TRIBHUVAN UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

A Project Report On

ENGLISH TEXT-TO-SPEECH SYNTHESIS USING TACOTRON2 FOR MEL SPECTROGRAM GENERATION

Submitted to:

Department of Computer Science and

Information Technology

Asian College of Higher Studies

Ekantakuna, Lalitpur

In Partial fulfillment of the requirements

For the Bachelor of Science in Computer Science

And Information Technology

Submitted by:

Albin Maharjan (TU Roll No.: 26704/077)

Diwash Joshi (TU Roll No.: 26712/077)

Nirayu Maharjan (TU Roll No.: 26721/077)

Under the supervision of

Mr. Pesal Rai

23rd January,2025

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by ALBIN MAHARJAN, DIWASH JOSHI, NIRAYU MAHARJAN entitled **"ENGLISH TEXT TO SPEECH SYNTHESIS USING TACTOTRON2 FOR MELSPECTROGRAM GENERATION"** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for the evaluation.

…………………………………………

Pesal Rai
Supervisor
Department of Computer Science and IT
Asian College of Higher Studies
Ekantakuna, Lalitpur

LETTER OF APPROVAL

This is to certify that this project prepared by ALBIN MAHARJAN, DIWASH JOSHI, NIRAYU MAHARJAN entitled **"ENGLISH TEXT TO SPEECH SYTHESIS USING TACOTRON2 FOR MELSPECTROGRAM GENERATION"** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion it is satisfactory in scope and quality as a project for the required degree.

-- **SIGNATURE of Supervisor** Mr. Pesal Rai Asian College of Higher Studies Ekantakuna, Lalitpur	-- **SIGNATURE of HOD/Coordinator** Mr. Pranaya Nakarmi Asian College of Higher Studies Ekantakuna, Lalitpur
-- **SIGNATURE of Internal Examiner**	-- **SIGNATURE of External Examiner**

ACKNOWLEDGEMENT

We would like to extend our sincere gratitude to Mr. Pesal Rai, our supervisor, for his invaluable assistance and guidance throughout the development of our project. We also wish to convey our appreciation to all the individuals who, whether directly or indirectly, contributed to the successful completion of this project. Your support, whether in the form of ideas or motivation, played a significant role in this endeavor.

This project would not have been achievable without the collective effort and support of everyone involved. Lastly, we would like to express our thanks to Asian College of Higher Studies for their continuous guidance, supervision, and provision of essential project-related information, as well as their support in bringing this project to fruition.

Project Members:

Albin Maharjan (TU Roll No.: 26704/077)

Diwash Joshi (TU Roll No.: 26712/077)

Nirayu Maharjan (TU Roll No.: 26721/077)

Date: January 23, 2025

ABSTRACT

This project focuses on the development of a Text-to-Speech (TTS) system using Tacotron2 for Mel-spectrogram generation and HiFi-GAN as a vocoder for high-quality speech synthesis. The system converts input text into natural and expressive speech through a robust pipeline that includes text preprocessing, phoneme conversion, and attention-based sequence modeling. Tacotron2 is utilized to generate Mel-spectrograms, capturing the prosody and tonal nuances of human speech, while HiFi-GAN efficiently reconstructs audio waveforms from the spectrograms to produce realistic and high-fidelity output. The implementation is designed for flexibility, allowing fine-tuning of parameters such as decoder steps, learning rates, and audio sampling rates to meet diverse application needs. This TTS system demonstrates its potential in applications such as virtual assistants, audiobook narration, and accessibility tools for visually impaired users.

Keywords: Text-to-Speech, Tacotron2, Mel-Spectrogram, HiFi-GAN, Speech Synthesis, Natural Language Processing.

TABLE OF CONTENTS

LIST OF ABBREVIATION

API	Application Programming Interface
CSS	Cascading Style Sheets
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
HiFi-GAN	High-Fidelity Generative Adversarial Network
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
MOS	Mean Opinion Score
NLP	Natural Language Processing
RNN	Recurrent Neural Network
SDLC	Software Development Life Cycle
Seq2Seq	Sequence-to-Sequence
TTS	Text-to-Speech
UI	User Interface
VS Code	Visual Studio Code

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1. Introduction

The English language, the dominant and most widely spoken in the world, plays a crucial role in text-to-speech (TTS) technology. Developments in this area have opened applications so wide that they show potential for TTS systems. TTS systems are designed to process written input to produce human talk-like speech. Text-to-speech is an important conversion in natural language processing (NLP), especially for languages with the richness and complexity of the phonetic structure of English. Now, TTS systems are vital to providing accessibility and interaction on various platforms during the current digital era.

TTS technology is commonly applied in a wide range of areas, from virtual assistants like Siri and Alexa to adaptive technology for people with challenges in reading, especially those with problems like dyslexia [1]. In such cases, TTS systems help by reading the text out loud. TTS is also actively implemented in automotive navigation for spoken directions and in customer service applications for automation, which helps to achieve greater efficiency.

As technology in TTS develops further, it is expected that this will change the way humans interact with digital content, making information more accessible and thereby helping better levels of user engagement. The development of an English TTS system shall strive to excel in contextual understanding, naturalness, and intelligibility. The latter is produced with the Tacotron2 model and has a multilayer LSTM network encoded in sequence-to-sequence decoders that generates the Mel spectrogram and is fed to the vocoder like Hifi-GAN to generate human like speech waveforms. This system keeps the meaning and tone of the input text and ensures smooth and flowing speech with the correct pronunciation, making the spoken output both clear and natural.

Technically, TTS systems solve the problem of people facing dyslexia, old age, and the disabled in reading and writing because barriers to text access usually exist worldwide. Speech synthesis increases opportunities for people to access information, therefore improving their abilities to interact with the environment. [2]

1.2. Problem Statement

Globally, 10-20% of the population is affected by dyslexia [3], while more than 12% experience difficulties with literacy [4] many are due to deteriorating vision as a consequence of age. These people face barriers in learning, information processing, and communication, given their most important means of interaction with the digital world: TTS. Despite these technological developments, most of the systems currently available do not have optimum accessibility for these groups, hence putting these populations at an extreme disadvantage in comparison.

The project aims to address this gap by developing an English Text-to-Speech (TTS) system using the Tacotron2 model, which leverages advanced deep learning techniques like sequence-to-sequence (seq2seq) modeling and Long Short-Term Memory (LSTM) networks. The goal is to generate natural-sounding speech that enhances digital content accessibility for individuals with dyslexia, older adults with vision impairments, and those facing literacy challenges. By doing so, the system seeks to provide a critical digital resource for the people who depend on such tools to navigate the digital world more effectively.

1.3. Objectives

The main aim of this project is to develop an English text to Text-to-speech system using Tacotron2 and HiFi-GAN:

i. To create a web interface that enables user to interact with the Text to Speech system by entering text and receiving its corresponding audio.

1.4. Scope and Limitations

The primary scope of a TTS system built with Tacotron2 and HiFi-GAN is to produce high-fidelity, natural-sounding speech. Tacotron2 converts text to Mel-spectrograms, which are then used by HiFi-GAN to generate high-quality, human-like speech waveforms. This allows for synthesis with rich prosody, emotional variation, and accurate pronunciation Speech Generation. One of the key advantages of using this combination is real-time speech synthesis. Both Tacotron2 and HiFi-GAN have been optimized for efficient computation, enabling faster speech generation suitable for applications like virtual assistants, audiobooks, and accessibility tools.

One significant limitation of this TTS system is the large volume of data required to train the models effectively. High-quality, labeled datasets are essential for achieving natural-sounding speech. Moreover, the training process demands considerable computational power, which may be a barrier for smaller organizations or research groups. HiFi-GAN excel in producing natural speech, they can still struggle with certain accents or non-standard pronunciations if not trained on diverse datasets. For instance, they may not handle homophones or nuanced linguistic features perfectly without additional fine-tuning

1.4. Development Methodology

To develop the Text to Speech system we have used the waterfall model. The Waterfall Model is a type of SDLC that is used for small projects whose requirements are well-known and don't require performing previous steps or backtracking to previous steps.

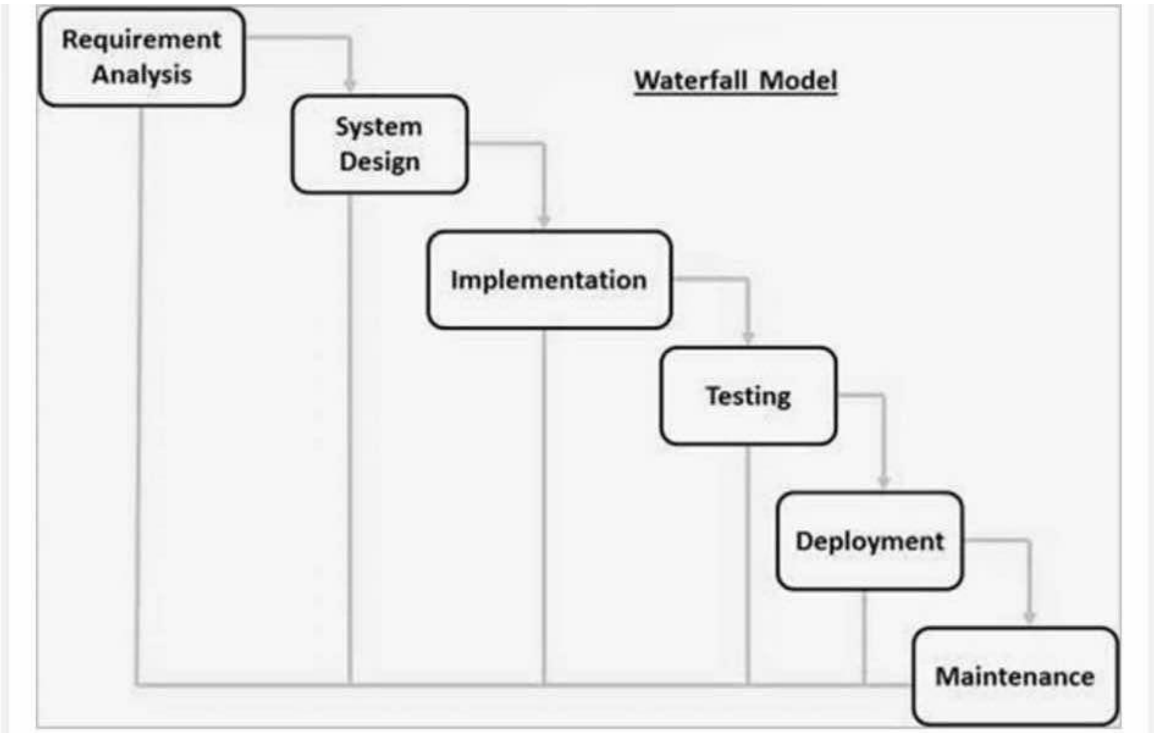

Figure 1.1 Waterfall Model of Text to Speech System

The project Text-to-speech system is developed using waterfall model because the project is comparatively small, the requirements are well-known, and the resources are available. As in waterfall models, it is hard to backtrack to previous stages when the testing phase is emphasized with a feedback loop. If the outcome after testing is not satisfactory, the process loops back to earlier stages (typically design or implementation) to make necessary adjustments. This loop continues until the system meets the desired level of quality and performance. Once satisfied with the result we will move forward to further stages.

