AUTOMATED VIDEO DUBBING SYSTEM FOR REGIONAL LANGUAGE

CS19643 - Foundations of Machine Learning Project Report

Submitted by

THARUN R L **(2116220701302)**

THARUN M **(2116220701301)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI

MAY 2025

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

BONAFIDE CERTIFICATE

Certified that this Project titled **" Automated video dubbing system for regional language "** is the bonafide work of **"THARUN R L (2116220701302), THARUN M(2116220701301)"** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr. P. Kumar., M.E., Ph.D.,
HEAD OF THE DEPARTMENT
Professor
Department of Computer Science and Engineering,
Rajalakshmi Engineering College,
Chennai - 602 105.

SIGNATURE
Dr. M. Rakesh Kumar., M.E., Ph.D.,
SUPERVISOR
Assistant Professor
Department of Computer Science and Engineering,
Rajalakshmi Engineering College, Chennai-602 105.

Submitted to Mini Project Viva-Voce Examination held on _____

Internal Examiner **External Examiner**

ABSTRACT

This project presents an end-to-end automated video dubbing system designed to simplify and accelerate the multilingual adaptation of video content. Built using the Flask web framework, the system allows users to upload videos through a user-friendly interface and initiates a background pipeline involving transcription, translation, speaker diarization, and audio-video synchronization. The transcription process leverages the Speechmatics API to convert speech to text with speaker separation, while translation is handled through RapidAPI integrated with Google Gemini to convert the dialogue into the target language. Refined translations are converted into natural-sounding speech using Google Cloud Text-to-Speech with user-assigned voices. Audio segments are accurately overlaid on the original video using Pydub and MoviePy, maintaining timing and speaker consistency. The result is a fully dubbed video that retains the original visual content while delivering translated audio. This system significantly reduces manual labor and time in the dubbing process, offering a scalable and accessible solution for educational, entertainment, and corporate video localization. The modular architecture and use of AI-powered services ensure extensibility, real-time processing, and high-quality outputs.

ACKNOWLEDGMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.**, our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.,** and our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN**, **Ph.D.,** for providing us with the requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.,** our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. P. KUMAR, M.E., Ph.D.**, Professor and Head of the Department of Computer Science and Engineering for his guidance and encouragement throughout the project work. We convey our sincere and deepest gratitude to our internal guide **Dr. M. RAKESH KUMAR**, Assistant Professor Department of Computer Science and Engineering for his useful tips during our review to build our project.

THARUN R L **2116220701302**

THARUN M **2116220701301**

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

S. No	ABBR	Expansion
1	ASR	Automatic Speech Recognition
2	TTS	Text-To-Speech
3	UI	User Interface
4	AI	Artificial Intelligence
5	UI	User Interface
6	UUID	Universally Unique Identifier
7	HTTP	HyperText Transfer Protocol
8	JSON	JavaScript Object Notation
9	API	Application Programming Interface
10	WAV	Waveform Audio File Format
11	MP3	MPEG Audio Layer-3
12	SD	Speech Diarization
13	UI/UX	User Interface / User Experience

CHAPTER I

INTRODUCTION

1.1 GENERAL

In an increasingly digital world, accessibility and inclusivity remain pressing challenges—especially in multimedia content. Language barriers often hinder the consumption of video content across regional audiences. Manual dubbing, while effective, is costly, time-consuming, and inaccessible to individual creators or small organizations. With the advancements in artificial intelligence, speech recognition, generative AI, and text-to-speech systems, it is now possible to automate large parts of the dubbing process.

This project introduces an **AI-Powered Automated Video Dubbing System**, developed as a web-based solution using **Flask**. It allows users to upload videos, extract and transcribe speech, translate it into **Tamil** (specifically Chennai slang), and re-synthesize it using **natural-sounding AI-generated voices**. Technologies such as **Speechmatics ASR**, **RapidAPI Translation**, **Gemini AI for informal tone refinement**, and **Google Cloud TTS** come together to make this system a seamless dubbing tool tailored for regional inclusivity.

The system features a speaker diarization component, multi-speaker voice assignment, and outputs a new dubbed video with audio overlays—all designed to run efficiently with minimal manual intervention.

