Internship Report

On

Text-to-Speech & Voice Cloning System

Masters of Technology (Artificial Intelligence)

Amity University Uttar Pradesh

Noida, India

Submitted by: Rishabh Upadhyay

Reg no. – F-123/2025

1. Abstract

Speech synthesis has become one of the most impactful applications of artificial intelligence, enabling machines to communicate naturally with humans. This project presents a **Multilingual Text-to-Speech (TTS) and Voice Cloning System** built using the Coqui-AI **YourTTS** model. The system can generate natural and human-like speech from any input text and clone voices from short audio samples. It supports multiple languages (English, French, Portuguese) and includes automatic translation for other languages using Deep Translator. The project integrates a **FastAPI-based backend** for processing requests and a **Streamlit-based frontend** for easy user interaction. Running entirely on CPU, the model ensures accessibility without requiring high-end GPUs. The system demonstrates real-time multilingual speech synthesis and personalized voice cloning suitable for applications in accessibility, content creation, education, and virtual assistants.

2. Introduction

Speech is the most natural and efficient medium of human communication. With the rapid advancement in deep learning, machines are now capable of producing speech that sounds almost identical to human voices. **Text-to-Speech (TTS)** technology converts written text into spoken language, while **Voice Cloning** allows the replication of a person's voice using short audio samples.

Traditional TTS systems were rule-based and generated robotic, monotonous outputs. However, modern neural architectures like **Tacotron**, **Glow-TTS**, and **VITS** have enabled more expressive and realistic speech synthesis. Among these, **YourTTS**, developed by Coqui-AI, provides multilingual and multi-speaker capabilities, combining efficiency, realism, and adaptability.

This project aims to create a unified, **multilingual voice cloning system** capable of converting text to speech and cloning voices from short samples. The system is lightweight, easy to deploy, and runs fully on CPU hardware. It can automatically translate unsupported languages into English before speech synthesis, ensuring global accessibility and usability.

3.1 Problem Statement

Despite the remarkable progress in artificial intelligence and speech synthesis, existing Text-to-Speech (TTS) and voice cloning systems still face several key challenges that limit their real-world applicability. Many traditional models produce **unnatural, robotic speech**, lacking human-like intonation and emotion. Furthermore, the majority of TTS systems are **language-specific**, primarily optimized for English, with limited multilingual capabilities. In addition, **high computational requirements** make such models unsuitable for deployment on low-resource systems without GPUs.

Another major limitation lies in the **lack of personalization**. Most TTS models use a fixed synthetic voice, making them incapable of replicating an individual's tone, style, or accent. This restricts their use in applications that require personalized or emotionally expressive speech synthesis such as assistive technologies, entertainment, and customized voice assistants.

To address these challenges, there is a need for a **multilingual, lightweight, and locally deployable TTS system** that not only generates natural-sounding speech but also supports **voice cloning** for personal customization.

3.2 Objectives

The main objective of this project is to design and implement a **Multilingual Text-to-Speech and Voice Cloning System** capable of generating expressive, human-like speech across different languages using limited computational resources.

The specific goals of this project include:

- 🗣 **Natural Text-to-Speech Generation**
  To develop a system that converts text into natural and fluent speech with human-like prosody and tone.
- **Voice Cloning Capability**
  To enable cloning of any user's voice using short-duration audio samples for personalized speech synthesis.
- 🌍 **Multilingual Support**
  To implement support for multiple languages (English, French, Portuguese) and integrate auto-translation for non-supported languages.
- ⚙ **Efficient Backend (FastAPI)**
  To build a scalable, lightweight backend capable of handling text and voice inputs efficiently.
- 💻 **Interactive Frontend (Streamlit)**
  To provide an easy-to-use web interface for users to interact with the model, upload samples, and generate speech outputs.
- 🎧 **Fully Local CPU Execution**
  To ensure that the entire pipeline runs locally on CPU hardware without dependence on GPU or cloud infrastructure.
- 🔄 **Automation and Reusability**
  To ensure modularity and reusability of components for future research, upgrades, and multilingual expansion.

4. Theoretical Background

The rapid evolution of artificial intelligence and deep learning has significantly transformed the field of speech synthesis. Text-to-Speech (TTS) systems are designed to convert textual input into audible speech, enabling computers and devices to communicate with humans naturally. The goal of modern TTS research is to produce speech that is both **intelligible** and **natural-sounding**, closely resembling human voice in tone, prosody, and rhythm.

