тисячі .двадцять чотири".

2. Розшифровка абревіатур. Наприклад, "NASA" перетворюється на "Національне управління з аеронавтики і дослідження космічного простору".

3. Замінник спеціальних символів. Наприклад, "$" перетворюється на "долар".

Цей етап є критично важливим для забезпечення правильного вимовляння та інтонації синтезованого мовлення.

Токенізація і визначення фонем. Токенізація включає поділ тексту на менші одиниці, зазвичай слова або фрази. Кожне слово потім конвертується у

фонеми – найменші одиниці звуку в мові. Цей процес складається з таких кроків: лексичний аналіз (визначення меж слів і розпізнавання основних граматичних категорій); фонетична транскрипція (конвертація текстових символів у фонетичні символи, які представляють звукові одиниці).

Визначення фонем є важливим етапом, оскільки від точності цього процесу залежить якість та природність синтезованого мовлення.

Прогнозування просодії. Просодія включає ритм, інтонацію і наголос, які надають мовленню природності та виразності. Прогнозування просодії є складним завданням, яке вимагає розуміння контексту і сенсу тексту. До основних аспектів прогнозування просодії можна віднести:

– Інтонація – зміна висоти тону, яка передає питання, вигук або твердження;

– Ритм – часові інтервали між словами та фразами;

– Наголос – виділення певних слів або складів для передачі значення.

Сучасні системи використовують нейронні мережі для прогнозування просодії, що дозволяє створювати більш природне та виразне мовлення.

Останнім етапом є сам синтез мовлення, де фонеми перетворюються в аудіо сигнал. Існує кілька методів синтезу, зокрема:

1) Конкатенативний синтез: Використовує записані фрагменти реального мовлення. Перевагою цього методу є висока якість мовлення для конкретних випадків, однак він менш гнучкий при зміні інтонацій або контексту.

2) Параметричний синтез: Генерує мовлення на основі математичних моделей, які описують характеристики звукових хвиль. Хоча цей метод забезпечує більшу гнучкість у порівнянні з конкатенативним синтезом, він часто поступається в якості звучання.

3) Нейронний синтез: Базується на глибокому навчанні і використовує складні моделі, такі як трансформери, для генерації мовлення. Цей метод дозволяє створювати високоякісне, натуральне мовлення, яке може адаптуватися до різних інтонацій і контекстів. Нейронні мережі навчаються на

великих обсягах даних, що дозволяє їм створювати високоякісні синтетичні голоси.

Формантний синтез є однією з основних технологій синтезу мовлення, яка використовує лінгвістичні та акустичні знання для генерації звуків, що імітують людську мову. Цей метод був розроблений на основі вивчення формантів – резонансних частот, які визначають характер звуків у мовленні. Давайте розглянемо його принципи, переваги та недоліки.

Формантний синтез базується на моделюванні фізичних характеристик мовлення. Нижче розглянемо основні етапи цього процесу.

Аналіз фонем – система, яка визначає фонеми, які складають слова. Для кожної фонеми визначаються параметри формантів – частоти, ширина та рівень.

Генерація звуку використовуючи генератори частот, система створює синтетичний звук, який відповідає заданим параметрам формантів. Цей процес дозволяє контролювати різні аспекти звучання, такі як висота тону, тривалість та амплітуда.

З точки зору технологічної структури формантний синтез може складатися з наступних компонентів:

- Генераторів формантів, які створюють основні звукові хвилі на основі заданих частот;
- Генератору шуму для відтворення приголосних звуків;
- Моделювання мовного тракту – система моделює фізичну структуру людського мовного тракту для досягнення більш природного звучання.

До основних переваг формантного синтезу можна віднести наступне:

– Формантний синтез дозволяє повністю контролювати генеровані сигнали, що забезпечує універсальність у створенні різних типів мовлення (наприклад, дитяче або доросле).

– Низькі вимоги до ресурсів. Ця технологія може бути реалізована на малопотужних системах, що робить її доступною для широкого використання.

Окрім переваг формантного синтезу існують і недоліки. Нижче розглянемо деякі з них:

– Штучність звучання незважаючи на контроль над параметрами, результати часто звучать менш природно порівняно з конкатенативними методами, які використовують записані зразки людської мови.

– Складність налаштування для досягнення якісного звучання може знадобитися значний обсяг налаштувань і оптимізації.

Формантний синтез часто порівнюють з конкатенативним синтезом. Останній використовує фрагменти записаного мовлення для створення нових речень. Хоча конкатенативний синтез може забезпечити більш природне звучання, він має обмеження щодо варіативності і потребує значних обсягів пам'яті для зберігання записаних фрагментів.

Таблиця 2.1 Порівняння формантного та конкатенативного синтезів

Характеристика	**Формантний синтез**	**Конкатенативний синтез**
Природність звучання	Середня	Висока
Контроль параметрів	Високий	Низький
Вимоги до пам'яті	Низькі	Високі
Гнучкість	Висока	Обмежена

Формантний синтез є важливим методом у сфері технологій TTS, який забезпечує контроль над акустичними параметрами та можливість генерації різноманітних звукових сигналів. Хоча він має свої недоліки у вигляді штучності звучання, його переваги роблять його корисним інструментом у багатьох застосуваннях. Подальші дослідження та вдосконалення можуть призвести до покращення якості звучання та більшої натуральності формантного синтезу.

Конкатенативний синтез, також відомий як компілятивний синтез, є одним з найпоширеніших методів синтезу мовлення, який базується на

поєднанні (конкатенації) заздалегідь записаних мовних одиниць для створення нових звукових фрагментів. Цей метод дозволяє генерувати природне звучання, яке близьке до людської мови, завдяки використанню реальних голосових записів.

Принципи роботи конкатенативного синтезу:

1. Запис мовних одиниць. У системах конкатенативного синтезу створюється велика база даних, що містить записи різних мовних одиниць. Це можуть бути фонеми, дифони (пари фонем), трифони (групи з трьох фонем) або навіть цілі слова. Записи повинні бути зроблені в контрольованих умовах, щоб забезпечити високу якість звуку.

2. Вибір одиниць. Під час синтезу тексту система аналізує вхідний текст і визначає, які мовні одиниці необхідно використовувати для його відтворення. На основі контексту та фонетичних правил вибираються найбільш підходящі елементи з бази даних.

3. Конкатенація. Обрані мовні одиниці з'єднуються в один звуковий сигнал. Цей процес може включати обробку для згладжування переходів між елементами, щоб уникнути чутних артефактів на місцях з'єднання.

До основних переваг конкатенативного синтезу можна віднести наступне:

– Природність звучання оскільки цей метод використовує реальні записи людської мови, результати звучать більш природно і виразно в порівнянні з формантним синтезом.

– Гнучкість це конкатенативний синтез може бути адаптований для різних мов і акцентів шляхом додавання нових записів до бази даних.

– Висока якість завдяки використанню записаних фрагментів, якість синтезованого мовлення може бути дуже високою.

Недоліки технології синтезу мовлення можуть включати вимоги до пам'яті, оскільки для зберігання великої кількості мовних одиниць потрібно значну кількість оперативної пам'яті. Це може стати проблемою для пристроїв з обмеженими ресурсами, де обсяг доступної пам'яті є критичним. Іншим важливим

недоліком є шум на місцях з'єднання, коли переходи між фрагментами мовлення не оброблені належним чином. Це може призвести до появи чутних артефактів у звучанні, що погіршує якість сприйняття. Крім того, існує обмеження словника: зміст синтезованого тексту обмежений обсягом бази даних, і якщо в словнику немає потрібної мовної одиниці, система не зможе її відтворити, що обмежує можливості роботи з певними термінами або рідкісними словами.

Порівняння з іншими методами конкатенативний синтез часто порівнюють із формантним і параметричним синтезом. У табл. 2.2. подано коротке порівняння конкатенативного, формантного та параметричного синтезів.

Таблиця 2.2. Порівняння конкатенативного, формантного та параметричного синтезів

Характеристика	**Конкатенативний синтез**	**Формантний синтез**	**Параметричний синтез**
Природність звучання	Висока	Середня	Висока
Гнучкість	Обмежена	Висока	Висока
Вимоги до пам'яті	Високі	Низькі	Низькі
Спосіб генерації	Конкатенація записів	Генерація звуку	Моделювання параметрів

Конкатенативний синтез є потужним методом у сфері технологій TTS, який забезпечує високу якість та природність звучання завдяки використанню реальних голосових записів. Хоча він має свої недоліки, такі як вимоги до пам'яті та можливість виникнення артефактів на місцях з'єднання, його переваги роблять його популярним вибором у багатьох комерційних системах. Подальший розвиток технологій обробки звуку та алгоритмів вибору можуть допомогти подолати деякі з цих недоліків у майбутньому.

До основних технологічних викликів конкатенативного синтезу можна віднести:

- Емоційний синтез – створення мовлення, яке передає різні емоції, вимагає глибокого розуміння контексту та інтонацій;
- Адаптація до різних мов та акцентів – розробка моделей, які легко адаптуються до різних мов та діалектів, забезпечуючи високу якість мовлення.
- Реалістичне моделювання голосу - створення моделей, які точно відтворюють індивідуальні особливості голосу, такі як тембр, ритм і наголос.

З поширенням штучного інтелекту та глибокого навчання якість TTS продовжує покращуватися. Основні напрямки розвитку включають:

1. Імітація емоцій. Створення моделей, здатних передавати емоції у синтезованому мовленні.

2. Персоналізація голосів. Розробка моделей, які можуть персоналізувати мовлення під індивідуальні потреби користувачів.

3. Інтеграція з іншими технологіями. Використання TTS у поєднанні з іншими технологіями, такими як розпізнавання мови та природномовні інтерфейси, для створення більш інтерактивних та адаптивних систем.

Таким чином, системи синтезу мовлення продовжують еволюціонувати, забезпечуючи більш натуральне, гнучке та адаптивне мовлення, що знаходить застосування у все більшій кількості сфер та додатків.

2.3 Огляд схожих програм та їх недоліки

Веб-застосунки для синтезу мовлення (TTS) набули широкого поширення завдяки їхній здатності забезпечувати доступність інформації для різних категорій користувачів, зокрема для осіб з обмеженими можливостями. Огляд сучасних програм, схожих на розроблюваний веб-застосунок, включає:

1) Natural Reader – це онлайн-сервіс, що дозволяє перетворювати текст на мовлення за допомогою високоякісних синтезованих голосів. Програма підтримує різні формати тексту, такі як PDF, DOCX та TXT. Основними

недоліками є обмеження на безкоштовне використання та залежність від інтернет-з'єднання.

2) Read Aloud – це розширення для браузера, яке озвучує текст зі сторінок веб-сайтів. Хоча воно забезпечує зручність у використанні та підтримує різні мови, його функціональність обмежена браузерним контекстом і не дозволяє завантажувати власні документи для озвучування.

3) Voice Dream Reader. Цей додаток для мобільних пристроїв дозволяє озвучувати текст з різних джерел, включаючи PDF, DOCX і веб-сторінки. Однак, він є платним і може бути дорогим для деяких користувачів, особливо якщо необхідні додаткові функції.

4) Google Cloud Text-to-Speech. Google Cloud TTS пропонує понад 220 голосів у більш ніж 40 мовах і діалектах. Система використовує технології машинного навчання для забезпечення високої якості звучання та можливостей налаштування.

5) Amazon Polly дозволяє розробникам інтегрувати синтез мовлення у свої додатки та сервіси. Polly підтримує SSML для детального налаштування вимови і має можливість генерувати аудіо в реальному часі.

6) IBM Watson Text to Speech. IBM Watson TTS пропонує гнучкі можливості інтеграції та підтримує багато мов і голосів. Система також включає функції для налаштування інтонації та темпу.

