МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Онлайн-система синтезованого голосового озвучування текстів»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студентка групи ІТ.м-01 Оношко Вікторія Валеріївна

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «_» грудня 2021 р.

Науковий керівник _____ к.т.н., Кузнєцов Е. Г.
(підпис)

Голова комісії _____ Шифрін Д.М.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2021

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. кафедри ІТ

_____ В. В. Шендрик
«__» _____ 2021 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Оношко Вікторія Валеріївна
(прізвище, ім'я, по батькові)

1 Тема проекту *Онлайн-система синтезованого голосового озвучування текстів*

затверджена наказом по університету від « 29 » 10 2021 р. № 0787-IV

2 Термін здачі студентом закінченого проекту «_10_» ___грудня___ 2021 р.

3 Вхідні дані до проекту технічне завдання на розробку онлайн-системи синтезованого голосового озвучування текстів

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити аналіз предметної області, моделювання та проектування, реалізація онлайн-системи синтезованого голосового озвучування текстів

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність проблеми, мета та задачі проекту, аналогічні онлайн-системи, функціональні вимоги, інструменти та засоби реалізації, моделювання програмного продукту, етапи розробки онлайн-системи синтезованого голосового озвучування текстів, висновки

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Ознайомлення та аналіз предметної області	22.9.21 – 28.9.21	
2	Дослідження аналогів	28.9.21 – 29.9.21	
3	Визначення ідеї проекту	30.9.21 – 1.10.21	
4	Аналіз документації PyTorch	1.10.21 – 8.10.21	
5	Аналіз моделей TTS	8.10.21– 14.10.21	
6	Планування WBS	14.10.21 – 18.10.21	
7	Планування OBS	18.10.21 – 19.10.21	
8	Створення календарного плану	19.10.21 – 22.10.21	
9	Визначення ризиків	22.10.21 – 26.10.21	
10	Розробка структури проекту	27.10.21 – 2.11.21	
11	Створення набору даних	2.11.21 – 9.11.21	
12	Розробка та навчання кодера	9.11.21 – 19.11.21	
13	Розробка та навчання синтезатора	9.11.21 – 19.11.21	
14	Розробка та навчання вокодера	9.11.21 – 19.11.21	
15	Реалізація структури проекту	19.11.21 – 26.11.21	
16	Тестування продукту	26.11.21 – 2.12.21	
17	Проектна документація	1.10.21 – 10.12.21	

Магістрант _____ Оношко В.В.

Керівник роботи _____ к.т.н., Кузнєцов Е.Г.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Онлайн-система синтезованого голосового озвучування текстів».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 30 найменувань, додатків. Загальний обсяг роботи − 69 сторінок, у тому числі 48 сторінок основного тексту, 3 сторінки списку використаних джерел, 18 сторінок додатків.

Кваліфікаційну роботу магістра присвячено розробці онлайн-системи синтезованого голосового озвучування текстів.

В роботі проведено аналіз існуючих аналогів рішень у сфері онлайн-систем голосового озвучування текстів.

Виконано проектування продукту проекту за допомогою UML на основі проведеного структурно-функціонального аналізу продукту проекту.

Крім того, описано практичну частину реалізації проекту, детально зображено процес створення системи на основі нейронних мереж для синтезу мовлення та процес розробки інтерфейсу та ітеграції із системою.

Результатом проведеної роботи є розроблена онлайн-система синтезованого голосового озвучування текстів.

Практичне значення роботи полягає у використанні синтезу мовлення як допоміжного засобу:

– для людей з порушенням зору;
– для людей із захворюваннями, які впливають на їх голос;
– для людей з обмеженими можливостями навчання та ін.

Ключові слова: онлайн-система, озвучування текстів, синтез голосу, нейромережі, Python, PyTorch.

ЗМІСТ

ВСТУП

Останнім часом все більше штучний інтелект (Artificial Intelligence) змінює робочі процеси у всіх галузях у всьому світі. Все частіше управління компаніями проводиться на основі великих даних, тому потреба в технологіях штучного інтелекту зростає. Технології надають значні обчислювальні можливості, інструменти та алгоритми для реалізації різноманітних проектів: від систем розпізнавання мови та рекомендацій до медичної візуалізації та покращеного управління постачанням.

Завдяки штучному інтелекту, глибокому навчанню (Deep Learning) та аналізу даних (Data Analysis) стали можливими створення розумних міст, революція у сфері аналізу даних та інші досягнення. Ці технології дозволяють втілювати найамбітніші проекти. Зважаючи на це, можемо вважати розробку додатків на основі штучного інтелекту актуальною.

У світі, де нові технології з’являються з експоненційною швидкістю, а наше повсякденне життя все більше оточується динаміками та звуковими хвилями, технологія перетворення тексту на мову розширює способи нашого спілкування.

Моделі глибокого навчання знедавна стали переважати у багатьох сферах машинного навчання (Machine Learning). Процес синтезу мовлення, також названий як технологія Text-To-Speech (TTS), не є винятком.

Ця технологія є новаторською, і великі прориви у цій галузі відбуваються регулярно. Технологія перетворення тексту поступово впроваджується і в наше повсякденне життя. Популярні пристрої вже включають в себе віртуальних помічників на основі штучного інтелекту, таких як Amazon Alexa і Google Assistant. Технологія перетворення тексту на мову була використана в рекламних цілях з появою інтерактивних голосових оголошень. TTS також може бути оптимальним інструментом для перетворення величезних масивів тексту на відтворювані аудіодані.

Можемо придумати безліч застосувань даної технології, що дозволять вирішити ту чи іншу проблему. Перетворення тексту на мову використовується в різних цілях як допоміжна технологія, яка допомагає зробити світ доступнішим, коли

йдеться про те, як ми говоримо і слухаємо. Наведемо кілька основних способів використання технології синтезу мовлення як допоміжного засобу:

– для людей з порушенням зору;

– для людей із захворюваннями, які впливають на їх голос;

– для людей з обмеженими можливостями навчання (від 15 до 20% населення світу страждають від нездатності до навчання через неграмотність);

– для вивчення нової мови;

– для використання в дорозі (слухати будь-який текст, перебуваючи у дорозі або одночасно виконуючи кілька завдань);

Побачивши дані проблеми, стало очевидним, що людині в настільки сучасному світі можна допомогти зробити його доступнішим. Більше того, ідея проекту полягає в тому, щоб за допомогою штучного інтелекту отримати змогу не лише використати один із наданих голосів, але і синтезувати озвучування тексту власним голосом. Щоб маючи зразок аудіозапису голосу змогти в повній мірі відтворити будь-який текст.

Об’єктом дослідження дипломного проектування є перетворення тексту на мову (TTS).

Предметом дослідження є використання методів машинного навчання для синтезованого озвучування текстів.

На основі визначеної ідеї роботи було сформовано мету проекту, яка полягає у розробці онлайн-системи синтезованого голосового озвучування текстів. Основними задачами для досягнення даної мети з якими ми зустрінемося в ході виконання розробки продукту проекту є:

– детальне вивчення предметної області;

– вибір технологій розробки продукту;

– розробка системи на основі нейронних мереж для синтезу мовлення (TTS), яка може генерувати звук різними голосами, у тому числі тими, що не були відомі під час навчання;

– розробка інтерфейсу онлайн системи;

– інтеграція системи із інтерфейсом;

– проведення тестування продукту.

Практичне значення даної роботи полягає у застосуванні для озвучування текстів, вона допоможе користувачам слухати книги, статті, навчальні документи або будь-які твори з великою кількістю слів.

Дана розробка буде корисна як звичайним користувачам, які можуть використовувати будь-який контент у будь-якому місці. Так і користувачам із описаними раніше проблемами, такими як порушення зору чи захворювання, які впливають на голос.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальна характеристика предметної області

Технологія перетворення тексту в мовлення та синтез мовлення є одними з найсучасніших досягнень штучного інтелекту. Крім того, що людина може просто вводити текст для відтворення комп'ютером, голосові обчислення дають змогу з'явитися повністю оригінальним синтетичним голосам.

Ці голоси дозволяють людям відновити втрачений голос або створити голос для бренду, урізноманітнити способи взаємодії з комп'ютерами та перетворювати будь-яку кількість тексту в мовлення природного звучання.

Наведемо декілька галузей, у яких перетворення тексту на мову дуже вдало використовується. Однією із них є клас, у якому учні, які поки не вміють читати, але можуть зрозуміти коли з ними розмовляють. Замість того, щоб вчитель проводив час з кожною дитиною і читав для неї, дуже корисним може бути просте перетворення тексту на мову. Крім того, понад 750 мільйонів молодих людей і дорослих у всьому світі не мають навичок читання або мають справу з неписьменністю, а від 15 до 20% населення світу страждають від нездатностей до навчання, які пов'язані з мовою [8]. Серед таких проблем найчастіше зустрічається дислексія. Надання аудиторії можливості почути будь-який фрагмент контенту, прочитаного вголос, спрощує доступ знань для людей з широким діапазоном рівнів грамотності та проблем з навчанням.

Ще один приклад використання TTS, на якому хочеться наголосити, є можливість сприймать інформацію людям із вадами зару. За оцінками WHO, 135 мільйонів людей у всьому світі мають ту чи іншу форму порушення зору, 40 – 45 мільйонів із яких сліпі [9]. Технологія перетворення тексту на мовлення дозволяє тим, хто не може читати з екрана, отримати доступ до письмового контенту, слухаючи його. Якщо людина не має будь-якої форми порушення зору, читання протягом тривалого часу може викликати значну зорову напругу. У такому випадку, технологія

перетворення тексту на мову є цінним інструментом, який пропонує читачам відірватися від погляду екран, не роблячи паузи у взаємодії з текстовим матеріалом.

Також дана технологія може допомогти в передачі голосу тим, хто страждає порушенням мови або страждає захворюванням, яке вплинуло на їхню здатність говорити.

Технологія перетворення тексту на мовлення і синтез голосу – одні з передових досягнень, які стали можливими завдяки штучному інтелекту. Окрім простого надання людині можливості вводити текст для озвучування на комп'ютері, голосові обчислення дозволяють створювати повністю оригінальні синтетичні голоси. Ці голоси дозволяють людям відновити втрачений голос, вести більш реалістичні розмови з комп'ютерами і перетворювати будь-який обсяг розмовного тексту на мову, що звучить досить природно.

1.2 Огляд останніх досліджень і публікацій

Перетворення тексту на мовлення або синтез мовлення, метою якого є синтез розбірливої та природної мови із заданим текстом, є гарячою темою досліджень у спільнотах мовлення, мови та машинного навчання, а також має широке застосування в галузі.

У 2016 році почали з'являтися моделі глибокого навчання, які створювали більш природне звучання, ніж традиційні підходи конкатенації. З того часу велика частина дослідницької діяльності була зосереджена на тому, щоб зробити ці моделі більш ефективними, звук більш природним або навчати їх наскрізними методами [1].

Стосовно якості синтезованого мовлення, у дослідженні 2017 року [2] автори демонструють мову, подібну на людську, максимально природну. Цікавим є те, що природність мовлення найкраще оцінюється суб'єктивними показниками, а порівняння з реальним людським мовленням приводить до висновку, що може

існувати таке явище, як мовлення, що звучить природніше за людську мову. Хоча насправді деякі стверджують, що межа природності людини вже досягнута [3].

Також, очевидно, що набори професійно записаних даних є дефіцитним ресурсом. Синтез голосу з правильною вимовою, інтонацією та мінімумом фонового шуму вимагає навчальних даних з такими ж властивостями. Крім того, ефективність даних залишається основною проблемою глибокого навчання. Дослідження призвели до створення фреймворків для перетворення голосу та клонування голосу. Вони відрізняються тим, що перетворення голосу є формою передачі стилю на мовному сегменті від одного голосу до іншого, тоді як клонування голосу полягає у захопленні голосу мовця для виконання перетворення тексту в мовлення [4].

Хоча повне навчання моделі TTS з одним спікером технічно є моделлю клонування голосу, інтерес досліджувачів скоріше полягає в створенні фіксованої моделі, здатної генерувати нові голоси із невеликої кількості даних. Загальноприйнятий підхід полягає в тому, щоб обумовити модель TTS, навчену узагальнювати нові голоси мовців на вбудовуванні голосу для клонування [5-7]. Вбудовування є малорозмірним і отримане за допомогою моделі кодера звуку, яка приймає еталонне мовлення як вхідну інформацію.

Цікаво, що існує велика розбіжність між тривалістю еталонного мовлення, необхідного для клонування голосу, серед різних методів, починаючи від пів години на одного мовця до всього кількох секунд. Цей фактор зазвичай визначає відповідність згенерованого голосу щодо справжнього голосу мовця.

1.3 Аналіз програмних продуктів – аналогів

Перед початком роботи над власним продуктом, було вирішено провести аналіз аналогічних онлайн-систем озвучування тексту. В результаті здійсненого пошуку за заданою нами темою роботи, було знайдено продукти-аналоги.

Сайт VoxWorker.com – це онлайн-сервіс для озвучування тексту, який може перекладати текст в аудіозапис. Текст може бути озвучено англійською або українською мовами. Для синтезу мови може бути обрано жіночий або чоловічий голоси з різним тембром або акцентом, налаштувати швидкість та висоту голосу. Результат озвучки можна зберегти в файл формату mp3, що є найпопулярнішим форматом для аудіозаписів. Проте, сервіс не зберігає написані тексти. А також, всі аудіо файли видаляються з сервера через годину. За необхідності можна налаштувати постійне зберігання файлів [10].

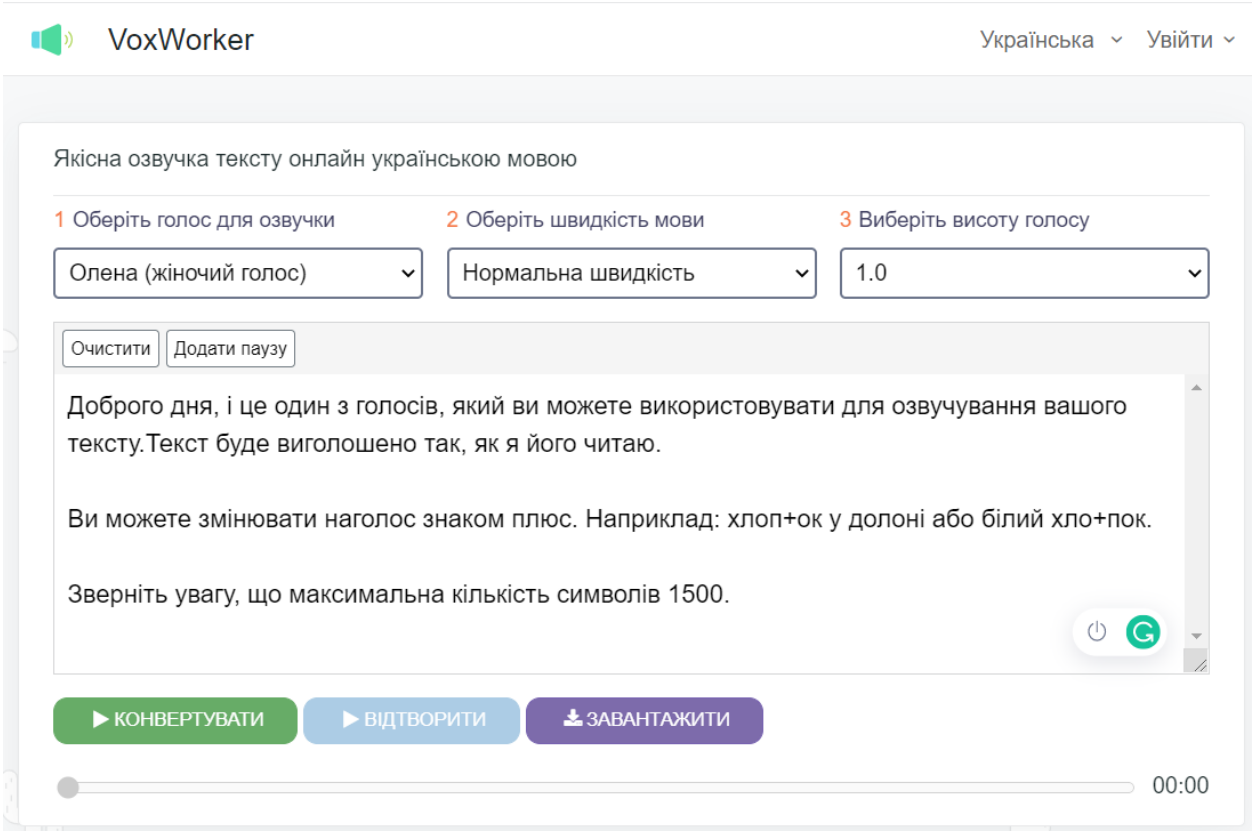


Рисунок 1.1 – Головна сторінка сайту VoxWorker

Сайт textfromtospeech.com – це схожий на попередній, проте більш розширений сервіс. Даний сервіс для голосового відтворення текстів дозволяє озвучити надрукований текст різними мовами [11], надає реалістичне звучння голосів, можливість регулювати гучність, швидкість та висоту відтворення звуку [11]. Результат можна зберігати в аудіоформаті. В його базі набагато більше пропонованих голосів, та підтримка більшого різноманніття мов для озвучення, проте немає

української. Крім того, є можливість озвучення довгих текстів, яку надає не кожен сервіс, і що також важливо, надає безкоштовний онлайн-доступ. Також в функціоналі даного сайту є не тільки перетворення тексту на голос, а і навпаки, перетворення мовлення на текст, а також звук до тексту і відео в текст.

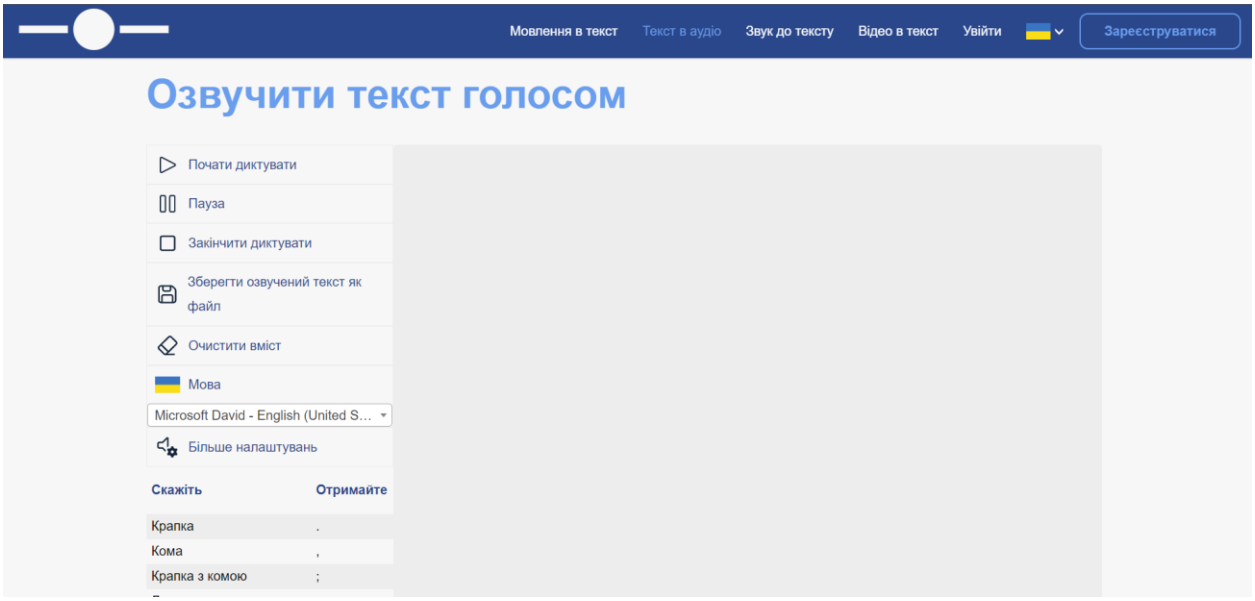


Рисунок 1.2 – Сторінка текст в аудіо сайту textfromtospeech.com

Сервіс ZVUKOGRAM не має додаткових функцій, як попередній сервіс, але також має велику кількість доступних мов та голосів. Можна відразу озвучити велику статтю, і все це буде в одному файлі. Можна також створювати діалоги різними голосами.

На відміну від попереднього безкоштовного сервісу, даний сервіс має обмежений обсяг безкоштовного озвучення текстів, асортимент голосів також є обмеженим, крім того присутній ліміт символів.