1.5. Report Organization

Table 1.1 Outline of Document

Chapter 1: Introduction	The introductory chapter establishes the foundation for the document, offering a thorough introduction of the text to the speech system. It addresses the problem statement, articulates objectives, defines the project's scope and limitations, elucidates the chosen development methodology, and outlines the organizational structure of the subsequent report.
Chapter 2: Background Study and literature review	his chapter delves into the background study of Text-to-Speech (TTS) systems, providing insights into the evolution from traditional methods to modern deep learning models. The literature review critically examines Tacotron2 and HiFi-GAN, highlighting their strengths in generating high-quality, natural-sounding speech while addressing the limitations of earlier TTS approaches. It also emphasizes the role of Mel-spectrograms as key representations for speech synthesis. By exploring advancements in TTS technology, particularly in under-resourced languages, this chapter identifies the strengths and challenges of existing systems, setting the foundation for leveraging Tacotron2 and HiFi-GAN to enhance speech synthesis quality and efficiency.
Chapter 3: System Analysis	This chapter presents the system analysis for the Text-to-Speech (TTS) translator project, detailing the requirements, feasibility, and modeling approaches. The Requirement Analysis categorizes functional requirements, such as model training and speech generation, and non-functional requirements, including scalability and high-quality output. The Feasibility Study evaluates the project's technical, operational, economic, and schedule viability, affirming its practicality. The chapter also includes Analysis Diagrams, such as class, state, sequence, and activity diagrams, which illustrate the system's architecture, interactions, and processes. These elements

	collectively outline a structured approach to developing a robust and efficient TTS system using Tacotron2 and HiFi-GAN.
Chapter 4: System Design and Algorithm Details	This chapter outlines the system design and algorithm details for the Text-to-Speech (TTS) translator project. The Design section presents refined diagrams, including class, component, and deployment diagrams, to illustrate the system's modular architecture and physical infrastructure. The Algorithm Details section describes the core methodologies: Tacotron2 for generating Mel-spectrograms using sequence-to-sequence learning with attention, and HiFi-GAN for converting these spectrograms into high-fidelity audio waveforms using GAN-based techniques. Together, these design and algorithmic elements provide a comprehensive framework for implementing a scalable, efficient, and natural-sounding TTS system.
Chapter 5: Implementation and Testing	This chapter provides a detailed account of the implementation of the Text-to-Speech (TTS) system, focusing on the tools and technologies used, as well as the step-by-step development of key modules. The section also outlines the testing procedures, including both unit and system testing, to ensure the system's functionality, robustness, and performance. The implementation covers the integration of Tacotron2 and HiFi-GAN for speech synthesis, while the testing phase ensures that each component operates correctly and meets the project's requirements.
Chapter 6: Conclusion and Future Recommendations	This chapter concludes the report by summarizing the successful development of the Text-to-Speech (TTS) system, highlighting its use of Tacotron2 for Mel-spectrogram generation and HiFi-GAN for audio synthesis. The system delivers high-quality, human-like speech with an impressive Mean Opinion Score (MOS) of 4.48 ± 0.89. While the system meets key performance goals, recommendations for future improvements include optimizing backend processes for scalability, enhancing UI responsiveness, strengthening input validation, and implementing Continuous Integration (CI) pipelines for stability.

CHAPTER 2: BACKGROUND STUDY AND LITERATURE REVIEW

2.1. Background Study

Text-to-Speech Systems (TTS systems) aim to convert written text into spoken words. Traditional methods involve rule-based models or concatenative synthesis, where recorded speech fragments are joined together. However, these methods struggle with generating natural prosody and variability. Recent advances in deep learning, particularly with models like Tacotron2 and HiFi-GAN, have significantly improved the quality and fluidity of synthesized speech by learning from large datasets of paired text and audio.

I. Concepts and Techniques

a) Overview of Tacotron2:

Tacotron2 is a neural network architecture for end-to-end TTS. It consists of two main components: an encoder-decoder with attention for generating Mel-spectrograms from text, and a vocoder that converts these spectrograms into audio. Unlike earlier models, Tacotron2 bypasses the need for phonetic transcripts and directly works with text input. It improves upon Tacotron by simplifying the architecture and enhancing the quality of spectrogram generation. [5]

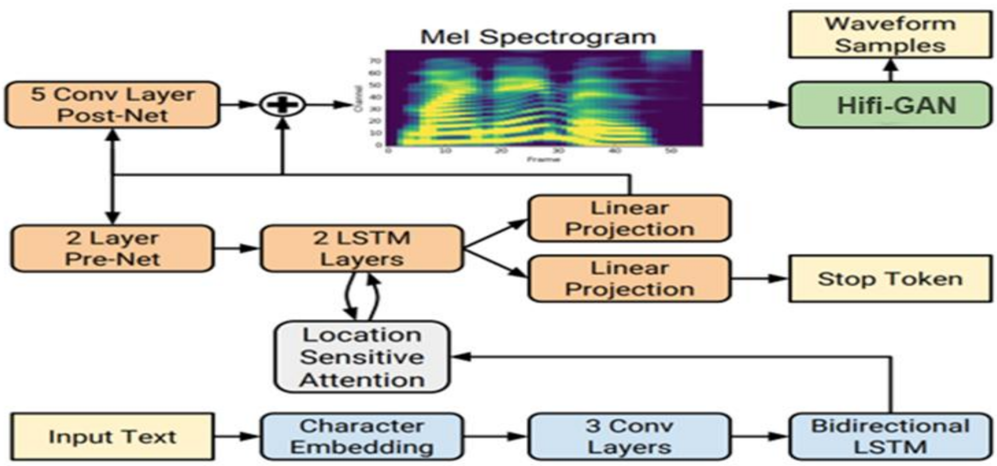

Figure 2.1 Working of Tacotron2 Model

b) Overview of HiFi-Gan:

HiFi-GAN is a generative adversarial network (GAN) designed for high-quality, real-time speech synthesis. It converts Mel-spectrograms into natural-sounding waveforms with minimal distortion, making it a highly effective vocoder for TTS systems. HiFi-GAN uses a generator and discriminator architecture to improve the realism of the generated speech, reducing noise and artifacts. Compared to traditional vocoders, it offers faster speech generation with improved fidelity and is suitable for applications like virtual assistants and audiobooks. Its architecture enhances prosody and intonation, resulting in more expressive and lifelike speech. The model's efficiency and high output quality have made it a leading choice in modern TTS systems like those using Tacotron2 for Mel-spectrogram generation. [6]

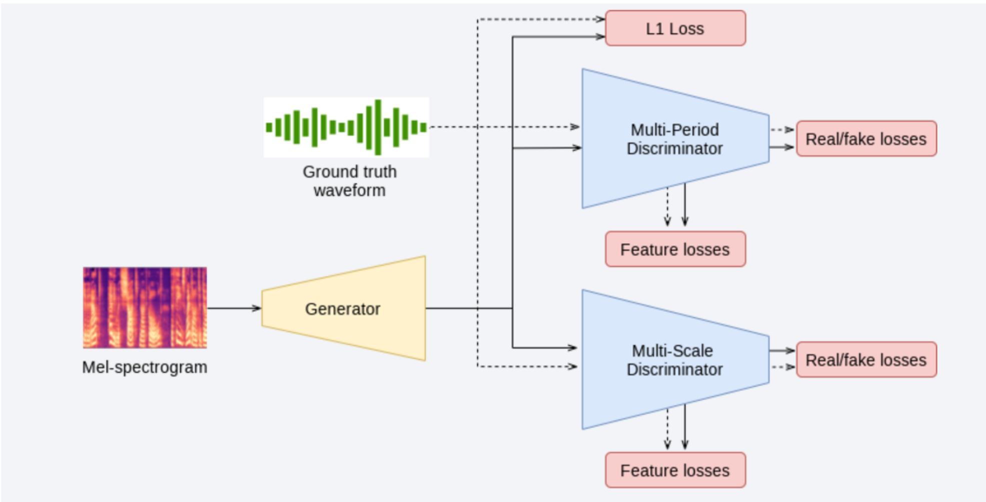

Figure 2.2 Architecture of HiFi-GAN

c) Mel-Spectrograms:

A Mel-spectrogram is a time-frequency representation of sound, commonly used in speech synthesis systems. It captures the intensity of frequencies in a sound signal and converts it into a 2D representation that reflects how human hearing perceives different frequencies (the Mel scale). Tacotron2 generates Mel-spectrograms, which are then converted into speech by HiFi-Gan. [7]

2.2. Literature Review

The research by Khadka et al. presents a method for generating high-quality synthesized Nepali speech using the Tacotron2 model for Mel spectrogram generation. The project involves two main phases: Mel spectrogram generation and vocoder output, with text preprocessing and tokenization as initial steps. The Tacotron2 model is trained on the OpenSLR dataset and fine-tuned using a new dataset created by the authors, which includes approximately 1.2 hours of self-recorded data to enhance prosody. Incremental learning techniques are employed to continuously update the model with new data, improving its adaptability. The generated Mel spectrograms are processed through HiFiGAN and WaveGlow vocoders, with HiFiGAN selected as the preferred option based on qualitative evaluations. The final synthesized speech achieved a Mean Opinion Score (MOS) of 4.03, marking the highest score in Nepali TTS tasks to date. This research addresses limitations of existing TTS systems for under-resourced languages like Nepali, paving the way for improved accessibility and communication technologies. [8].

Recent research in speech synthesis has leveraged Generative Adversarial Networks (GANs) to generate raw waveforms, improving sampling efficiency and memory usage. However, these methods have not yet matched the quality of autoregressive and flow-based models. To address this, Kong et al. introduced HiFi-GAN, a model that achieves both high efficiency and high-fidelity speech synthesis. By focusing on the periodic patterns of speech, HiFi-GAN enhances the sample quality significantly. It generates high-fidelity audio 167.9 times faster than real-time on a single GPU, with subjective evaluations showing human-like quality. The model also generalizes well to Mel spectrogram inversion for unseen speakers and end-to-end speech synthesis. Additionally, a compact version of HiFi-GAN generates audio 13.4 times faster than real-time on a CPU, maintaining comparable quality to autoregressive models. This work sets a new benchmark in both efficiency and fidelity for TTS systems. [9]

Tacotron 2, developed by Shen et al. is a neural network architecture designed for text-to-speech synthesis that directly converts text into speech. The system consists of a recurrent sequence-to-sequence feature prediction network that generates Mel-scale spectrograms, followed by a modified HiFi-GAN vocoder that synthesizes time-domain waveforms from these spectrograms. The model achieves a mean opinion score (MOS) of 4.53, which is comparable to the 4.58 MOS for professionally recorded speech. By employing Mel spectrograms as an intermediate representation, Tacotron 2 simplifies the traditional TTS pipeline and reduces the complexity associated with linguistic and acoustic feature extraction. The architecture includes an encoder that processes character sequences using convolutional layers and a bidirectional LSTM, while the decoder predicts spectrogram frames autoregressively with attention mechanisms. The modified HiFi-GAN vocoder utilizes dilated convolutions to generate high-quality audio from the predicted mel spectrograms. Training involves first optimizing the feature prediction network and then training the HiFi-GAN on its outputs, utilizing maximum likelihood estimation for both components. Overall, Tacotron 2 represents a significant advancement in TTS technology, producing natural-sounding speech that is difficult to distinguish from human voice (Shen et al). [10]

This project aims to enhance text-to-speech (TTS) synthesis by integrating Tacotron2 for generating Mel-spectrograms and HiFi-GAN as the vocoder for high-quality audio output. Tacotron2 utilizes sequence-to-sequence modeling and attention mechanisms to convert input text into smooth, natural-sounding Mel-spectrograms, effectively capturing the nuances of human speech. HiFi-GAN complements this by transforming the Mel-spectrograms into realistic audio waveforms with minimal latency and high fidelity. By combining Tacotron2's robust text-to-speech capabilities with HiFi-GAN's efficiency in waveform generation, this system seeks to produce natural, high-quality speech suitable for applications such as accessibility tools, virtual assistants, and educational platforms.

CHAPTER 3: SYSTEM ANALYSIS

3.1. System Analysis

3.1.1. Requirement Analysis

This section especially focuses on identifying and understanding the specific need to develop our Text-to-speech translator. To clarify the various requirements needed to develop and implement the system, we have categorized them as:

i. **Functional Requirement**

- Admin shall train the TTS model with new datasets to enhance accuracy.
- Admin shall update the system with improved versions or features.
- Admin shall evaluate the model to ensure quality and performance.
- User shall provide text input for conversion into speech.
- User shall trigger the system to convert text into speech.
- User shall play the generated speech audio directly within the interface.
- User shall download or access the final speech audio output.