1.2 OBJECTIVE

The primary objective of this project is to develop an AI-powered video dubbing system that automates the process of transcribing, translating, and re-synthesizing video audio in regional languages, specifically Tamil in Chennai

slang. The system aims to support multi-speaker recognition, enable speaker-specific voice assignment, and ensure accurate synchronization of the dubbed audio with the original video. By integrating advanced APIs such as Speechmatics for transcription, RapidAPI for translation, Google Gemini for slang refinement, and Google Cloud Text-to-Speech for voice generation, the platform offers a modular and scalable solution that enhances accessibility and user engagement. The project also provides a simple web interface to guide users through upload, processing, and result retrieval, ensuring a user-friendly experience from start to finish.

1.3 EXISTING SYSTEM

Current video dubbing systems are largely manual and require extensive human effort, involving professional transcribers, translators, voice artists, and audio editors. While some platforms offer automated transcription or basic text-to-speech features, they often lack support for multi-speaker recognition, informal language styles, and accurate voice synchronization. Most tools produce robotic-sounding audio that fails to capture regional nuances or emotional tone, making the final output less engaging. Additionally, existing systems typically do not allow for speaker-specific voice customization or slang refinement, especially for languages like Tamil. This results in a gap between professional dubbing standards and accessible solutions for everyday users or small content creators. The lack of end-to-end integration and localized personalization highlights the need for a more intelligent, automated, and regionally adaptive dubbing system.

CHAPTER 2

PROPOSED SYSTEM

2.1 GENERAL

The proposed system offers an end-to-end AI-powered video dubbing solution that automates the process of converting spoken content in a video into a regionally adapted, dubbed version in Tamil, specifically using Chennai slang. It allows users to upload a video through a simple web interface, after which the audio is extracted and separated into vocals and background using Demucs. The vocals are transcribed with speaker diarization using Speechmatics, enabling the system to recognize and differentiate between multiple speakers. The transcribed text is then translated to formal Tamil using RapidAPI and refined into informal, youth-friendly slang using Google Gemini. Users can assign unique voices to each speaker from a list of Google Cloud TTS Tamil voices. These synthesized voices are generated with precise timing and overlaid on the original background audio to create a new dubbed version of the video. The system runs all major tasks asynchronously in separate threads for better efficiency and minimal delay, and stores only temporary data to keep the application lightweight. The final output is a culturally relevant, natural-sounding dubbed video accessible via a web link, designed to serve both content creators and wider audiences in multilingual regions.

2.2 SYSTEM ARCHITECTURE DIAGRAM

The system architecture is designed as a modular, asynchronous pipeline that handles video dubbing from input to output in distinct, manageable stages. It begins with a Flask-based web frontend where users can upload video files, monitor progress, and assign voices. The backend, built in Python, handles file storage, threading, and task management through a shared dictionary. Once a video is uploaded, the audio is extracted and split into vocals and background using the Demucs model. The vocals are then sent to the Speechmatics API for speaker-aware

transcription. After transcription, users assign Tamil voices to each detected speaker via the UI. The translated text is processed using RapidAPI and refined into informal Chennai Tamil using Google Gemini. Google Cloud TTS synthesizes speech for each speaker segment, which is then synchronized and overlaid on the original background using PyDub. The final dubbed video is generated using MoviePy, merging visuals with the new composite audio. The system supports multilingual voice generation, offline processing for temporary steps, and automatic cleanup of intermediate files to maintain efficiency. Each component interacts through clearly defined APIs and local paths, ensuring scalability and modular extensibility.