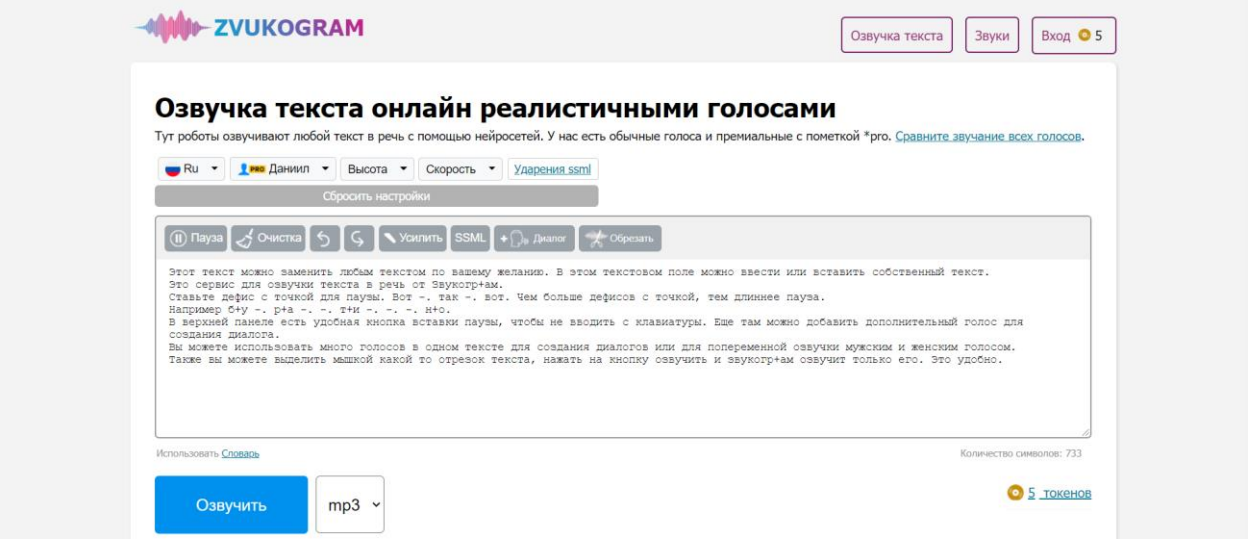


Рисунок 1.3 – Сторінка сервісу ZVUKOGRAM

На основі опису програмних продуктів-аналогів, було сформовано порівняльну таблицю з метою зрівнити характеристики аналогів, яку представлено в табл. 1.1.

Таблиця 1.1 – Порівняльна характеристика аналогів

Критерій	voxworker.com	textfromtospeech.com	zvukogram.com
Багатомовність	+ (3)	+ (15)	+ (32)
Різноманіття голосів	Невелике	Невелике	+
Налаштування характеристик звуку	+	+	+
Безкоштовне використання	+	+	Дуже обмежене
Ліміт символів	+	-	+
Збереження в популярних аудіо форматах	+	+	+

З порівняльної характеристики видно, що онлайн системи озвучування текстів можуть бути багатомовними, мати різноманітне звучання, і в той же час бути дуже обмеженими і лімітованими. Безкоштовні системи часто не дають якісного звучання, та не є достатньо універсальними та різноманітними.

На відміну від вже існуючих аналогів очікуваним результатом є онлайн система озвучування текстів, особливістю якої можна вважати те, що користувачі матимуть можливість створювати власне різноманіття голосів маючи зразки запису голосу. Із зразків аудіо користувач зможе синтезувати власний голос та озвучувати тексти, використовуючи його, не залежно від розміру контенту.

2 ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Мета та задачі дослідження

Метою даного проекту є розробка онлайн системи синтезованого голосового озвучування текстів, за допомогою якого користувач зможе легко озвучити необхідні тексти. Вона допоможе користувачам слухати книги, статті, навчальні документи або будь-які твори з великою кількістю слів. Система зможе в режимі реального часу синтезувати нові для неї голоси за допомогою нейронних мереж. Маючи зразок голосу, система зможе зафіксувати цифрове представлення голосу та відтворити його на іншому фрагменті тексту.

В ході використання програмного продукту користувач матиме можливість як просто запустити перетворення тексту на мову, обравши один із завчасно створених голосів, так і разпочати синтез нового голосу. Для цього користувач, маючи завчасно створений файл із зразком голосу, або ж записавши голос одразу на місці, додає цей приклад звучання і система згенерує новий варіант голосу. Після цього користувачу необхідно ввести фрагмент тексту для озвучування та почати відтворення.

Основними задачами для досягнення мети в ході виконання розробки продукту проекту, є:

– детальне вивчення предметної області та програмних продуктів-аналогів, вибір технологій розробки;

– розробка системи на основі нейронних мереж для синтезу мовлення, яка може генерувати звук різними голосами, у тому числі тими, що не були відомі під час навчання;

– розробка інтерфейсу та інтеграція із системою;

– проведення тестування продукту.

Більш детальний опис наводиться у технічному завданні на розробку продукту проекту у додатку А.

Одним з найбільш важливих моментів в розробці системи перетворення тексту на мову є створення відповідної системи на основі нейронних мереж.

Система синтезу тексту в мову зазвичай складається з кількох етапів, таких як інтерфейс аналізу тексту, акустична модель та модуль синтезу звуку. Після огляду найбільш відомих методів перетворення тексту на мовлення в машинному навчанні було обрано методи, що будуть реалізовані у під час розробки продукту. Наша система складатиметься з трьох компонентів, якими будуть:

Мережа енкодерів мовців, яка навчена на задачі верифікації мовця, використовуючи незалежний набір даних. Він містить набір мовлення від тисяч мовців, не очищеного від шумів та без транскриптів. Мережа навчена для генерації мовлення цільового мовця. Мережа, буде основуватися на певній моделі TTS, тож для реалізації було обрано модель Generalized End-to-End Loss for Speaker Verification (GE2E). Дана модель, на відміну від попередньої моделі втрат на основі кортежів (TE2E), робить навчання моделей перевірки мовця більш ефективним. Це перевірено і підтверджено як теоретичними, так і експериментальними результатами [12].

Мережа синтезу послідовності на основі Tacotron, яка генерує спектрограму безпосередньо з тексту. Оскільки Tacotron генерує мовлення на рівні кадрів, це є значно швидшим, ніж авторегресійні методи на рівні вибірки [13];

Рекурентна мережа вокодерів для послідовного моделювання звуку на основі методу WaveNet, яка перетворює спектрограму на звукову доріжку. Компактна форма мережі дає змогу генерувати сигнал на частоті дискретизації безперервного звуку 24 кГц у 16-бітному форматі у 4 рази швидше, ніж у реальному часі на графічному процесорі [14].

2.2 Методи дослідження

Для вирішення поставленої задачі було виобрано мову Python. Вона є мовою програмування для інтелектуальних систем, тож вибір можна вважати очевидним. Python є мовою високого рівня, яка наразі стає дуже популярною, адже сфера її застосування постійно розширюється.

Широко використовуються в галузі глибокого навчання бібліотеки Python, такі як TensorFlow, PyTorch та scikit-learn, для відстеження проектів машинного навчання.

PyTorch – це бібліотека машинного навчання з відкритим вихідним кодом, заснована на бібліотеці Torch, яка використовується для таких програм, як комп'ютерний зір та обробка природної мови, в основному розроблена дослідною лабораторією Facebook AI Research (FAIR) [17].

PyTorch надає дві високорівневі функції:

– Тензорні обчислення (як наприклад, NumPy) із сильним прискоренням за допомогою графічних процесорів (GPU);

– Глибокі нейронні мережі, побудовані на системі автоматичного диференціювання на основі типів.

Для безпосереднього створення веб-додатку було обрано Flask – мікрофреймворк для веб-додатків, створений з використанням Python. Частина «мікро» в мікрофреймворку означає, що Flask прагне зробити ядро додатку простим, але розширюваним. Flask підтримує розширення для додавання функціональності до програми, так як би вона була реалізована в самому Flask.

Для розробки клієнтської частини додатку, було обрано найпопулярнішу платформу HTML, CSS та JavaScript для розробки адаптивних мобільних веб-сайтів Bootstrap.

3 МОДЕЛЮВАННЯ ТА ПРОЕКТУВАННЯ

3.1 Проектування онлайн системи

Після виконання аналізу предметної області, аналізу програмних продуктів аналогів, визначення мети та задач дослідження, та опису методів дослідження, обраних для вирішення поставленої задачі, наступним етапом є проектування онлайн-системи. Функціонал майбутньої розробки буде представлено у вигляді діаграм Use Case та IDEF0.

3.1.1 Моделювання використання онлайн системи

Моделювання використання надає інформацію про акторів, варіанти використання та діаграми сценаріїв використання в UML (діаграма Use Case). Даний тип діаграм представляє систему, яка описана як група акторів, які взаємодіють із даною системою за допомогою варіантів використання. Діаграма варіантів використання відображає функціональне призначення програмної системи, що проектується.

В даному випадку було визначено наступні актори:

– Користувач – замовник, або користувач, що використовує систему.

– Система синтезу мовлення – навчена система синтезу мовлення із тексту.

– Хмарне сховище – сховище даних, де зберігаються схеми голосів користувача.

Після визначення акторів, що взяємодіють із системою, було сформовано перелік усіх варіантів використання, необхідних для функцонування системи. Варіанти використання показують виконання певних операцій або дій та повністю відповідають вимогам до системи, які представлено в Додатку А.

Варіантами використання в даному випадку було визначено:

– Перегляд списку можливих голосів

– Генерація нового голосу
– Вибір голосу для озвучення
– Введення тексту для озвучення
– Керування паузами
– Синтезоване озвучення введеного тексту
– Прослуховування озвученого тексту

На основі сформованих акторів системи та варіантів використання було розроблено Use Case діаграму, яку наведено на рис. 3.1.

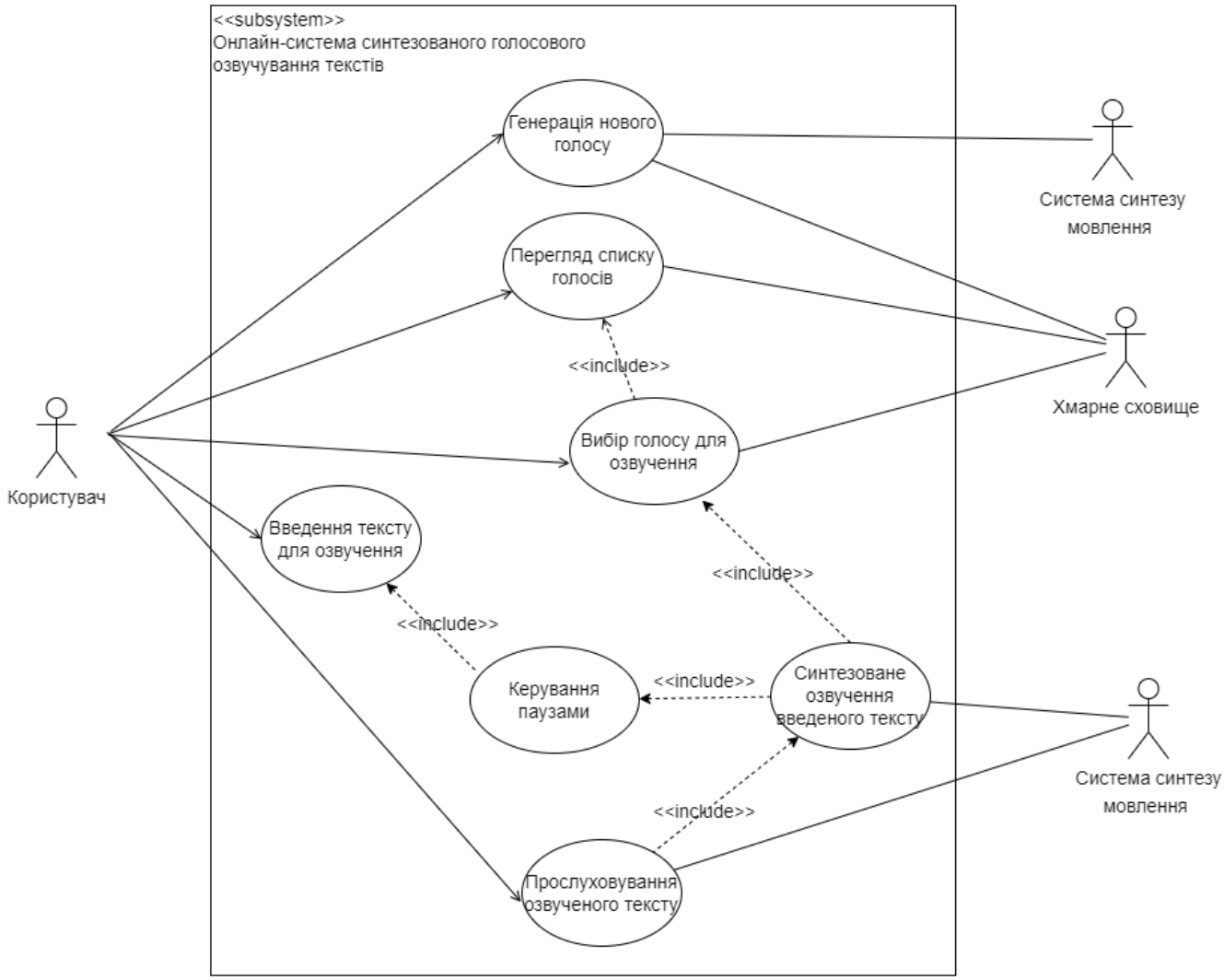


Рисунок 3.1 – Діаграма варіантів використання

3.1.2 Моделювання онлайн системи в IDEF0

Контекстна діаграма включає в себе загальний опис системи та її взаємодії з середовищем. Основними елементами даних, які представлені на даній діаграмі є:

– Вхідні: ввідні дані, які на стадії ініціалізації продукту визначають певні задачі.

– Вихідні: дані, що визначають результат, отриманий під час реалізації.

– Управління: дані, механізми управління (інструкції, положення, тощо), які використовуються під час реалізації.

– Механізми: усе що необхідне для реалізації (апаратне, програмне забезпечення, команда проекту).

Контекстна діаграма є діаграмою найвищого рівня, яка представляє систему в загальному вигляді, та співвідносить її із іншими компонентами інтерфейсу за допомогою стрілок.

В ході проведення аналізу було сформовано перелік даних для контекстної діаграми:

– Вхідні: технічне завдання на розробку онлайн-системи та вимоги користувача.

– Вихідні: розроблена онлайн-система та проектна документація, яка надається як звіт про виконану роботу.

– Управління: методологія імплементації моделей TTS, методологія створення проектів із PyTorch, набір даних для навчання.

– Механізми: команда проекту, і необхідне технічне та апаратне забезпечення.

Розроблену контекстну діаграму процесу створення онлайн-системи синтезованого голосового озвучування текстів наведено на рис. 3.2.

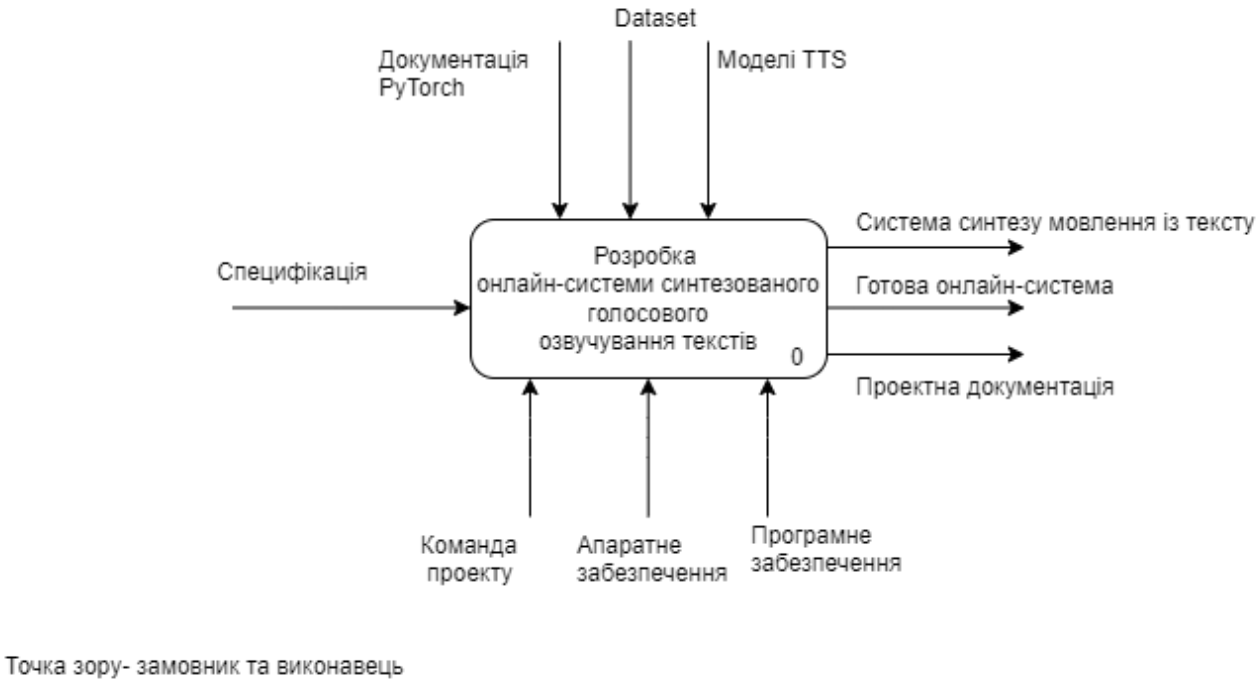


Рисунок 3.2 – Контекстна діаграма

Після цього наступним етепом є проведення процесу декомпозиції, який представляє послідовність виконання робіт проекту. В ході декомпозиції було виділено 4 основних етапи, які складатимуть діаграму другого рівня, а саме:

– Розробка та навчання кодера
– Розробка та навчання синтезатора
– Розробка та навчання вокодера
– Реалізація структури проекту у вигляді онлайн сторінки

Під час опису першого етапу було сформовано наступний перелік даних:

– Вхідні: технічне завдання на розробку проекту, вимоги користувача;
– Вихідні: навчена модель кодера;
– Управління: методологія імплементації моделі кодера, методологія створення проектів із PyTorch, набір даних для навчання;
– Механізми: команда проекту, технічне забезпечення, VS Code, PyTorch.

Далі таким самим чином було сформовано перелік даних другого етапу:

– Вхідні: технічне завдання на розробку проекту та вимоги користувача;
– Вихідні: навчена модель синтезатора;

– Управління: методологія імплементації моделі синтезатора, методологія створення проектів із PyTorch, набір даних для навчання;

– Механізми: команда проекту, технічне забезпечення, VS Code, PyTorch.

Після цього було сформовано перелік даних третього етапу:

– Вхідні: технічне завдання на розробку проекту, вимоги користувача;

– Вихідні: навчена модель вокодера;

– Управління: методологія імплементації моделі вокодера, методологія створення проектів із PyTorch, набір даних для навчання;

– Механізми: команда проекту, технічне забезпечення, VS Code, PyTorch.

Четвертий етап є завершальним і представляє процес реалізації структури проекту у вигляді онлайн сторінки і по суті формування фінального готового продукту. При формуванні четвертого етапу було виділено наступний перелік даних:

– Вхідні: технічне завдання на розробку проекту, вимоги користувача.

– Вихідні: розроблена онлайн-система та набір проектної документації, як звіт із результатом виконаної роботи.

– Управління: методологія створення проектів із PyTorch.

– Механізми: команда проекту, технічне забезпечення, VS Code.

Діаграму другого рівня наведено на рис. 3.3.

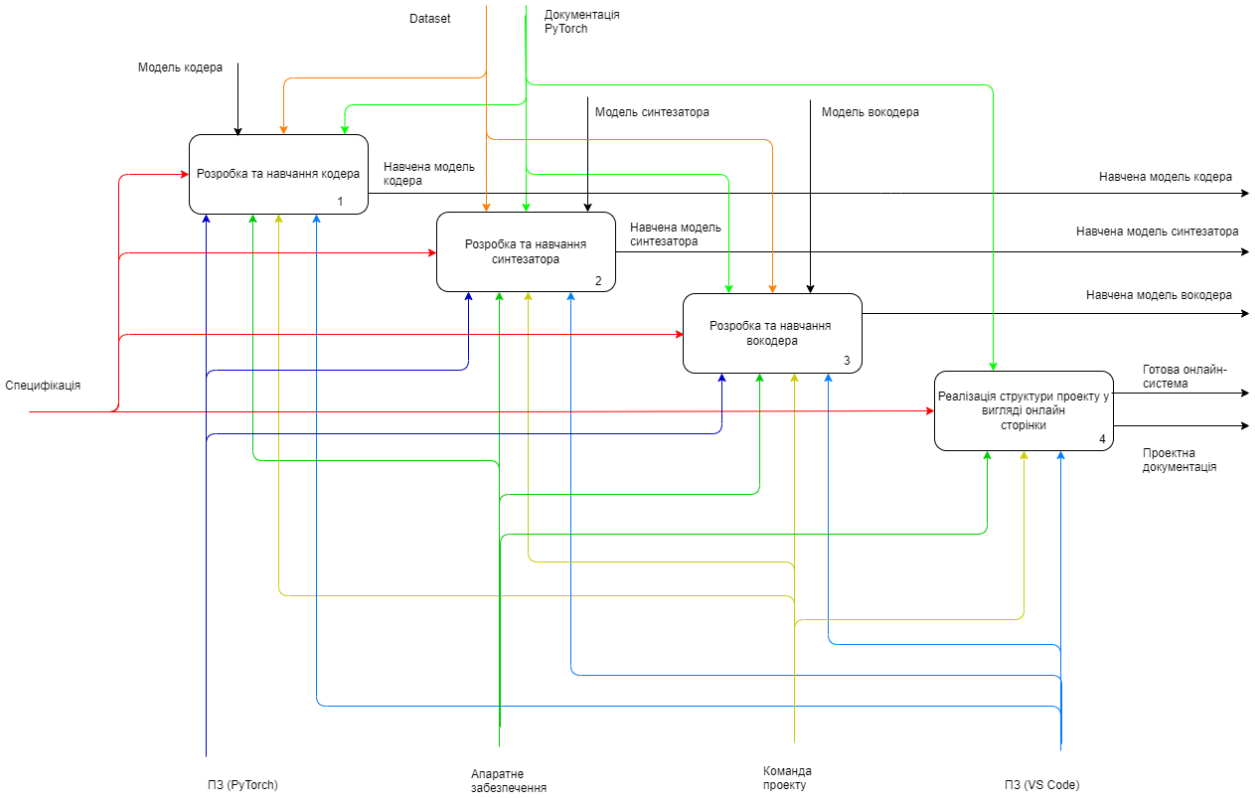


Рисунок 3.3 – Декомпозиція діаграми А-0

Разом із моделюванням використання та моделюванням онлайн системи в IDEF0 також проводилось планування робіт даного проекту, під час якого були сформувані ієрархічна структура робіт (WBS) та організаційна структура проекту (OBS), які описують роботу учасників команди проекту на кожному етапі виконання робіт. Розроблювався також календарний план виконання проекту у вигляді діаграми Ганта і PDM-мережі. А також було визначено ризики та сформовано план управління ризиками. Результат планування робіт проекту наведено у Додатку Б.