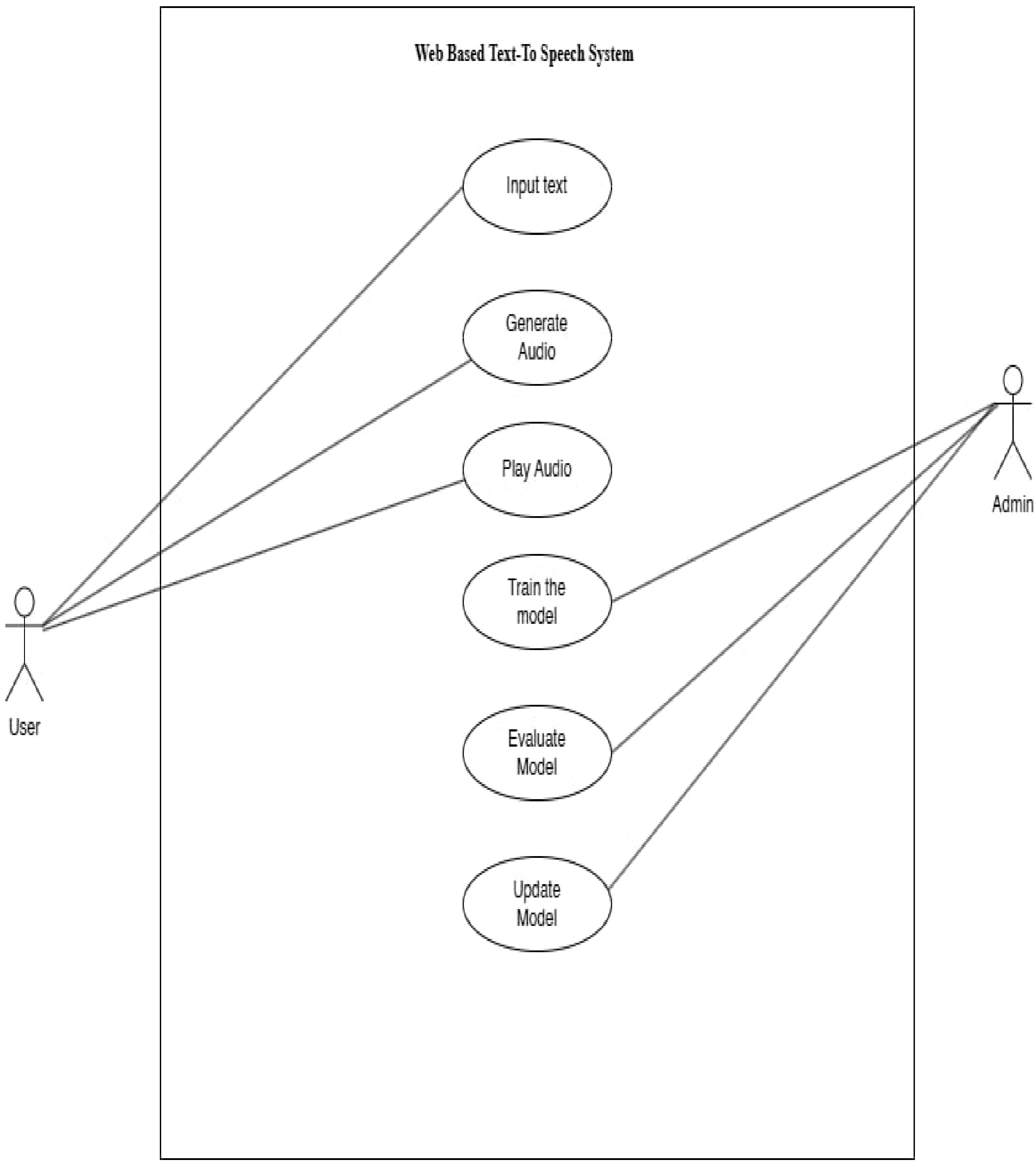

Figure 3.1 System Use Case Diagram of Text to Speech System

ii. Non-Functional Requirement

- The system must be scalable to handle varying loads, from personal use to commercial datasets.
- The system must generate speech with high accuracy, correctly pronouncing complex words.
- The system must reliably produce high-quality speech output with minimal errors and handle diverse inputs.
- The system must be maintainable, allowing for easy updates and improvements without disrupting existing features.
- The system must have a user-friendly interface that is simple, intuitive, and non-technical for easy navigation and usage.
- The system must synthesize speech with minimal latency and use memory efficiently for long sentences.

3.1.2. Feasibility Study

A feasibility Study can be defined as a detailed analysis that considers the aspects such as technical, operational, economic, schedule, etc. in the project to determine the probability of success of developing a Text-to-Speech system using Tacotron2 for Mel-spectrogram generation.

i. Technical Feasibility

The project of text to Speech System is technically feasible as it leverages widely used and well-supported technologies like Python for backend development and HTML, CSS, and JavaScript for the frontend. Using Django for the backend allows for rapid web application development and scalability, while Python in VS code and Google Colab are excellent tools for training and implementing machine learning models. Google Colab is an ideal environment for training complex models like those used in text-to-speech systems due to its powerful GPU support and integration with Python-based machine learning libraries. This combination of technologies enables a smooth development process while providing flexibility and scalability for your system.

ii. Operational Feasibility

This project can always be effectively managed by a small team of three developers working on the system. This project primarily aims to develop a robust and accurate TTS system through implementation of a Tacotron2 model using Seq2Seq encoder-decoder architecture. The operational requirements are manageable, and the team size is big enough for handling model development, training, and testing while focused and efficient development is ensured.

iii. Economic Feasibility

The project is economically feasible since it uses free, open-source software, such as Python. These significantly reduce the development cost. The existing computing resources, including all required hardware, are adequate for the project. There are free or low-cost cloud hosting services that can be used in the development and deployment to keep the cost of this project within budget while still providing great value additions to the TTS system in terms of improved accessibility and automation.

iv. Schedule Feasibility

The development of the project will be completed within the given timeframe. So, it passes the schedule feasibility.

Table 3.1 Project Schedule of Text to Speech System

Name	Start Date	End Date	Duration
Requirement Gathering	Aug 20,2024	Sep 03,2024	11 Days
System Design	Sep 04,2024	Sep 13,2024	8 Days
Data Collection and Preprocessing	Sep 16,2024	Sep 27,2024	10 Days
Model Building	Sep 30,2024	Oct 11,2024	10 Days
Model Training and Testing	Oct 14,2024	Oct 30,2024	13 Days
UI Design	Oct 31,2024	Nov 04,2024	3 Days
System Integration	Nov 05,2024	Nov 25,2024	15 Days
Deployment	Nov 26,2024	Dec 05,2024	8 Days
Project review and closure	Dec 06,2024	Dec 20,2024	11 Days
Documentation	Aug 20,2024	Jan 22,2025	112 Days

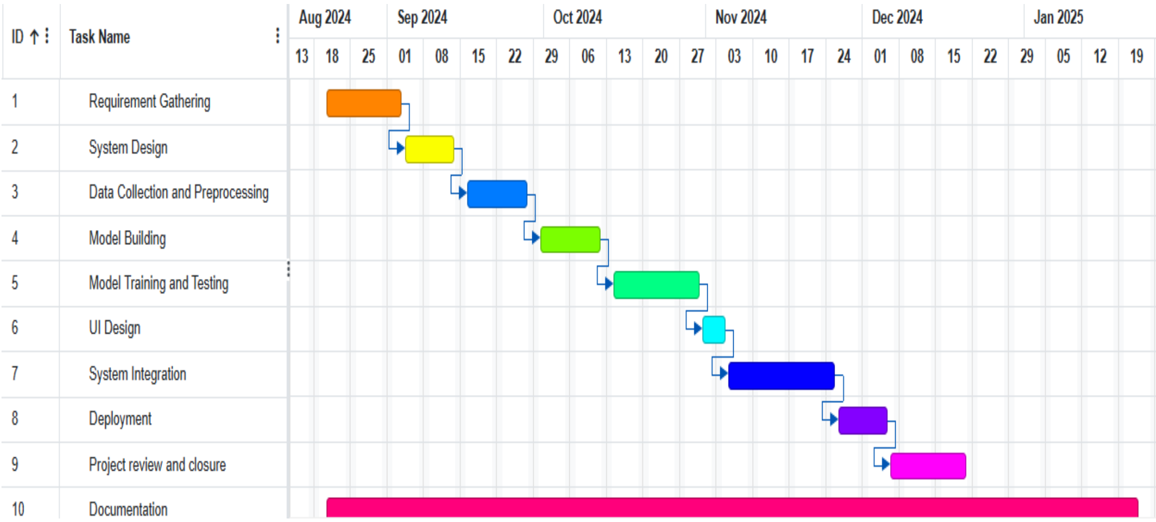

Figure 3.2 Gantt Chart of Text to Speech System

The Gantt chart presents a meticulously planned project timeline, starting in mid-September 2024 with Requirement Gathering, which sets the foundation for the subsequent stages. It transitions into System Design, overlapping with Data Collection and Preprocessing to maximize efficiency. In early October, Model Building begins, followed by the critical phase of Model Training and Testing, extending through November. Concurrently, UI Design is undertaken and completed before System Integration, ensuring a focus on both functionality and user experience. System Integration, starting in late November, leads into Deployment in December, marking the culmination of the technical implementation. The project concludes with a comprehensive Project Review and Closure phase in late December 2024, ensuring a polished and thoroughly evaluated outcome. Throughout the project, Documentation is maintained as a parallel and ongoing task, capturing each stage's processes, outputs, and learnings to ensure seamless knowledge transfer and future reference. The chart's clear dependencies, indicated by arrows, highlights the logical flow of tasks, emphasizing a strategic and professional approach to timely project completion.