7) Microsoft Azure Text to Speech Microsoft Azure TTS забезпечує високу якість синтезу з можливістю налаштування параметрів голосу та інтеграції з іншими сервісами Azure.

Системи тексту в мовлення (TTS) продовжують розвиватися, забезпечуючи нові можливості для взаємодії між людьми та технологіями.

Незважаючи на численні переваги, існуючі програми для синтезу мовлення мають кілька суттєвих недоліків, які можуть обмежувати їх використання. Одним з основних обмежень є наявність обмежень на безкоштовне використання, оскільки багато програм пропонують обмежену кількість функцій або часу роботи

в безкоштовних версіях, що може бути неприємним для користувачів, які потребують регулярного доступу до TTS. Крім того, платні версії або додаткові функції часто мають високу вартість, що може бути фінансово неприємним для окремих користувачів або невеликих компаній, які не готові витрачати значні кошти на ці сервіси. Ще одним недоліком є складність налаштування, оскільки деякі програми мають складний інтерфейс або вимагають додаткових налаштувань, що може бути занадто складним для новачків або людей, які не мають досвіду в налаштуванні подібних програм. Це може створювати додаткові труднощі для користувачів, які шукають прості та зручні рішення.

РОЗДІЛ 3.
РОЗРОБКА ВЕБ-ЗАСТОСУНКУ

3.1 Вибір інструментів та технологій

У рамках цього проєкту було застосовано низку сучасних технологій, кожна з яких відіграє ключову роль у його реалізації. Далі розглянемо їх детальніше, аналізуючи особливості та переваги кожної.

React – це ефективна та гнучка JavaScript бібліотека для створення користувацьких інтерфейсів. Вона дозволяє розробникам створювати веб-додатки, що можуть змінювати дані без перезавантаження сторінки. Основною одиницею в React є компоненти – маленькі та ізольовані частини коду, які управляють власним станом.[1]

Декларативний підхід React використовує декларативний підхід до програмування, який спрощує процес проектування інтерфейсів. Розробники описують, якими частинами повинен бути інтерфейс у різних станах, а React ефективно оновлює та відображає тільки ті компоненти, які потребують оновлення. Це веде до більш прогнозованого коду та меншого обсягу помилок.

React – це популярна JavaScript-бібліотека для створення інтерфейсів користувача, яка була розроблена компанією Facebook у 2013 році. Вона дозволяє створювати динамічні та інтерактивні веб-додатки з використанням компонентів і реактивного підходу до управління даними.

Нижче подані основні принципи роботи бібліотеки React, зокрема:

- Компонентний підхід. React ґрунтується на ідеї розбиття інтерфейсу на окремі компоненти, які можуть бути багаторазово використані. Кожен компонент – це функція або клас, який повертає React-елементи (JSX).
- JSX (JavaScript XML). Це синтаксис, що дозволяє описувати структуру інтерфейсу в JavaScript-коді. JSX виглядає як HTML, але підтримує динамічні вставки через JavaScript.

- Односпрямований потік даних. Дані передаються вниз через властивості (props) від батьківського компонента до дочірніх. Це забезпечує передбачуваність поведінки додатка.

- Virtual DOM. React використовує віртуальну модель DOM для оптимізації оновлень. Це дозволяє ефективно змінювати тільки ті частини сторінки, які потрібно оновити, без повного перерендера.

Основні поняття в React включають компоненти, які є будівельними блоками інтерфейсу, стан (state) і пропси (props) для управління даними, а також Virtual DOM, що забезпечує швидке та ефективне оновлення користувацького інтерфейсу. Існують два типи компонентів:

Функціональні компоненти – це функції, які приймають props і повертають JSX. Завдяки хукам функціональні компоненти стали основним підходом.

Класові компоненти- класи, які успадковуються від React.Component і мають метод render().

1. Props (Властивості). Це дані, які передаються від батьківського компонента до дочірнього. Props є незмінними, що забезпечує стабільність даних.

2. State (Стан). Це динамічні дані, які належать конкретному компоненту. Стан можна оновлювати за допомогою методу setState() у класових компонентах або через useState у функціональних.

3. Hooks (Хуки). Це функції, що дозволяють використовувати стан та інші можливості React у функціональних компонентах. До популярних хуків можна віднести: useState для управління станом; useEffect для виконання побічних ефектів, наприклад, запитів до API; useContext для доступу до глобальних даних через контекст.

4. Context API. Дозволяє передавати дані через дерево компонентів без використання props. Це корисно для управління глобальним станом.

Далі розглянемо життєвий цикл компонентів. Класові компоненти React мають такі стадії життєвого циклу:

1. Монтування (Mounting) - constructor(); render(); componentDidMount()
2. Оновлення (Updating) - componentDidUpdate(prevProps, prevState)
3. Розмонтування (Unmounting) - componentWillUnmount()

Для функціональних компонентів життєвий цикл реалізується через useEffect().

React має кілька ключових переваг, які роблять його популярним вибором серед розробників. Швидкість є однією з таких переваг, оскільки завдяки використанню Virtual DOM оновлення інтерфейсу відбувається швидко та ефективно, що покращує продуктивність додатків. Гнучкість React дозволяє використовувати його для створення як простих додатків, так і складних SPA (Single Page Applications), забезпечуючи необхідні інструменти для реалізації різних типів проєктів. Масштабованість є ще однією сильною стороною React, оскільки компонентна архітектура дає змогу легко підтримувати та розширювати кодову базу, що робить його ідеальним для великих проєктів. Підтримка спільноти є важливим фактором, оскільки велика спільнота розробників і багата екосистема бібліотек, таких як Redux, React Router, і Material-UI, дозволяють значно розширити можливості React. Окрім того, React є універсальним інструментом, який можна використовувати для створення як веб-додатків, так і мобільних додатків за допомогою React Native, що робить його ще більш зручним для розробки кросплатформених рішень.

Попри численні переваги, React має кілька недоліків, які можуть створювати труднощі для розробників. По-перше, для початківців може бути високий поріг входження, оскільки React вимагає розуміння ряду складних концепцій, таких як JSX, хуки та контекст, що може зробити перші кроки у використанні фреймворку складними. По-друге, часті оновлення бібліотеки та її швидкий розвиток іноді викликають проблеми з підтримкою старих проєктів, адже нові версії можуть змінювати або видаляти функціональність, що потребує постійного оновлення коду для сумісності з новими релізами. Крім того, відсутність строгої структури проекту в React може призвести до того, що реалізація буде залежати від розробника, і без належної документації проєкти

можуть стати складними для підтримки, особливо в командній роботі або у великих проектах, де важлива організація коду.

До екосистема React входять наступні компоненти:

– React Router – бібліотека для управління маршрутизацією у SPA-додатках;

– Redux – інструмент для управління станом у великих додатках;

– Next.js – фреймворк для створення серверних рендерингових додатків на базі React;

– Styled Components – інструмент для написання стилів у JS;

– React Native – фреймворк для створення мобільних додатків на основі React.

React знаходить широке застосування в різних сферах розробки завдяки своїй універсальності та ефективності, а саме:

1. Веб-додатки – створення складних SPA-додатків, таких як адмін-панелі, онлайн-магазини.
2. Мобільні додатки – використання React Native для створення додатків на iOS та Android.
3. Компонентні бібліотеки – розробка UI-компонентів, які можуть бути багаторазово використані.[2]

Flask – це популярний мікрофреймворк для веб-розробки на Python, який дозволяє швидко створювати веб-додатки. Його головною особливістю є мінімалістичний підхід: Flask надає основні функції для розробки веб-додатків, дозволяючи розробнику самому вибирати, які додаткові компоненти або бібліотеки використовувати. Це робить Flask ідеальним для невеликих проєктів і прототипів, але також дозволяє створювати масштабовані додатки за допомогою розширень.

До основних характеристик Flask можна віднести:

- Мінімалізм і легкість – Flask не нав’язує певної структури проєкту чи інструментів, залишаючи вибір за розробником. Він ідеально підходить для тих, хто хоче мати повний контроль над архітектурою свого додатка.
- WSGI та Jinja2 – використовує WSGI (Web Server Gateway Interface) як протокол взаємодії між сервером і додатком. Використовує шаблонізатор Jinja2, який дозволяє створювати HTML-шаблони з інтегрованим Python-кодом.
- Розширюваність – Flask має вбудовану підтримку розширень, таких як Flask-SQLAlchemy (для роботи з базами даних), Flask-WTF (форми), Flask-Migrate (міграції) тощо.
- Режим налагодження – Flask ї має інтерактивний дебагер і можливість автоматичного перезапуску сервера під час внесення змін у код.
- Масштабованість – хоча Flask задумувався як мікрофреймворк, він підтримує модульний підхід, що дозволяє створювати масштабовані проєкти.

Основи роботи з Flask включають кілька ключових елементів. Перш за все, для початку розробки необхідно встановити Flask за допомогою пакетного менеджера pip:

```
pip install flask
```

Простий приклад додатка: Ось базовий приклад додатка на Flask:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def home():
    return 'Hello, Flask!'

if __name__ == '__main__':
    app.run(debug=True)
```

Flask(__name__): створює екземпляр додатка.

@app.route('/'): визначає маршрут для URL /.

app.run(debug=True): запускає сервер розробки з увімкненим режимом налагодження.

Flask складається з кількох основних компонентів, які забезпечують його функціональність та гнучкість. Перший компонент — це об'єкт додатку, створений за допомогою класу Flask, який виступає основою для всього веб-додатку. Маршрутизація Flask дозволяє створювати маршрути для URL за допомогою декораторів.

```
@app.route('/about')
def about():
   return 'This is the About page.'
```

Підтримуються динамічні URL:

```
@app.route('/user/<username>')
def profile(username):
        return f'Hello, {username}!'
```

1. Шаблони (Jinja2) Flask підтримує шаблони Jinja2 для створення динамічних HTML-сторінок.

   HTML-шаблон: (файл templates/index.html)

   ```
   <!DOCTYPE html>
   <html>
   <head>
      <title>{{ title }}</title>
   </head>
   <body>
      <h1>Welcome, {{ user }}!</h1>
   </body>
   </html>
   ```

   - Рендеринг шаблону:
   - from flask import render_template
   - @app.route('/')
   - def home():
   - return render_template('index.html', title='Home', user='John Doe')

2. Форми Flask-WTF дозволяє легко працювати з формами.

```
from flask import request
```

```
@app.route('/submit', methods=['POST'])
def submit():
    name = request.form['name']
    return f'Hello, {name}!'
```

3. Робота з базами даних Flask-SQLAlchemy спрощує інтеграцію з базами даних.

```
from flask_sqlalchemy import SQLAlchemy

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///example.db'
db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), nullable=False)

db.create_all()
```

4. Реалізація API Flask може використовуватися для створення RESTful API.