4 РЕАЛІЗАЦІЯ ОНЛАЙН-СИСТЕМИ

Для реалізації мети продукту проекту необхідно провести реалізацію системи на основі нейронної мережі для синтезу мовлення, реалізацію інтерфейсу та інтеграцію із системою. Створювана онлайн-система має назву SpeakUp.

Онлайн-система призначена для синтезованого озвучення будь-яких текстів, із використанням наданих або згенерованих нових голосів, за допомогою нейронних мереж.

4.1 Реалізація системи синтезу мовлення

Створення системи синтезу мовлення відбувалося із використанням можливостей фреймворку PyTorch та було реалізовано необхідні моделі. Наша система складається з трьох компонентів, кожен з яких є навченою нейронною мережею. Для повного навчання трьох моделей необхідний великий обсяг звуку, для цього є потреба у завантаженні великої кількості даних та формування датасету.

В результаті пошуку потрібних даних, було завантажено набори даних із записами мовлення багатьох різних мовців, кожного із яких поміщено в окрему папку. Приклад того, як виглядає один із наборів даних, показано на рис. 4.1. Дані розміщені в папках у вигляді аудіофайлів із описом у тектових файлах із мета даними. Приклад того, як розміщені дані наведено на рис. 4.2.

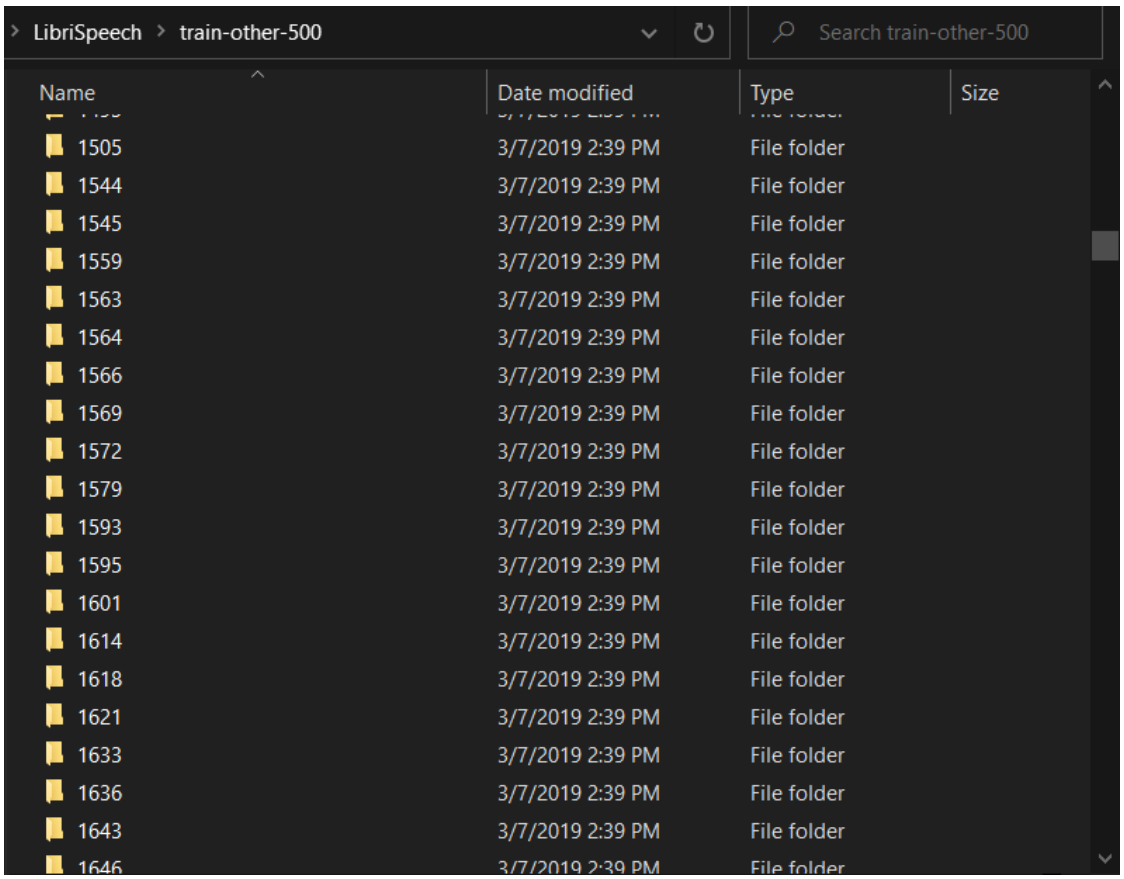


Рисунок 4.1 – Набір аудіо даних різних мовців

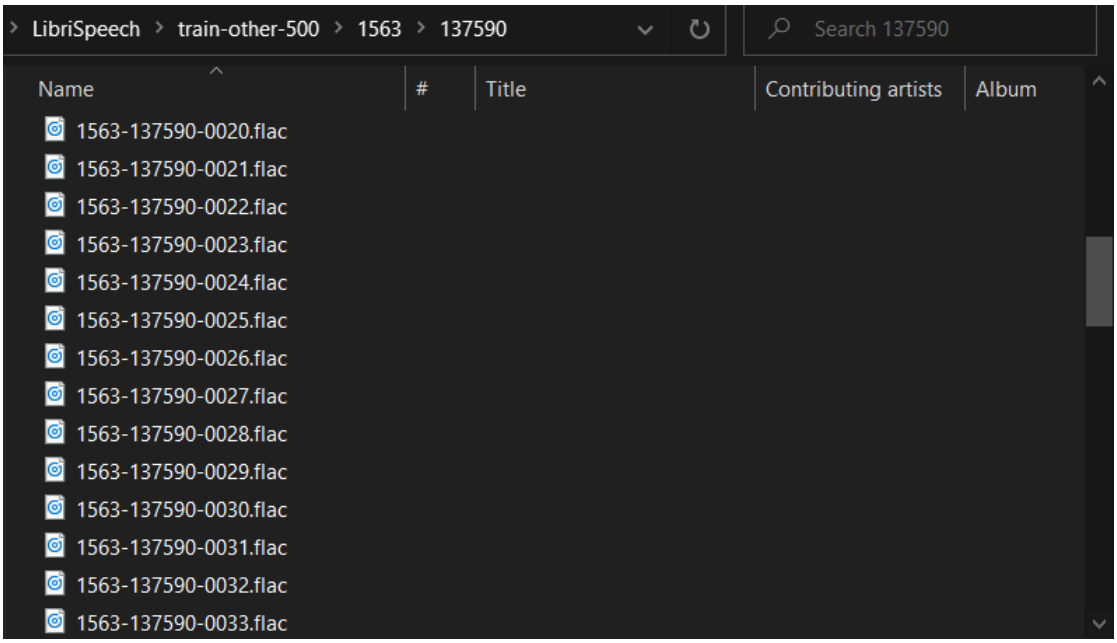


Рисунок 4.2 – Розміщення звуку в наборі даних

Кожен з трьох модулів містить код для попередньої обробки даних, візуалізації даних, завантаження даних, навчання моделі, визначення гіперпараметрів і подальшого використання моделі.

Енкодер – модель мережі, що навечена для того щоб генерувати мовлення цільового мовця. Для навчання кодера потрібно дуже багато звуку різних мовців, хоча, можна використовувати нерозмічені дані із шумами. Модель кодера та процедура її навчання описані в кількох роботах [4, 7]. Дана модель була відтворена із реалізацією PyTorch.

Процес навчання моделі починається з підготовки даних. Для цього запускається скрипт попередньої обробки. Процес попередньої обробки даних представлено на рисунку 4.3.

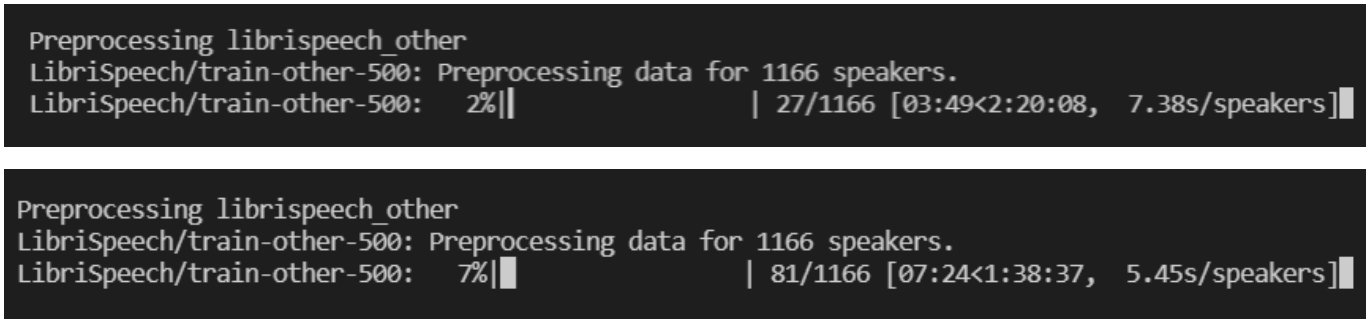


Рисунок 4.3 – Процес попередньої обробки даних

Після цього відбувається процес тренування. Для цього запускається скрипт навчання моделі. Процес навчання кодера візуалізується за допомогою Visdom сервера, тому перед запуском скрипта необхідно запустити даний сервер. За допомогою візуалізації ми можемо явно побачити процес навчання моделі, який предсталено на рис. 4.4 – 4.5. Результат навчання наведено на рис. 4.6 та зберігається у вихідній директорії у форматі .pt та використовується під час роботи онлайн-системи.

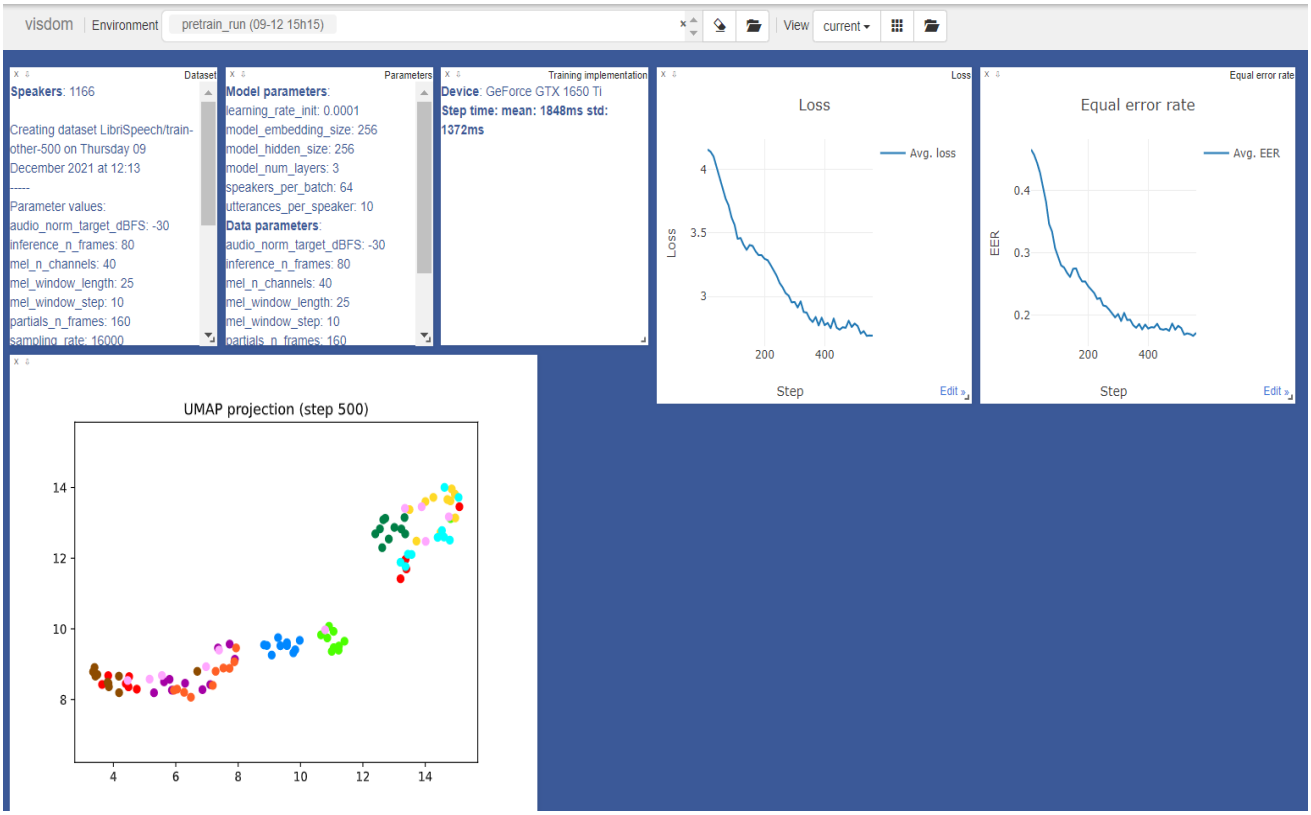


Рисунок 4.4 – Приклад процесу навчання моделі кодера

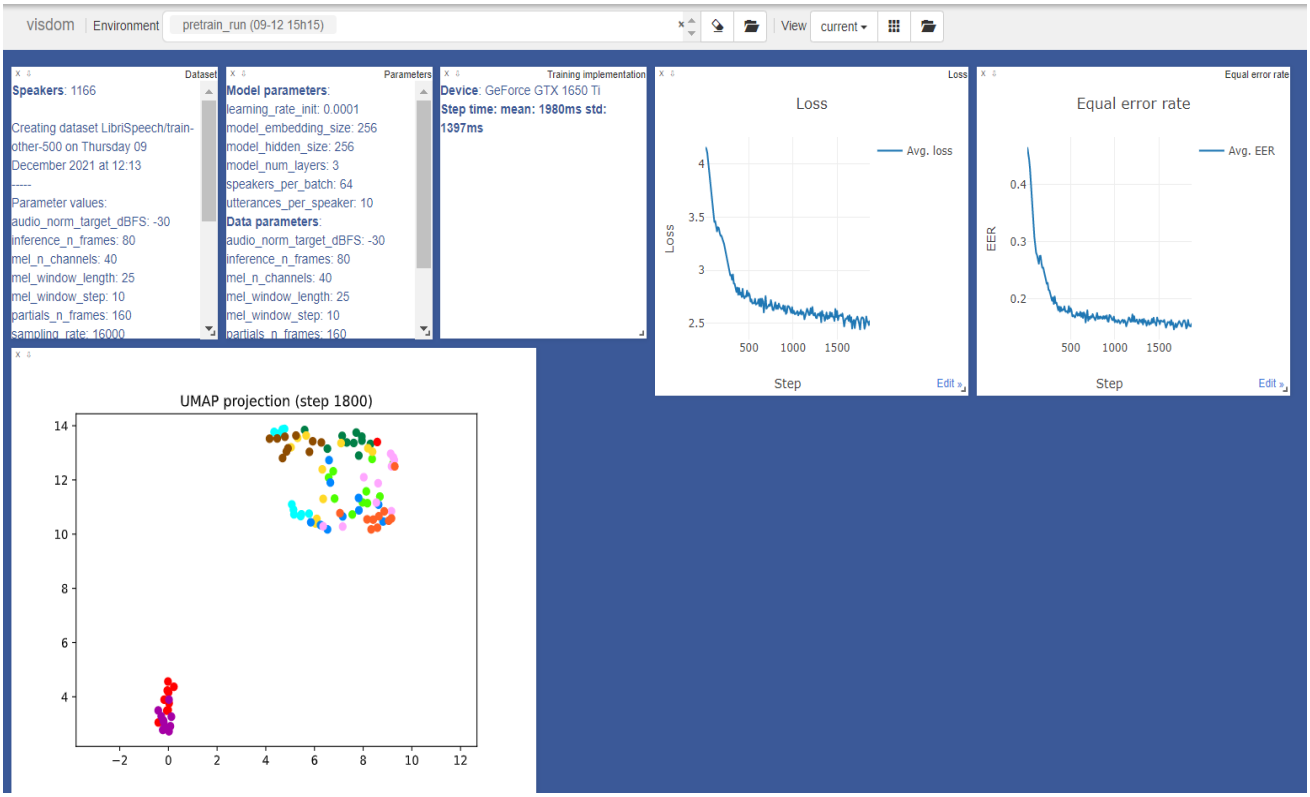


Рисунок 4.5 – Приклад процесу навчання моделі кодера

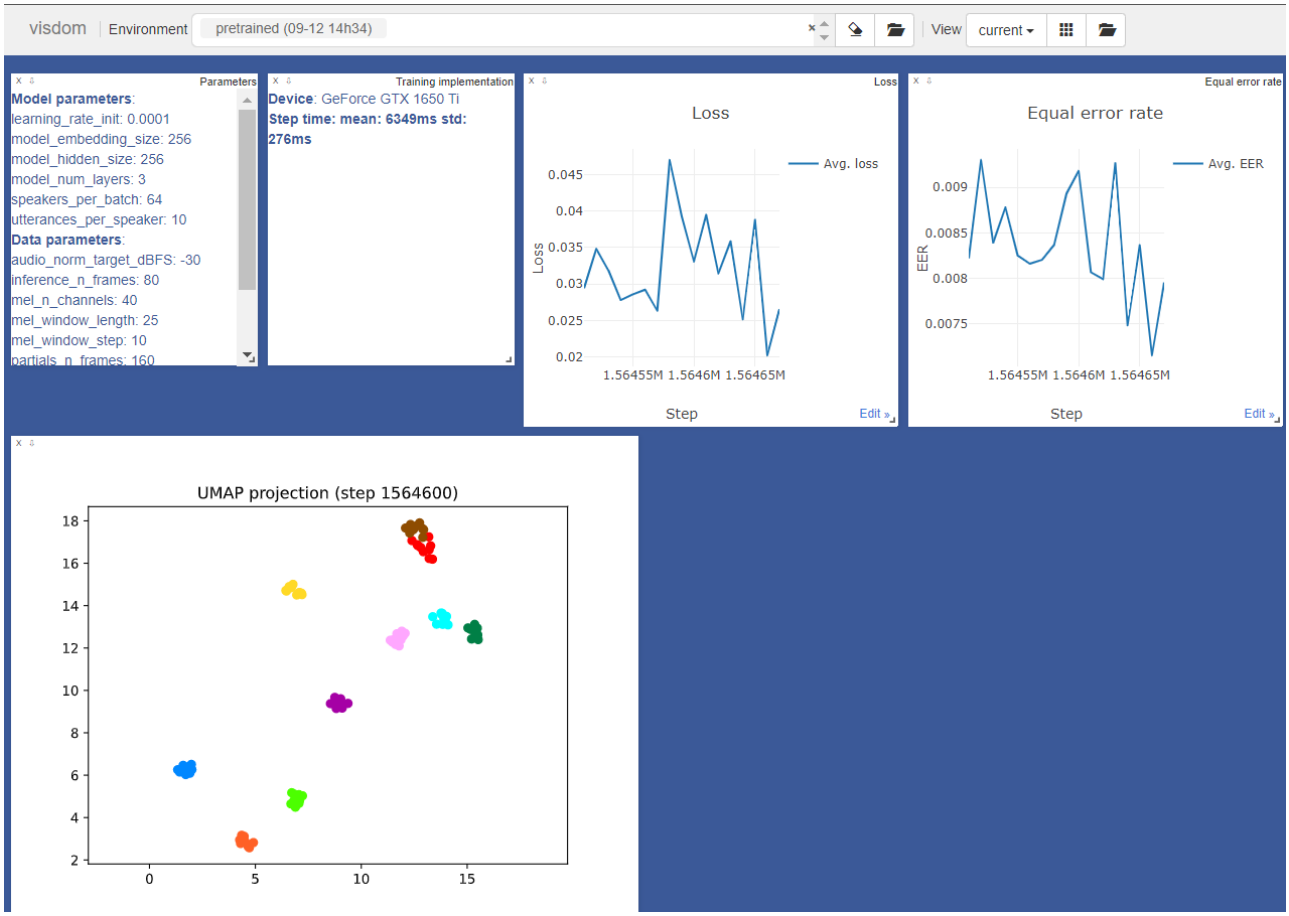


Рисунок 4.6 – Приклад завершеного процесу навчання моделі кодера

Синтезатор – модель мережі синтезу послідовності на основі Tacotron, яка генерує спектрограму безпосередньо з тексту. Для навчання синтезу потрібно багато чистого, добре розміченого звуку різних мовців.