3.1.3. Analysis

i. Object modelling using Class and Object Diagrams

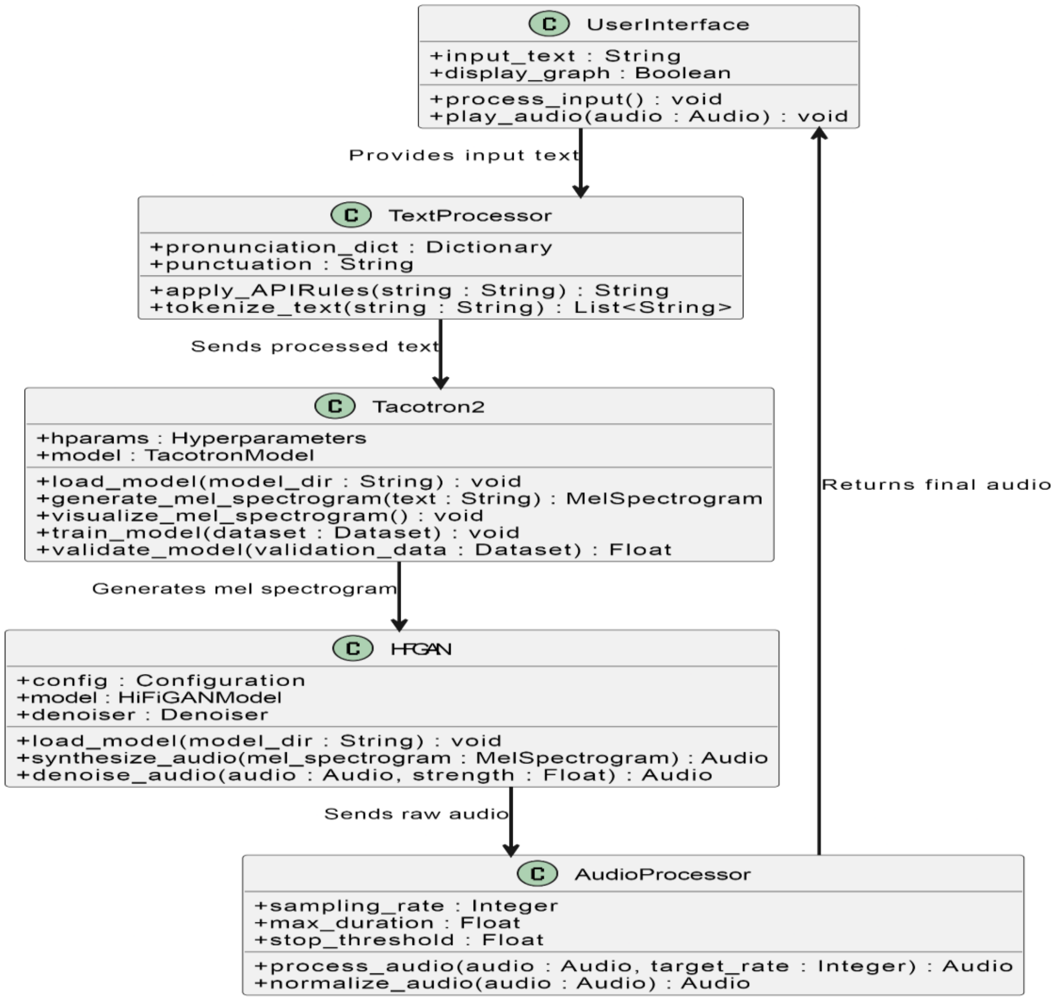

Figure 3.3 Class Diagram of Text to Speech System

This class diagram represents the core components of a Text-to-Speech (TTS) system using Tacotron2 and HiFi-GAN. The **Tacotron2** class generates mel spectrograms from text, while the **HiFi-GAN** class converts these spectrograms into high-quality audio, with denoising features. The **Text Preprocessor** class handles tasks like tokenization and ARPAbet conversion, optimizing the text input. The **Audio Processor** class improves the audio output through normalization, resampling, and filtering. The **User Interface** class allows users to submit input, view real-time graphs, and play audio. These components work together to create a modular and efficient TTS system.

ii. Dynamic modelling using State and Sequence Diagrams

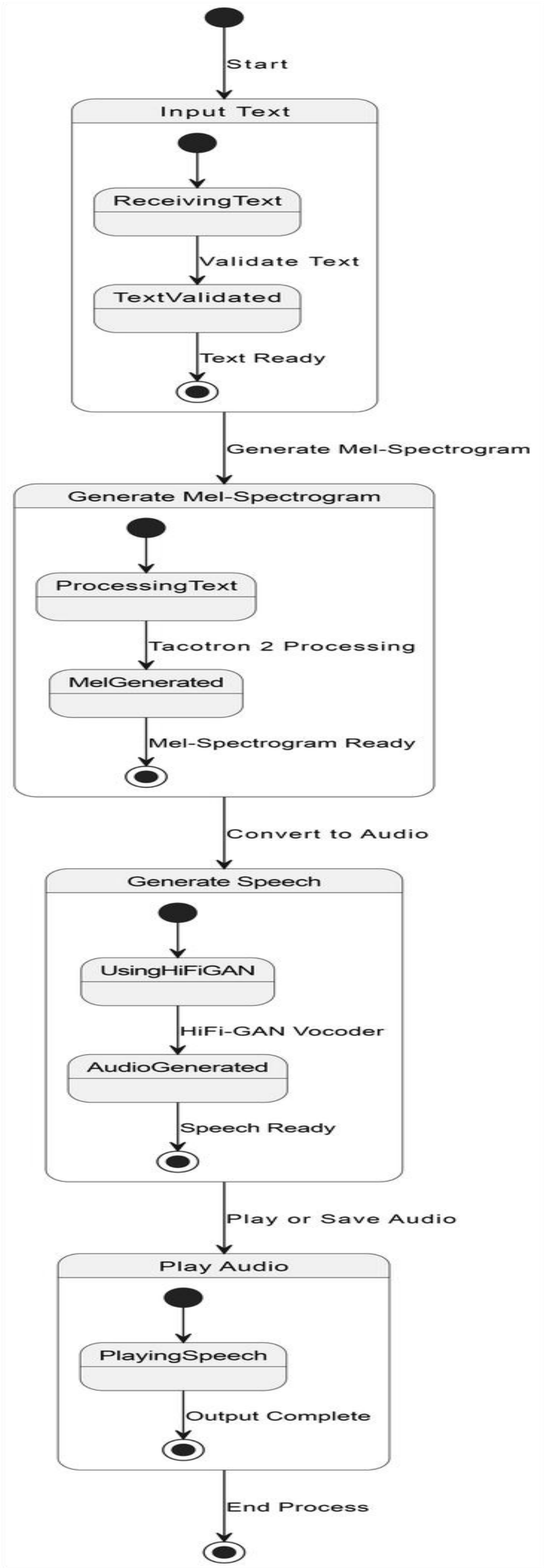

Figure 3.4 State Diagram of Text to Speech System

The state diagram represents the process of transforming input text into synthesized speech using a Text-to-Speech (TTS) system. The workflow begins with Input Text, where the system receives and validates the text for processing. Next, the validated text moves into the Generate Mel-Spectrogram phase, where Tacotron2 processes the text to create a Mel spectrogram, a visual representation of sound frequencies over time. In the Convert to Audio step, the Mel spectrogram is passed through a HiFi-GAN vocoder, converting it into an audio waveform. Finally, the process concludes with Play Audio, where the generated speech is either played back or saved for later use. Each step ensures smooth transitions and validates outputs to produce high-quality, natural-sounding speech.

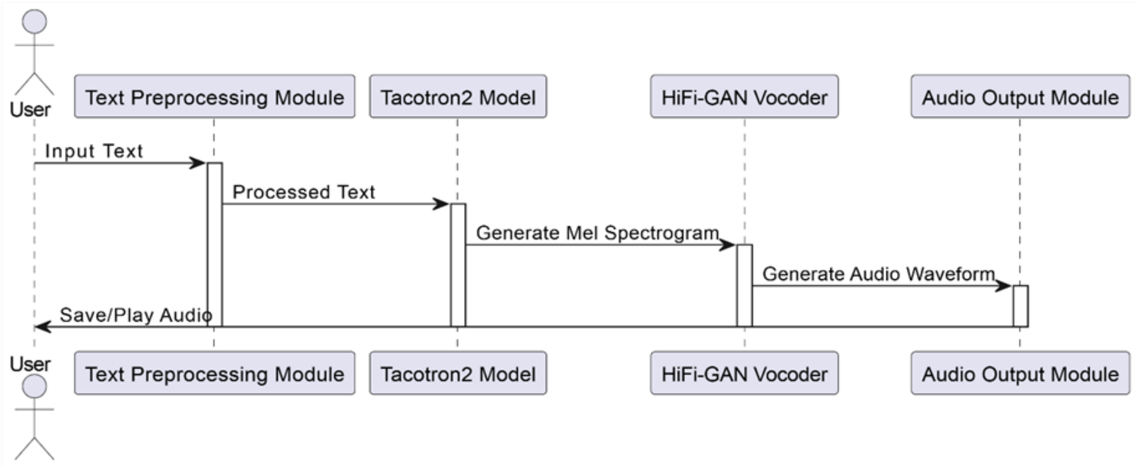

Figure 3.5 Sequence Diagram of Text to Speech System

This sequence diagram illustrates the interaction between components in a Text-to-Speech (TTS) system. The user initiates the process by providing input text, which is passed to the Text Preprocessing Module for validation and formatting. The processed text is then sent to the Tacotron2 Model, where it is transformed into a Mel spectrogram, a key intermediate representation of speech. This spectrogram is forwarded to the HiFi-GAN Vocoder, which converts it into an audio waveform. Finally, the Audio Output Module either plays or saves the generated speech, completing the interaction. The diagram highlights the logical sequence of operations and the collaboration among the system's modules to produce natural-sounding audio.

iii. Process modeling using Activity Diagrams

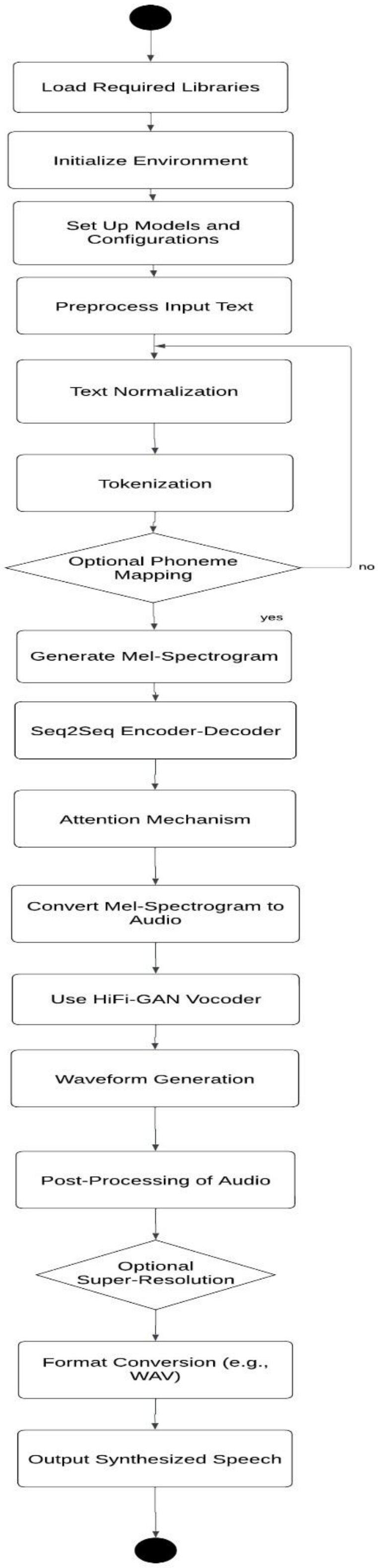

Figure 3.6 Activity diagram of the Text to Speech System

This activity diagram illustrates the process of converting text into synthesized speech using a text-to-speech (TTS) system with Tacotron2 and HiFi-GAN. The process begins by loading the necessary libraries and initializing the environment, The system then downloads the Tacotron2 model for spectrogram generation and the HiFi-GAN vocoder for waveform synthesis. Once the models are set up and configured, the input text undergoes preprocessing, including normalization, tokenization, and optional phoneme mapping. The preprocessed text is then converted into a Mel spectrogram through a sequence-to-sequence (Seq2Seq) encoder-decoder framework and an attention mechanism. The Mel spectrogram is passed to the HiFi-GAN vocoder, which generates the audio waveform. Post-processing steps may include optional super-resolution and format conversion, such as converting the audio to WAV format. Finally, the system outputs synthesized speech, providing high-quality audio based on the input text.