```
from flask import jsonify

@app.route('/api/data')
def data():
    return jsonify({'key': 'value', 'status': 'success'})
```

Flask має низку переваг, які роблять його популярним вибором серед розробників. По-перше, його простота та зрозуміла документація роблять його ідеальним для початківців, дозволяючи швидко освоїти основи розробки веб-додатків. Крім того, Flask відзначається високою гнучкістю, оскільки розробники можуть повністю контролювати структуру додатка та використовувати лише необхідні компоненти, без обмежень. Широка екосистема Flask підтримує безліч розширень, що дає змогу додавати нові функції, такі як аутентифікація, ORM або кешування, що значно розширює можливості фреймворка. Flask також забезпечує швидкість розробки,

дозволяючи швидко створювати прототипи та невеликі проєкти. Завдяки великій спільноті розробників, Flask має багату базу прикладів і підтримки, що робить його ще більш зручним для використання в реальних проєктах.

Також, поряд із перевагами Flask має і кілька недоліків, які можуть стати проблемою для певних проєктів. По-перше, він надає менше інтегрованих функцій у порівнянні з більшими фреймворками, такими як Django, що означає, що багато завдань, як-от аутентифікація або робота з базами даних, потрібно реалізовувати вручну. Це веде до того, що розробникам доводиться виконувати більше ручної роботи, а деякі функції, які в інших фреймворках йдуть "з коробки", у Flask потрібно додавати за допомогою сторонніх бібліотек. Крім того, масштабування великих проєктів може бути складним завданням, оскільки Flask не має вбудованих інструментів для управління складними архітектурами, і для забезпечення продуктивності та підтримки структури може знадобитися більше зусиль.

Незважаючи на ці недоліки, Flask є відмінним вибором для різних застосувань. Він ідеально підходить для швидкого створення прототипів веб-додатків, оскільки дозволяє швидко реалізувати основну функціональність без зайвих налаштувань. Завдяки своїй легкості та гнучкості Flask також популярний у створенні мікросервісів, де важлива модульність і можливість швидко змінювати окремі компоненти. Крім того, він є чудовим вибором для створення RESTful API, завдяки зручному налаштуванню маршрутизації та підтримці простих інтеграцій. Flask також часто використовують для розробки невеликих веб-додатків, таких як блоги або панелі адміністратора, де не потрібно складних функцій і великої інфраструктури.

Flask можна застосовувати у таких проєктах:

1. Прототипи – Flask ідеально підходить для швидкого створення прототипів веб-додатків.
2. Мікросервіси – завдяки легкості й гнучкості Flask часто використовується для створення мікросервісів.
3. API - Flask є відмінним вибором для створення RESTful API.

4. Невеликі веб-додатки – Flask популярний серед розробників, які створюють невеликі сайти, блоги або панелі адміністратора.

Flask – це чудовий інструмент для тих, хто цінує мінімалізм і гнучкість. Завдяки простоті у використанні, широкій екосистемі та активній спільноті, Flask залишається одним із найкращих виборів для веб-розробників.

PostgreSQL – це потужна об'єктно-реляційна система управління базами даних. Вона широко відома своєю надійністю, міцністю функціоналу та підтримкою великої кількості даних. PostgreSQL використовується в проекті для збереження даних користувачів. Вона включає такі функції, як транзакції, надійне збереження даних і високий рівень сумісності зі стандартами SQL, що робить її ідеальним вибором для додатків, які потребують надійного управління даними.[3]

AWS S3 – сервіс дозволяє ефективно управляти доступом до файлів, їх збереженням та дистрибуцією, забезпечуючи високу доступність та масштабованість. Це особливо важливо для функціональності, що дозволяє користувачам зберігати велику кількість аудіофайлів, створених з використанням TTS, без зайвого навантаження на основний сервер.[4]

Identity and Access Management (IAM) від AWS є ключовим інструментом для управління доступом до AWS ресурсів. Ця система дозволяє адміністраторам веб-застосунку безпечно керувати доступом до AWS сервісів та ресурсів для користувачів та систем. IAM забезпечує детальне керування ідентифікацією, автентифікацією та авторизацією через політики, які визначають, які дії дозволені різним користувачам. У контексті веб-застосунку, IAM використовується для регулювання доступу до S3 бакетів, що використовуються для зберігання аудіофайлів, забезпечуючи, що лише авторизовані користувачі можуть завантажувати або отримувати доступ до своїх медіафайлів.[5]

Архітектура веб-застосунку включає наступні основні компоненти:

1) Клієнтська частина – реалізована за допомогою React. Це забезпечує динамічний та інтерактивний інтерфейс користувача.[7]

2) Серверна частина – побудована на Flask, обробляє всі серверні запити, включаючи автентифікацію користувачів, управління сесіями та обробку тексту для синтезу мовлення.

3) API для озвучування тексту – використання Flask для інтеграції з gTTS, який перетворює текст на мовлення.

3.2 Реалізація функціоналу для озвучування тексту

Реалізація функції озвучування тексту в сучасних додатках та веб-інтерфейсах дозволяє значно підвищити доступність сервісів для широкої аудиторії. Зокрема, використання бібліотеки gTTS (Google Text-to-Speech) є одним із найпоширеніших підходів до інтеграції можливостей перетворення тексту на мовлення. У цій статті ми детально розглянемо всі аспекти реалізації такої функціональності: від вибору інструментів і технологій до технічних деталей, тестування та прикладів використання.

Розробка функціоналу озвучування тексту потребує вибору відповідної технології, яка надасть наступні переваги:

- Якість синтезованого мовлення. Голос має бути природним, чітким і приємним для слухача;
- Швидкодію. Система повинна оперативно відповідати на запити користувача;
- Інтеграцію з іншими компонентами додатку. Це забезпечує зручність використання для розробників;
- Підтримку різних мов та акцентів. Сучасні додатки часто орієнтовані на міжнародну аудиторію.

Для проєкту було обрано бібліотеку gTTS з огляду на його переваги, зокрема:[6]

1. Доступність. Це безкоштовна бібліотека, яка використовує API від Google.
2. Простота інтеграції. Вона легко імплементується в Python-проекти.

3. Різноманіття мов. Підтримуються десятки мов, включаючи українську.

4. Якість звуку. Голоси, які генерує gTTS, звучать природно завдяки використанню сучасних алгоритмів синтезу мовлення.

Далі розглянемо процес інтеграції gTTS у веб-інтерфейс.

Першим кроком у розробці є встановлення бібліотеки gTTS. Це можна зробити за допомогою менеджера пакетів `pip`:

```
pip install gTTS
```

Після встановлення необхідно імпортувати бібліотеку у ваш проект:

```
from gtts import gTTS
```

Для генерації мовлення з тексту використовується наступний простий код:

```
text = "Ласкаво просимо до нашого додатку!"
tts = gTTS(text=text, lang='uk')
tts.save("output.mp3")
```

У цьому прикладі:

- `text` — текст, який потрібно озвучити.
- `lang` — код мови (наприклад, `'uk'` для української).
- Метод `save()` створює аудіофайл, який потім можна відтворити.

Щоб надати можливість користувачам взаємодіяти з функцією озвучування через веб-інтерфейс, використовуються фреймворки на зразок Flask або Django. Ось приклад використання Flask:

```
from flask import Flask, request, send_file
from gtts import gTTS

app = Flask(__name__)

@app.route('/speak', methods=['POST'])
def speak():
    text = request.form.get('text')
    if not text:
        return "Будь ласка, введіть текст!", 400

    tts = gTTS(text=text, lang='uk')
    tts.save("output.mp3")
```

```
    return send_file("output.mp3", as_attachment=True)

if __name__ == '__main__':
    app.run(debug=True)
```

У цьому прикладі:

1. Користувач вводить текст через форму в браузері.
2. Сервер обробляє запит, генерує аудіофайл і повертає його користувачеві.
3. Озвучення виконується в режимі реального часу.

Щоб підтримувати багато мов, можна додати параметр вибору мови у веб-форму. Код для цього виглядає так:

```
    lang = request.form.get('lang', 'uk')    # За замовчуванням використовується українська
    tts = gTTS(text=text, lang=lang)
```

Хоча gTTS забезпечує високу якість, існують способи підвищення швидкодії (її оптимізації), зокрема:

1. Кешування результатів. Якщо користувач запитує озвучення одного й того ж тексту, система може зберігати попередньо створені аудіофайли.
2. Попереднє створення файлів. Часто використовувані фрази (наприклад, привітання) можна озвучувати заздалегідь.
3. Паралельна обробка. У багатокористувацькому середовищі корисно використовувати багатопотоковість.

Для впевненості в коректній роботі функції озвучування необхідно провести всебічне тестування:

1. Функціональне тестування. Перевірка, чи коректно генерується аудіо для різних текстів.
2. Тестування мов. Переконатися, що система правильно озвучує тексти різними мовами.
3. Перевірка продуктивності. Аналіз швидкості відповіді системи на запити.

4. Юзабіліті-тестування. Оцінка зручності використання функції для кінцевих користувачів.

Переваги використання технології синтезу мовлення включають в себе простоту впровадження, оскільки реалізація функціоналу не потребує складної інфраструктури, що робить процес інтеграції швидким і доступним. Крім того, якість мовлення, яку надають сучасні системи TTS, є дуже високою: тексти звучать природно і зрозуміло, що покращує сприйняття інформації. Важливою перевагою є також підтримка багатьох мов, що дає змогу реалізувати проєкти для міжнародної аудиторії та забезпечує гнучкість у використанні технології для різних мовних груп.

Недоліки використання технології синтезу мовлення, зокрема gTTS, включають залежність від Інтернету, оскільки для роботи цієї системи необхідний доступ до мережі, що може бути проблемою для автономних або ізольованих систем, де стабільне з’єднання відсутнє. Крім того, у великих системах, що обробляють великий обсяг даних або мають високу частоту запитів, можуть виникати обмеження продуктивності, що вимагатиме додаткового розширення серверних ресурсів для забезпечення належної роботи без затримок.

Функцію озвучування тексту можна використовувати у різних сферах. Нижче наведемо декілька прикладів її застосування:[8]

- Освітні платформи – інтерактивні тренажери для вивчення мов.
- Додатки для людей з вадами зору – текст озвучується для полегшення взаємодії.
- Мобільні додатки – системи навігації або голосові повідомлення.

Інтерфейс користувача розроблений так, щоб бути інтуїтивно зрозумілим та зручним для користувачів різного віку та досвіду використання веб-технологій. Це включає лаконічне меню, чітко виділені кнопки для озвучування тексту та збереження аудіозаписів, а також поліпшені опції для візуального сприйняття інформації.

РОЗДІЛ 4.
ПРОГРАМНА РЕАЛІЗЯЦІЯ ВЕБ-ЗАСТОСУНКУ

4.1 Робота з аудіофайлами

Функціональність веб-застосунку включає перетворення тексту в аудіо, збереження аудіофайлів для кожного користувача із заданою назвою, а також можливість відтворення цих файлів.

Застосунок використовує бібліотеку gTTS (Google Text-to-Speech) для перетворення тексту в мовлення. Це Python бібліотека, яка інтерфейсує Google Translate's текст у мовлення API. Далі представлено код функції, основними завданнями якої є надсилання запиту на озвучування тексту, через запит POST на сервер за допомогою функції fetch(), в якому передається текст для озвучування, після чого функція обробляє відповідь сервера: якщо вона успішна, встановлює посилання на новий аудіофайл, який містить озвучений текст. Якщо запит не був успішним або виникла помилка під час виконання, виводить помилку у консоль.