Було використано реалізацію Tensorflow з відкритим вихідним кодом для Tacotron 2 [18], з якої було вилучено логіку, що стосується WaveNet і додано модифікації необхідні даній системі. На рисунку 4.7 наведено архітертуру моделі, що представлена у роботі [2]. Tacotron є рекурентною моделлю, яка генерує спектрограму Mel з тексту. Вона має структуру кодер-декодер, яка об’єднана механізмом уваги, чутливим до розташування. Окремі символи з текстової послідовності спочатку вбудовуються як вектори. Далі йдуть згорткові шари, щоб збільшити діапазон одного кадру кодера. Ці кадри пропускаються через двонаправлений LSTM для створення вихідних кадрів кодера.

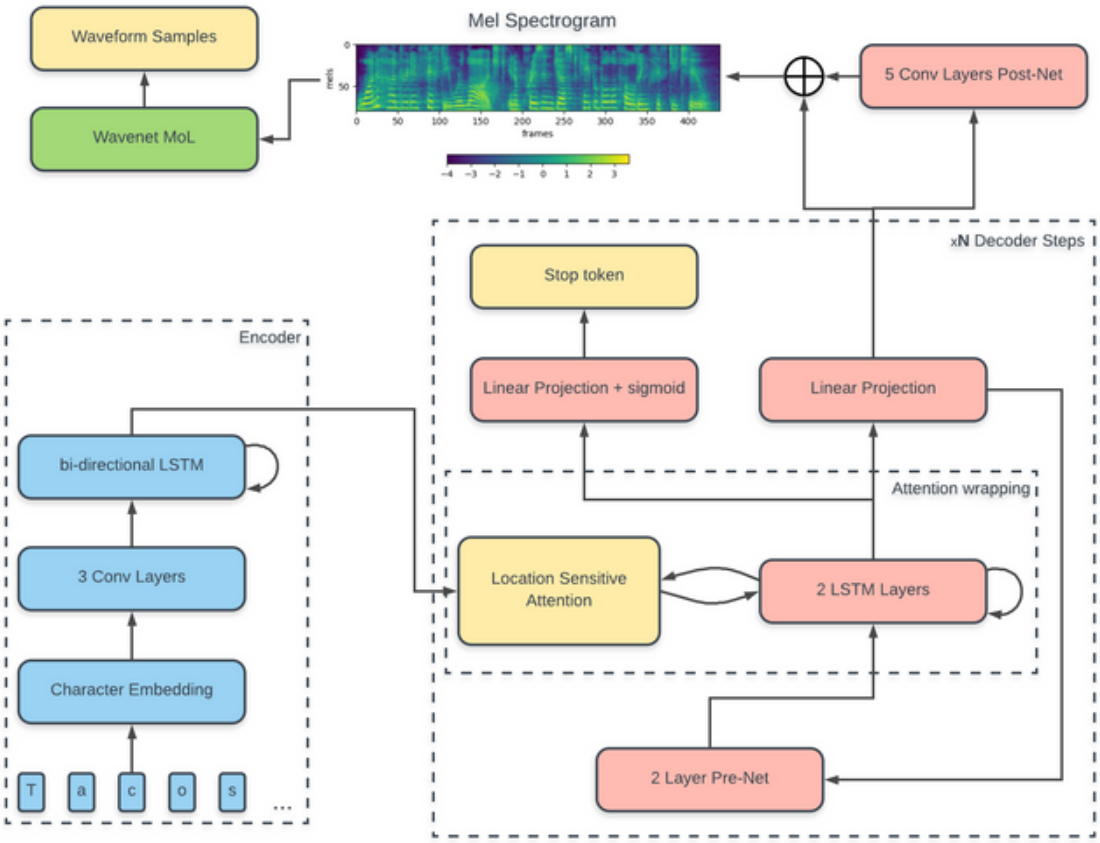
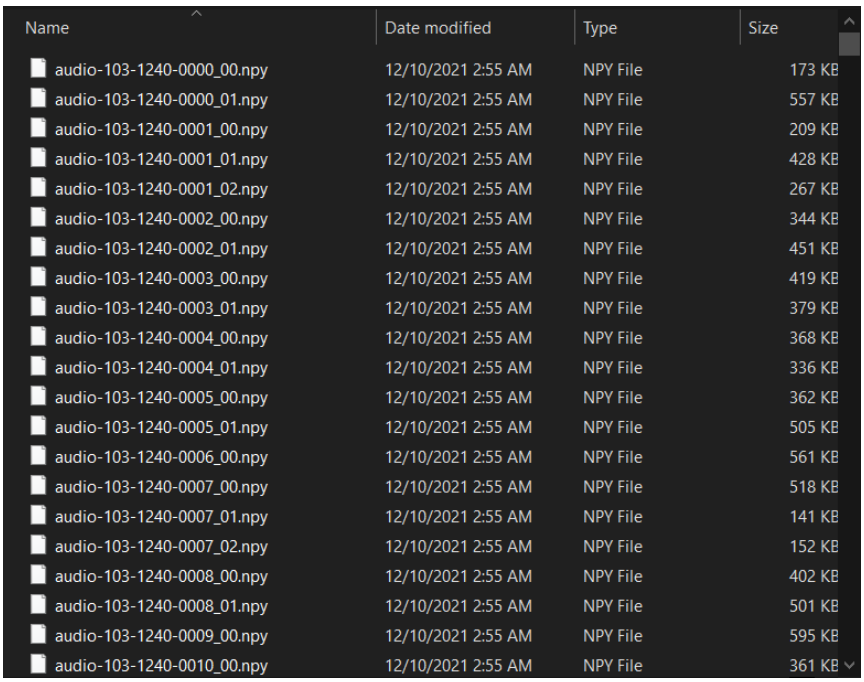


Рисунок 4.7 – Архітектура моделі Tacotron [2]

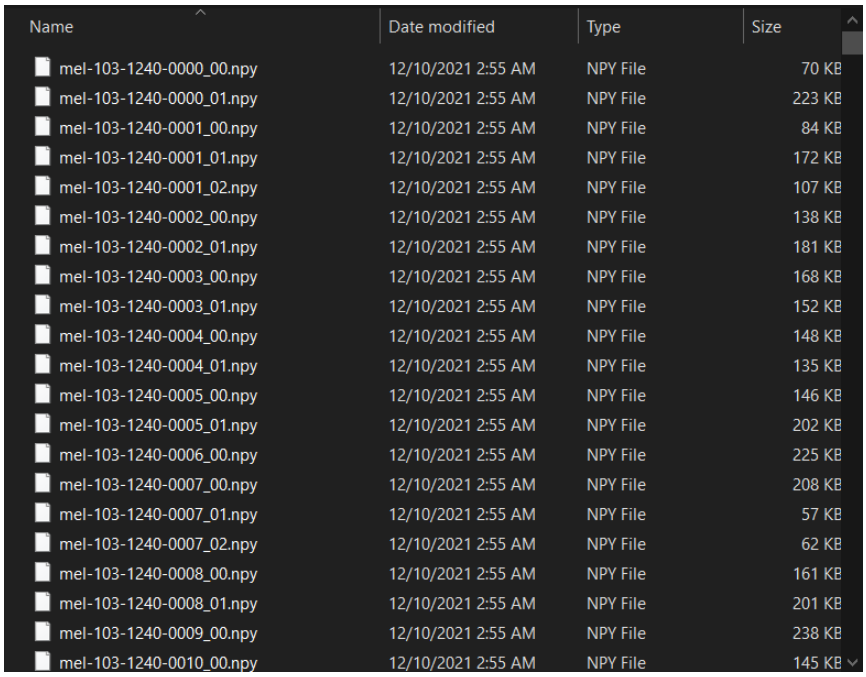
Зміни в даній архітектурі полягають в тому, що даються вбудовування спікера, що з'єднується з кожним кадром, який виводить кодер Tacotron. Механізм уваги бере вихідні кадри кодера, щоб генерувати вхідні кадри декодера. Кожен вхідний кадр декодера об'єднується з вихідним кадром попереднього декодера, пропущеним через попередню мережу, що робить модель авторегресивною [2]. Цей конкатенований вектор проходить через два односпрямовані шари LSTM перед проектуванням на один кадр спектрограми Mel. Інша проекція того ж вектора на скаляр дозволяє мережі самостійно передбачити, що вона повинна припинити генерувати кадри, надаючи значення вище певного порогу. Вся послідовність кадрів пропускається через залишкову пост-мережу, перш ніж вона стане спектрограмою Mel.

Процес навчання моделі починається з попередньої обробки даних та запуску скрипта попередньої обробки. В результаті попередньої обробки було створено директорії із обробленими даними для звуку та спектрограм (рис. 4.8 – 4.9).



Name	Date modified	Type	Size
audio-103-1240-0000_00.npy	12/10/2021 2:55 AM	NPY File	173 KB
audio-103-1240-0000_01.npy	12/10/2021 2:55 AM	NPY File	557 KB
audio-103-1240-0001_00.npy	12/10/2021 2:55 AM	NPY File	209 KB
audio-103-1240-0001_01.npy	12/10/2021 2:55 AM	NPY File	428 KB
audio-103-1240-0001_02.npy	12/10/2021 2:55 AM	NPY File	267 KB
audio-103-1240-0002_00.npy	12/10/2021 2:55 AM	NPY File	344 KB
audio-103-1240-0002_01.npy	12/10/2021 2:55 AM	NPY File	451 KB
audio-103-1240-0003_00.npy	12/10/2021 2:55 AM	NPY File	419 KB
audio-103-1240-0003_01.npy	12/10/2021 2:55 AM	NPY File	379 KB
audio-103-1240-0004_00.npy	12/10/2021 2:55 AM	NPY File	368 KB
audio-103-1240-0004_01.npy	12/10/2021 2:55 AM	NPY File	336 KB
audio-103-1240-0005_00.npy	12/10/2021 2:55 AM	NPY File	362 KB
audio-103-1240-0005_01.npy	12/10/2021 2:55 AM	NPY File	505 KB
audio-103-1240-0006_00.npy	12/10/2021 2:55 AM	NPY File	561 KB
audio-103-1240-0007_00.npy	12/10/2021 2:55 AM	NPY File	518 KB
audio-103-1240-0007_01.npy	12/10/2021 2:55 AM	NPY File	141 KB
audio-103-1240-0007_02.npy	12/10/2021 2:55 AM	NPY File	152 KB
audio-103-1240-0008_00.npy	12/10/2021 2:55 AM	NPY File	402 KB
audio-103-1240-0008_01.npy	12/10/2021 2:55 AM	NPY File	501 KB
audio-103-1240-0009_00.npy	12/10/2021 2:55 AM	NPY File	595 KB
audio-103-1240-0010_00.npy	12/10/2021 2:55 AM	NPY File	361 KB

Рисунок 4.8 – Результат попередньої обробки аудіо



Name	Date modified	Type	Size
mel-103-1240-0000_00.npy	12/10/2021 2:55 AM	NPY File	70 KB
mel-103-1240-0000_01.npy	12/10/2021 2:55 AM	NPY File	223 KB
mel-103-1240-0001_00.npy	12/10/2021 2:55 AM	NPY File	84 KB
mel-103-1240-0001_01.npy	12/10/2021 2:55 AM	NPY File	172 KB
mel-103-1240-0001_02.npy	12/10/2021 2:55 AM	NPY File	107 KB
mel-103-1240-0002_00.npy	12/10/2021 2:55 AM	NPY File	138 KB
mel-103-1240-0002_01.npy	12/10/2021 2:55 AM	NPY File	181 KB
mel-103-1240-0003_00.npy	12/10/2021 2:55 AM	NPY File	168 KB
mel-103-1240-0003_01.npy	12/10/2021 2:55 AM	NPY File	152 KB
mel-103-1240-0004_00.npy	12/10/2021 2:55 AM	NPY File	148 KB
mel-103-1240-0004_01.npy	12/10/2021 2:55 AM	NPY File	135 KB
mel-103-1240-0005_00.npy	12/10/2021 2:55 AM	NPY File	146 KB
mel-103-1240-0005_01.npy	12/10/2021 2:55 AM	NPY File	202 KB
mel-103-1240-0006_00.npy	12/10/2021 2:55 AM	NPY File	225 KB
mel-103-1240-0007_00.npy	12/10/2021 2:55 AM	NPY File	208 KB
mel-103-1240-0007_01.npy	12/10/2021 2:55 AM	NPY File	57 KB
mel-103-1240-0007_02.npy	12/10/2021 2:55 AM	NPY File	62 KB
mel-103-1240-0008_00.npy	12/10/2021 2:55 AM	NPY File	161 KB
mel-103-1240-0008_01.npy	12/10/2021 2:55 AM	NPY File	201 KB
mel-103-1240-0009_00.npy	12/10/2021 2:55 AM	NPY File	238 KB
mel-103-1240-0010_00.npy	12/10/2021 2:55 AM	NPY File	145 KB

Рисунок 4.9 – Результат попередньої обробки спектрограм

Після цього відбувається безпосередньо процес навчання моделі, що запускається за допомогою скрипта. Результат навчання наведено на рис. 4.10 – 4.11 та зберігається у вихідній директорії синтезатора у форматі .pt та використовується в ході роботи із онлайн-системою.

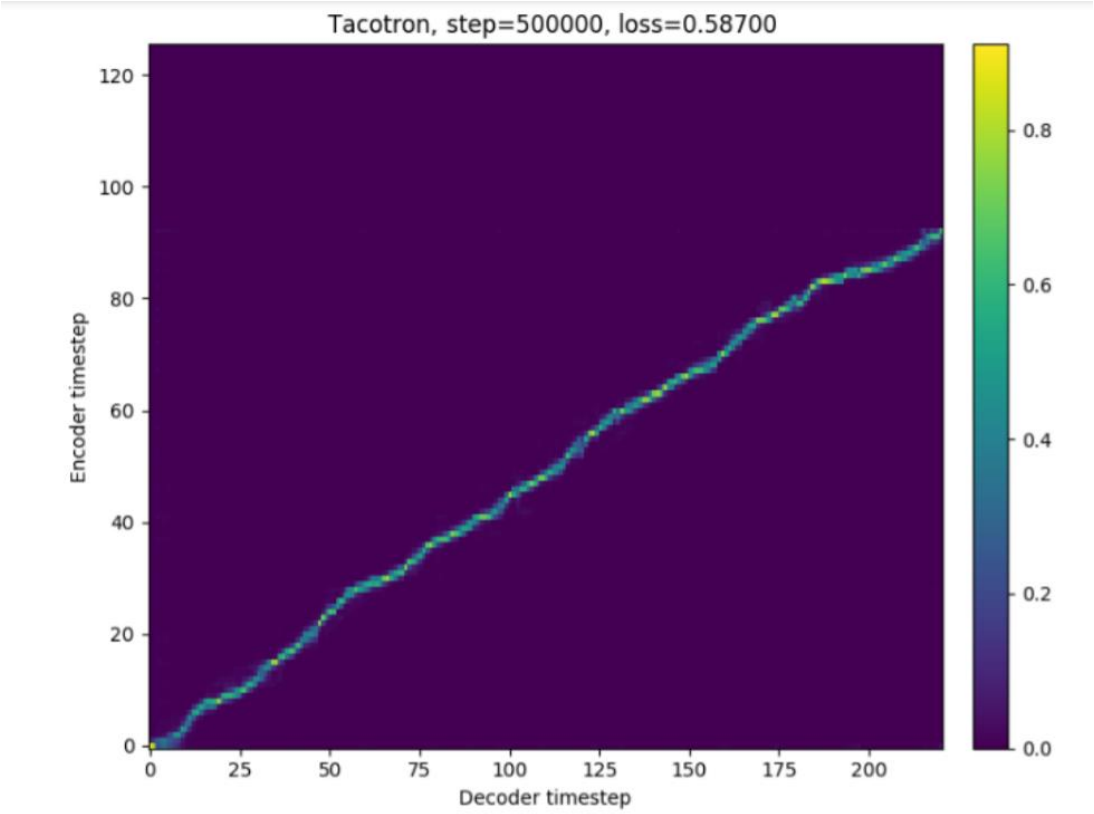


Рисунок 4.10 – Діаграма відношення між кроками кодера та кроками декодера

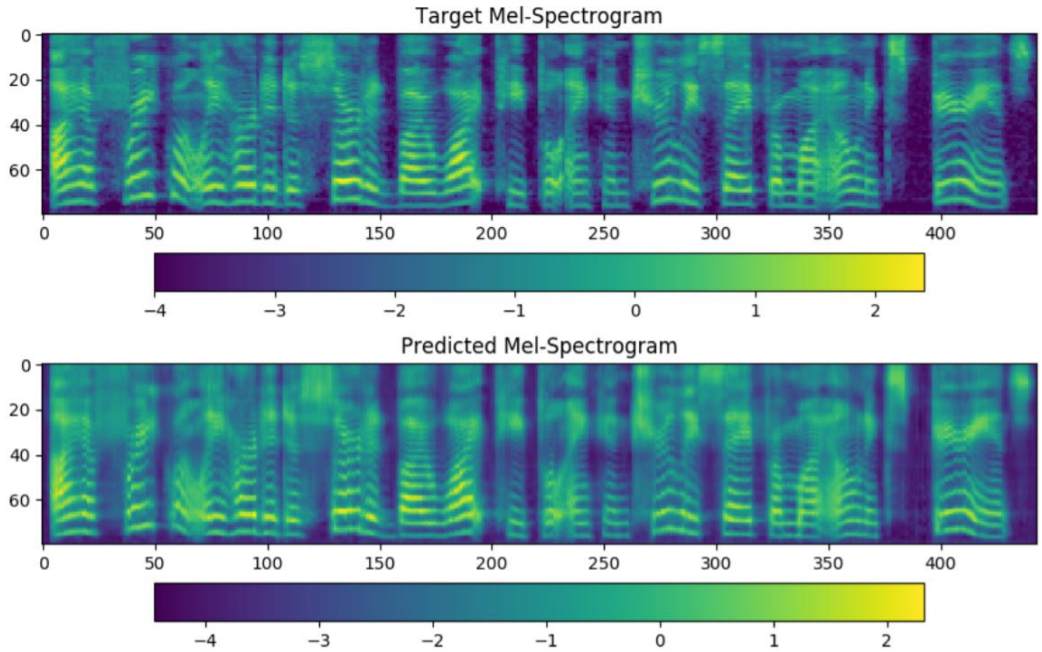


Рисунок 4.11 – Порівняння цільової та передбачуваної спектрограми

Вокодер – модель мережі послідовного моделювання звуку на основі методу WaveNet, яка перетворює спектрограму на звукову доріжку. WaveNet був в центрі

глибокого навчання, яке стосується звуку з моменту його випуску і залишається популярним і досі, коли справа доходить до природності голосу. Вокодер теж потребує навчання на чистих, добре розмічених даних.

Модель вокодера, яка використовується в даному проекті, описана в роботі [7] та є реалізацією PyTorch, яка базується на WaveRNN. Дана модель не залежить безпосередньо від результату кодера для мовця. Спектрограма Mel, передбачена моделлю синтезатора, фіксує всі деталі, необхідні для високоякісного синтезу різноманітних голосів, що дозволяє створити вокодер з багатьма спікерами шляхом простого навчання на даних від багатьох спікерів. При навчанні даної моделі не було використано візуалізацію. Результатом навчання, так само я і в попередніх моделях є файл .pt, який необхідний при використанні робочих моделей.

Використовуючи можливості фреймворку PyTorch було розроблено раніше описані моделі. Завдяки ним відбувається синтез мовлення із тексту. Після того як було навчено описані мережі, можна переходити до їх використання та створення інтерфейсу користувача. В нашому випадку це онлайн сторінка.

4.2 Реалізація інтерфейсу онлайн-системи

Перед тим як розпочати роботу над дизайном онлайн-системи було проаналізовано тренди поточного року та тенденції у сучасному дизайні онлайн сторінок. Було обрано палітру кольорів (рис 4.12), що буде використано як основні кольори логотипу та веб-сторінки. Під час створення логотипу, що представлений на рис. 4.13, було використано обрану назву проекту та звукову хвилю в якості основи логотипу, також на логотипі можна помітити напис Get your voice, що також є ключовою фразою проекту, адже розроблювана онлайн система надає змогу користувачу конвертувати текст у власний голос.

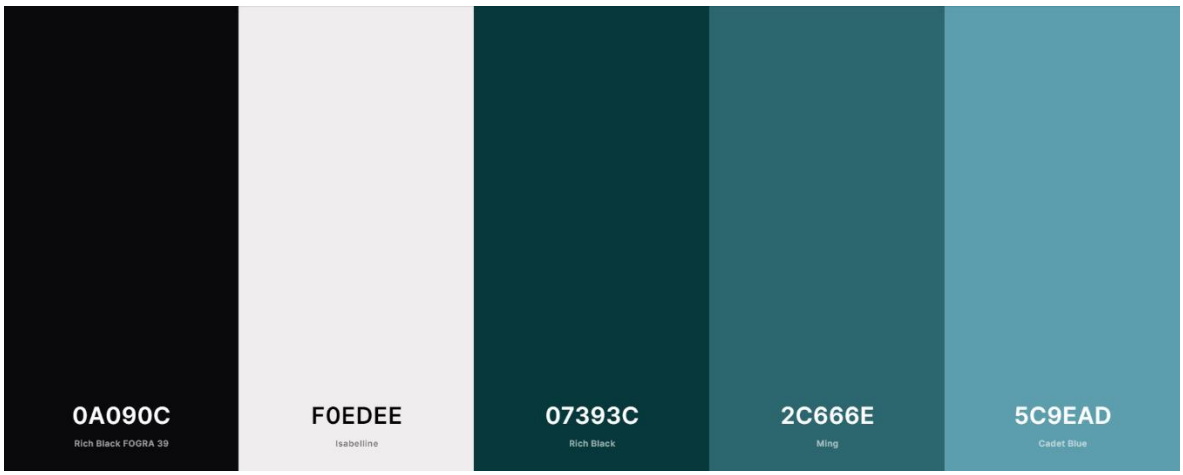


Рисунок 4.12 – Палітра кольорів розроблюваної онлайн-системи

Рисунок 4.13 – Логотип розроблюваної онлайн-системи

Розроблюваний веб додаток складається із однієї сторінки, яка поділена на секції, що відображені зверху на панелі навігації (рис. 4.14), крім цього на даній

панелі розміщено слоган проекту, який також є посиланням на початок сторінки при навігації.