CHAPTER 4: SYSTEM DESIGN AND ALGORITHM DETAILS

4.1. Design

i. Refinement of Class Diagram

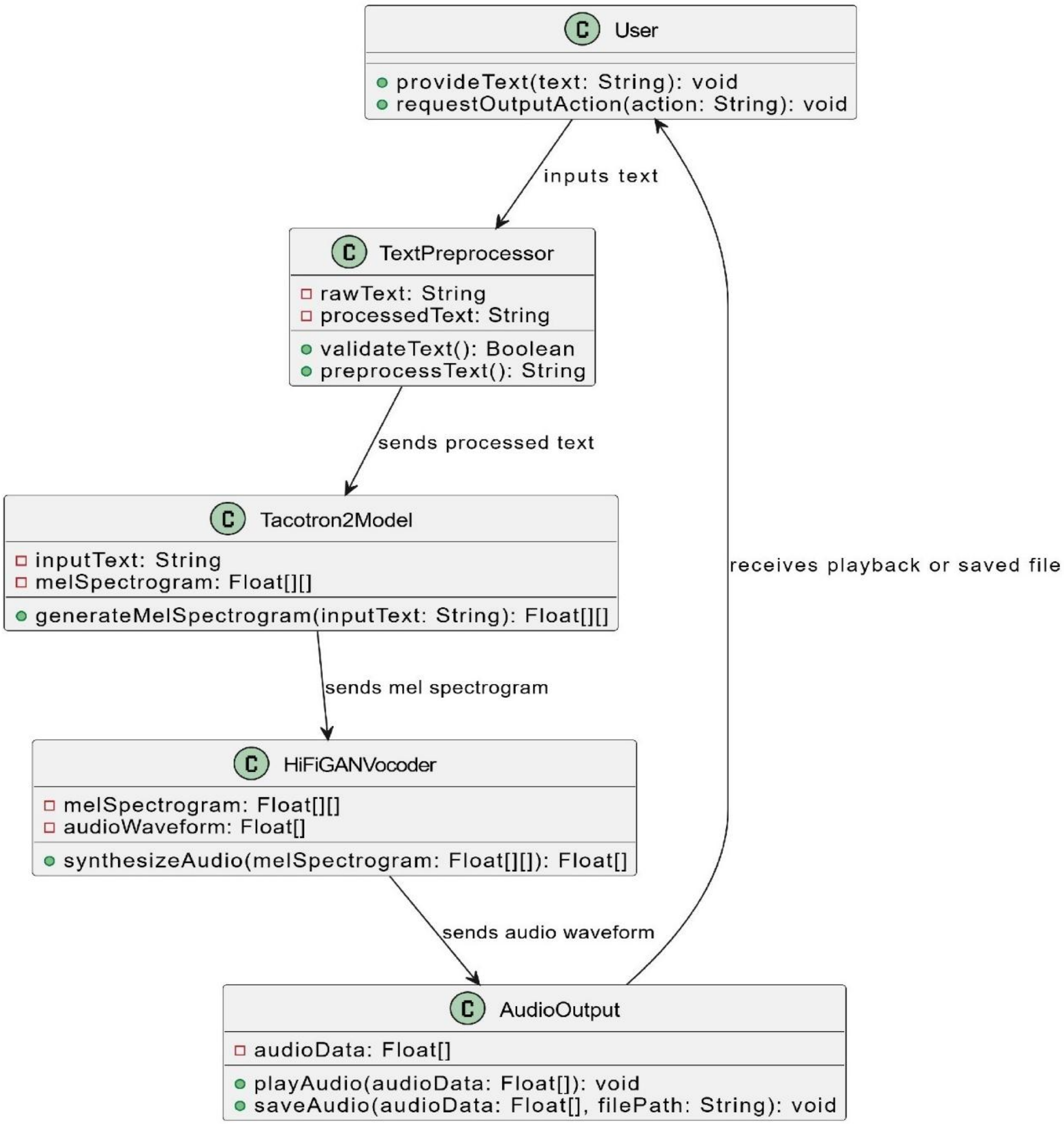

Figure 4.1 Refinement of Class Diagram Text to Speech System

This refined class diagram represents the architecture of a Text-to-Speech (TTS) system. The User interacts with the system by providing input text and choosing to save or play the audio. The Text Preprocessing Module ensures the input is valid and processes it into a suitable format. The Tacotron2Model takes the processed text to generate a Mel spectrogram, a visual representation of audio. This spectrogram is converted into a realistic audio waveform by the HiFiGAN Vocoder. Finally, the Audio Output Module delivers the audio to the user, allowing it to be played or saved as desired. This modular structure enhances maintainability and scalability.

ii. Component Diagram

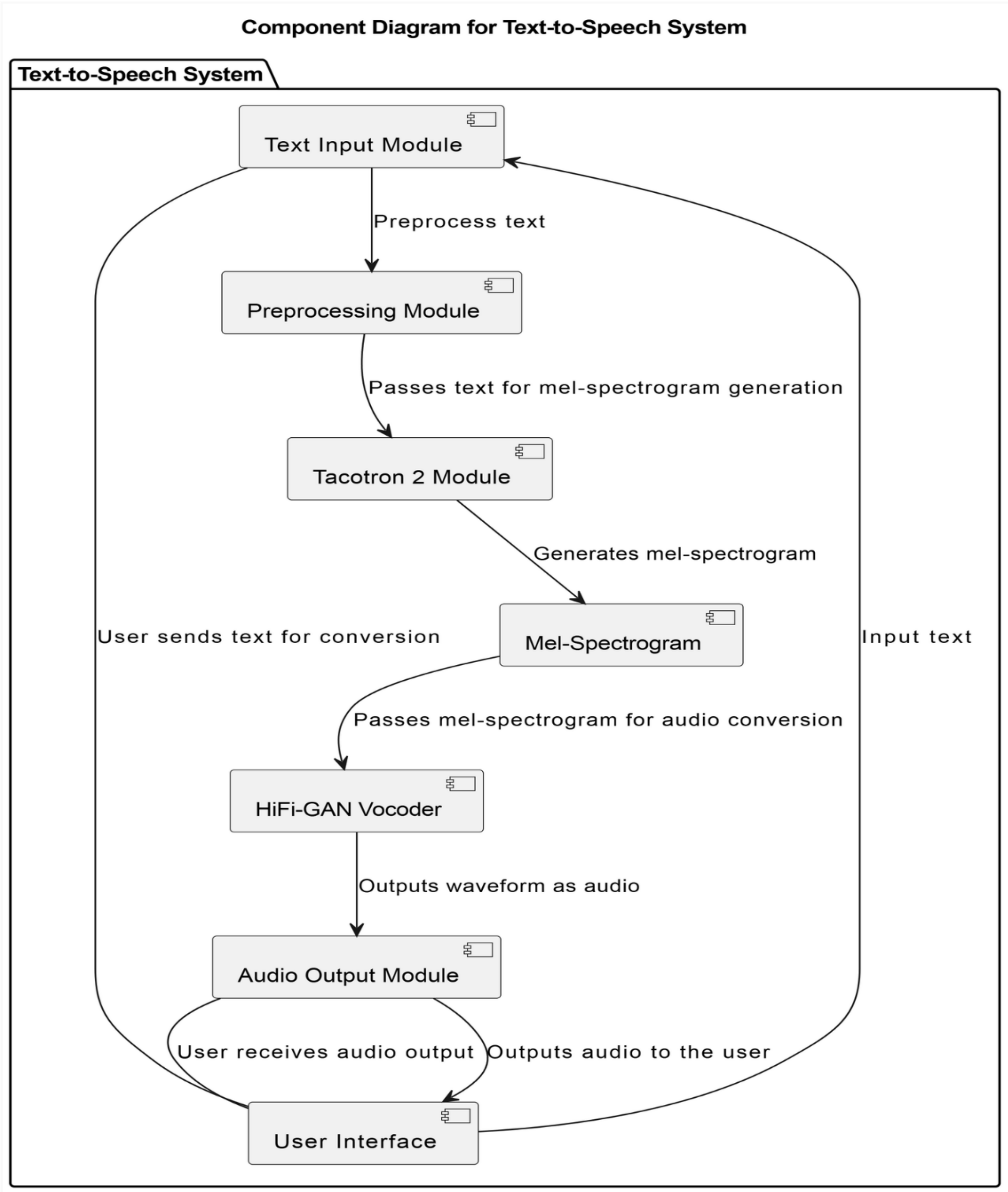

Figure 4.2 Component Diagram of Text to Speech System

The Component Diagram illustrates the internal structure of the text-to-speech system by breaking it into functional modules. It includes a Text Input Module for receiving user input, a Preprocessing Module for normalizing the text, a Tacotron2 Module for generating Mel-spectrograms, and a HiFi-GAN Vocoder for converting spectrograms into audio waveforms. The Audio Output Module delivers synthesized speech back to the user, while the User Interface acts as the primary interaction point for text input and audio playback. This diagram focuses on the logical architecture and the flow of data between these components.

iii. Deployment Diagram

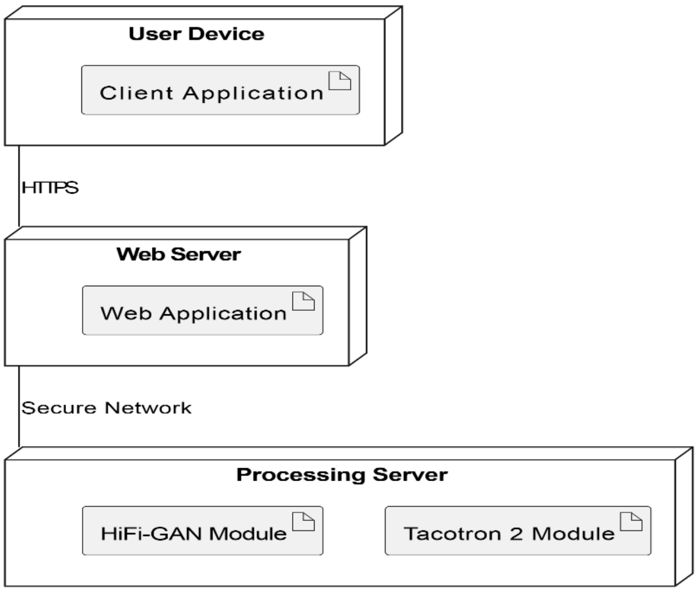

Figure 4.3 Deployment Diagram of Text to Speech System

The Deployment Diagram depicts the physical architecture of the text-to-speech system, showing how its components are deployed across various hardware nodes. The User Device hosts the client application for text input and audio playback, which communicates securely with the Web Server hosting the web application. The Processing Server contains the core components—Tacotron2 for generating Mel-spectrograms and HiFi-GAN for producing audio waveforms. Communication pathways include HTTPS for secure interactions between the user device and the web server and a secure network connection between the web and processing servers. This diagram highlights the system's physical infrastructure and communication flow.

4.2. Algorithm Details

1.Tacotron2: Mel-Spectrogram Generation

Tacotron2 is an end-to-end TTS model that converts text into mel-spectrograms using two main components:

Algorithm: Sequence-to-Sequence with Attention

1. **Encoder:**
   - Converts input text (phonemes or characters) into a series of hidden representations.
   - Uses convolutional layers and bidirectional LSTM layers to capture sequential and contextual information.
2. **Attention Mechanism:**
   - Aligns input text features (encoder outputs) with the decoder steps, enabling the decoder to focus on the most relevant parts of the input at each step.
   - Typically employs location-sensitive attention to ensure smooth alignment between text and audio frames.
3. **Decoder:**
   - Generates Mel-spectrogram frames sequentially using autoregressive LSTM layers.
   - Predicts Mel-spectrograms frame by frame until the end of the sequence is reached.
   - Includes post-processing with a convolutional layer to refine the spectrogram output.

1. **Encoder**:

The encoder extracts local features from the input and processes them with a bidirectional LSTM.

Convolutional Layers:

Each convolutional layer applies a 1D convolution followed by ReLU activation and dropout:

$$\mathbf{X}^{(i+1)} = \mathrm{Dropout}(\mathrm{ReLU}(\mathrm{BatchNorm}(\mathrm{Conv1D}(\mathbf{X}^{(i)}))))$$

where $\mathbf{X}^{(i)}$ is the input at layer i.

Bidirectional LSTM:

A bidirectional LSTM processes the output sequentially:

$$\mathbf{H}_{\text{enc}} = \text{BiLSTM}(\mathbf{X})$$

$$\mathbf{H}_{\text{enc}} = [\overrightarrow{\mathbf{H}}_{\text{enc}}; \overleftarrow{\mathbf{H}}_{\text{enc}}]$$

where $\overrightarrow{\mathbf{H}}_{\text{enc}}$ and $\overleftarrow{\mathbf{H}}_{\text{enc}}$ are the forward and backward LSTM outputs.

2. **Decoder**:

The decoder predicts mel-spectrogram frames and includes attention mechanisms.