```
const handleSynthesize = async () => {
    setLoading(true);
    console.log("Button clicked");
    try {
        const response = await fetch('/synthesize', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ text: text })
        });
        if (!response.ok) {
            throw new Error('Network response was not ok');
        }
        const data = await response.json();
        console.log("Response from server:", data);

        const newAudioSrc = `/audio_files/${data.filename}?${Date.now()}`;
        setAudioSrc(newAudioSrc);
    } catch (error) {
        console.error('Failed to fetch:', error);
    } finally {
        setLoading(false);
    }
};
```

Рисунок 4.1 - Функція надсилання запиту на озвучування

```python
@app.route('/synthesize', methods=['POST'])
def synthesize_text():
    try:
        data = request.get_json()
        text = data.get('text', 'Текст не надано')
        tts = gTTS(text=text, lang='uk')
        filename = secure_filename("output.mp3")
        filepath = os.path.join('audio_files', filename)
        tts.save(filepath)
        print(1)
        return jsonify({"message": "Audio created successfully", "filename": filename})
    except Exception as e:
        app.logger.error(f"Error: {str(e)}")
        return jsonify({"error": str(e)}), 500
```

Рисунок 4.2 - Функція опрацювання запиту на озвучування

На головній сторінці застосунку користувач бачить наступне вікно:

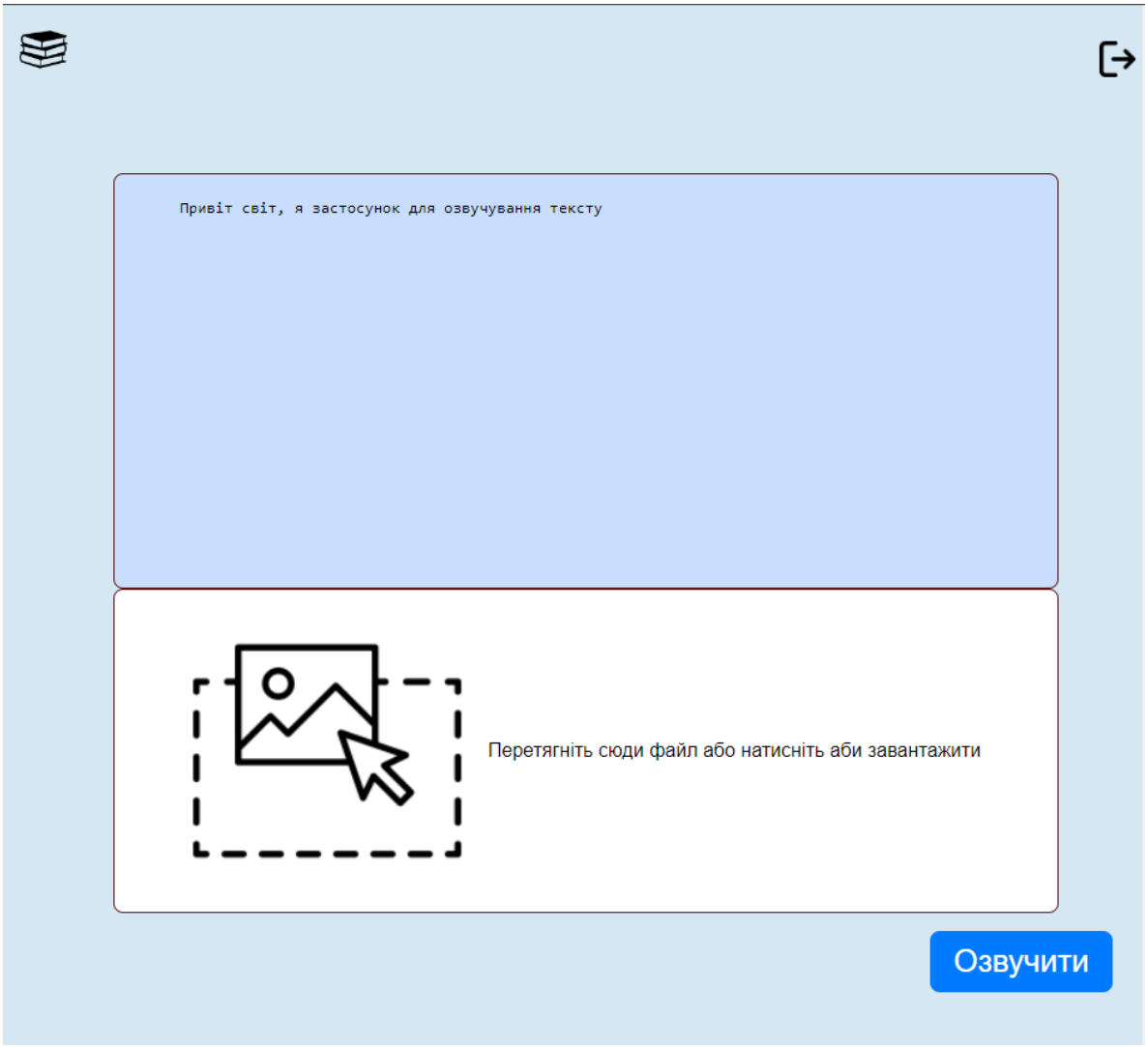

Рисунок 4.3 - Головне вікно користувача до виконання озвучування

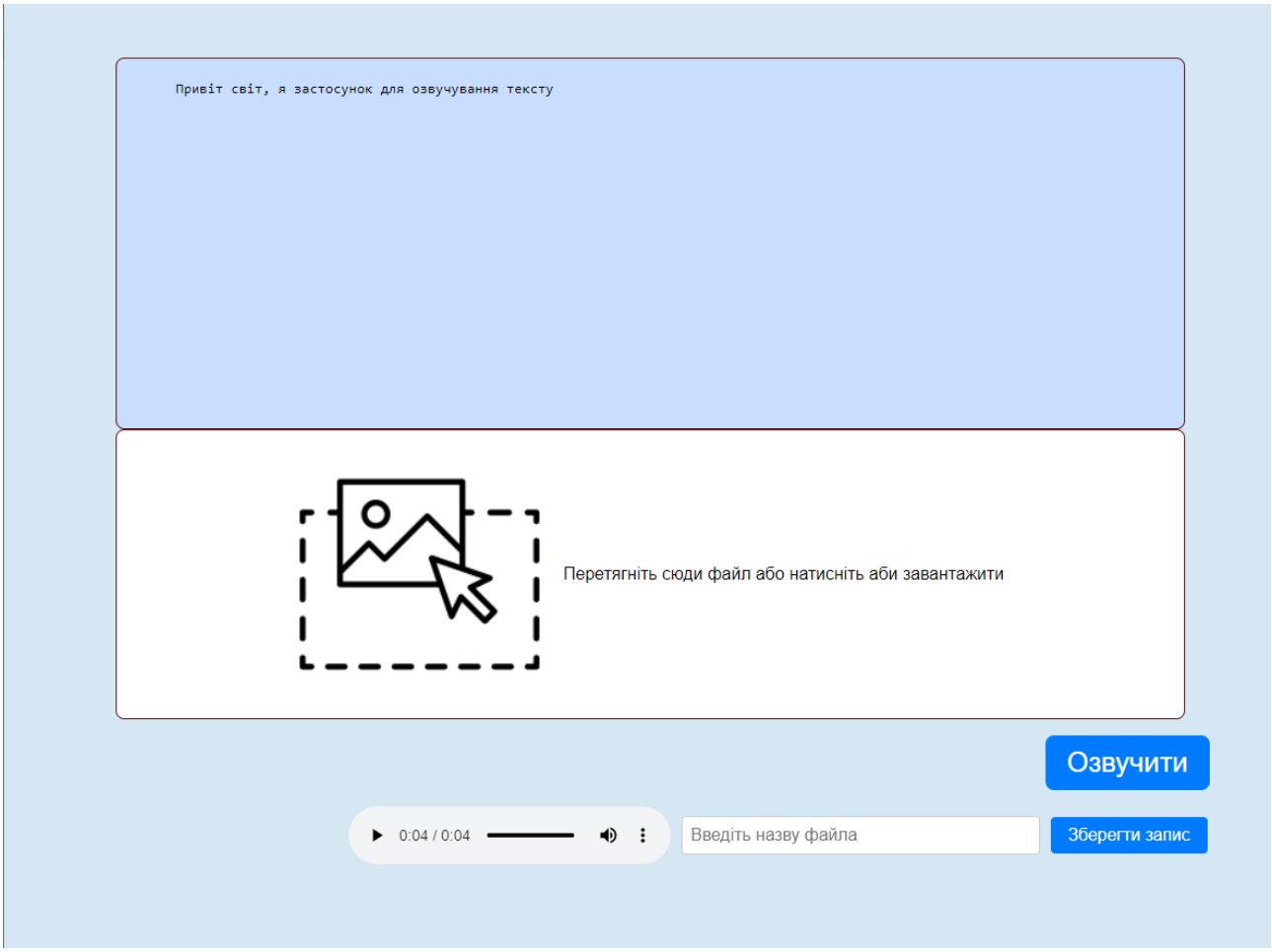

Рисунок. 4.4 - Головне вікно користувача після виконання озвучування

Ця функціональність веб-застосунку відповідає за зчитування текстових даних з файлів, завантажених користувачами. Вона дозволяє користувачам перетворити вміст файлів у різних форматах, таких як PDF, DOCX і TXT, на текст, який подальше може бути озвучений за допомогою системи TTS. Зчитування файлів булореалізовано наступним чином:

- Підтримка форматів підтримуються основні текстові та документні формати. Наприклад, для файлів DOCX використовується бібліотека `mammoth`, яка дозволяє екстрагувати текст з документів Word, зберігаючи при цьому структурну цілісність тексту. Для файлів PDF застосовується бібліотека pdfjs-dist, яка дозволяє асинхронно зчитувати текстовий вміст з PDF-документів;

- Зчитування файлу під час завантаження файлу на сервер, файл спочатку зберігається в тимчасовому сховищі. Потім застосунок ініціює процес

зчитування тексту з файлу. Цей процес залежить від типу файлу і включає в себе перетворення файлу в бінарний формат, що може оброблятися відповідною бібліотекою;

- Оптимізація для швидкого зчитування програмний код оптимізовано для мінімізації затримок при зчитуванні файлів, що забезпечує користувачам можливість отримувати текст для озвучування без значних затримок.

```
const handleDrop = async (event) => {
    event.preventDefault();
    const file = event.dataTransfer.files[0];
    if (file) {
        setFileName(file.name);
        if (file.type === 'text/plain') {
            const reader = new FileReader();
            reader.onload = (e) => {
                setText(e.target.result);
            };
            reader.readAsText(file);
        } else if (file.type === 'application/vnd.openxmlformats-officedocument.wordprocessingml.document') {
            const reader = new FileReader();
            reader.onload = async (e) => {
                const arrayBuffer = e.target.result;
                const result = await mammoth.extractRawText({ arrayBuffer: arrayBuffer });
                setText(result.value);
            };
            reader.readAsArrayBuffer(file);
        } else if (file.type === 'application/pdf') {
            const reader = new FileReader();
            reader.onload = async (e) => {
                const arrayBuffer = e.target.result;
                const pdf = await getDocument(arrayBuffer).promise;
                let pdfText = '';
                for (let i = 1; i <= pdf.numPages; i++) {
                    const page = await pdf.getPage(i);
                    const textContent = await page.getTextContent();
                    const strings = textContent.items.map(item => item.str);
                    pdfText += strings.join(' ');
                }
                setText(pdfText);
            };
            reader.readAsArrayBuffer(file);
        } else {
            setText("Unsupported file type");
        }
    }
};
```

Рисунок 4.5 - Функція опрацьовування переміщення користувачем файлу у відповідне поле на вікні

Оскільки одним з найпоширеніших форматів передачі навчальних матеріалів є PDF, то варто розглянути функцію перетворення вмісту файлів, з таким форматом, у текст окремо:

```
const processPDF = async (file) => {
    const reader = new FileReader();
    reader.onload = async (e) => {
        const typedarray = new Uint8Array(e.target.result);
        const pdf = await getDocument(typedarray).promise;
        let pdfText = '';
        for (let i = 1; i <= pdf.numPages; i++) {
            const page = await pdf.getPage(i);
            const textContent = await page.getTextContent();
            const strings = textContent.items.map(item => item.str);
            pdfText += strings.join(' ');
        }
        setText(pdfText);
    };
    reader.readAsArrayBuffer(file);
};
```

Рисунок 4.6 - Функція опрацьовування переміщення користувачем файлу у відповідне поле на вікні

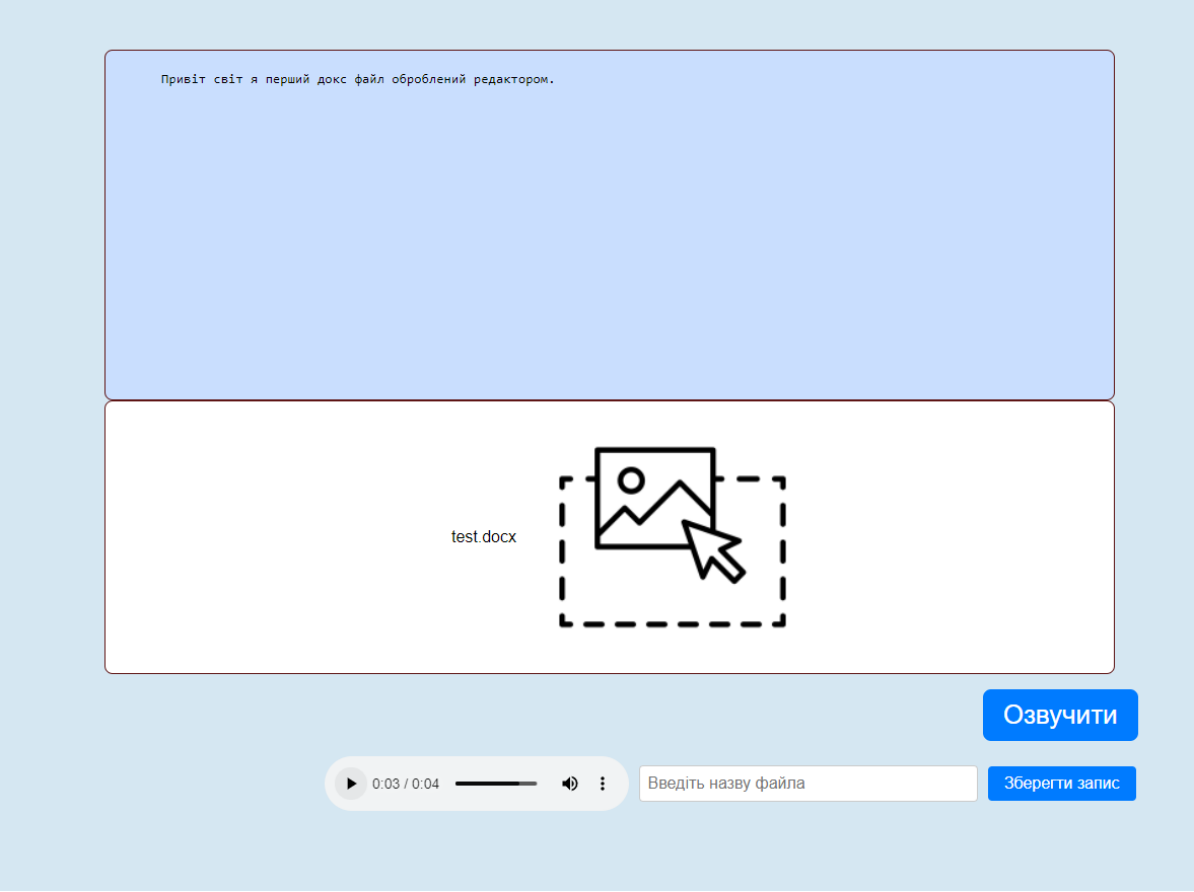

Рисунок 4.7 - Головне вікно користувача після виконання зчитування файлу користувача та його озвучування

Це дозволяє користувачам зберігати важливу інформацію у зручному аудіо форматі, і з легкістю доступити до неї в будь-який час. Зберігання відбувається на сервері за допомогою сервісу Amazon S3, що забезпечує надійне

та масштабоване сховище. Далі розглянемо порядок роботи, а саме, як це працює:

- Створення аудіофайлу. Після того, як текст було перетворено на мовлення за допомогою TTS, генерований аудіофайл тимчасово зберігається на сервері. Користувач може вказати назву файла, під якою він буде збережений, або використовувати назву за замовчуванням.
- Завантаження в S3. Аудіофайл автоматично відправляється у визначений "bucket" на Amazon S3 з наданою користувачем назвою. Кожен "bucket" асоціюється з конкретним користувачем, що дозволяє зберігати персоналізовану бібліотеку аудіофайлів для кожного користувача.
- Безпека. Всі аудіофайли захищені від несанкціонованого доступу, а доступ до них можливий лише для авторизованих користувачів за допомогою системи управління доступом, реалізованої у веб-застосунку.

Функціонал збереження аудіофайлу реалізовується в кількох функціях, як показано нижче:

```
const saveAudioFile = async () => {
    if (!audioSrc) {
        alert('Немає аудіо для збереження');
        return;
    }

    setFileSaved(true);
    const response = await fetch(audioSrc);
    const blob = await response.blob();

    if (!blob) {
        alert('Помилка при завантаженні аудіо');
        return;
    }

    const customPath = userEmail + '/' + (customFileName || 'audio_file') + '.mp3';
    uploadFileToS3(blob, customPath);
};
```

Рисунок 4.8 – Функція перевірки наявності аудіо та його назви

```
const uploadFileToS3 = (blob, filename) => {
    const AWS = window.AWS;
    const s3 = new AWS.S3();

    const params = {
        Bucket: 'voiceedu',
        Key: `audio/${filename}`,
        Body: blob,
        ContentType: 'audio/mpeg'
    };

    s3.upload(params, function(err, data) {
        if (err) {
            console.error("Error uploading file:", err);
        } else {
            console.log("Upload successful:", data.Location);
        }
    });
};
```

Рисунок 4.9 – Функція завантаження створеного аудіо-файлу на AWS S3

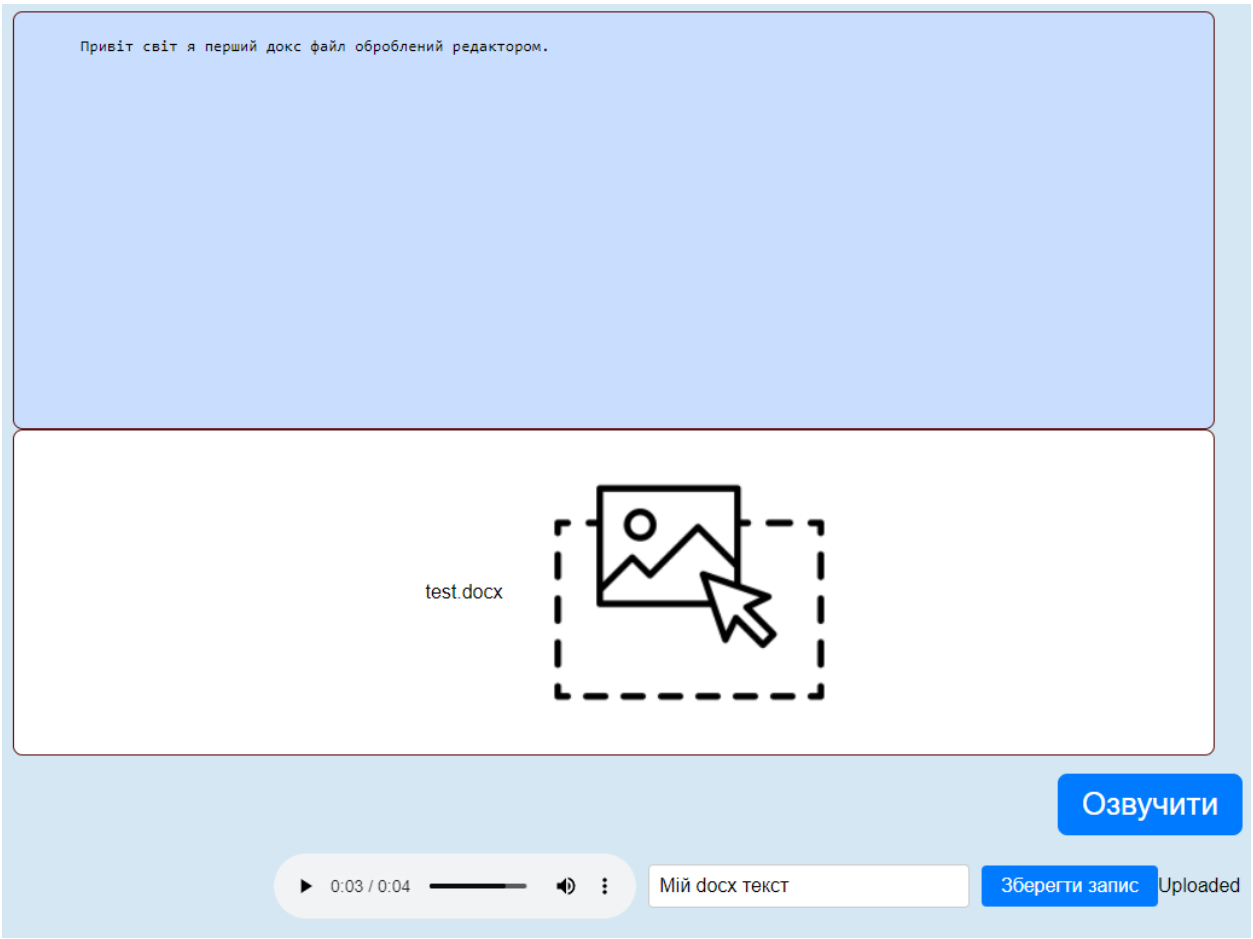

Рисунок 4.10 - Головне вікно користувача після збереження файлу

Функція прослуховування та менеджменту створених аудіо надає користувачам зручний інструмент для ефективного управління та прослуховування створених аудіофайлів. Вона включає в себе можливості для перегляду списку згенерованих аудіо, їх відтворення, а також функції для організації та збереження файлів, таких як перейменування, видалення або завантаження на пристрій. Це дозволяє користувачам зручно працювати з їх аудіо-контентом, забезпечуючи оптимальну організацію та доступність матеріалів.

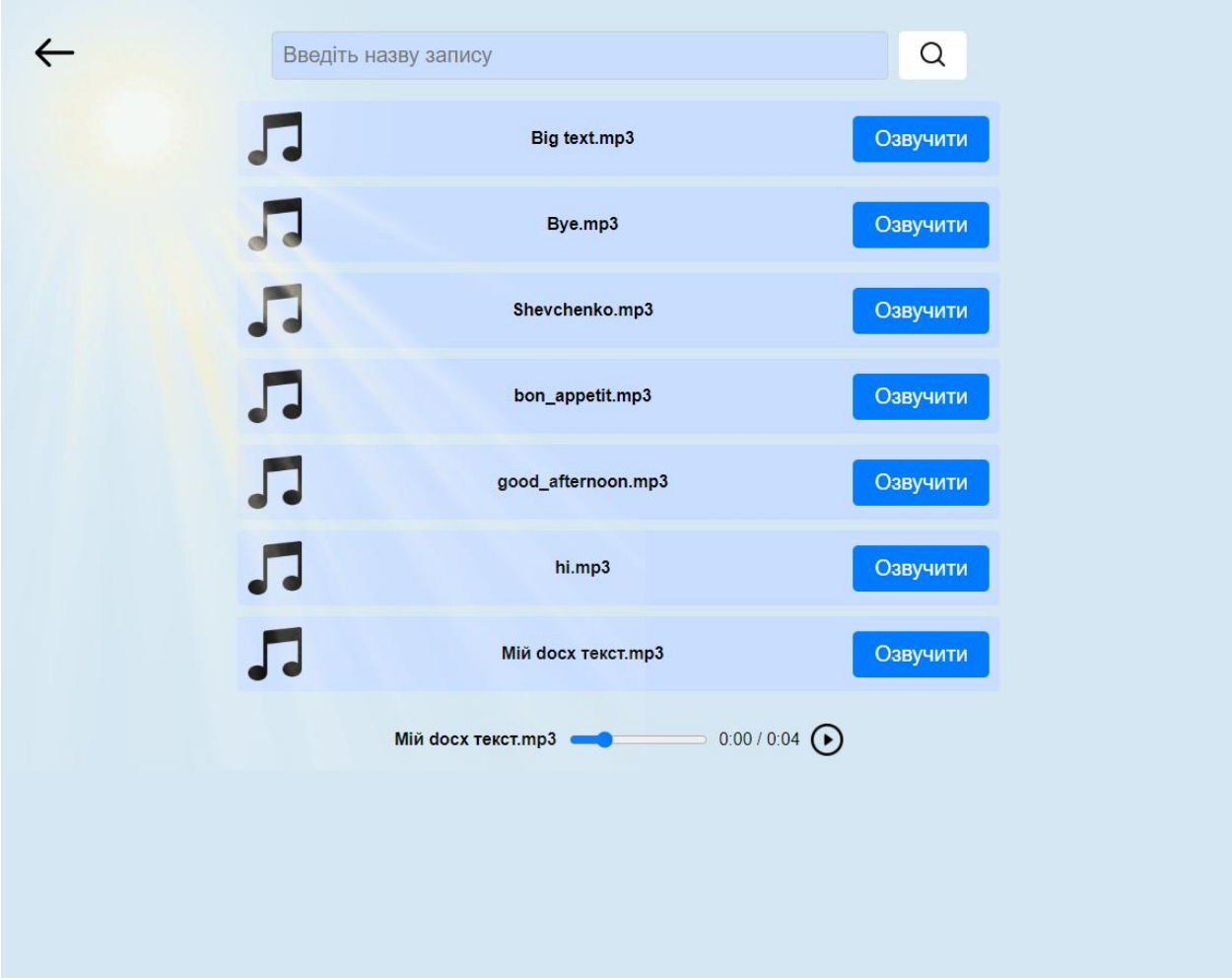

Рисунок 4.11 – Вікно перегляду збережених користувачем аудіофайлів, під час відтворення “Мій docx текст.mp3”

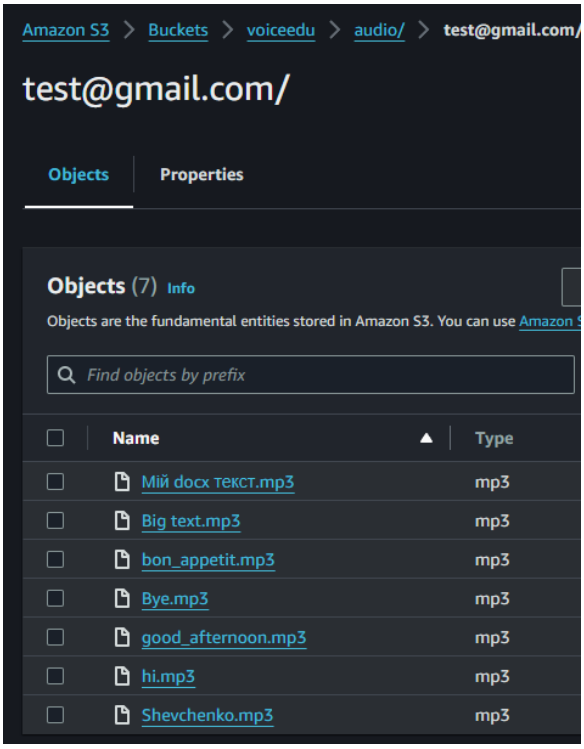

Рисунок 4.12 – Наявність відображених у користувацькому вікні аудіофайлів у відповідному “bucket”