Рисунок 4.14 – Початок сторінки та навігаційна панель

Головним елементом онлайн-сторінки є секція конвертації тексту в аудіо (рис. 4.15). В даній секції розташовано форму для конвертації. Задля зручності користувачам було надано декілька варіантів надання цільового аудіозапису. Користувач може обрати голос для озвучування із прикладів, завантажити власний аудіо файл, або ж записати звук на місці, перебуваючи на сторінці розробленої онлайн системи. Після завантаження звуку, його буде відображено для прослуховування. Також відведено місце для введення тесту для озвучення. Центральна кнопка секції призначена для початку конвертації тексту в аудіо. Після виконання конвертації системою, користувачу буде відображено результат у вигляді програвача готового звуку.

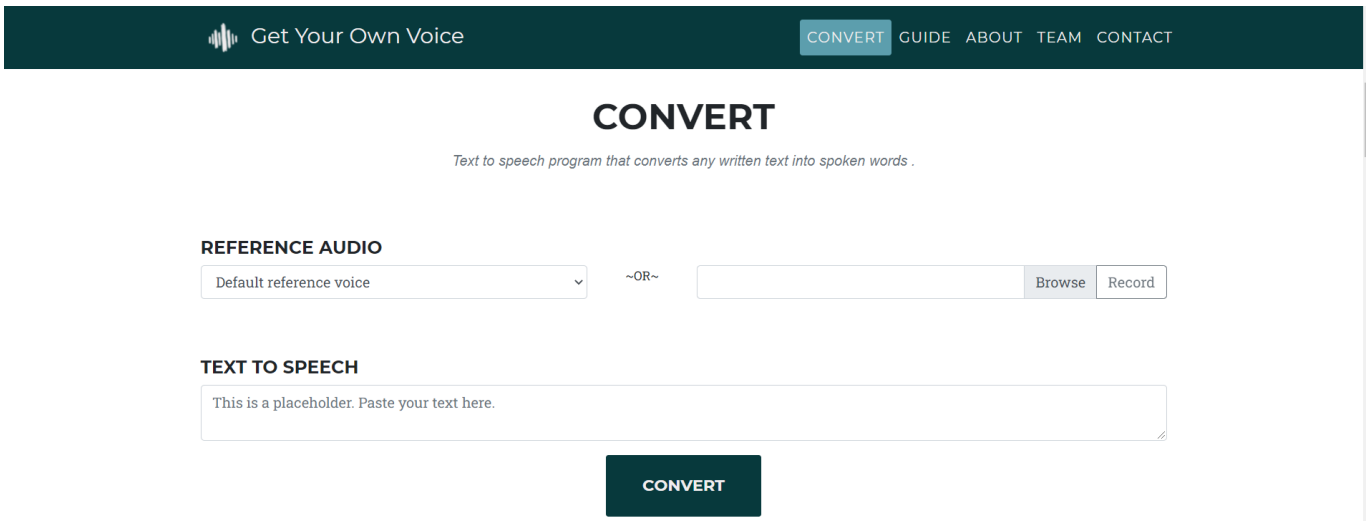


Рисунок 4.15 – Секція конвертації тексту в аудіо

Крім цього, сторінка містить керівництво користувача, опис призначення системи, інформацію про команду та контактну інформацію. Усі додаткові секції веб додатку наведено на рис. 4.16 – 4.20.

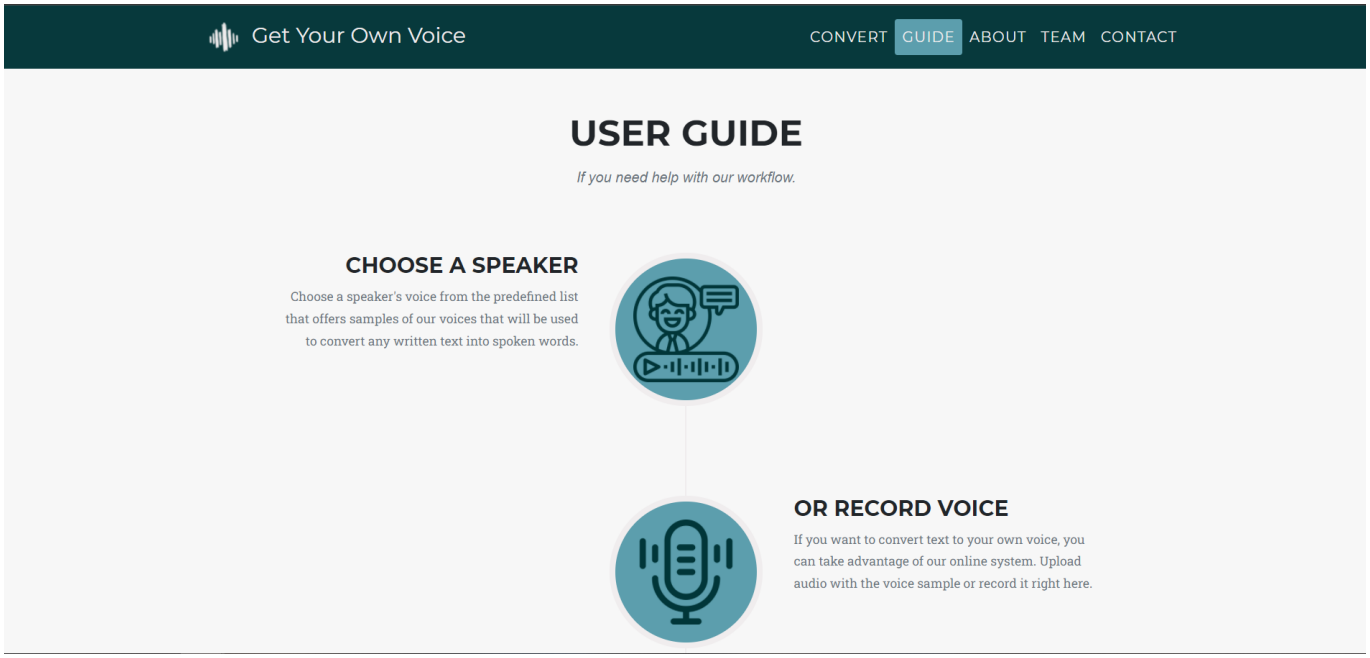


Рисунок 4.16 – Секція керівництво користувача

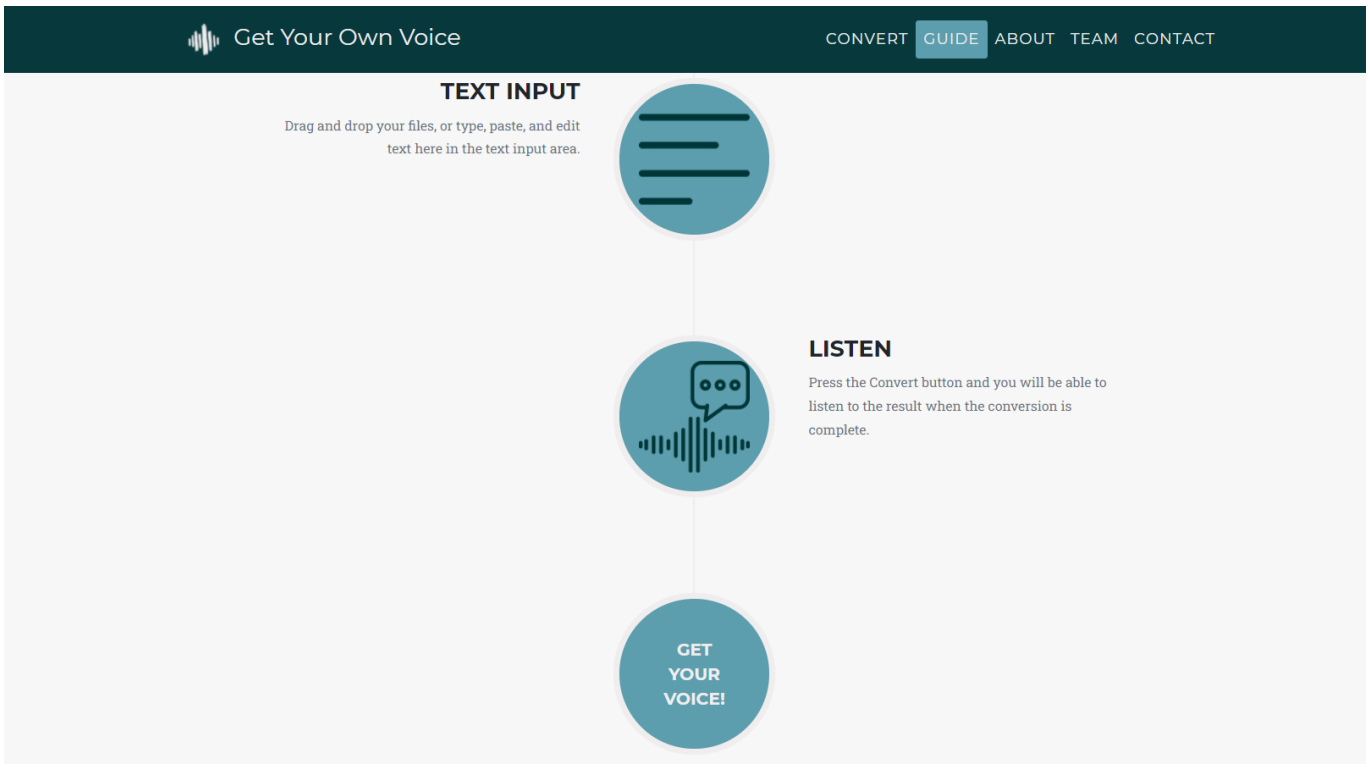


Рисунок 4.17 – Секція керівництво користувача

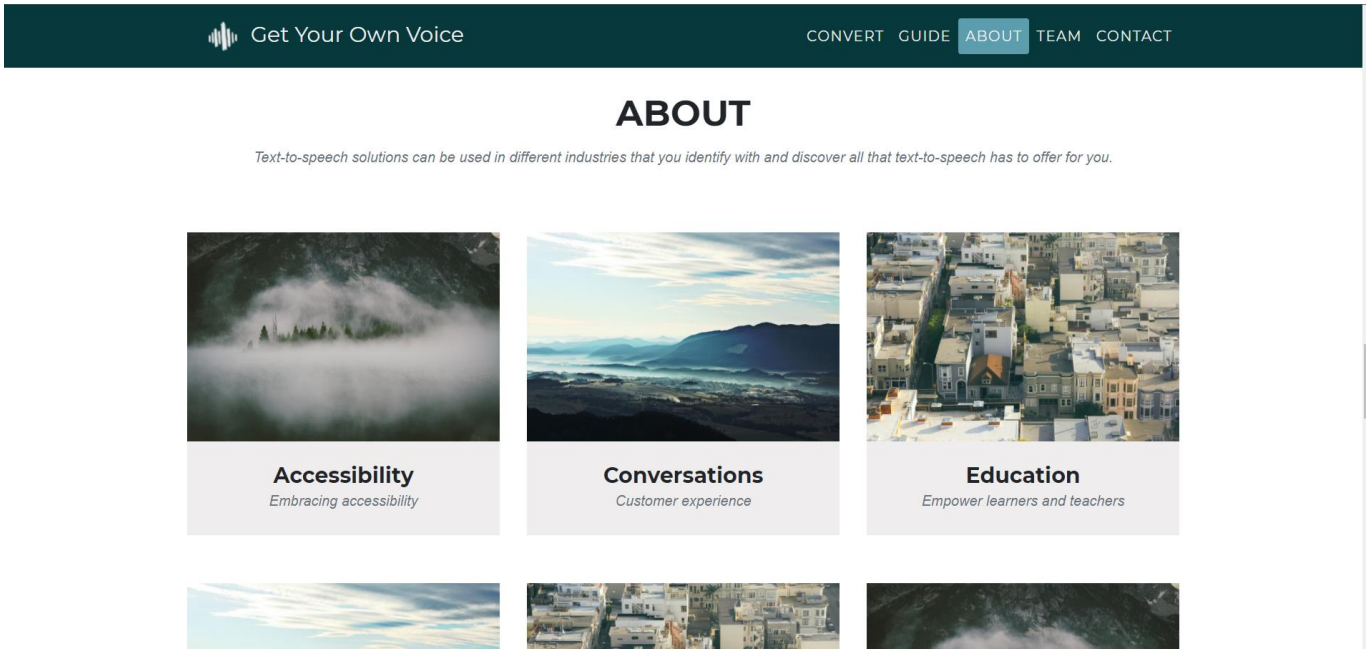


Рисунок 4.18 – Секція опису призначення системи

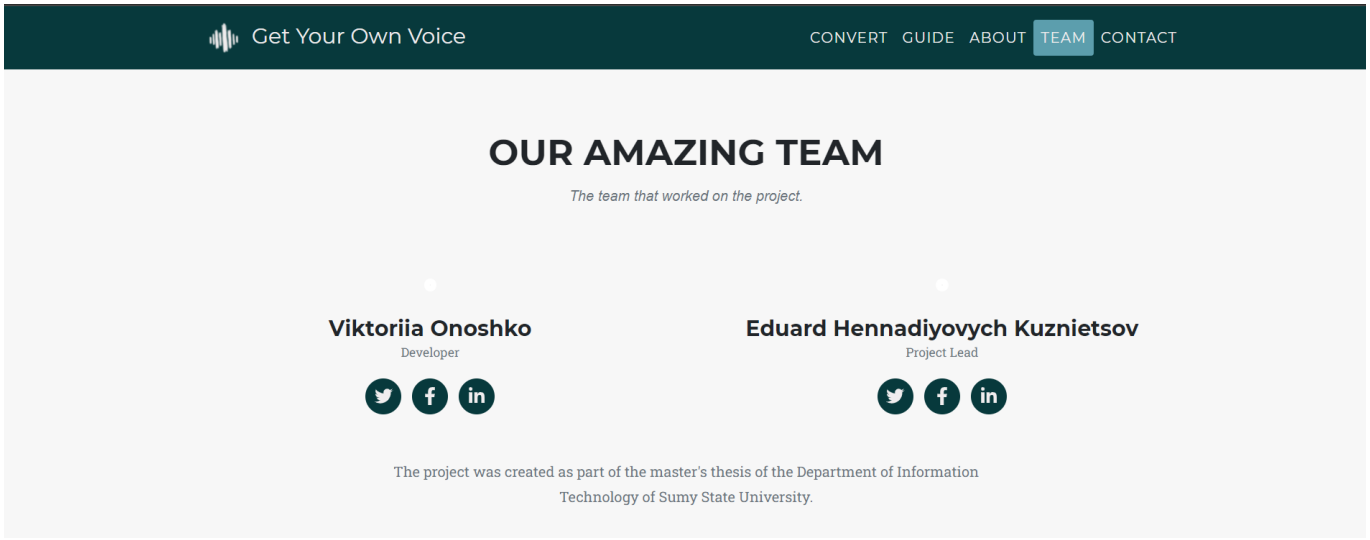


Рисунок 4.19 – Секція інформації про команду проекту

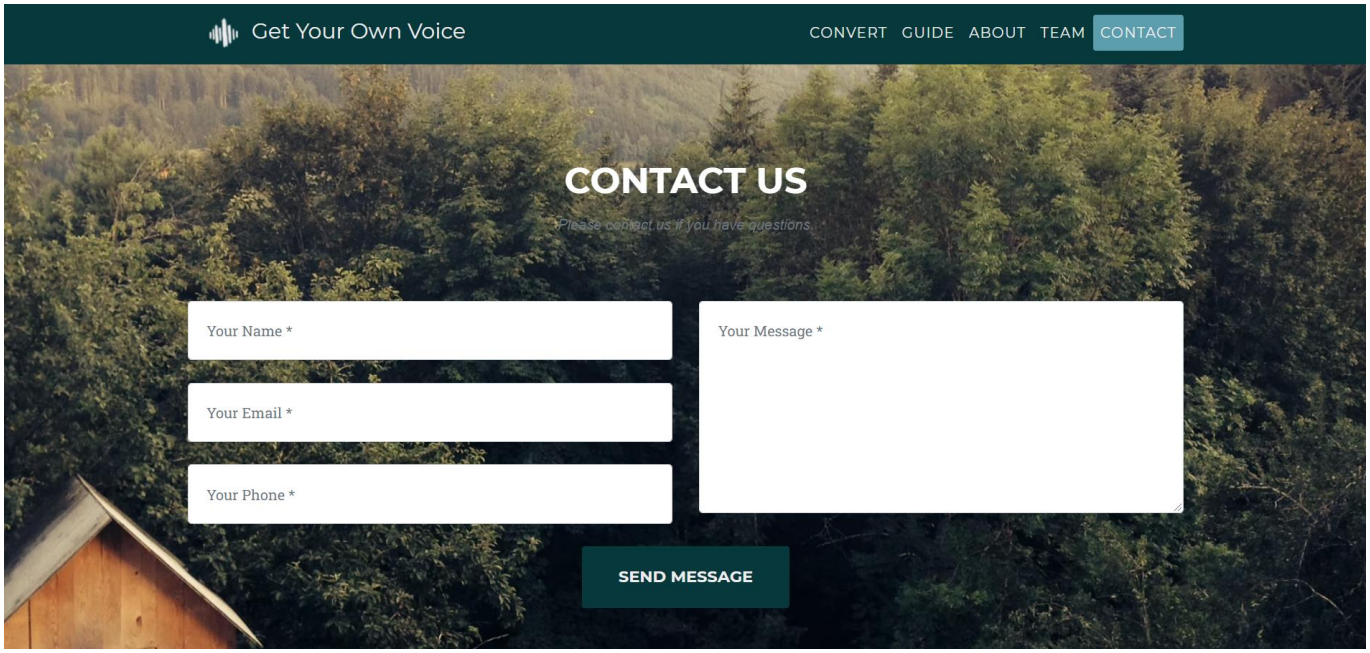


Рисунок 4.20 – Секція контактної інформації

Реалізація створюваного інтерфейсу проходила за допомогою Bootstrap із використанням компонентів для створення інтерфейсу. Основними компонентами, що використовувались в ході розробки є Section, Container, Navbar, Scrollspy, Forms, Input group.

Інтерфейс будь-якого сучасного веб додатку має бути адаптивним для різноманітних форматів екранів, в тому числі мобільних. В даному випадку це також

досягається за допомогою можливостей Bootstrap. Приклад адаптивності інтерфейсу в іншому форматі екрану наведено на рис. 4.21.

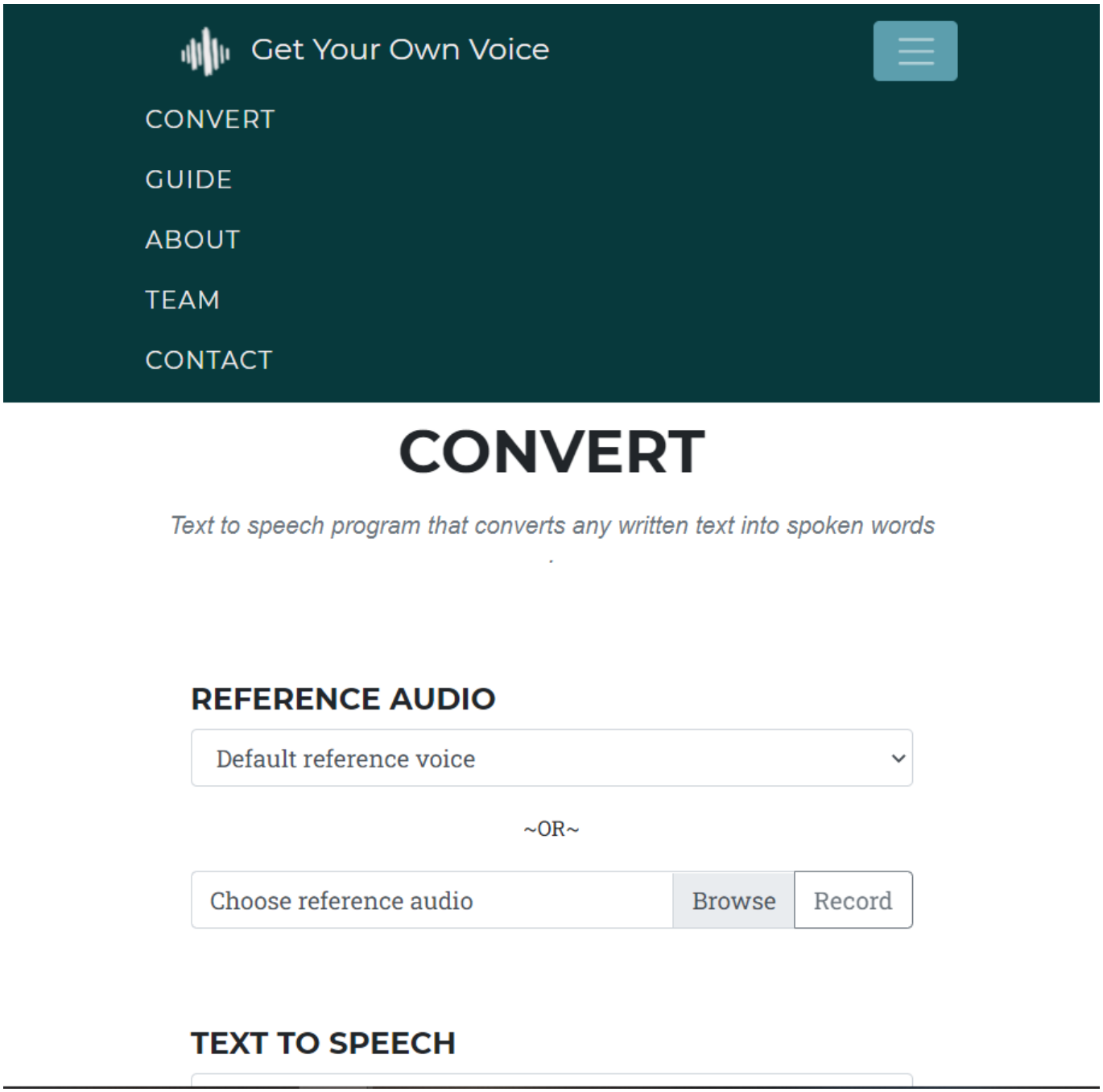


Рисунок 4.21 – Приклад адаптивності інтерфейсу

4.3 Реалізація інтеграції із системою

Наступним етапом після створення інтерфейсу користувача є реалізація функціоналу веб додатку та інтеграція із інтерфейсом.