Prenet:

The prenet processes the input mel-spectrogram frames:

$$\mathbf{X}_{\text{pre}}^{(i+1)} = \text{Dropout}(\text{ReLU}(\mathbf{W}^{(i)}\mathbf{X}_{\text{pre}}^{(i)}))$$

where $\mathbf{W}^{(i)}$ is the weight matrix for layer i.

Attention RNN:

$$\mathbf{h}_a, \mathbf{c}_a = \text{LSTMCell}([\mathbf{X}_{\text{pre}}, \mathbf{C}_a], (\mathbf{h}_a, \mathbf{c}_a))$$

Attention Mechanism:

The alignment energies are computed as:

$$e_t = \mathbf{v}^T \tanh(\mathbf{W}_q \mathbf{h}_a + \mathbf{W}_k \mathbf{M} + \mathbf{W}_f \mathbf{F})$$

where $\mathbf{M}$ is the memory, $\mathbf{F}$ is the location-based features, and $\mathbf{v}$ is a learnable vector.

Softmax computes the attention weights:

$$\alpha_t = \mathrm{Softmax}(e_t)$$

The context vector $\mathbf{C}_a$ is:

$$\mathbf{C}_a = \sum_t \alpha_t \mathbf{M}_t$$

Decoder RNN:

The decoder uses LSTM cells:

$$\mathbf{h}_d, \mathbf{c}_d = \mathrm{LSTMCell}([\mathbf{h}_a, \mathbf{C}_a], (\mathbf{h}_d, \mathbf{c}_d))$$

Output Projection:

The mel output is projected:

$$\mathbf{Y}_{\text{mel}} = \mathbf{W}_{\text{proj}} [\mathbf{h}_d, \mathbf{C}_a]$$

The gate output:

$$\mathbf{g} = \sigma(\mathbf{W}_g [\mathbf{h}_d, \mathbf{C}_a])$$

3. Postnet:

Applies a series of convolutions to refine the mel-spectrogram prediction.

$$\mathbf{Y}_{\text{post}} = \mathbf{Y}_{\text{mel}} + \sum_i \mathrm{Conv1D}(\mathbf{Y}^{(i)})$$

2. HiFi-GAN: Vocoder

HiFi-GAN is a generative adversarial network (GAN) used to convert the mel-spectrograms into waveform audio. It focuses on producing high-quality, natural-sounding audio efficiently.

Algorithm: Generative Adversarial Network (GAN)

1. **Generator:**
   - Converts mel-spectrograms into audio waveforms.
   - Uses multi-receptive field fusion (MRF) modules with multiple convolutional kernels of different sizes to capture fine-grained details in the audio.
2. **Discriminator:**
   - Consists of multiple sub-discriminators operating at different scales (e.g., raw audio, shorter segments, or down sampled versions).
   - Evaluates the realism of the generated audio by distinguishing between real and fake samples.
3. **Adversarial Loss:**
   - Combines standard GAN loss with additional feature matching and mel-spectrogram loss to ensure high-quality synthesis.

1. Generator

Input Processing:

- **Input (Mel-spectrogram):** $x \in \mathbb{R}^{80 \times T}$
- Initial convolution:

$$x_1 = \mathrm{LeakyReLU}(\mathrm{Conv1d}(x))$$

Upsampling and ResBlock:

- For each upsampling layer i:

$$x_{i+1} = \mathrm{LeakyReLU}(\mathrm{ConvTranspose1d}(x_i))$$

  - Pass through $\mathrm{ResBlocks}$:

$$x_{i+1} = \frac{1}{K} \sum_{j=1}^{K} \mathrm{ResBlock}(x_i)$$

  where K is the number of kernels.

Final Layers:

- Final convolution and Tanh activation:

$$y = \mathrm{Tanh}(\mathrm{Conv1d}(x_L))$$

2. ResBlock

Each ResBlock performs:

- Dilated convolutions with residual connections:

$$x' = \mathrm{LeakyReLU}(x)$$

$$x'' = \mathrm{LeakyReLU}(\mathrm{Conv1d}(\mathrm{LeakyReLU}(\mathrm{Conv1d}(x'))))$$

  - Residual addition:

$$x_{\text{out}} = x + x''$$

3. Multi-Period Discriminator (MPD)

The MPD processes periodic reshaped inputs x for each period p:

- Reshaped to $(B, C, T/p, p)$
- Convolutions:

$$x' = \mathrm{Conv2d}(\mathrm{LeakyReLU}(\mathrm{Conv2d}(\ldots(x)\ldots)))$$

  - Final flattening:

$$y_d = \mathrm{Flatten}(\mathrm{Conv2d}(x'))$$

4. Multi-Scale Discriminator (MSD)

The MSD processes the input x at multiple scales:

- Downsample with average pooling:

$$x_{i+1} = \mathrm{AvgPool1d}(x_i)$$

- Apply convolutional discriminator as in MPD.

5. Loss Functions

- **Generator Loss:**

$$\mathcal{L}_G = \sum_i \mathbb{E}[(1 - D(G(x)))^2]$$

- **Discriminator Loss:**

$$\mathcal{L}_D = \sum_i \mathbb{E}[(1 - D(y))^2 + D(G(x))^2]$$

- **Feature Loss:**

$$\mathcal{L}_{\text{feat}} = \sum_{i,j} \mathbb{E}[\|D_i^j(y) - D_i^j(G(x))\|_1]$$

Summary of Algorithms:

- **Tacotron2:** Sequence-to-sequence learning with attention for mel-spectrogram generation.
- **HiFi-GAN:** GAN-based approach with multi-receptive field convolution for waveform generation.

This combination ensures that the TTS system can produce highly intelligible and natural-sounding speech with efficient audio synthesis.

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1. Implementation

5.1.1. Tools Used

- Programming Languages: Python and Django were the primary programming languages utilized for the development of the Text to Speech Systems.
- Front-end Development: HTML, CSS, JavaScript were used for designing and structuring the user interface of the application, providing a visually appealing and user-friendly front end.
- Code Editor: Visual Studio Code (VS code) and Google Colab served as the integrated development environment (IDE) for coding, debugging, and collaborative development.
- Diagram Creation: Draw.io was used for creating visual diagrams, and documentation of the project's components and structure. Also, Figma was used to design the user interface.

5.1.2. Implementation Details of Modules

The Text-to-Speech (TTS) system integrates Tacotron2 for text-to-spectrogram conversion and HiFi-GAN for generating high-quality speech from spectrograms.

Tacotron2 Module

Purpose: Converts input text into a mel spectrogram using a sequence-to-sequence model.

Key Components:

- Model Loading: Loads pre-trained Tacotron2 weights.

- Mel Spectrogram Generation: Converts text into a mel spectrogram through a series of RNN layers.

```
class Tacotron2:
 def __init__(self):
   self.model = Tacotron2Model()

self.model.load_state_dict(torch.load(tacotron2_model_pa
th))
        def generate_mel(self, text: str):
            sequence = text_to_sequence(text)
            mel = self.model(sequence)
            return mel
```

HiFi-GAN Module

Purpose: Converts the mel spectrogram into a high-quality audio waveform.

Key Components:

- Model Loading: Loads pre-trained HiFi-GAN weights.

- Audio Generation: Uses a generator to convert mel spectrograms into audio.

```
class HiFiGAN:
        def __init__(self):
            self.model = HiFiGANModel()

self.model.load_state_dict(torch.load(hifigan_model_path
))

        def generate_audio(self, mel_spectrogram):
            audio = self.model(mel_spectrogram)
            return audio
```

TextToSpeech Integration Module

Purpose: Integrates Tacotron2 and HiFi-GAN to generate speech from text.

Key Components:

- Model Integration: First generates mel spectrogram using Tacotron2, then converts it to audio using HiFi-GAN.

Code:

```
class TextToSpeech:
        def __init__(self, tacotron, hifigan):
            self.tacotron = tacotron
            self.hifigan = hifigan

        def process_text(self, text):
            mel = self.tacotron.generate_mel(text)
            audio = self.hifigan.generate_audio(mel)
            return audio
```

HiFiGANConfig Module

Purpose: Manages HiFi-GAN configuration settings.

Code:

```
class HiFiGANConfig:
        def __init__(self, config_file):
            with open(config_file) as f:
                self.config = json.load(f)

        def get_config(self):
            return self.config
```

Audio Playback

Purpose: Handles the playback of the generated audio.

Code:

```
class Audio:
        def __init__(self, waveform):
            self.waveform = waveform

        def play(self):
            import IPython.display as ipd
            ipd.display(ipd.Audio(self.waveform,
rate=22050))
```

5.2. Testing

Testing can be defined as the process of checking or verifying if the system is ready to perform the task without any flaws and errors. The testing can be manual or via automation tools to verify each component of a system is working. After the project is ready its various components are tested in terms of quality and performance to make it error free and remove any sort of possible flaws or problems. Testing is needed on the development cycle of the system to ensure that the system's every component works fine.

5.2.1. Test Cases for Unit Testing

Table 5.1 Table of Unit Testing of Text to Speech System

Test ID	Component	Test Description	Test Data	Expected Output	Actual Output	Test Result
1.	Audio resampling	Validate Sampling Rate of the data	Wav file	All wav files resampled to 22050Hz	Expected output is matched	Pass
2.	Silence Trimming	Validate if the silence removed from audio	Wav file	Silence from beginning and end should be removed	Expected output is matched	Pass
3.	Audio Normalization	Validate if audio is normalized	Wav file	Audio should have same loudness through the file	Expected output is matched	Pass
4.	Text Preprocessing Module	Validate tokenization of input text using English cleaners	"Hello, world!"	["Hello", ",", "world", "!"]	["Hello", ",", "world", "!"]	Pass

5.	Text Normalization	Ensure special characters are handled properly	"Dr. Smith's $100 bill"	"Doctor Smith's one-hundred-dollar bill"	"Doctor Smith's one-hundred-dollar bill"	Pass
6.	Tacotron2 Model	Verify Mel-spectrogram generation for input text	"Good morning !"	A valid Mel-spectrogram array is generated	Error	Fail
7.	Tacotron2 Model	Verify Mel-spectrogram generation	"Good morning !"	A valid Mel-spectrogram array is generated	A valid Mel-spectrogram array is generated	Pass
8.	HiFi-GAN Vocoder	Convert Mel-spectrogram to audio waveform	Valid Mel-spectrogram input	High-quality audio waveform is produced	High-quality audio waveform is produced	Pass
9.	Frontend Integration	call for TTS conversion	Text input via web interface	API response with audio file	Previous audio played	Fail
10.	Frontend Integration	call for TTS conversion after updating code	Text input via web interface	API response with audio file	API response with audio file as of text	Pass
11.	Latency Calculation	Measure time taken for Mel-spectrogram generation	"Hello"	Mel-spectrogram generated in < 2 seconds	Mel-spectrogram generated	Pass

12.	Error Handling	Test handling of empty input	“ ”	Text input cannot be empty!		Pass
13.	Audio Playback Component	Validate playback of generated audio	Generated audio file	Audio is played without distortion	Audio is played without distortion	Pass

5.2.2. Test Cases for System Testing

Table 5.2 Table for System Testing of Text to Speech System

Test ID	Test Description	Test Data	Expected Output	Test Result
1.	Verify full text-to-speech workflow	"Good afternoon, everyone!"	High-quality audio file generated and played	Pass
2.	Test multiple concurrent requests	50 simultaneous text inputs	All requests processed without crash or significant delay	Fail
3.	Test multiple concurrent requests	10 simultaneous text inputs	All requests processed without crash or significant delay	Pass
4.	Validate UI and backend communication	Submit text via web interface	API request sent, audio file returned, and played through the web interface	Pass
5.	Test long text input	200-word paragraph	Complete audio generated without truncation or performance degradation	Pass
6.	Validate TTS output for punctuation handling	"Hello, how are you?"	Correct pauses and intonation in the synthesized speech	Pass

7.	Test UI responsiveness across devices	Test on desktop, tablet, and mobile	UI adapts correctly, functionality remains intact	Fail
8.	Error Handling Test	Enter unsupported text (e.g., emojis)	System returns an error message without crashing	Fail
9.	Validate latency of the entire system	"Hello, world!"	Audio output generated and played within 3 seconds	Pass
10.	Test audio quality for non-standard words	"Pneumonoultramicroscopicsilicovolcanoconiosis"	Accurate pronunciation with natural intonation	Pass

5.3. Result Analysis

The result analysis evaluates the outcomes of unit and system tests to ensure the developed text-to-speech (TTS) system meets functional and non-functional requirements. Additionally, it introduces the Mean Opinion Score (MOS) and includes visual results such as Mel-spectrograms and Tensor Board metrics to provide a comprehensive evaluation.