```
const fetchTracks = (userEmail) => {
    setLoading(true);
    setError('');
    const params = {
        Bucket: 'voiceedu',
        Prefix: `audio/${userEmail}/`
    };

    s3.listObjectsV2(params, (err, data) => {
        setLoading(false);
        if (err) {
            console.error("Error fetching the files: ", err);
            setError('Failed to fetch tracks. Please try again later.');
        } else {
            if (data.Contents.length === 0) {
                setError('No tracks found.');
            } else {
                const trackData = data.Contents.map((file) => {
                    return { title: file.Key.split('/').pop(), key: file.Key };
                });
                setTracks(trackData);
            }
        }
    });
};
```

Рисунок 4.13 - Функція видобування користувацьких записів із сховища

```
const handleSearch = (e) => {
    setSearch(e.target.value);
};

const filteredTracks = tracks.filter(track =>
    track.title.toLowerCase().includes(search.toLowerCase())
);
```

Рисунок 4.14 - Функції застосування пошуку за назвою запису

4.2 Забезпечення безпеки даних та автентифікація користувачів

Основні аспекти забезпечення безпеки даних та автентифікації користувачів у веб-застосунку включають наступні підпункти: реєстрація користувачів, логін користувачів, управління сесіями. Далі розглянемо детальніше кожен із них.

Реєстрація користувачів в системі здійснюється через введення базових даних: електронна пошта та пароль. Для збереження паролів використовується метод хешування з додаванням "солі", що забезпечує додатковий рівень безпеки. Цей метод запобігає можливості використання простих атак по словнику або атак з використанням готових таблиць хешів. Програмно спершу надсилається запит на реєстрацію.

```
const handleSubmit = async (event) => {
    event.preventDefault();
    if (!validateForm()) {
        setError('Please fill in all fields.');
        return;
    }

    try {
        const response = await fetch('http://localhost:5000/register', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ email, password }),
        });

        const data = await response.json();
        if (response.status === 201) {
            navigate('/main-window');
        } else {
            throw new Error(data.message || 'Registration failed');
        }
    } catch (error) {
        setError(error.message);
    }
};
```

Рисунок 4.15 – Функція надсилання запиту на реєстрацію

Після чого функція обробляє відповідь сервера: якщо вона успішна, то користувача перекидає на головне вікно програми. Якщо запит не був успішним або виникла помилка під час виконання, то про це буде повідомлено користувачу.

```python
@app.route('/register', methods=['POST'])
def register():
    try:
        data = request.get_json()
        email = data.get('email')
        password = data.get('password')

        if not validate_email(email) or not validate_password(password):
            return jsonify({"error": "Invalid email or password format"}), 400

        cur = conn.cursor()
        cur.execute("SELECT * FROM users WHERE email = %s", (email,))
        user = cur.fetchone()
        if user:
            return jsonify({"error": "User already exists"}), 400

        hashed_password = bcrypt.hashpw(password.encode('utf-8'), bcrypt.gensalt())
        cur.execute("INSERT INTO users (email, password) VALUES (%s, %s)", (email, hashed_password))
        conn.commit()
        cur.close()

        token = jwt.encode({
            'user_id': email,
            'exp': datetime.now(timezone.utc) + timedelta(hours=24)
        }, SECRET_KEY, algorithm='HS256')

        return jsonify({"message": "User registered successfully", "token": token}), 201

    except Exception as e:
        app.logger.error(f"Error: {str(e)}")
        return jsonify({"error": str(e)}), 500
```

Рисунок 4.16 – Функція опрацювання запиту на реєстрацію

В процесі опрацьовування запиту на реєстрацію відбувається перевірка, введених користувачем, електронної пошти та паролю на те, чи відповідають вони певним вимогам, адже формат пошти повинен відповідати стандартному вигляду, а пароль повинен бути двошим 8 символів, містити хоча б один великий символ, один малий символ та цифру.(рис. 4.17, 4.18).

```python
def validate_email(email):
    import re
    regex = r'^[^\s@]+@[^\s@]+\.[^\s@]+$'
    return re.match(regex, email)
```

Рисунок 4.17 - Функція валідації запропонованої користувачем електронної пошти

```python
def validate_password(password):
    import re
    regex = r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,}$'
    return re.match(regex, password)
```

Рисунок 4.18 - Функція валідації запропонованого користувачем паролю

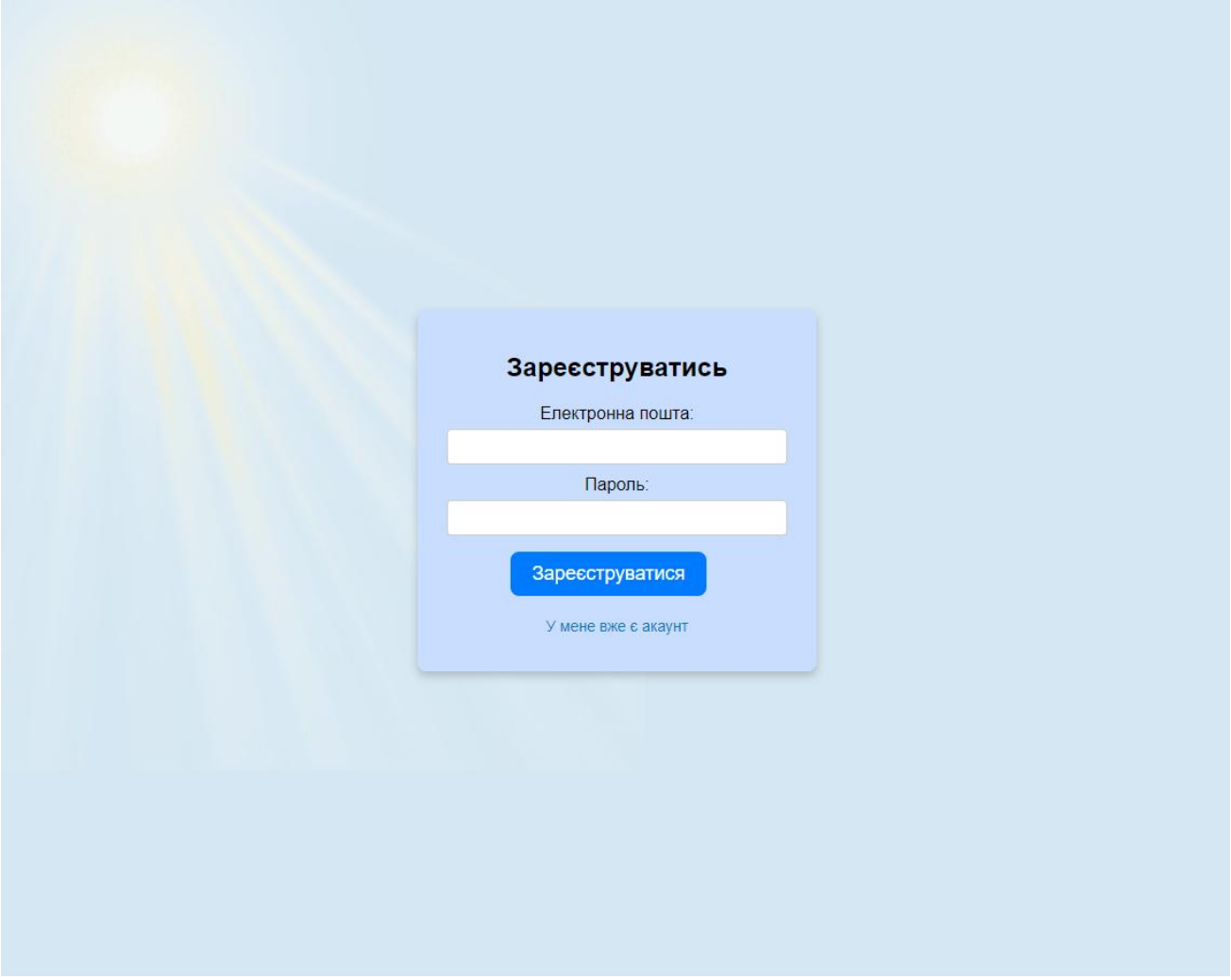

Рисунок 4.19 - Вікно реєстрації нового користувача

Авторизація користувачів відбувається за допомогою їхньої електронної адреси та пароля, що були задані при реєстрації. Система перевіряє наявність введених даних у базі, порівнює хеш введеного пароля з збереженим хешем. У разі успішної авторизації користувач отримує доступ до свого профілю і функціональних можливостей веб-застосунку. (рис. 4.19)