За допомогою мікрофреймворку Flask відбувалася реалізація веб додатку. Flask — це легкий фреймворк веб-додатків WSGI [16]. Він створений саме для того, щоб розпочати роботу над проектом швидко та легко, з можливістю масштабування до складних програм в майбутньому.

Для початку створення додатку у активованому середовищі необхідно скористатися командою, щоб встановити Flask: $ pip install Flask.

Мінімальний додаток Flask для інтеграції із інтерфейсом, що знаходиться в файлі index.html, наведено на рис. 4.22. Далі додаток розширювався реалізацією основних функцій.

```
# Importing flask module in the project is mandatory
# An object of Flask class is our WSGI application.
from flask import Flask, request, render_template
import os

port = int(os.environ.get("PORT", 5010))
app = Flask(__name__, template_folder="app/", static_folder="static/")

@app.route('/', methods=['GET', 'POST'])
def hello_world():
    return render_template("index.html", text="", output="", outputAudio="")

if __name__ == '__main__':
    app.run(debug=True, host='localhost', port=port)
```

Рисунок 4.22 – Мінімальний додаток Flask

Основною функцією онлайн-системи є конвертація тексту в аудіо із наданого цільового голосу, як власного, так і обраного із прикладів. Після того як користувачем було обрано цільовий аудіозапис, а також введено бажаний текст для озвучення та

натиснута кнопка «Конвертувати» відбувається процес перетворення тексту в мовлення. Обробка запиту з форми відбувається із використанням мікрофреймворку Flask. Всередині обробки запиту відбуваються процеси, що взаємодіють із моделями синтезу мовлення.

Спочатку відбувається перевірка файлу на відповідність аудіо-форматам та у разі проходження перевірки відбувається завантаження файлу для подальшої його обробки.

Після цього починається робота із моделями, спочатку підготуємо кодер, синтезатор і вокодер та завантажимо моделі для початку роботи з ними.

Далі за допомогою кодера проводимо попередню обробку звукової хвилі аудіофайлу, під час якої форма звукової хвилі буде дискретизована, щоб відповідати гіперпараметрам даних. До попередньої обробки входить нормалізація звуку та скорочення довгих пауз. На виході буде отримана форма сигналу у вигляді масиву чисел із плаваючою точкою.

Наступним кроком ми отримуємо вбудовування (вкладення). Вбудовування є одним із застосувань глибокого навчання, це метод, який використовується для представлення дискретних змінних у вигляді безперервних векторів. Ця техніка знайшла практичне застосування за допомогою вбудовування слів для машинного перекладу. На виході отримуємо вбудовування у вигляді numpy-масиву float32.

На цьому етапі від кодера ми переходимо до синтезатора і від вхідного аудіофайлу до тексту, наступним етапом є генерація спектрограми. Модель синтезатора генерує Mel спектрограму з тексту і вкладення спікера. Дана спектрограма є звичайною, з частотами, виміреними в одиницях мел, що є психофізичною одиницею висоти звуку при якій рівні по висоті звуки звучать однаково далеко за відстанню для слухача.

Останнім єтапом є перетворення спектрограми на результат у вигляді звукової хвилі. Для цього використаємо модель вокодера, яка виводить форму звукового сигналу спектрограми mel, створену раніше синтезатором.

Даний програмний код наведено у додатку В.

4.4 Використання онлайн-системи

Використання онлайн-системи полягає у взаємодії користувача із онлайн-системою через інтуїтивно зрозумілий інтерфейс. На випадок не достатнього розуміння процесу використання, на сторінці надано керівництво користувача. Використовуючи веб додаток для синтезованого голосового озвучування текстів, користувачі зможуть отримати допоміжну технологію у деяких аспектах життя, яка допомагає зробити світ доступнішим, коли справа доходить до того, як ми говоримо та слухаємо.

На рис. 4.23 – 4.27 наведено приклад використання користувачем розробленої онлайн-системи.

Розглянемо докладно секцію конвертації тексту. Користувач може перейти до даної секції, використавши навігаційну панель та натиснувши відповідний елемент, або проскролити сторінку до відповідної секції.

Користувачу надається форма, яку він має заповнити для конвертації тексту в аудіозапис. Спершу користувачу потрібно обрати цільовий звук, для цього йому надано декілька опцій. Натиснувши на елемент вибору зліва, користувач може обрати голос із наданих йому прикладів. Обравши будь-який із варіантів, користувач має змогу прослухати обраний звук та, у випадку, якщо йому не сподобається почуте, змінити свій вибір. Описана поведінка зображена на рис. 4.23.

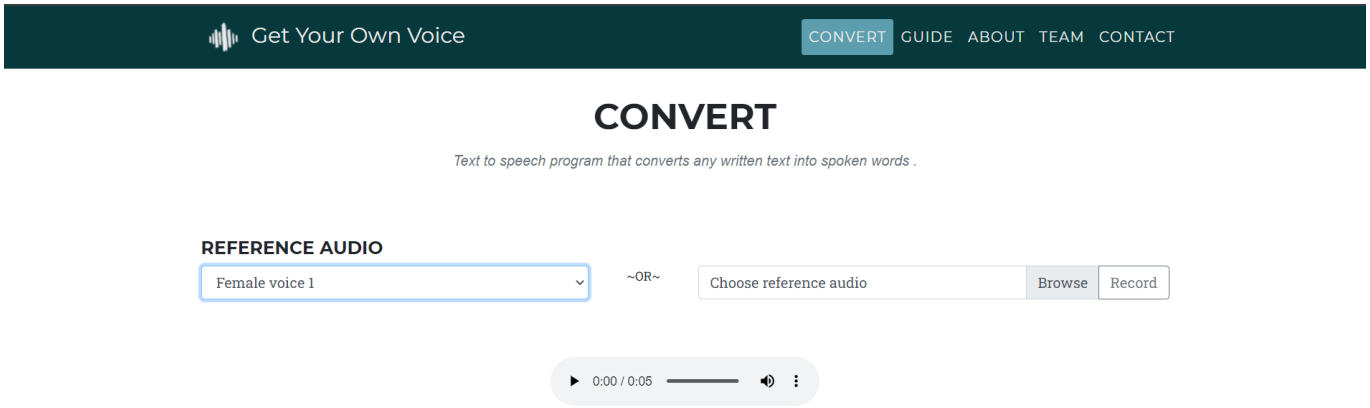


Рисунок 4.23 – Вибір цільового аудіозапису зі списку прикладів

Натиснувши на елемент вибору файлу користувач може обрати цільовий голос із власних аудіо файлів. Це може бути власний голос, записаний раніше, або приклад іншого голосу, записаного у файл. Поруч із полем вибору файлу розташована кнопка запису голосу, якщо передчасно записаний звук відсутній, або ж користувачу зручніше зробити запис прямо на сайті. Так само, як і в попередньому випадку, користувач може прослухати обраний звук та за потреби змінити вибір. У випадку, якщо заповнено обидва поля та створено запис, для прослуховування відображається той, який було обрано останнім. Приклад описаної поведінки наведено на рис. 4.24.

Рисунок 4.24 – Вибір цільового аудіозапису зі власних файлів

Після цього користувач має заповнити поле введення тексту та надати бажаний текст для перетворення на аудіо. Керувати паузами користувач може, переносячи текст на наступний рядок. Розроблювана система не обмежена в обсязі тексту. Проте, від довжини тексту залежить час обробки та озвучування. Приклади введеного тексту представлено на рис. 4.25 – 4.26.

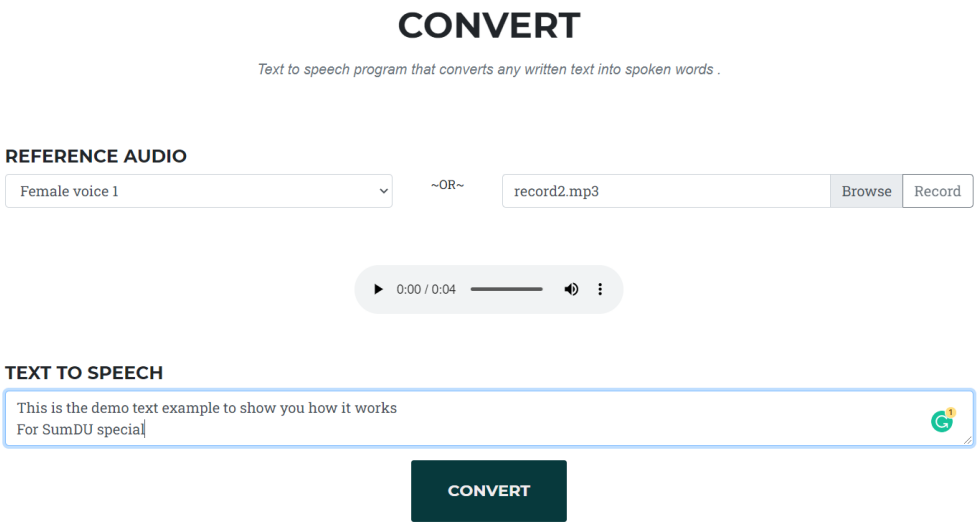


Рисунок 4.25 – Введення тексту для конвертації

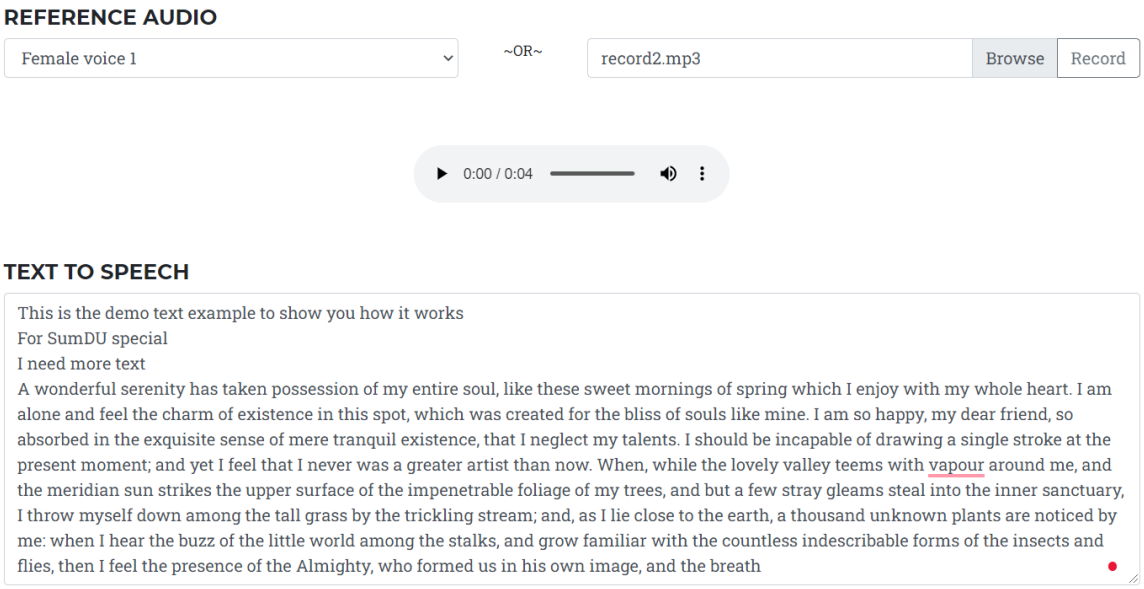


Рисунок 4.26 – Введення тексту для конвертації

Після введення та редагування тексту користувач має натиснути кнопку «Конвертувати». Результат буде відображено для прослуховування одразу після завершення конвертації (рис. 4.27).

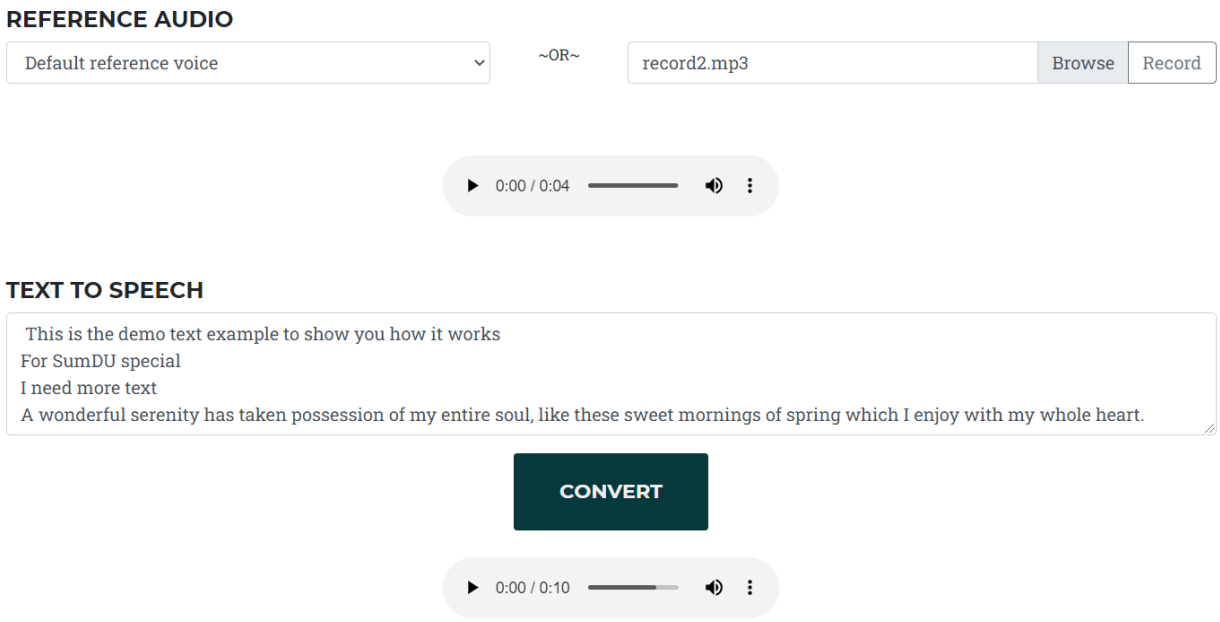


Рисунок 4.27 – Відображення результату озвучення тексту

ВИСНОВКИ

В ході роботи над проектом кваліфікаційної роботи магістра було розглянуто предметну область розроблюваного проекту, актуальність проблеми, проведено аналіз аналогічних онлайн-систем. В першу чергу було проведено ознайомлення та аналіз предметної області, досліджено характеристики, алгоритми, існуючі проблеми, та інше.

Було здійснено пошук та аналіз вже існуючих аналогічних онлайн систем озвучування тексту. Також, було детально вивчено актуальність проблеми.

Крім того, було сформовано технічне завдання на розробку на розробку онлайн-системи синтезованого голосового озвучування текстів, в якому зазначені усі визначені вимоги до продукту проекту.

Другий розділ було присвячено визначенню мети та задач проекту, також вибору методів дослідження та засобів реалізації, що будуть використані для реалізації мети проекту.

У третьому розділі було проведено моделювання онлайн-системи синтезованого голосового озвучування текстів, представлене у вигляді UML та IDEF0 діаграм, також було проведено планування робіт.

Четвертий розділ присвячено опису процесу реалізації системи на основі нейронної мережі для синтезу мовлення, розробки інтерфейсу та інтеграції інтерфейсу із системою.

Результатом проведеної роботи є аналіз предметної області, визначена мета та вимоги до продукту проекту, проведене моделювання онлайн-системи та виконано планування робіт проекту. В результаті реалізації, у відповідності з моделюванням було створено онлайн-систему синтезованого голосового озвучування текстів.

Метою даної роботи, було визначено розробку онлайн-системи синтезованого голосового озвучування текстів. Основними задачами в ході виконання розробки продукту проекту які було визначено на початку виконання реалізації продукту проекту були:

– детальний аналіз предметної області та аналогічних продуктів;

– вибір методів дослідження та технологій розробки продукту проекту;

– розробка системи на основі нейронних мереж для синтезу мовлення, яка може генерувати звук різними голосами, у тому числі тими, що не були відомі під час навчання;

– розробка інтерфейсу та інтеграція із системою;

– проведення тестування продукту.

Використовуючи можливості бібліотеки PyTorch та мікрофреймворку Flask було реалізовано системи на основі нейронних мереж для синтезу мовлення, а за допомогою Bootstrap разом із HTML, CSS та JS було розроблено інтерфейс веб додатку. В ході реалізації роботи було в повній мірі виконано задачі проекту.

Розроблена онлайн-система відповідає встановленим функціональним вимогам, а саме, було розроблено та навчено систему синтезу мовлення із тексту, що дозволяє виконувати перетворення тексту на мовлення голосом цільового мовця. Було надано передчасно створені голоси, а також можливість генерації нового голосу із завантаженого або записаного користувачем аудіо. Реалізовано функціонал синтезованого озвучення введеного тексту із передчасно створених голосів або згенерованого нового голосу. Крім того, створена онлайн-система не обмежує користувача в обсязі озвучуваного тексту та дозволяє керувати паузами. Після виконання синтезованого озвучування система дозволяє прослуховувати отриманий результат.

Оскільки, розроблена онлайн-система виконує перетворення тексту на мовлення, використовуючи не лише голоси, які було завчасно надано системою, а і будь-яким іншим голосом. Виходячи з цього, вона може бути корисна, як просто для перетворення бажаного тексту на аудіо, так і для більш цілеспрямованого використання саме синтезу власного голосу. Система буде корисною користувачам, які потребують озвучення із використанням конкретного голосу, наприклад, людям, які з певних причин втратили голос.

Крім вже описаного функціоналу, в подальших планах є вдосконалення продукту проекту завдяки розширення функціоналу шляхом одночасного використання кількох мов, що може бути оформлено в нову наукову роботу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wavenet: A generative model for raw audio [Електронний ресурс] / [A. van den Oord, S. Dieleman, H. Zen та ін.]. – 2016. – Режим доступу до ресурсу: http://arxiv.org/abs/1609.03499.

2. Natural TTS synthesis by conditioning wavenet on mel spectrogram predictions [Електронний ресурс] / [J. Shen, R. Pang, R. J. Weiss та ін.]. – 2017. – Режим доступу до ресурсу: http://arxiv.org/abs/1712.05884.

3. Shirali-Shahreza S. MOS Naturalness and the Quest for Human-Like Speech [Електронний ресурс] / S. Shirali-Shahreza, G. Penn // IEEE. – 2018. – Режим доступу до ресурсу: https://ieeexplore.ieee.org/document/8639599.

4. Tacotron: A fully end-to-end text-to-speech synthesis model [Електронний ресурс] / [Y. Wang, R. Skerry-Ryan, D. Stanton та ін.]. – 2017. – Режим доступу до ресурсу: https://www.researchgate.net/publication/315696313_Tacotron_A_Fully_End-to-End_Text-To-Speech_Synthesis_Model.

5. Deep voice 2: Multi-speaker neural textto-speech [Електронний ресурс] / [S. O. Arik, G. Diamos, A. Gibiansky та ін.]. – 2017. – Режим доступу до ресурсу: https://www.researchgate.net/publication/317163898_Deep_Voice_2_Multi-Speaker_Neural_Text-to-Speech.

6. Neural voice cloning with a few samples [Електронний ресурс] / [S. O. Arik, J. Chen, K. Peng та ін.]. – 2018. – Режим доступу до ресурсу: https://proceedings.neurips.cc/paper/2018/file/4559912e7a94a9c32b09d894f2bc3c82-Paper.pdf.

7. Transfer learning from speaker verification to multispeaker text-to-speech synthesis [Електронний ресурс] / [Y. Jia, Y. Zhang, R. J. Weiss та ін.]. – 2018. – Режим доступу до ресурсу: https://arxiv.org/abs/1806.04558.