5.3.1. Mean Opinion Score (MOS) Analysis

The MOS score, a widely used metric for evaluating the naturalness and intelligibility of synthesized speech, was calculated through subjective evaluations. The developed TTS system achieved an **Overall MOS of 4.48 ± 0.89**, indicating high-quality, human-like speech output. This score demonstrates that the system generates natural prosody, accurate pronunciations, and smooth transitions between phonemes, with room for further optimization to improve consistency.

5.3.2. Unit Testing Analysis

The unit testing phase consisted of 13 test cases targeting individual modules and components:

- **Pass Rate**: 11 out of 13 test cases passed, yielding an 85% success rate.
- **Failures and Actions**:
  - Tacotron2 Mel-Spectrogram Generation: Initially failed due to a coding error but resolved after debugging.
  - Frontend Integration: Failed to return the correct audio file, fixed through code modifications.
- **Key Observations**:
  - Core functionalities such as resampling, silence trimming, normalization, and tokenization performed as expected.
  - Robust handling of edge cases, including empty text inputs and special punctuation.

5.3.3. System Testing Analysis

System testing assessed the complete workflow, including integration with external components:

- **Pass Rate**: 7 out of 10 test cases passed, yielding a 70% success rate.
- **Failures and Actions**:
  - Concurrent Requests: Handling 50 simultaneous requests failed due to resource limitations; however, handling 10 concurrent inputs succeeded, demonstrating moderate scalability.
  - UI Responsiveness: Failed to adapt on tablets and mobile devices, highlighting the need for improved responsive design.
  - Unsupported Text Input: Crashed when processing emojis, requiring enhanced input validation.

- **Key Observations**:
    - High-quality audio generation for long and complex text inputs.
    - Audio generation latency remained under 3 seconds, meeting efficiency expectations.

5.3.4. Visual Analysis

Mel-Spectrogram Figures

The generated Mel-spectrograms demonstrate the system's ability to convert text into time-frequency representations accurately. These spectrograms visually depict how Tacotron2 captures the acoustic features of speech, such as pitch, duration, and intensity. A smooth transition between frames and consistent alignment with text input validates the effectiveness of the encoder-decoder attention mechanism.

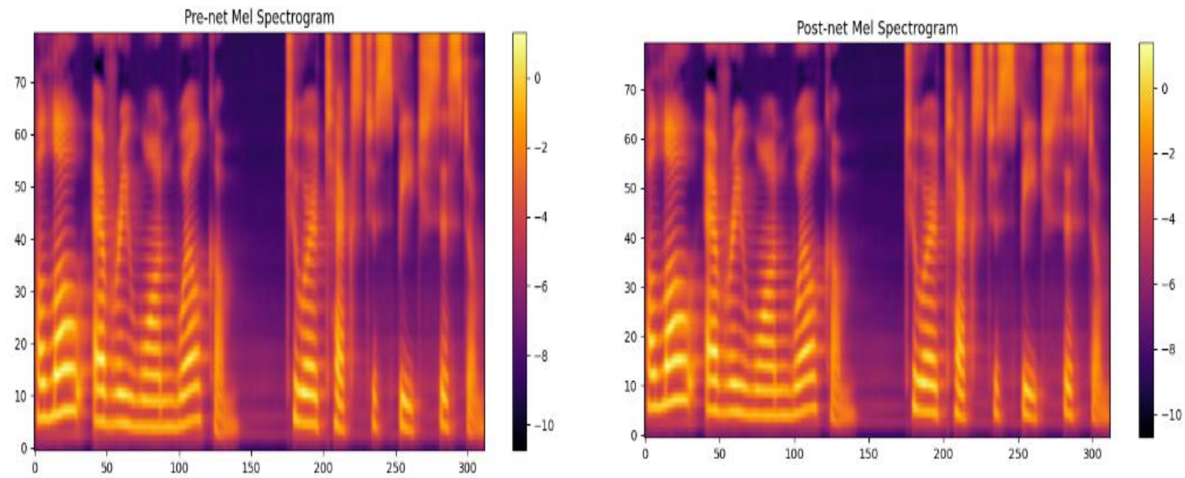

Figure 5.1 Generation of Mel spectrogram Text to Speech System

- **Pre-net Mel Spectrogram:** This figure showcases the raw Mel spectrogram output generated before post-processing by the post-net. While it provides an accurate representation of the acoustic features, some noise or incomplete frequency components may be present.
- **Post-net Mel Spectrogram:** After applying the post-net, the spectrogram exhibits refined acoustic features with reduced noise and enhanced continuity. This demonstrates the system's ability to synthesize natural and smooth transitions between speech segments.

Tensor Board Loss Curves

The **loss curves** from Tensor Board illustrate the convergence of the model during training. The gradual decline in training and validation loss indicates successful learning, while the gap between them highlights generalization. The stability in later iterations suggests that the model effectively balances underfitting and overfitting.

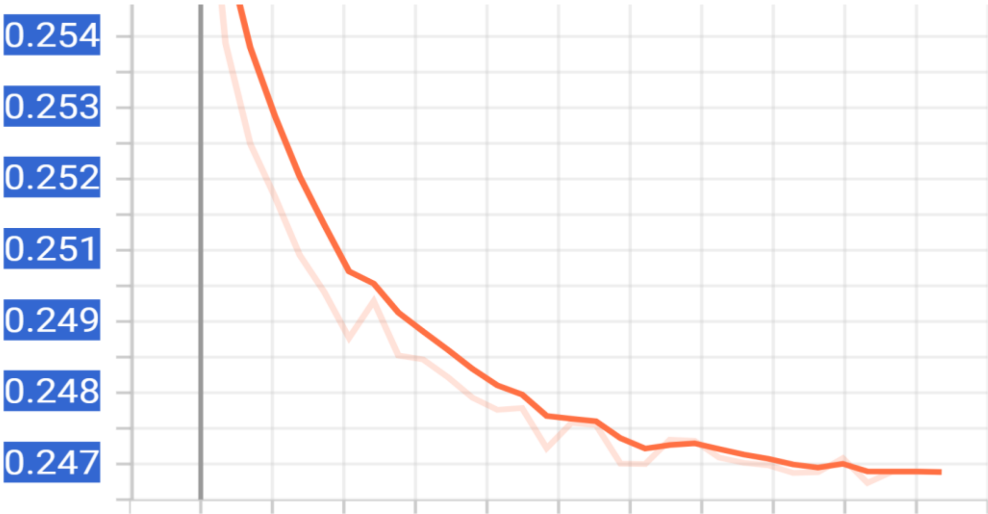

Figure 5.2 Validation Loss of Text to Speech System

This plot shows the validation loss decreasing steadily, indicating the model's ability to generalize well across unseen data.

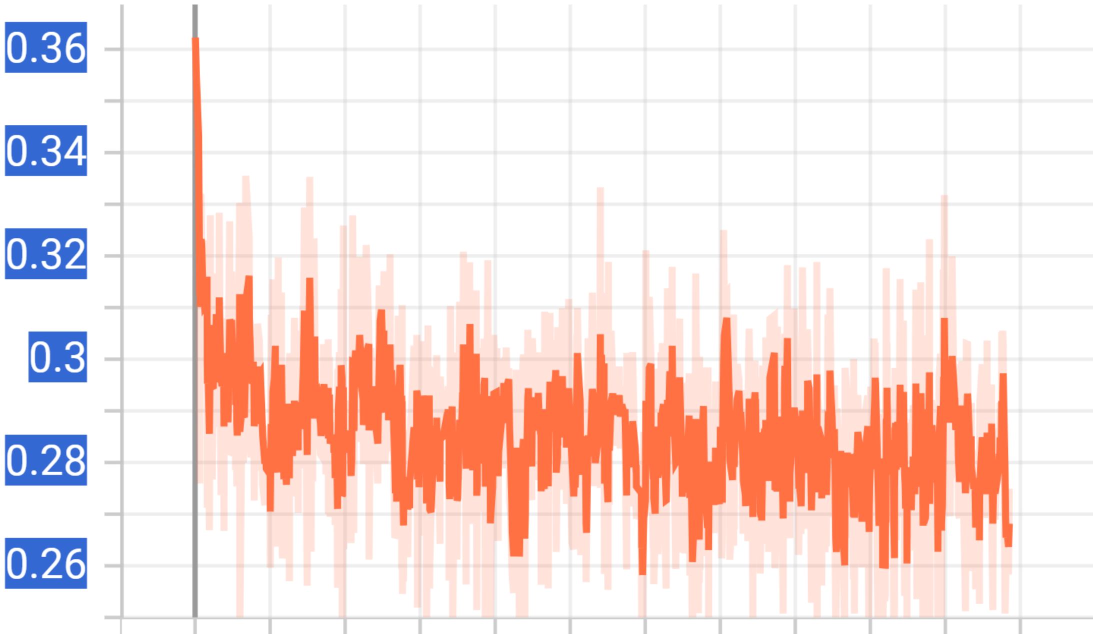

Figure 5.3 Training Loss of Text to Speech System

This plot depicts the decline in training loss, signifying the model's consistent learning from the provided data. The fluctuations reflect ongoing updates to the model parameters, while the overall downward trend confirms successful optimization.

5.3.5. Strengths

- High MOS score (4.48 ± 0.89) reflecting natural intonation and accurate pronunciation.
- Strong performance in core functionality, such as audio preprocessing and text-to-speech conversion.
- Scalable to handle moderate traffic loads effectively.

5.3.6. Weaknesses

- Limited concurrency handling under high loads.
- Insufficient input validation for unsupported characters like emojis.
- Poor UI adaptability on smaller screens, requiring further enhancements in responsive

CHAPTER 6: CONCLUSION AND FUTURE RECOMMENDATIONS

6.1. Conclusion

The Text-to-Speech (TTS) system developed using Tacotron2 for mel-spectrogram generation and HiFi-GAN as the vocoder effectively produces high-quality, human-like speech waveforms. Tacotron2 employs a sequence-to-sequence learning algorithm with attention, transforming text into mel-spectrograms through an encoder, attention mechanism, and autoregressive decoder. HiFi-GAN, a GAN-based vocoder, converts these spectrograms into natural-sounding audio using multi-receptive field fusion and adversarial loss techniques. Achieving an impressive Mean Opinion Score (MOS) of 4.48 ± 0.89, the system delivers clear, natural speech. While it meets key performance expectations, addressing scalability and UI responsiveness will enhance its reliability and user experience, ensuring a more robust and efficient TTS solution.

6.2. Future Recommendations

To enhance scalability, performance, and user satisfaction, key recommendations include optimizing backend processes with efficient queuing systems and leveraging cloud platforms like AWS or Google Cloud for dynamic resource scaling. Improving UI responsiveness through modern CSS frameworks, responsive design, and engaging user feedback elements will ensure a seamless experience across devices. Strengthening input validation by handling unsupported characters, performing language checks, and providing clear error messages will improve system reliability.

Additionally, implementing Continuous Integration (CI) pipelines with automated testing for core components and edge cases will ensure system stability. Accessibility features like adjustable speech speed and pitch, along with robust data privacy measures, will ensure inclusivity and security. Offering voice customization options will further personalize the user experience, ensuring the system remains efficient, scalable, and user-friendly.