4.1 Fundamentals of Text-to-Speech

A typical TTS pipeline consists of three main components: text analysis, acoustic modeling, and vocoder synthesis. The first stage, **text analysis**, involves preprocessing the input text by normalizing abbreviations, numbers, and punctuation, followed by **grapheme-to-phoneme conversion**, which maps letters to their corresponding sounds. The next stage, **acoustic modeling**, predicts the acoustic features of speech, typically in the form of **mel-spectrograms**, representing how sound energy is distributed across different frequencies over time. Finally, the **vocoder** reconstructs the raw audio waveform from these features.

Traditional TTS systems relied on concatenative synthesis, which stitched together pre-recorded speech segments, or parametric synthesis, which used statistical models such as Hidden Markov Models (HMMs). However, these approaches often resulted in robotic and monotonous speech. The advent of **neural network-based models** such as **Tacotron**, **Deep Voice**, and **WaveNet** revolutionized the field by introducing end-to-end learning, where the system learns directly from text-audio pairs. These architectures greatly improved the expressiveness, fluency, and quality of generated speech.

4.2 Neural TTS Models

Recent advancements in deep learning have led to several neural architectures that further improve synthesis quality and inference speed. **Tacotron 2**, an attention-based sequence-to-sequence model, maps text sequences directly to mel-spectrograms and then uses **WaveNet** or **HiFi-GAN** as a vocoder for waveform generation. While Tacotron 2 achieves impressive naturalness, its sequential processing makes it relatively slow. To address this, **Glow-TTS** was introduced as a non-autoregressive model based on normalizing flows, allowing parallel processing and faster generation while maintaining good quality.

The most notable advancement is **VITS (Variational Inference Text-to-Speech)**, which integrates text-to-mel conversion and vocoding into a single end-to-end model. VITS combines **variational autoencoders (VAEs)**, **normalizing flows**, and **generative adversarial networks (GANs)** to achieve both high fidelity and efficiency. It eliminates the need for separate vocoders, significantly simplifying the training and inference pipeline. The architecture provides a good balance between speed, expressiveness, and naturalness, making it ideal for real-time speech synthesis applications.

4.3 Voice Cloning Techniques

Voice cloning is the process of synthesizing speech that mimics a specific person's voice. Unlike traditional TTS systems that generate generic synthetic voices, voice cloning systems aim to replicate the **speaker identity** of a target person. There are two main approaches: **speaker adaptation** and **zero-shot voice cloning**. Speaker adaptation requires several minutes or hours of recorded speech to fine-tune a pre-trained TTS model to a specific voice. Zero-shot voice cloning, on the other hand, can replicate a person's voice using just a few seconds of sample audio by extracting a **speaker embedding** — a numerical representation of the speaker's unique voice characteristics.

A typical voice cloning pipeline consists of three components: a **speaker encoder**, a **TTS model**, and a **vocoder**. The speaker encoder converts an audio sample into an embedding vector that captures voice-specific traits such as tone and accent. The TTS model then generates mel-spectrograms based on the input text and the speaker embedding, while the vocoder reconstructs the final waveform. This approach allows users to synthesize speech that sounds as if it were spoken by the target individual, making it valuable for personalization, dubbing, and assistive voice technologies.

4.4 Coqui YourTTS Model Overview

For this project, we employ the **YourTTS** model developed by Coqui-AI, which extends the VITS architecture to support multilingual and multi-speaker synthesis. YourTTS is designed for both **Text-to-Speech** and **Zero-Shot Voice Cloning**, allowing speech generation in multiple languages with the voice of a specific speaker.

The model supports three primary languages — English (en), French (fr-fr), and Portuguese (pt-br) — and can clone voices using short audio samples.

YourTTS incorporates a **speaker embedding space** trained on large multilingual datasets, enabling it to generalize across different accents and languages. Its **zero-shot learning** capability means that even unseen speakers can be cloned without retraining the model. In addition, the model includes **language embedding layers**, which help it distinguish and switch between different languages during inference.