8. UNESCO Literacy [Електронний ресурс] – Режим доступу до ресурсу: https://en.unesco.org/themes/literacy.

9. Up to 45 million blind people globally – and growing [Електронний ресурс] – Режим доступу до ресурсу: https://www.who.int/news/item/09-10-2003-up-to-45-million-blind-people-globally---and-growing.

10. VoxWorker [Електронний ресурс] – Режим доступу до ресурсу: https://voxworker.com/.

11. TextFromToSpeech [Електронний ресурс] – Режим доступу до ресурсу: https://www.textfromtospeech.com/uk/text-to-voice/.

12. Generalized End-To-End Loss For Speaker Verification [Електронний ресурс] / L. Wan, Q. Wang, A. Papir, I. L. Moreno – Режим доступу до ресурсу: https://arxiv.org/pdf/1710.10467.pdf.

13. Tacotron: Towards End-To-End Speech Synthesis [Електронний ресурс] / [Y. Wang∗, R. J. Skerry-Ryan∗, Daisy Stanton та ін.] – Режим доступу до ресурсу: https://arxiv.org/pdf/1703.10135.pdf.

14. Efficient Neural Audio Synthesis [Електронний ресурс] / [N. Kalchbrenner, E. Elsen, K. Simonyan та ін.] – Режим доступу до ресурсу: https://arxiv.org/pdf/1802.08435.pdf.

15. Bootstrap Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/docs/4.6/getting-started/introduction/.

16. Flask Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://flask.palletsprojects.com/en/2.0.x/.

17. PyTorch Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://pytorch.org/docs/stable/index.html.

18. Tacotron 2 [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Rayhane-mamah/Tacotron-2.

19. Моделирование на UML. [Електронний ресурс] – Режим доступу до ресурсу: http://.uml3.ru/.

20. A Guide to the Project Management Body of Knowledge (PMBOK® Guide)– Sixth Edition, 2017. – 760 c.

21. Taylor P. Text-to-Speech Synthesis / Paul Taylor., 2009.

22. Narayanan S. Text to Speech Synthesis: New Paradigms and Advances / Shrikanth Narayanan., 2005.

23. Шолле Ф. Deep Learning with Python (Глубокое обучение на Python) / Франсуа Шолле., 2017.

24. Raschka S. Python Machine Learning / S. Raschka, V. Mirjalili., 2015.

25. Mueller A. Introduction to Machine Learning with Python: A Guide for Data / A. Mueller, S. Guido., 2016.

26. Howard J. Deep Learning for Coders with Fastai and PyTorch / J. Howard, S. Gugger., 2020.

27. Программируем с PyTorch: Создание приложений глубокого обучения, 2020.

28. Stevens E. Deep Learning with PyTorch / E. Stevens, L. Antiga, T. Viehmann., 2020.

29. Yuxi (Hayden) Liu. Python Machine Learning by Example – Third Edition / Yuxi (Hayden) Liu., 2020.

30. Grinberg M. Flask Web Development: Developing Web Applications with Python / Miguel Grinberg., 2018.

ДОДАТОК А
ТЕХНІЧНЕ ЗАВДАННЯ
на розробку «Онлайн-система синтезованого голосового озвучування текстів»

1 Призначення й мета створення програмного продукту

1.1 Призначення розробки

Додаток призначений для синтезованого озвучення будь-яких текстів, із використанням наданих або згенерованих нових голосів, за допомогою нейронних мереж.

1.2 Мета створення додатку

Мета полягає у розробці онлайн-системи синтезованого голосового озвучування текстів.

1.3 Цільова аудиторія

У цільовій аудиторії програмного продукту можна виділити наступні групи:

1. Клієнти, користувачі додатку, які зацікавлені в озвучуванні текстів для власного використання.
2. Клієнти, користувачі додатку, які у зв’язку зі здоров’ям потребують синтезу текстів у мовлення.
3. Компанії, які потребують створення голосу для бренду.
4. Інші зацікавлені користувачі.

2 Вимоги до програмного продукту

Розроблюваний програмний продукт повинен бути реалізований у вигляді онлайн-системи, (далі – продукт, програмний продукт), клієнтська частина якого є інтернет сторінка, з набором елементів керування (кнопки та інші) та забезпечує виконання функціональних можливостей, визначених у пункті 2.3.

2.1 Вимоги до технології розробки

Продукт розробляється ітеративно з урахуванням принципів та технологій уніфікованого процесу розроблення програмного забезпечення. Програма повинна бути реалізована мовою Python з використанням PyTorch. Також повинні бути використані моделі TTS: Tacotron – модель для генерації спектрограм з тексту, WaveRNN – vocoder для генерації звуку із спектрограм та GE2E (Generalized End-to-End Loss for Speaker Verification) – encoder для ефективної ідентифікації мовлення.

2.2 Вимоги до програмного забезпечення

Для коректної роботи всіх можливостей онлайн-системи програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам для веб-браузера: Mozilla Firefox – починаючи з версії 4.0, Google Chrome – 69 починаючи з версії 9.0, Opera – починаючи з версії 12.0, Safari – починаючи з версії 8.0, Interten Explorer – починаючи з версії 11.0.

2.3 Вимоги до функціональних характеристик

Додаток повинен забезпечувати виконання наступних функції:

• наявність навченої системи синтезу мовлення із тексту;

• наявність передчасно створених голосів;

• генерація нових голосів;

• введення тексту для озвучення;

• керування паузами;

• синтезоване озвучення введеного тексту;

• прослуховування озвученого;

3 Склад і зміст робіт зі створення додатку

Основні етапи розробки повинні складатися з наступних пунктів:

• Оформлення завдання для дипломної роботи;

• Планування роботи. Розроблення ТЗ, побудова мережевого графіку та діаграми Ганта;

• Розробка системи на основі нейронної мережі для синтезу мовлення;

• Розробка інтерфейсу та інтеграція із системою;

• Проведення тестування програмного продукту

• Розроблення інструкції користувача;

• Оформлення пояснювальної записки;

• Здача пояснювальної записки;

• Презентація роботи та її захист.

4 Вимоги до складу й змісту робіт із введення додатку в експлуатацію

В процесі розробки програмного продукту необхідно перевірити готовність продукту до передачі в експлуатацію, а також забезпечити його придатність та відповідність ТЗ.

5 Порядок контролю та приймання

Контроль коректності функціонування та придатності додатку здійснюється замовником на основі наданої пояснювальної записки до дипломної роботи та програмних файлів.Контроль ходу виконання проекту здійснюється на основі календарного плану виконання дипломної роботи:

• Перевірка завдання дипломної роботи.

• Перевірка ТЗ, мережевого графіка та діаграми Ганта.

• Перевірка структури додатку.

• Перевірка наявності необхідного функціоналу.

• Перевірка коректності тестування.

• Перевірка інструкції користувача.

• Здача ПЗ.

• Презентація.

ДОДАТОК Б
ПЛАНУВАННЯ РОБІТ

Мета проекту: розробити онлайн-систему синтезованого голосового озвучування текстів, за допомогою якого користувач зможе легко перетворити бажаний уривок тексту на мовлення, із використанням наданих або згенерованих нових голосів, за допомогою нейронних мереж. Нові голоси мають генеруватися із наданих користувачем екземплярів аудіо-записів голосу.

Результати деталізації мети проекту методом SMART представлені у табл. Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

S – Specific	Розробити онлайн-систему синтезованого голосового озвучування текстів
M – Measurable	Оскільки даний проект є некомерційним та не має оцінювачів. Інструментами для виміру результатів роботи в даному випадку є експертна група, що надасть оцінку.
A – Achievable	Мету проекту можна вважати досяжною оскільки вона була ретельно вивчена розробником, обговорена командою проекту та узгоджена із вимогами до програмного продукту.
R – Relevant	Усе необхідне для реалізації проекту апаратне та програмне забезпечення є доступним та коректно працює (необхідне для розробки GPU – NVIDIA Geforce GTX 1650, середовище розробки – Visual Studio Code та Firebase – хмарне сховище). Команда проекту є достатньо кваліфікованою та обізнаною в необхідних інструментах для виконання поставленої мети.
T – Time-framed	Даний проект розрахований у встановлені часові обмеження, які визначені керівником проекту та описані в календарному плані та матриці відповідальності. Проект буде виконано у встановлені терміни, що підтверджено календарним планом проекту.

Планування змісту структури робіт проекту (WBS)

WBS (Work Break Structure) служить одним із основних інструментів для планування змісту структури робіт IT-проекту. Він є графічним представленням

проекту, що описано у вигляді ієрархічної структури, яка досягається шляхом розбиття робіт проекту на окремі результати, що повинні бути досягнуті на конкретних етапах для досягнення мети. WBS безумовно є найефективнішим способом візуалізації всього проекту, а також дозволяє зосередити увагу команди проекту на очікуваному результаті.

Оскільки описаний результат є бажаним досягненням проекту, після планування змісту структури робіт, буде отримано узгоджений із командою достатньо стабільний набір категорій, на які вона може спиратися в майбутньому. Отримана діаграма WBS представлена на рис. Б.1.

Планування структури організації, для впровадження готового проекту (OBS)

Після того, як було сформувано WBS починається етап виконання розробки організаційної структури виконавців проекту – OBS (Organization structure). Дана структура є графічним представленням проекту, описаним у вигляді переліку осіб, що задіяні в процесі реалізації продукту проекту, та є відповідальними за певні процеси чи роботи проекту.

Сформована організаційна структура описує розподіл робіт між учасниками проекту і те, як вони взаємодіють між собою в загальній системі. Простіше кажучи, вона описує, хто що робить, для досягнення мети проекту. Сформована організаційна структура проекту представлена на рис. Б.2.

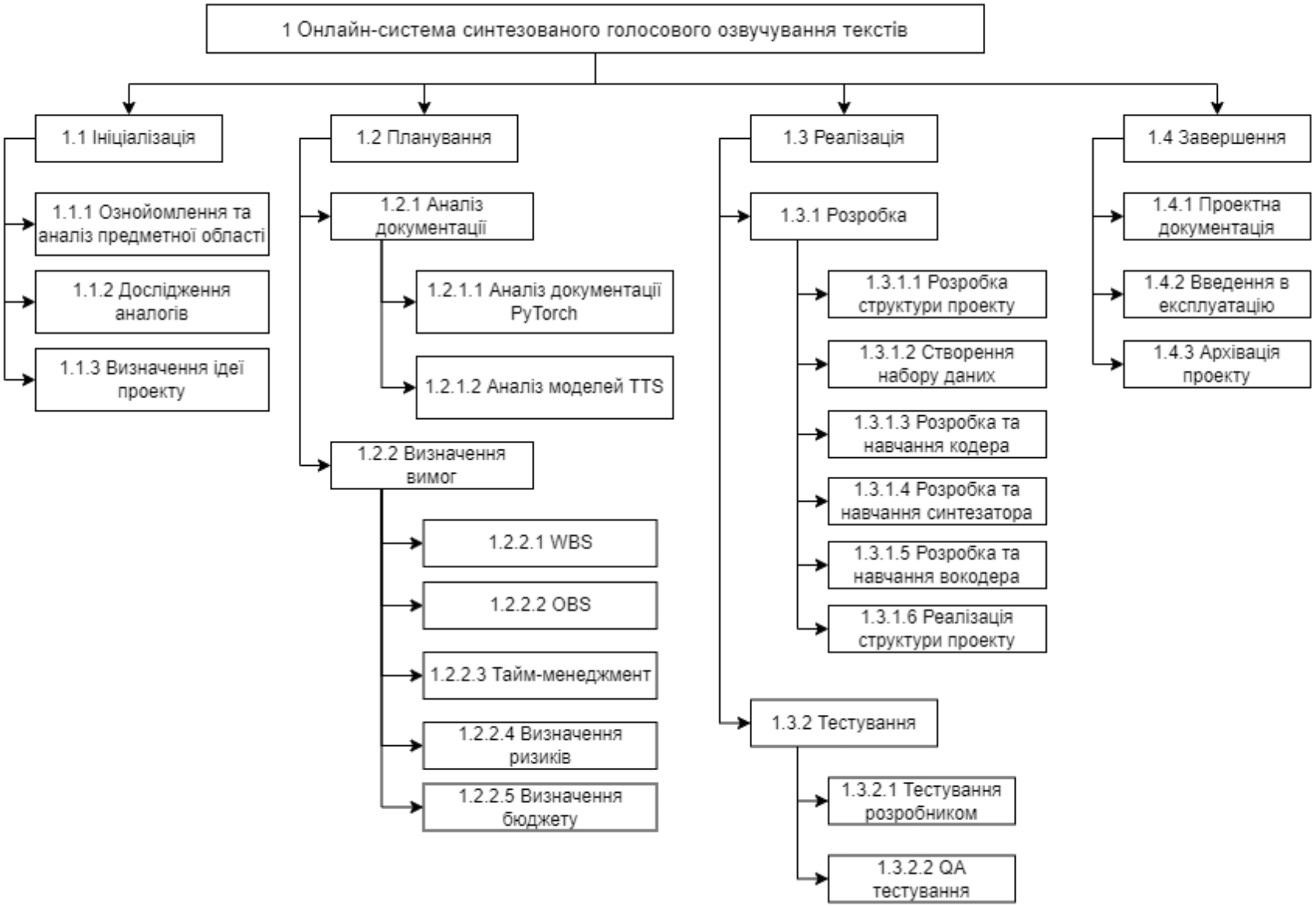


Рисунок Б.1 – Ієрархічна структура робіт проекту

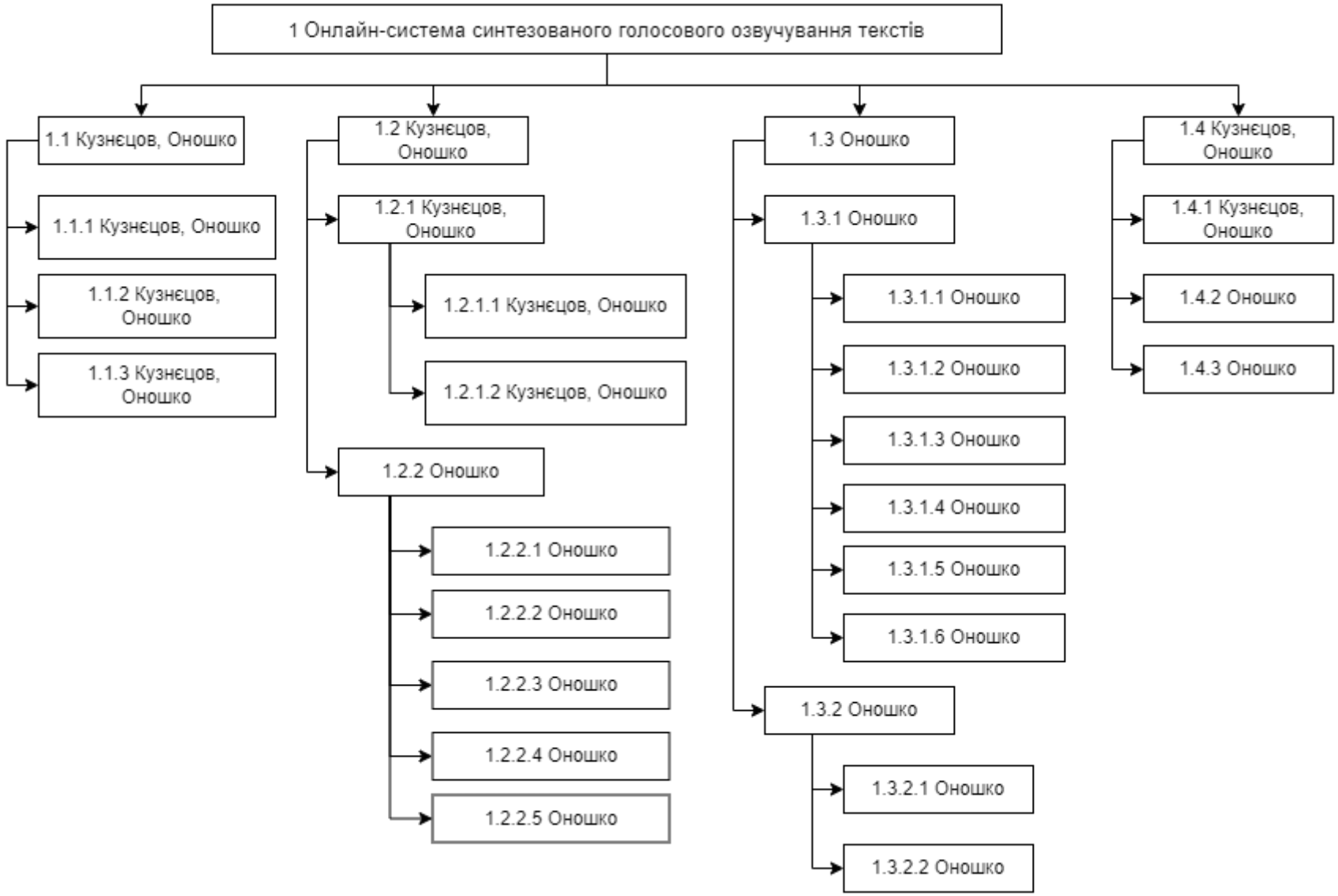


Рисунок Б.2 – Організаційна структура проекту

Побудова матриці відповідальності (виконавців пакетів робіт)

На основі двох раніше сформованих структур проекту (WBS та OBS) було побудовано документ, що формалізує розподіл ролей або представників команди проекту, щодо відповідільності за виконання робіт та досягнення кінцевих цілей проекту – мариця відповідальності, або Responsibility Assignment Matrix (RAM). Серед матриць відповідальності існує кілька моделей RAM, в залежності від типу залученості до проекту. Сформована матриця відповідальності представлена на рис Б.3.

RACI є моделлю, яка робить проект більш організованим, інформує про завантаженість команди проекту, оскільки показує, яка роль відведена кожному учаснику команди.

- Responsible – відповідальний за роботу та досягнення поставлених цілей, виконавець завдання.
- Accountable – відповідальний за якість виконання на фінальному етапі виконання робіт, затверджує результат того чи іншого завдання, тобто відповідальний за результати та якість.
- Consulted – відповідальний за інформацію, із цим учасником відбувається комунікація в ході виконання робіт проекту, його повідомляють про рішення та обговорюють із ним завдання.
- Informed – учасник, якого необхідно інформувати про хід і результат процесу виконання завдань, із ним найчастіше встановлена лише одностороння комунікація.

ID	WBS Element Description	PM	Team Leader	Developer	Analyst	QA
1.1.1	Ознайомлення та аналіз предметної області	R	A	-	-	-
1.1.2	Дослідження аналогів	C	A/R	I	C	-
1.1.3	Визначення ідеї проекту	C	A/R	C	C	-
1.2.1.1	Аналіз документації PyTorch	-	A	R	-	-
1.2.1.2	Аналіз моделей TTS	-	A	R	-	-
1.2.2.1	Планування WBS	R	C	C	A	-
1.2.2.2	Планування OBS	R	C	C	A	-
1.2.2.3	Тайм-менеджмент	C	A	I	R	-
1.2.2.4	Визначення ризиків	C	A/R	C	C	-
1.2.2.5	Визначення бюджету	A/C	-	-	R	-
1.3.1.1	Розробка структури проекту	R	A	C	C	I
1.3.1.2	Створення набору даних	I	A/C	R	-	I
1.3.1.3	Розробка та навчання кодера	I	A/C	R	-	I
1.3.1.4	Розробка та навчання синтезатора	I	A/C	R	-	I
1.3.1.5	Розробка та навчання вокодера	I	A/C	R	-	I
1.3.1.6	Реалізація структури проекту	I	A/C	R	-	I
1.3.2.1	Тестування розробником	I	A	R	C	I
1.3.2.2	QA тестування	I	A	I	-	R
1.4.1	Проектна документація	R	A	C	C	C
1.4.2	Введення в експлуатацію	C	A/R	I	I	I
1.4.3	Архівація проекту	I	A/R	I	I	I

Рисунок Б.3 – Матриця відповідальності проекту (RAM)

Розробка PDM-мережі (розгорнутий вигляд мережевих діаграм Ганта)

Мережеві моделі створюються для кожного пакету робіт, що було визначено під час розробки структури WBS. В таких моделях встановлюються зв’язки між усіма задачами проекту. Таким чином завдяки мережевим моделям можна отримати певну визначену тривалість проекту, а також тривалість окремих процесів. PDM-мережі (Product Data Management) на даний момент використовуються найчастіше. Це мережі типу «вершина-робота», вони складаються з робіт, що розташовані у вузлах, та логічних зв’язків – стрілок, які являють собою переходи між роботами проекту. Дана PDM-мережа була розроблена за допомогою пакету MS Project, результат створення представлено на рис. Б.4-Б.6.