REFERENCES

[1] J. S. a. J. Z. Y. Li, "Advances in Text-to-Speech Synthesis: A Review,," 2019.

[2] A. K. Singh, "Text-to-Speech Technology for Accessibility and Inclusion," *Journal of Assistive Technologies,* 2022.

[3] Y. Diena, "73 Dyslexia Statistics and Facts: How Many People Have Dyslexia?," 2023.

[4] U. I. f. Statistics, "Literacy," 2024.

[5] PyTorch, "The Tacotron 2 model for generating mel spectrograms from text," 2017.

[6] S. Y. J. B. J. L. a. J. K. J. Kim, "HiFi-GAN: Generative adversarial networks for efficient and high-quality speech synthesis,," 2020.

[7] S. D. Aäron van den Oord, "WaveNet: A generative model for raw audio," 2016.

[8] S. G. R. P. P. S. R. J. B. Khadka, "Nepali Text-to-Speech Synthesis using Tacotron2 for Melspectrogram Generation," 2023.

[9] J. K. J. B. e. a. J. Kong, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," 2020.

[10] J. Shen1, "NATURAL TTS SYNTHESIS BY CONDITIONING WAVENET ON MEL SPECTROGRAM," 2018.

APPENDICES

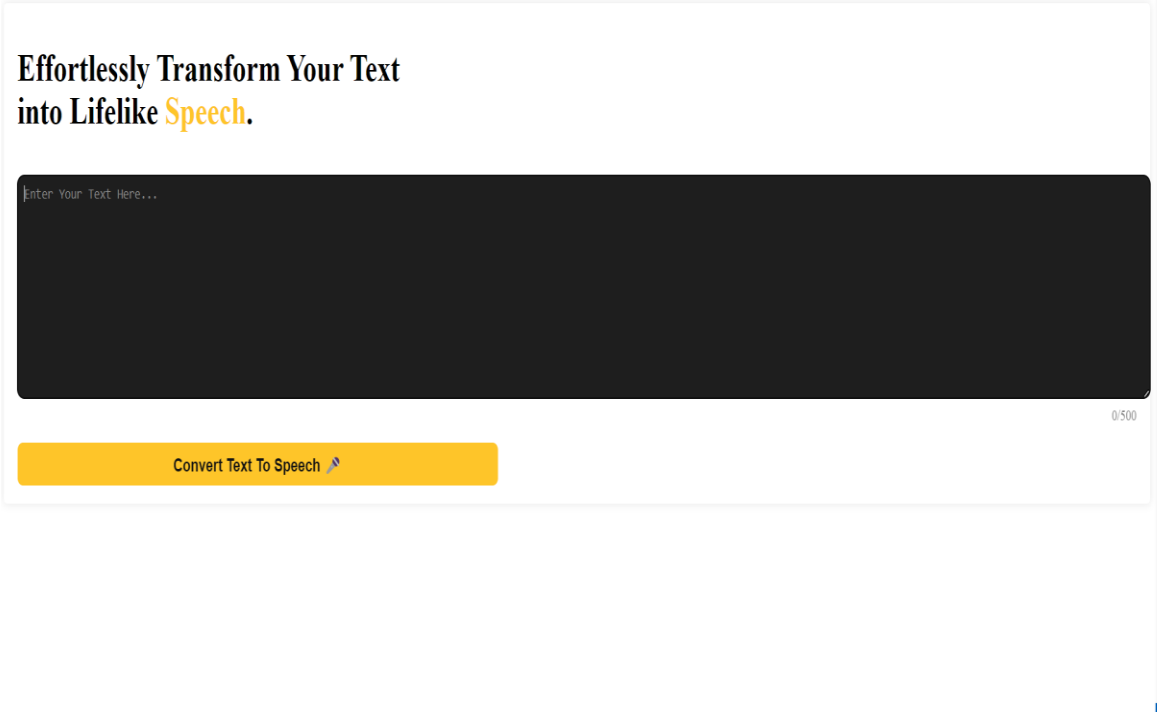

Figure 1 Front Page of the Text to Speech System

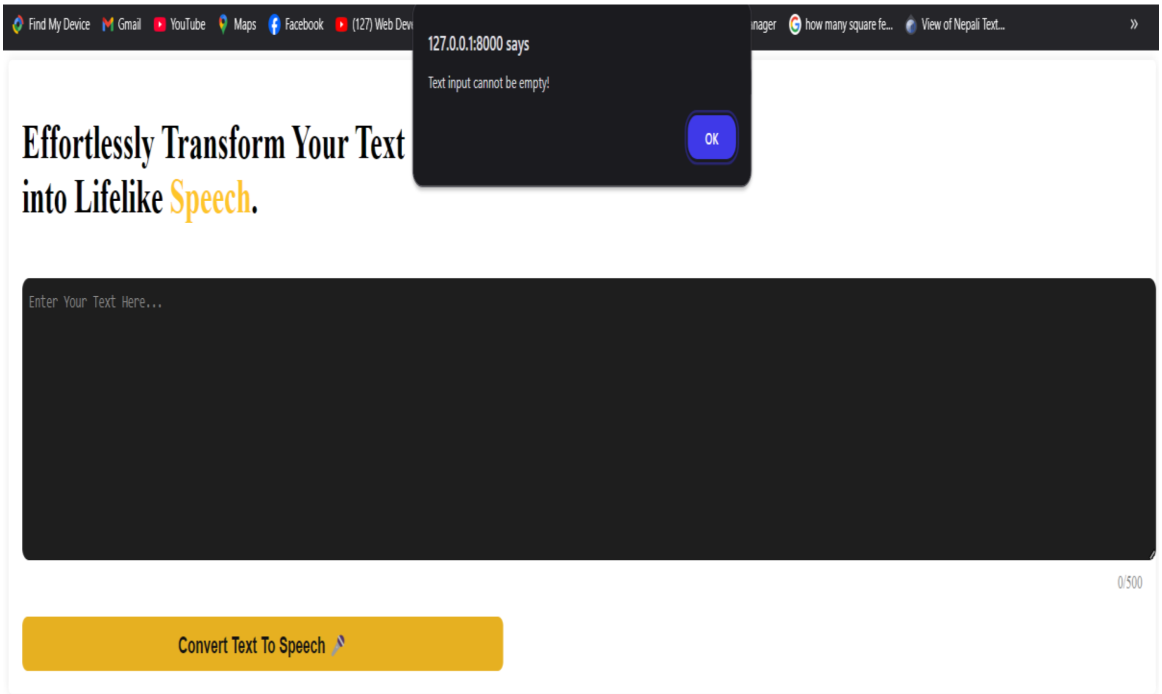

Figure 2 Error Popup When Empty Text Given

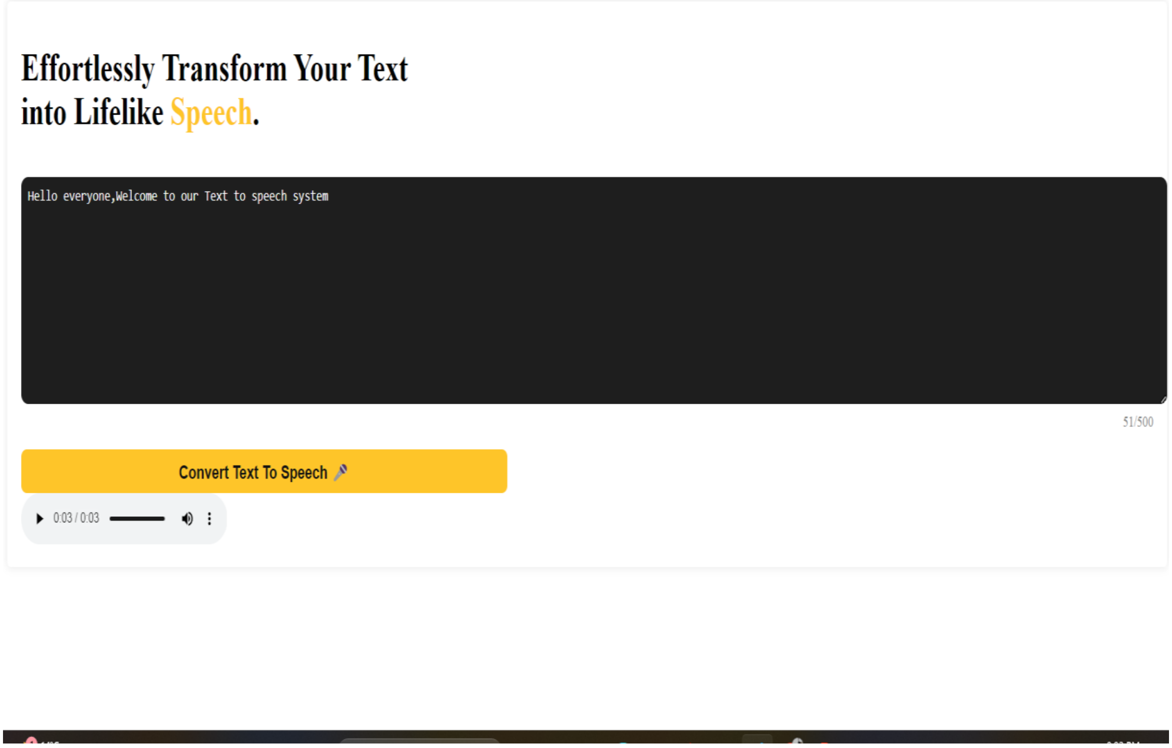

Figure 3 Option for Audio Playback After Generation of Audio

```
model_filename = 'Text_To_Speech'
Training_file = "/content/TTS-TT2/filelists/train.txt"
Validation_file= "/content/TTS-TT2/filelists/val.txt"
hparams.training_files = Training_file
hparams.validation_files = Validation_file
hparams.p_attention_dropout=0.1
hparams.p_decoder_dropout=0.1
# Learning Rate
hparams.decay_start = 15000         # wait till decay_start to start decaying learning rate
#### Lower learning rates will take more time but will lead to more accurate results:
# Start/Max Learning Rate
hparams.A_ = 3e-6
hparams.B_ = 8000                   # Decay Rate
hparams.C_ = 0                      # Shift learning rate equation by this value kasari effect garxa
hparams.min_learning_rate = 1e-5    # Min Learning Rate
# Quality of Life
generate_mels = True
hparams.show_alignments = True
alignment_graph_height = 600
alignment_graph_width = 1000
hparams.batch_size =  30
hparams.load_mel_from_disk = True
hparams.ignore_layers = []
#use cmudict dictionary to map text to phonemes
use_cmudict = True
#### Amount of epochs before sto (variable) hparams: Any h amount to not stop.
hparams.epochs =  150
torch.backends.cudnn.enabled = hparams.cudnn_enabled#speecd up training
torch.backends.cudnn.benchmark = hparams.cudnn_benchmark#select best algo for gpu
#### Where to save your model when training:
output_directory = '/content/drive/MyDrive/colab/outdir'
log_directory = '/content/TTS-TT2/logs' # Location to save Log files locally
log_directory2 = '/content/drive/My Drive/colab/logs' # Location to copy log files (done at the end of each epoch to cut down on I/O)
checkpoint_path = output_directory+(r'/')+model_filename
#### Train the model from scratch?
warm_start=False
hparams.text_cleaners=["english_cleaners"] + (["cmudict_cleaners"] if use_cmudict is True else [])
```

Figure 4 Configuration and Hyperparameter of Model During Training

```
MOS Scores per Audio File:
Audio File
1.wav     5.000000
10.wav    5.000000
11.wav    4.800000
12.wav    2.933333
13.wav    3.533333
14.wav    2.466667
15.wav    3.466667
2.wav     5.000000
3.wav     5.000000
4.wav     5.000000
5.wav     5.000000
6.wav     5.000000
7.wav     5.000000
8.wav     5.000000
9.wav     5.000000
Name: Rating, dtype: float64

Overall MOS for the TTS system: 4.48 ± 0.89
```

Figure 5 Overall MOS Score of the Text to Speech System