```
const handleSubmit = async (event) => {
    event.preventDefault();
    if (!validateForm()) {
        setError('Please fill in all fields.');
        return;
    }

    try {
        const response = await fetch('http://localhost:5000/login', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({ email, password }),
        });

        const data = await response.json();
        if (response.status === 200) {
            localStorage.setItem('token', data.token);
            localStorage.setItem('email', data.email);
            console.log('sdf')
            navigate('/main-window');
        } else {
            throw new Error(data.message || 'Login failed');
        }
    } catch (error) {
        setError(error.message);
    }
};
```

Рисунок 4.20 - Функція надсилання запиту на авторизацію

```
@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    email = data.get('email')
    password = data.get('password')
    cur = conn.cursor()
    cur.execute("SELECT email, password FROM users WHERE email = %s", (email,))
    user = cur.fetchone()
    cur.close()

    if user:
        hashed_password = user[1]
        if isinstance(hashed_password, memoryview):
            hashed_password = hashed_password.tobytes()

        if bcrypt.checkpw(password.encode('utf-8'), hashed_password):
            token = jwt.encode({'user_id': user[0], 'exp': datetime.now(timezone.utc)
                                + timedelta(hours=24)}, SECRET_KEY, algorithm='HS256')
            return jsonify({'token': token, 'email': user[0]}), 200
        else:
            return jsonify({'error': 'Invalid login credentials'}), 401
    else:
        return jsonify({'error': 'User not found'}), 404
```

Рисунок 4.21 – Функція опрацьовування запиту на авторизацію

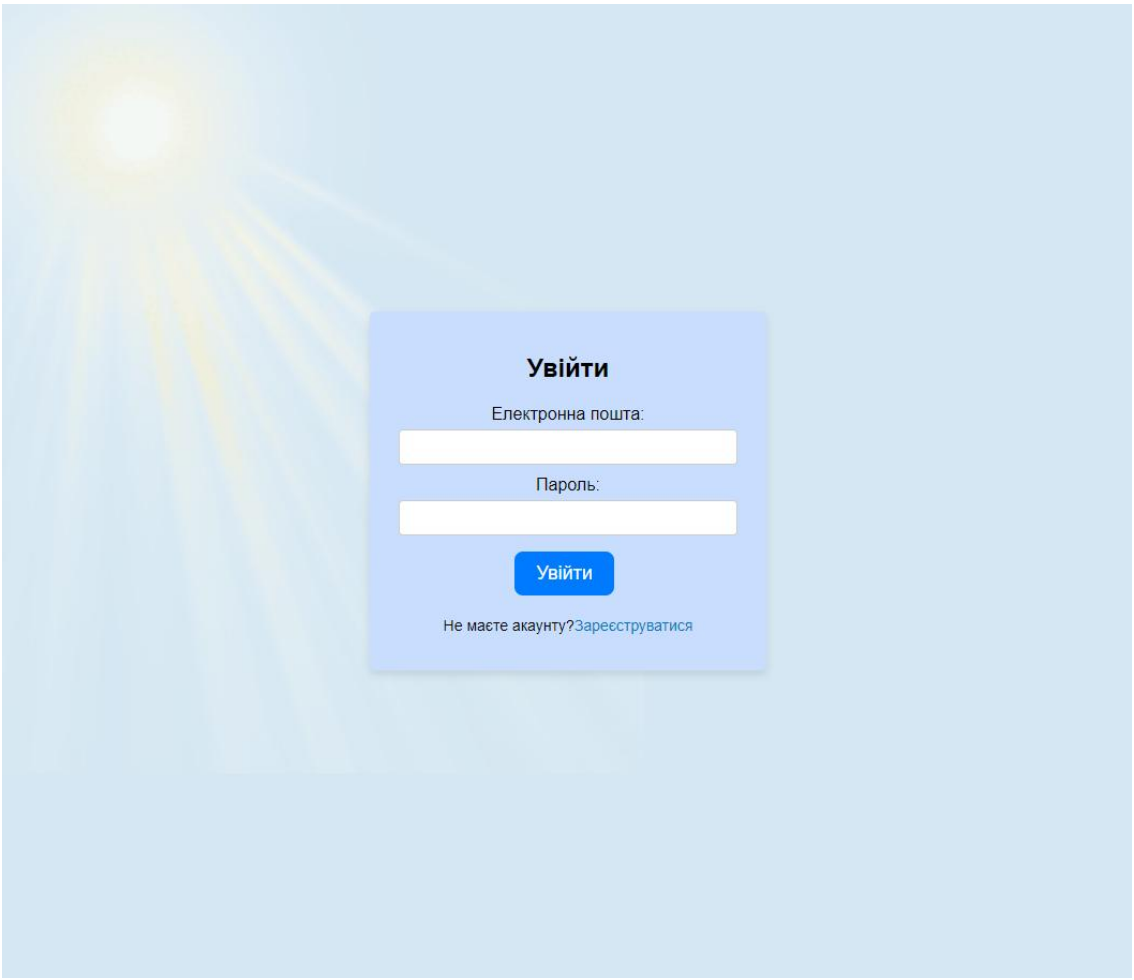

Рисунок 4.22 – Вікно авторизації користувача

Після успішної авторизації, в системі генерується токен сесії, який використовується для ідентифікації користувача у подальших запитах до сервера. Токен забезпечує засоби безпеки, обмежуючи доступ до ресурсів веб-застосунку та зберігаючи стан авторизації користувача.

```
if bcrypt.checkpw(password.encode('utf-8'), hashed_password):
    token = jwt.encode({'user_id': user[0], 'exp': datetime.now(timezone.utc)
                        + timedelta(hours=24)}, SECRET_KEY, algorithm='HS256')
    return jsonify({'token': token, 'email': user[0]}), 200
```

Рисунок 4.23 – Генерація токена при успішній авторизації

```python
def token_required(f):
    @wraps(f)
    def decorated(*args, **kwargs):
        token = None

        if 'Authorization' in request.headers:
            token = request.headers['Authorization'].split(" ")[1]

        if not token:
            return jsonify({'message': 'Token is missing!'}), 401

        try:
            data = jwt.decode(token, SECRET_KEY, algorithms=["HS256"])
            current_user = data['user_id']
        except:
            return jsonify({'message': 'Token is invalid!'}), 401

        return f(current_user, *args, **kwargs)

    return decorated
```

Рисунок 4.24 – Функція перевірки наявності токена

```python
@app.route('/api/verifyToken', methods=['POST'])
@token_required
def verify_token(current_user):
    return jsonify({
        'user_id': current_user,
        'message': 'Token verified successfully.'
    }), 200
```

Рисунок 4.25 – Функція відповіді на запит перевірки наявності токена

```javascript
const handleLogout = () => {
    localStorage.removeItem('token');
    localStorage.removeItem('email');
    navigate('/welcome-window');
};
```

Рисунок 4.26 – Функція завершення сесії

Завершення сесії – користувач може вийти з системи, при цьому токен на клієнті видаляється. Хоча токен залишається технічно дійсним до закінчення часу його дії, будь-які подальші запити з використанням видаленого токена будуть відхилені, оскільки клієнт не відправляє його.(рис.4.26)

4.3 Тестування та виправлення помилок

Тестування функцій веб-застосунку відбувалось за допомогою ручної перевірки нашого веб-застосунку, що включало у себе наступне:

- Тестування реєстрації нового користувача;

- Тестування авторизації користувача;
- Тестування перетворення тексту в мову;
- Тестування збереження перетвореного тексту за заданим іменем;
- Тестування пошуку за назвою;
- Тестування при спробі відтворити вибраний аудіофайл;
- Тестування при спробі завершити сесію;

Тестування проводилось з урахування усіх можливих варіацій створення, збереження, пошуку та відтворення аудіофайлів.

Впродовж усього процесу розробки веб-застосунку було виявленно та виправлено чимало помилок.

Однією з останніх помилок було неправильне складання шляху для збереження запису користувачем, що в свою чергу після зчитування, при спробі відтворити було неможливим, через невідповідність вмісту та формату.

```
customFileName += ".mp3"
const uniqueFilename = `${customFileName}-${Date.now()}.mp3`;
uploadFileToS3(blob, uniqueFilename);
```

Рисунок 4.27 – Код пов’язаний з неправильним формуванням шляху збереження