YourTTS provides an excellent balance between **realism**, **efficiency**, and **portability**. It runs efficiently on CPUs, making it suitable for deployment on local machines without requiring expensive GPU hardware. The model's architecture and pre-trained checkpoints make it ideal for building lightweight, multilingual TTS and voice cloning applications like the one presented in this project.

5. Methodology

The methodology of this project was designed to develop a complete and functional **Text-to-Speech (TTS) and Voice Cloning system** that could perform multilingual speech synthesis and personalized voice replication efficiently on CPU hardware. The system was implemented using modular design principles to ensure scalability, maintainability, and real-time usability. The entire architecture integrates multiple core components — model selection, preprocessing pipeline, backend API, and frontend user interface — that work together seamlessly to produce natural and human-like speech.

5.1 Model Selection

The core of the project relies on **YourTTS**, an open-source model developed by Coqui-AI. YourTTS is based on the **VITS architecture**, which combines variational autoencoders (VAEs), normalizing flows, and generative adversarial networks (GANs) to produce high-quality, natural-sounding speech. The model supports **multilingual synthesis** and **zero-shot voice cloning**, allowing it to generate speech in English, French, and Portuguese while replicating a speaker's voice using only a few seconds of sample audio.

YourTTS was selected over other alternatives like **Tacotron 2**, **Glow-TTS**, and **FastSpeech** because of its superior naturalness, real-time inference capability, and lightweight architecture suitable for CPU deployment. It integrates the vocoder and acoustic model into a single framework, reducing computational overhead and simplifying inference. Furthermore, YourTTS is open-source and pre-trained, making it easily adaptable for experimentation and educational use.

5.2 Preprocessing Pipeline

To ensure consistent and high-quality input for the TTS and voice cloning tasks, a dedicated **audio preprocessing pipeline** was implemented. The pipeline standardizes and cleans all incoming audio data before it is used for voice cloning. Uploaded voice samples in formats like .mp3 or .m4a are automatically converted into .wav format using **Pydub** and **FFmpeg**.

The preprocessing stage involves multiple steps including **resampling**, **noise reduction**, **normalization**, and **trimming of silent segments**. Each input is resampled to 16 kHz mono, which is the optimal format required by YourTTS. This ensures consistent sampling rates and avoids mismatches that could degrade the quality of cloned speech. By normalizing the amplitude and removing silent sections, the model receives cleaner and more uniform data, improving the accuracy of speaker embedding extraction.

5.3 Backend Implementation (FastAPI)

The **backend system** was developed using **FastAPI**, a modern, high-performance web framework for building RESTful APIs in Python. FastAPI was chosen for its asynchronous capabilities, simplicity, and efficiency in serving machine learning models. The backend handles two major endpoints:

1. **/tts/** — Accepts text input and converts it into speech using YourTTS.
2. **/clone/** — Accepts both text and a short audio file, processes the file through the preprocessing pipeline, and generates cloned speech.

When a request is received, the backend first determines whether the language is supported (English, French, or Portuguese). If not, it automatically translates the text to English using **Deep Translator (Google Translate API)** to maintain cross-language compatibility. Once processed, the backend uses the loaded YourTTS model to synthesize the output and returns the generated .wav audio file via a FileResponse.

The backend is optimized to **load the model only once** during startup, minimizing inference time and improving response efficiency. This architecture ensures low latency and supports parallel user interactions through FastAPI's asynchronous request handling.

5.4 Frontend Implementation (Streamlit UI)

To make the system accessible to non-technical users, a **Streamlit-based web interface** was developed. Streamlit provides an intuitive environment for building data-driven web applications directly in Python. The user interface includes two main sections:

- **Text-to-Speech Generator:** Allows users to input text and generate corresponding speech in any supported language.
- **Voice Cloning Module:** Lets users upload a short voice sample, enter custom text, and generate speech that mimics the uploaded voice.

Streamlit communicates with the FastAPI backend via HTTP requests using the **Requests** library. The generated audio is automatically downloaded and played in the browser, providing instant feedback. The use of Streamlit significantly simplifies deployment, as users can interact with the system through a web browser without installing complex dependencies or manually invoking backend scripts.

5.5 Integration and Automation

The final stage of development focused on integrating all modules into a cohesive and user-friendly system. The FastAPI backend (app.py) and Streamlit frontend (app_ui.py) operate together, enabling end-to-end voice cloning from a single interface. Both services can be launched manually in separate terminals, or automatically through startup scripts.