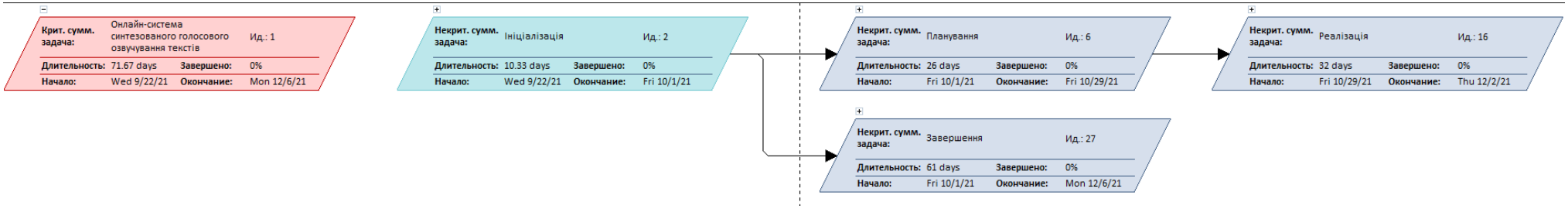


Рисунок Б.4 – Сумарні задачі PDM-мережі

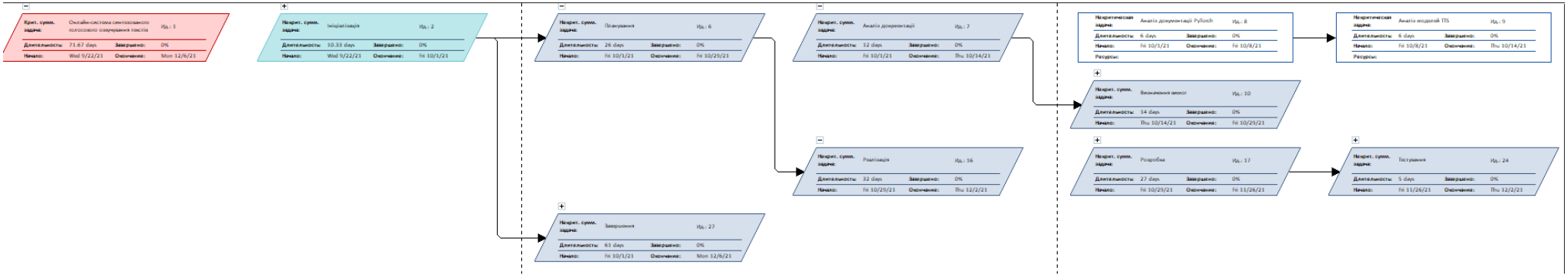


Рисунок Б.5 – PDM-мережа у згорнутому до основних моментів вигляді

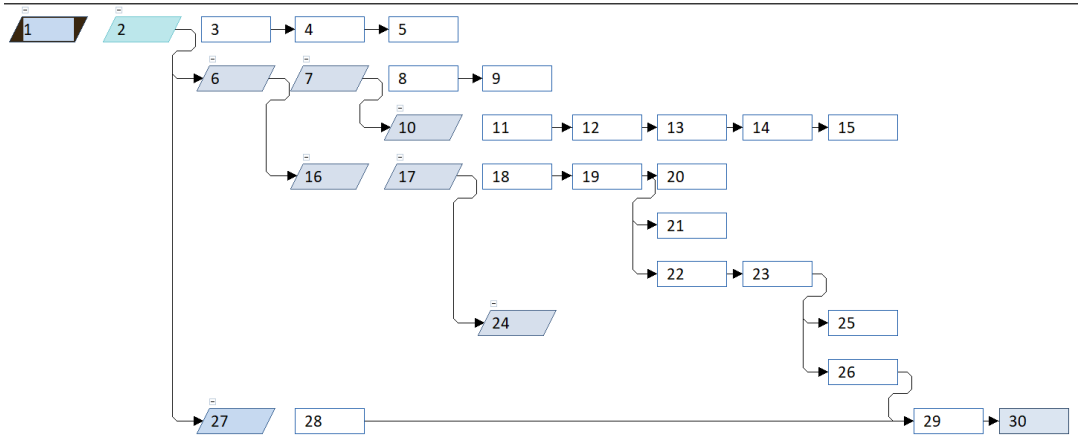


Рисунок Б.6 – PDM-мережа у повному вигляді

Побудова діаграми Ганта

На відміну від попередньо розробленої PDM-мережі, діаграма Ганта дозволяє отримувати реальну тривалість кожного процесу проекту з урахуванням вихідних і святкових днів, та обмежених ресурсів, а також відслідковувати відсоток виконаних робіт. Вона являє собою гістограму, що містить горизонтальні відрізки, розташовані в системі між осями: переліком робіт та часовою шкалою. По розташуванню кожного процесу можна побачити початок, кінець та тривалість кожної із робіт, крім того можуть бути відображені зв’язки між роботами та відсоток виконаних робіт по кожному завданню одразу на гістограмі.

Список робіт діаграми, що сформовано на основі раніше розробленої WBS, представлений на рис. Б.7. Діаграма Ганта була створена за допомогою пакету MS Project та представлена на рис. Б.8.

Онлайн-система синтезованого голосового озвучування текстів	**71.67 days**	**Wed 9/22/21**	Mon 12/6/21
Ініціалізація	**10.33 days**	**Wed 9/22/21**	**Fri 10/1/21**
Ознойомлення та аналіз предметної області	6 days	Wed 9/22/21	Tue 9/28/21
Дослідження аналогів	2 days	Tue 9/28/21	Wed 9/29/21
Визначення ідеї проекту	2 days	Thu 9/30/21	Fri 10/1/21
Планування	**26 days**	**Fri 10/1/21**	**Fri 10/29/21**
Аналіз документації	**12 days**	**Fri 10/1/21**	**Thu 10/14/21**
Аналіз документації PyTorch	6 days	Fri 10/1/21	Fri 10/8/21
Аналіз моделей TTS	6 days	Fri 10/8/21	Thu 10/14/21
Визначення вимог	**14 days**	**Thu 10/14/21**	**Fri 10/29/21**
Планування WBS	2 days	Thu 10/14/21	Mon 10/18/21
Планування OBS	2 days	Mon 10/18/21	Tue 10/19/21
Тайм-менеджмент	4 days	Tue 10/19/21	Fri 10/22/21
Визначення ризиків	3 days	Fri 10/22/21	Tue 10/26/21
Визначення бюджету	3 days	Wed 10/27/21	Fri 10/29/21
Реалізація	**32 days**	**Fri 10/29/21**	**Thu 12/2/21**
Розробка	**27 days**	**Fri 10/29/21**	**Fri 11/26/21**
Розробка структури проекту	3 days	Fri 10/29/21	Tue 11/2/21
Створення набору даних	7 days	Tue 11/2/21	Tue 11/9/21
Розробка та навчання кодера	10 days	Tue 11/9/21	Fri 11/19/21
Розробка та навчання синтезатора	10 days	Tue 11/9/21	Fri 11/19/21
Розробка та навчання вокодера	10 days	Tue 11/9/21	Fri 11/19/21
Реалізація структури проекту	7 days	Fri 11/19/21	Fri 11/26/21
Тестування	**5 days**	**Fri 11/26/21**	**Thu 12/2/21**
Тестування розробником	3 days	Fri 11/26/21	Tue 11/30/21
QA тестування	5 days	Fri 11/26/21	Thu 12/2/21
Завершення	**61 days**	**Fri 10/1/21**	**Mon 12/6/21**
Проектна документація	58 days	Fri 10/1/21	Thu 12/2/21
Введення в експлуатацію	2 days	Thu 12/2/21	Fri 12/3/21
Архівація проекту	1 day	Fri 12/3/21	Mon 12/6/21

Рисунок Б.7 – Список робіт діаграми Ганта

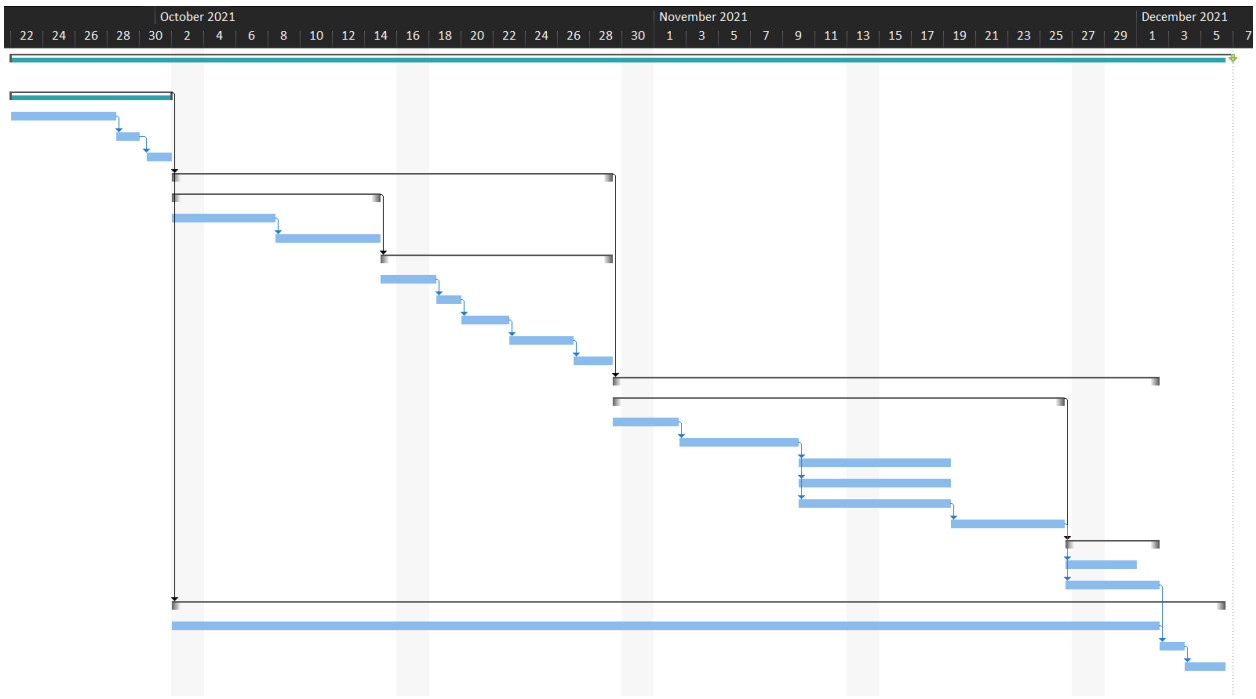


Рисунок Б.8 – Діаграма Ганта

Управління ризиками

Ризик проекту є ймовірнісною подією чи умовою, що може мати як позитивний так і негативний вплив на проект вцілому. Серед процесів управління ризиками проекту є процеси ідентифікації, аналізу, планування реагування та контролю ризиків на проекті.

При ідентифікації ризиків визначаються ризики, що в змозі вплинути на проект, а також визначаються характеристики цих ризиків. В ході виконання ідентифікації було визначено ряд ризиків, що зможуть виникнути в даному випадку на проекті. Наступним етапом є проведення аналізу ризиків та оцінка ймовірності їх виникнення на проекті та вплив їх на результат за шкалою від 1 до 5, де 1 – мінімальна оцінка, а 5 – відповідно, максимальна.

В результаті проведеного аналізу ймовірності та впливу кожного з визначених ризиків в рамках проекту було сформовано матрицю (рис. Б.10), за допомогою якої можна визначити рівень ризику (Probability / Impact Matrix). Результати аналізу ризиків представлено на рис. Б.9.

Risk ID	Risk description	Probability	Impact	Level of risk	Risk responce strategy
R1	Зміна визначених вимог	2	3	M	Підготовка детальної специфікації. Обговорення і погодження із клієнтом
R2	Неможливість дотримуватися календарного плану	3	4	M	Створення плану реалізації проекту на основі ретельного аналізу всіх робіт.
R3	Поява незапланованих робіт	3	5	H	Перепланування та оптимізація часу на випадок незапланованих робіт
R4	Проблеми з програмним забезпеченням	4	5	H	Своєчасне оновлення системи та необхідного ПЗ, використання ліцензійного ПЗ, створення резервних копій на зовнішніх/хмарних носіях
R5	Проблеми з апаратним забезпеченням	3	5	H	Здійснення профілактичної перевірки, забезпечення резервних копій проекту на зовнішніх/хмарних носіях
R6	Непорозуміння у колективі	2	2	L	Вміння швидко виявити причину непорозумінь і усунути її.
R7	Неправильна розстановка пріоритетів задач	3	3	M	Правильний розподіл пріоритетів завдань. Оптимізація пріоритетів завдань

Рисунок Б.9 – Risk Register

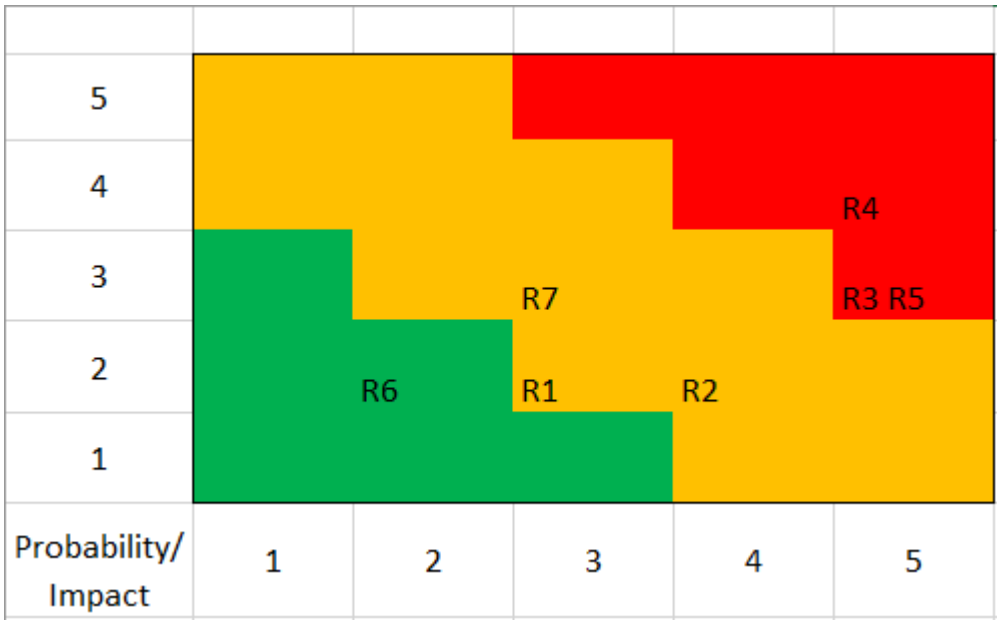


Рисунок Б.10 – Probability Impact Risk Matrix

ДОДАТОК В
ФАЙЛИ КОДУ РЕАЛІЗАЦІЇ

index.py

```
from flask import Flask, request, render_template
import sys
import soundfile as sf
import os
from werkzeug.utils import secure_filename

app = Flask(__name__, template_folder="app/", static_folder="static/")

port = int(os.environ.get("PORT", 5010))

@app.after_request
def header(response):
    response.headers['Cache-Control'] = 'no-store, no-cache, must-revalidate, post-check=0, pre-check=0,
max-age=0'
    response.headers['Pragma'] = 'no-cache'
    response.headers['Expires'] = '-1'
    return response

def htmbuilder(outputaudio):
    x = ""
    x+="<audio controls>"
    x+="  <source src='"+str(outputaudio)+"' type='audio/wav'>"
    x+="</audio><br>"
    return x

def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
def upload_file():
    fil = []
    if request.method == 'POST':
        request.form["textarea"]
        select = request.form.get('select')
        if select != '':
            fil.append("File selected from defaults")
            fil.append(select)
            return fil

        record = request.form.get('record')
        if record != '':
            fil.append("File recorded and uploaded")
            fil.append(record)
            return fil

        file = request.files['file']
        if allowed_file(file.filename):
            file.save(UPLOAD_FOLDER+secure_filename(file.filename))
            fil.append("File uploaded successfully")
            fil.append(UPLOAD_FOLDER+secure_filename(file.filename))
```

```
            return fil
        else:
            fil.append("Not An Expected File")
            return fil


@app.route('/',methods=['GET', 'POST'])
def hello_world():
    uploaded_file = upload_file()
    lig = "This is a demo text. This will be there if you do not add any text example."
    if request.method == 'POST':
        textarea = request.form["textarea"]
    print(str(textarea))
    #return mainpage()
    # if List is empty
    if not uploaded_file:
        return render_template("index.html", text="", output="", outputAudio="")
    else:
        try:
            ## Init parser
            parser = argparse.ArgumentParser(description=' Process some models arguments.'
formatter_class=argparse.HelpFormatter)
            parser.add_argument("-enc",                    "—e-trained-path",                    type=Path,
default="encoder/trainedmodel/run_pretrained.pt")
            parser.add_argument("-sync",                    "—s-trained-path",                    type=Path,
default="synthesizer/trainedmodels/run_pretrained/run_pretrained.pt")
            parser.add_argument("-voc",                    "—v-trained-path",                    type=Path,
default="vocoder/trainedmodels/run_pretrained/run_pretrained.pt")

            ## Parse arguments
            parsed_args = parser.parse_args()

            ## Load the models
            encoder.load(parsed_args.enc-trained-path)
            synthesizer = Synthesizer(parsed_args.sync-trained-path)
            vocoder.load(parsed_args.voc-trained-path)

            # Get the reference audio filepath
            in_path = uploaded_file[1]
            print(str(in_path))

            # Load directly from the filepath:
            preprocessed_form = encoder.preprocess(in_path)
             # If already loaded:
            initial_wav, freq_sampling = librosa.load(str(in_path))
            preprocessed_form = encoder.preprocess(initial_wav, freq_sampling)

            # Deriving of the embedding
            embedding = encoder.embedding_utterance(preprocessed_form)
            print("Embedding created successfully")

            ## The spectrogram generation
            text = str(textarea)
            texts = [text]
            embeddings = [embedding]

            spectrograms = synthesizer.synthesize_spectrograms(texts, embeddings)
```

```
        spec = spectrograms[0]
        print("Mel spectrogram created successfully")

        ## The waveform generation
        print("Start synthesis:")
        # The waveform synthesizing.
        created_wav = vocoder.infer_wav(spec)
        ## The post generation process
        created_wav = np.pad(created_wav, (0, synthesizer.freq_sample), mode="constant")
        # Run preprocees to remove silences
        created_wav = encoder.preprocess(created_wav)

        path = "static/output.wav"
        print(created_wav.dtype)
        sf.write(path, created_wav, synthesizer.freq_sample)
        return render_template("index.html", text=text, output=htmbuilder(path))
    except Exception as e:
        return render_template("index.html", text=text, output="Caught exception: %s" % repr(e))


if __name__ == '__main__':
    app.run(debug=True, host='localhost', port=port)
```

index.html Converter section

```
<!-- Text to speech converting Section -->
    <section id="convert">
        <div class="container">
            <div class="text-center">
                <h2 class="section-heading">Convert</h2>
                <h3 class="section-subheading text-muted">Text to speech program that converts any written text into spoken words .</h3>
            </div>

            <form name="textConverter" id="textConvertForm" novalidate="" method="post" enctype="multipart/form-data">
            <div class="row atextarean-items-stretch mb-5">
                <div class="col-xl-12">
                    <!-- Audio input -->
                    <div class="col-xl-12">
                        <h5>Reference audio</h5>
                        <div class="row align-items-stretch mb-5" id="reference">
                            <!-- Dropdown -->
                            <div class="col-md-5 form-group">
                                <select class="form-control" name="select" id="refAudioSelect">
                                    <option value="" selected>Default reference voice</option>
                                    <option value="static/1320_00000.mp3">Male voice 1</option>
                                    <option value="static/3575_00000.mp3">Female voice 1</option>
                                    <option value="static/6829_00000.mp3">Female voice 2</option>
                                    <option value="static/8230_00000.mp3">Male voice 2</option>
                                    <option value="static/p240_00000.mp3">Female voice 3</option>
                                    <option value="static/p260_00000.mp3">Male voice 3</option>
                                </select>
                            </div>
                            <div class="col-md-1 text-center align-middle">
                                <p class="asd"> ~OR~ </p>
```

```
                    </div>
                    <!-- Audio file or record -->
                    <div class="col-md-6 form-group input-group">
                      <div class="custom-file">
                        <input type="file" class="custom-file-input" name="file" id="refAudioFile" accept="audio/*">
                        <label class="custom-file-label" for="refAudioFile">Choose reference audio</label>
                      </div>
                      <div class="input-group-append">
                        <button class="btn btn-outline-secondary" type="button" id="refAudioRecord" data-toggle="modal" data-target="#recordModal">
                          Record
                        </button>
                        <input id="refAudioRecordFile" type="hidden" name="record"/>
                        <!-- <div id="audio-url-preview"></div> -->
                      </div>
                    </div>
                  </div>
                </div>
                <!-- Reference Audio -->
                <div class="col-xl-12 text-center mb-5">
                  <audio id="refAudio" controls hidden></audio>
                </div>
                <!-- Text to speech -->
                <div class="col-xl-12">
                  <h5>Text To Speech</h5>
                  <div class="form-group" id="text-form">
                    <textarea class="form-control" name="textarea" id="text"
                    placeholder="This is a placeholder. Paste your text here."
                    required data-validation-required-message="You need to insert some text to start converting">{{ text }}</textarea>
                    <p class="form-text text-danger"></p>
                  </div>
                </div>
                <!-- Button -->
                <div class="clearfix"></div>
                <div class="col-xl-12 text-center">
                  <div id="success"></div>
                  <button type="submit" class="btn btn-xl">Convert</button>
                </div>
                <!-- Output audio -->
                <div class="col-xl-12 text-center">
                  <audio id="outputAudio" controls hidden>
                    {{ outputaudio }}
                  </audio>
                </div>
                {{ output | safe}}
              </div>
            </div>
            </form>

            <div class="row text-center"></div>
          </div>
        </div>
```