```
const customPath = userEmail + '/' + (customFileName || 'audio_file') + '.mp3';
uploadFileToS3(blob, customPath);
```

Рисунок 4.28 – Виправлений код формування шляху збереження

Виправлення помилки полягало у тому, що при зчитуванні користувацької назви для аудіофайлу filename додавалось закінчення mp3, що й перешкоджало формуванню правильного формату файлу.

РОЗДІЛ 5.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

5.1 Розробка логіко-імітаційної моделі виникнення травм і аварій

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об'єктивного критерію рівня небезпеки для конкретного об'єкта. Таким показником вибрана ймовірність виникнення аварії, травми залежно від явища, що досліджується.

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми в процесі створення мікрокліматичних умов у приміщенні оцінюють відповідні небезпечні події. Кожній з них присвоїмо ймовірність виникнення:

Шифр	Назва події	Ймовірність
P_1	Відсутність захисного заземлення	0,02
P_2	Пошкодження захисного заземлення	0,04
P_3	Спрацювання складових захисту	0,1
P_4	Неправильна експлуатація захисту	0,02
P_5	Відсутність профілактичних заходів	0,2
P_6	Відсутність захисного щита	0,12
P_7	Недотримання правил вибору взуття	0,15
P_8	Незнання правил техніки безпеки	0,1
P_9	Відсутність засобів індивідуального захисту	0,2
P_{10}	Легковажність	0,08

На основі наведених подій будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічна інтерпретація якої зображено на рис. 5.1.

Розрахуємо ймовірності виникнення подій, що формують логіко-імітаційну модель процесів створення мікрокліматичних умов. Розглянемо травмонебезпечну ситуацію, що виникає за умови роботи працівників із електронебезпекою.

Підставивши дані ймовірностей базових подій у формулу, отримаємо ймовірність події 13: $P_{13} = 0,2+ 0,4 – 0,2 \cdot 0,4 = 0,0592$.

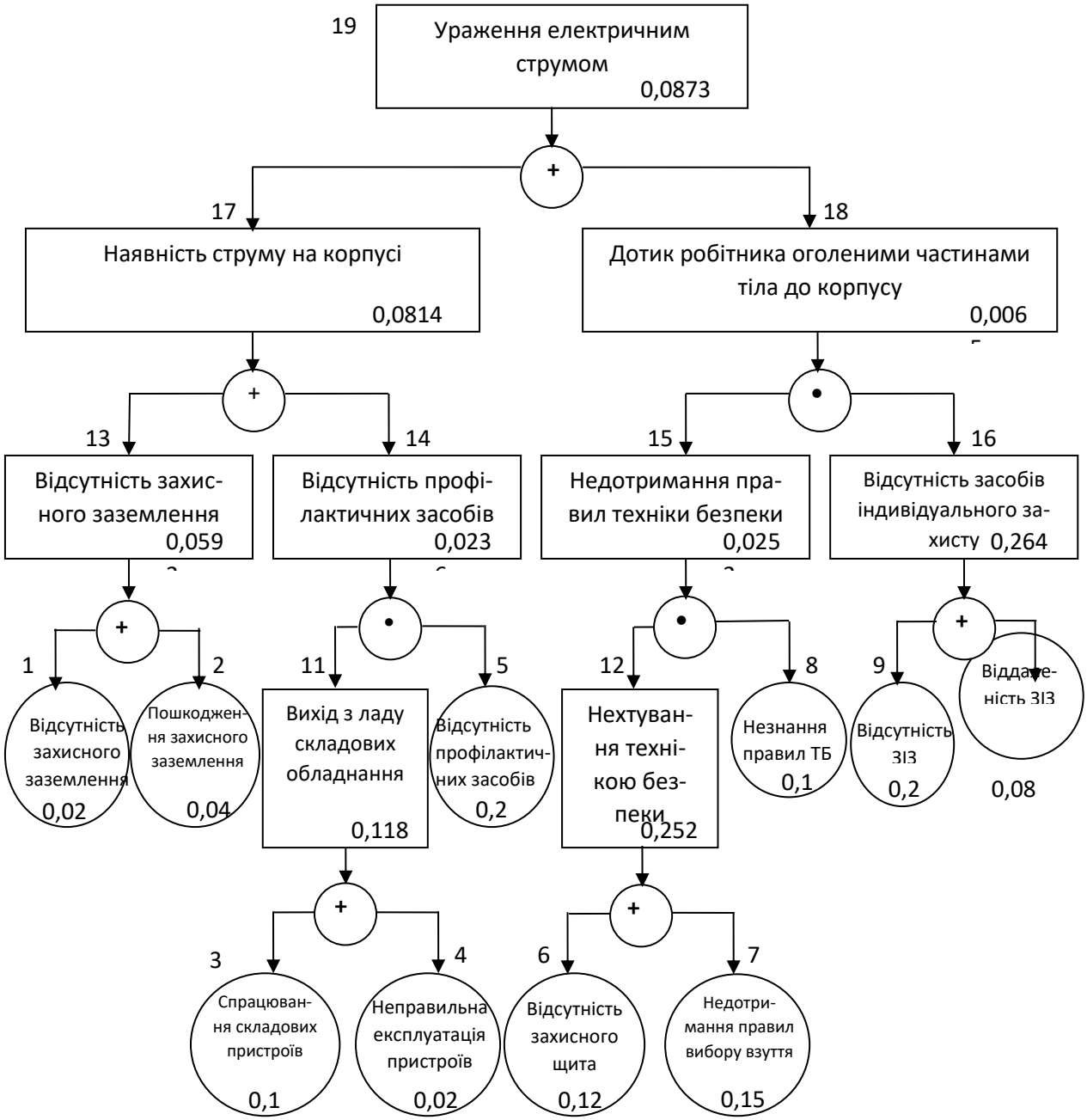

Рисунок 5.1 – Матриця логічних взаємозв’язків між окремими подіями травмонебезпечної ситуації

Аналогічно визначаємо ймовірність інших подій:

$$P_{11} = P_4 + P_5 - P_4P_5 = 0{,}3 + 0{,}4 - 0{,}3 \cdot 0{,}4 = 0{,}118.$$

$$P_{12} = P_6 + P_7 - P_6P_7 = 0{,}3 + 0{,}5 - 0{,}3 \cdot 0{,}5 = 0{,}252.$$

$$P_{16} = P_9 + P_{10} - P_9P_{10} = 0{,}2 + 0{,}15 - 0{,}2 \cdot 0{,}15 = 0{,}264.$$

$$P_{14} = P_{11} \cdot P_5 = 0{,}118 \cdot 0{,}2 = 0{,}0236.$$

$$P_{15} = P_{12} \cdot P_8 = 0{,}252 \cdot 0{,}1 = 0{,}0252.$$

$$P_{17} = P_{13} + P_{14} - P_{13} \cdot P_{14} = 0{,}592 + 0{,}0236 - 0{,}0592 \cdot 0{,}0236 = 0{,}0814.$$

$$P_{18} = P_{15} \cdot P_{16} = 0{,}264 \cdot 0{,}0252 = 0{,}0065.$$

$$P_{19} = P_{17} + P_{18} - P_{17} \cdot P_{18} = 0{,}0065 + 0{,}0814 - 0{,}0065 \cdot 0{,}0814 = 0{,}0873.$$

Таким чином, ймовірність перекидання машини та наслідкового виникнення травми працівника є досить мала і становить – $P_{19} = 0{,}0873$.

5.2. Планування заходів із покращення умов праці

До заходів щодо покращення умов праці належать всі види діяльності, спрямовані на попередження, нейтралізацію або зменшення негативної дії шкідливих і небезпечних виробничих факторів на працівників.

Рівень умов праці оцінюють порівнянням за фактичними і нормативними значеннями узагальнених (групових) показників.

Заходи щодо поліпшення умов праці здійснюють з метою створення безпечних умов праці шляхом:

- доведення до нормативного рівня показників виробничого середовища за елементами умов праці;
- захисту працівників від дії небезпечних і шкідливих виробничих факторів.

До показників ефективності заходів щодо поліпшення умов праці належать:

а) зміни стану умов праці:

- зміна кількості засобів виробництва, приведених у відповідність до вимог стандартів безпеки праці;
- покращання санітарно-гігієнічних показників;
- покращання психофізичних показників, зменшення фізичних і нервово-психічних навантажень, в т.ч. монотонних умов праці;

- покращання естетичних показників, раціональне компонування робочих місць і впорядкування робочих приміщень;

б) соціальні результати заходів:

- збільшення кількості робочих місць, що відповідають нормативним вимогам;
- зниження рівня виробничого травматизму;
- зменшення кількості випадків професійних захворювань;
- зменшення плинності кадрів через незадовільні умови праці;
- престиж та задоволення працею.

Отже, на покращення охорони праці потрібно виділити кошти на відновлення вентиляційних систем у ремонтних майстернях, естетично оформити приміщення офісу, відновити кабінет з охорони праці, поновити протипожежний інвентар.

5.3 Безпека в надзвичайних ситуаціях

Актуальність проблеми природно-техногенної безпеки для населення і території, зумовлена зростанням втрат людей, що спричиняється небезпечними природними явищами, промисловими аваріями та катастрофами. Ризик надзвичайних ситуацій природного та техногенного характеру невпинно зростає, тому питання захисту цивільного населення від надзвичайних ситуацій на сьогодні є дуже важливе.

У системі цивільної оборони окремого господарства необхідно забезпечити захист населення таким чином:

Укриття в захисних спорудах, якому підлягає усе населення відповідно до приналежності, досягається створенням фонду захисних споруд.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення його у позаміській зоні.

Медичний захист проводиться для зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення епідеміологічного благополуччя в районах надзвичайних ситуацій.

Радіаційний і хімічний захист включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного та хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об’єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення у позаміській зоні.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було успішно розроблено веб-застосунок для ефективного засвоєння навчального матеріалу через озвучування текстів. Застосунок інтегрує сучасні технології синтезу мовлення, забезпечуючи користувачам можливість перетворення текстової інформації в аудіоформат. Використання React і Flask дозволило створити надійний і масштабований інтерфейс, а інтеграція з AWS S3 забезпечила ефективне зберігання та керування аудіофайлами. Такий підхід сприяє більш глибокому залученню користувачів та покращує процес самонавчання.

Використання веб-застосунку для синтезу мовлення має велику доцільність у навчанні учнів та студентів, оскільки сприяє покращенню засвоєння матеріалу через слухове сприйняття. Можливість використання аудіоформату для кращого засвоєння інформації учнями з різними стилями навчання є також важливим фактором. Аудіоматеріали можна використовувати під час виконання інших завдань, що дозволяє ефективніше використовувати час.

Проект має великий потенціал для подальшого розвитку та оптимізації, з численними можливостями для вдосконалення. Одним із напрямків розвитку є розширення підтримуваних форматів документів, що дозволить збільшити універсальність застосунку та забезпечити можливість роботи з різноманітними типами матеріалів. Вдосконалення інтерфейсу користувача, зокрема покращення навігації та доступності функціональних елементів, також є важливим кроком, який зробить застосунок зручнішим для широкої аудиторії. Крім того, розвиток системи персоналізованих налаштувань дозволить кожному користувачеві кастомізувати параметри синтезу мовлення відповідно до своїх особистих уподобань, що покращить досвід використання. Створення голосів для озвучування, що базуються на модельному навчанні, стане ще одним значним кроком у персоналізації застосунку, дозволяючи користувачам вибирати найбільш підходящий голос для покращення сприйняття інформації. Ці удосконалення не лише покращать поточний функціонал, але й значно розширять можливості застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. React Documentation - офіційна документація бібліотеки React, що надає інформацію про основи роботи, компоненти, стани та життєвий цикл компонентів. Режим доступу: https://react.dev/learn

2. Flask Documentation - офіційна документація веб-фреймворку Flask, яка містить інструкції з налаштування, розробки API та інтеграції з іншими сервісами. Режим доступу: https://flask.palletsprojects.com/en/1.1.x/

3. PostgreSQL Documentation – документація системи управління базами даних PostgreSQL, що описує основні принципи роботи, налаштування та операції з базами даних. Режим доступу: https://www.postgresql.org/docs/

4. AWS S3 Documentation - документація сервісу Amazon Web Services S3, яка надає інформацію про налаштування сховищ, управління доступом та інтеграцію з іншими AWS сервісами. Режим доступу: https://docs.aws.amazon.com/s3/

5. AWS IAM Documentation - документація сервісу Amazon Web Services IAM, яка надає інформацію про управління доступом користувачів. Режим доступу: https://docs.aws.amazon.com/iam/?icmpid=docs_homepage_security

6. gTTS Documentation - документація Python бібліотеки gTTS, що дозволяє перетворювати текст у мовлення за допомогою Google Text-to-Speech API. Режим доступу: https://gtts.readthedocs.io/

7. React Router Documentation - документація бібліотеки React Router, яка описує маршрутизацію в односторінкових додатках (SPA). Режим доступу: https://reactrouter.com/en/6.23.1/start/tutorial

8. Інновації у сфері TTS технологій - статті та дослідження про сучасні досягнення у технологіях синтезу мовлення, з акцентом на персоналізацію та якість голосу. Режим доступу: https://fliki.ai/blog/future-text-to-speech

9. Azat MardanMorten Barklund. React Quickly, Second Edition 2nd ed. Edition. Manning Publications, 2023. 456 p.

10. "Deep Learning for Natural Language Processing" by Palash Goyal, Sumit Pandey, Karan Jain, Apress. 2018, 512c.

11. "Statistical Methods for Speech Recognition" by Frederick Jelinek, MIT Press, 1997. 646c.

12. "Neural Text-to-Speech Synthesis" by Xu Tan, Tao Qin, Sheng Zhao, Tie-Yan Liu, Springer, 2023, 238c.

13. "Speech and Language Processing" by Daniel Jurafsky, James H. Martin, Pearson, 3d edition, 2023. 816c.

14. " Stephan Raaijmakers. Deep Learning for Natural Language Processing. Manning Publications, 2022. 296 p.

15. Text-to-Speech Synthesis" by Paul Taylor, Cambridge University Press, 2009. 528c.

16. Tejas Kumar. Fluent React: Build Fast, Performant, and Intuitive Web Applications 1st Edition. O'Reilly Media, 2024. 334p/

17. https://www.free-tts-for-streamers.com/uk/help/what-is-difference-between-concatenative-and-neural-text-to-speech.