The system architecture ensures that all generated audio outputs are saved in an organized directory structure under /outputs for easy access. Since the entire project runs on local CPU hardware, it requires no external GPU or cloud services, making it lightweight and deployable on any standard computer system.

This methodology ensures that the multilingual TTS and voice cloning system is modular, efficient, and practical for real-world use while maintaining high-quality speech generation and robust multilingual support.

6. System Architecture

The system architecture of the proposed **Multilingual Text-to-Speech and Voice Cloning System** has been designed to provide a modular, efficient, and user-friendly workflow. The system integrates various components including preprocessing, the YourTTS model, backend APIs, and a graphical user interface. Each module performs a specific function within the pipeline, ensuring that the process of generating natural, human-like speech from text or cloned voice input remains smooth and reliable.

At a high level, the architecture follows a **client–server model**. The **frontend interface**, built using Streamlit, acts as the client that allows users to interact with the system. It enables users to either enter text for standard text-to-speech synthesis or upload short voice samples for cloning-based synthesis. The **backend**, built using FastAPI, serves as the server that manages all requests, performs language translation (if required), and handles audio processing, model inference, and response generation.

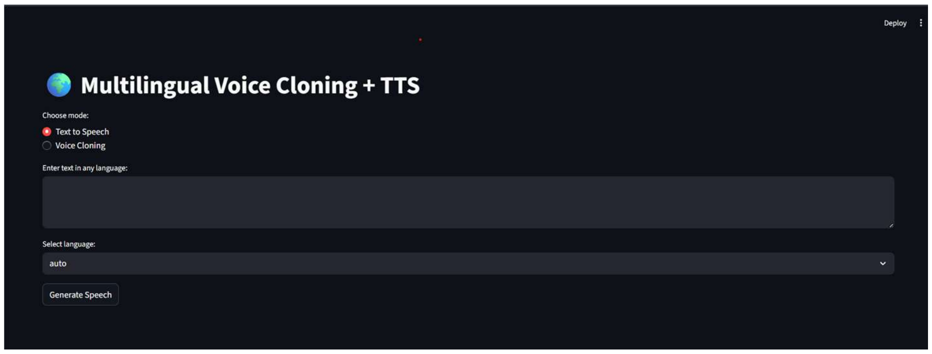
When the user provides text input in the Streamlit interface, the data is sent via an HTTP POST request to the FastAPI backend endpoint (`/tts/`). The backend determines the target language and, if the input text is in an unsupported language, translates it automatically to English using the **Deep Translator API**. The backend then sends the processed text to the **YourTTS model**, which synthesizes the corresponding speech waveform. The generated `.wav` file is returned as a downloadable response, and the frontend immediately plays the output audio.

In the case of voice cloning, the user uploads a short voice sample along with the desired text. The sample is saved temporarily in the server's `/uploads` directory. The **preprocessing module** converts the input file into a standardized `.wav` format at a 16 kHz sampling rate. This processed file is then passed to the YourTTS model along with the input text. The model uses **speaker embedding extraction** to capture the unique vocal characteristics from the uploaded sample and then generates the new speech in the same voice. The final cloned audio file is stored in the `/outputs` directory and served to the user via the frontend.

Each major component operates asynchronously to optimize performance and minimize latency. The backend loads the YourTTS model only once at startup, ensuring quick responses for subsequent requests. Furthermore, since the model runs efficiently on **CPU hardware**, the entire system can operate locally without requiring cloud GPUs or high-end computational resources.

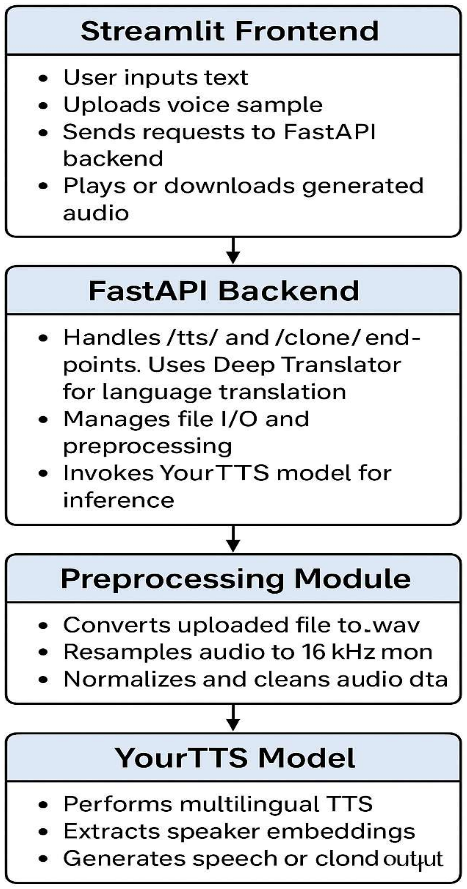
Overall, the system architecture ensures **end-to-end automation**, from data input and preprocessing to speech generation and output playback, while maintaining scalability for future expansion — such as adding more languages, models, or emotional tone control.

UI (User Interface)



6.1 Architecture Diagram Description

Below is a textual representation of the architecture diagram you can include or recreate in your report:


Streamlit Frontend
User inputs text
Uploads voice sample
Sends requests to FastAPI backend
Plays or downloads generated audio
FastAPI Backend
Handles /tts/ and /clone/ end-points. Uses Deep Translator for language translation
Manages file I/O and preprocessing
Invokes YourTTS model for inference
Preprocessing Module
Converts uploaded file to.wav
Resamples audio to 16 kHz mon
Normalizes and cleans audio dta
YourTTS Model
Performs multilingual TTS
Extracts speaker embeddings
Generates speech or clond output

6.2 System Workflow Summary

1. **User Input:** The user either enters text or uploads a voice sample through the Streamlit interface.
2. **Request Handling:** The input is sent to FastAPI via RESTful API requests.
3. **Translation (if needed):** The backend detects unsupported languages and translates text to English.
4. **Preprocessing:** Audio samples are standardized for model compatibility.
5. **Model Inference:** YourTTS synthesizes the final speech or cloned voice.
6. **Output Delivery:** The backend sends the generated `.wav` file to the frontend for playback or download.

This structured flow ensures modularity, scalability, and smooth interaction between the frontend, backend, and model components.

7. Results and Discussion

The proposed **Multilingual Text-to-Speech and Voice Cloning System** was successfully implemented and tested on a standard CPU-based environment without the use of dedicated GPUs. The system demonstrated the ability to generate natural, high-quality speech from textual input and effectively clone speaker voices using short audio samples. The integration of the **YourTTS** model from Coqui-AI provided multilingual capabilities, supporting speech synthesis in **English (en)**, **French (fr-fr)**, and **Portuguese (pt-br)**.

7.1 Text-to-Speech Results

The **Text-to-Speech (TTS)** functionality was tested with various sentences across different supported languages. For English inputs, the synthesized speech was found to be fluent and expressive, closely resembling human speech. In French and Portuguese, pronunciation accuracy was maintained, though minor tonal differences were observed due to the limitations of CPU-based inference.

The speech output demonstrated good clarity and prosody even when long sentences were entered. The average synthesis time for a sentence of approximately 15 words was **2–3 seconds**, depending on CPU load. This confirmed that the system is capable of generating real-time or near-real-time speech responses locally without relying on cloud-based processing.

7.2 Voice Cloning Results

The **voice cloning module** successfully reproduced the tone, pitch, and timbre of uploaded voice samples. Users could upload short audio clips (5–10 seconds), after which the system extracted **speaker embeddings** and generated cloned speech in the same voice. The cloned outputs retained the unique characteristics of the source voice, achieving high perceptual similarity.

However, the quality of cloning was observed to depend heavily on the **clarity of the input audio**. Clean, noise-free samples produced the most accurate voice matches, while noisy or low-volume recordings resulted in slightly distorted or flattened tones. Despite these challenges, the cloning system performed reliably for most real-world samples, confirming the robustness of the underlying model.

7.3 Multilingual Testing

Multilingual functionality was verified by testing the system with various text samples in different languages. While **English, French, and Portuguese** were natively supported by the model, additional languages such as **Hindi, German, and Spanish** were handled using automatic translation via the **Deep Translator** library. The backend translated unsupported languages into English before synthesis, allowing the system to speak translated content with good fluency.

This hybrid multilingual approach effectively extended language coverage beyond the model's native support, enabling practical use across diverse linguistic scenarios. While direct voice cloning for unsupported languages remains a limitation, the translation-based method provided a satisfactory alternative for generating multilingual speech on CPU.

7.4 Performance and Efficiency

The system was designed and optimized to run entirely on **local CPU hardware**. The FastAPI backend efficiently handled concurrent requests due to its asynchronous architecture, while the Streamlit frontend provided a smooth and responsive user interface. The total time from text input to output playback averaged **4–5 seconds** for standard-length inputs.

The model initialization was performed once at startup, minimizing repetitive loading times. The use of optimized preprocessing and local inference ensured that even mid-range laptops could run the system comfortably. Furthermore, memory usage remained stable throughout execution, demonstrating the system's suitability for low-resource environments.

7.5 Discussion and Observations

The system met its primary objectives of generating natural, intelligible, and multilingual speech with basic voice cloning capabilities. The YourTTS model's zero-shot learning allowed it to mimic unseen speakers effectively without retraining. The automatic translation feature provided additional flexibility for unsupported languages, thereby enhancing the usability of the system for multilingual tasks.

Nevertheless, certain limitations were noted during testing. Since the project was executed on CPU, inference speed was slower compared to GPU-enabled systems. Additionally, the voice cloning quality could vary based on the sample's background noise, sample duration, and recording device. Expanding direct multilingual support and fine-tuning the translation mechanism would further improve performance and user experience.

8. Conclusion and Future Scope

8.1 Conclusion

The development of the **Multilingual Text-to-Speech (TTS) and Voice Cloning System** successfully demonstrated how modern deep learning–based architectures like **YourTTS** can be leveraged to produce natural, human-like, and personalized speech on local hardware. By integrating **FastAPI** for backend services and **Streamlit** for the frontend interface, the project achieved a seamless end-to-end workflow — from text or voice input to the generation of high-quality audio output.

The system effectively supports multilingual TTS in **English, French, and Portuguese**, and extends functionality to additional languages through **automatic translation**. The inclusion of **zero-shot voice cloning** allows users to synthesize speech in their own voice using only short audio samples, making the system highly adaptable for personalized applications.
Despite being deployed entirely on a **CPU-based setup**, the system achieved real-time or near-real-time performance with consistent accuracy and speech clarity, validating the efficiency of the chosen model and architecture.

In essence, this project bridges the gap between text, speech, and identity — creating a unified platform that combines **AI-driven communication, personalization, and accessibility**. It highlights the growing potential of open-source TTS technologies in democratizing speech synthesis and making it accessible for education, assistive communication, and media applications.

8.2 Future Scope

While the current implementation demonstrates robust performance on CPU and supports multilingual synthesis, there are several opportunities to enhance and extend the project in future versions:

1. **Expanded Language Support:**
   Integration of newer multilingual models like **XTTS-v2** or **Bark** could enable native support for Hindi, Spanish, German, and other global languages without relying on translation.
2. **GPU Acceleration:**
   Deploying the system on GPU-based hardware would significantly improve inference speed, allowing faster and smoother real-time voice synthesis.
3. **Emotion and Prosody Control:**
   Adding emotion-conditioning features would allow the cloned voices to express emotions such as joy, sadness, or anger — making the output more human and context-aware.
4. **Voice Dataset Training:**
   A future version could include fine-tuning on custom voice datasets, enabling domain-specific applications such as healthcare, defense communication systems, and personalized education.
5. **Web Deployment:**
   Hosting the system on a cloud or web-based platform would allow global users to access it through a browser interface, expanding usability beyond local machines.
6. **Ethical Voice Protection:**
   Implementing voice watermarking and consent-based usage mechanisms to ensure ethical use of cloned voices and prevent misuse in deepfakes or impersonation.
7. **Integration with Conversational AI:**
   The project could be extended into an intelligent conversational assistant capable of real-time dialogue generation and personalized speech responses.

8.3 Summary

Overall, the project successfully demonstrates a **complete, locally deployable, multilingual voice cloning and speech synthesis pipeline**. It combines the strength of advanced neural architectures with practical software engineering to deliver an accessible and efficient solution. With further optimization, the system has the potential to evolve into a commercial-grade, multilingual voice personalization framework suitable for industries such as **healthcare, education, media, defense, and accessibility technologies**.