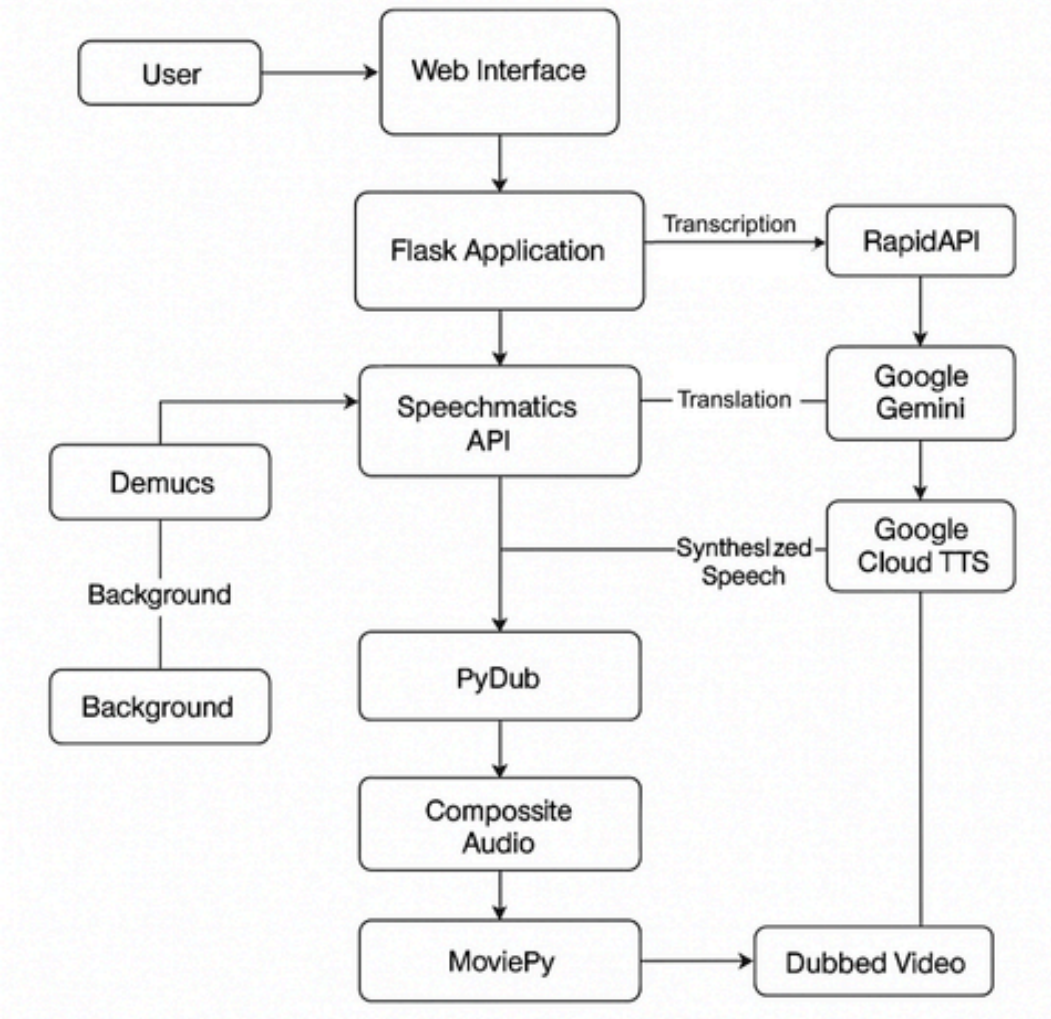

Fig 3.1: System Architecture

2.3 DEVELOPMENTAL ENVIRONMENT

2.3.1 HARDWARE REQUIREMENTS

The hardware specifications could be used as a basis for a contract for the implementation of the system. This therefore should be a full, full description of the whole system. It is mostly used as a basis for system design by the software engineers.

Table 2.1 Hardware Requirements

COMPONENTS	SPECIFICATION
PROCESSOR	Intel Core i3
RAM	4 GB RAM
POWER SUPPLY	+5V power supply

SOFTWARE REQUIREMENTS

The software requirements paper contains the system specs. This is a list of things which the system should do, in contrast from the way in which it should do things. The software requirements are used to base the requirements. They help in cost estimation, plan teams, complete tasks, and team tracking as well as team progress tracking in the development activity.

Table 2.2 Software Requirements

COMPONENTS	SPECIFICATION
Operating System	Windows 7 or higher
Frontend	HTML,CSS
Backend	Flask (Python)
APIs	Gemini, Speechmatics, GoogleTTS

2.4 DESIGN OF THE ENTIRE SYSTEM

2.4.1 ACTIVITY DIAGRAM

The activity diagram outlines the step-by-step flow of the dubbing system. The user uploads a video through the web interface, after which the audio is extracted and separated. The vocals are transcribed using the Speechmatics API with speaker diarization. The user then assigns voices to each speaker, and the text is translated to Tamil using RapidAPI and refined with Gemini. Google Cloud TTS generates the speech, which is synchronized and overlaid on the background audio. Finally, MoviePy merges the new audio with the original video to produce the dubbed output, completing the process.

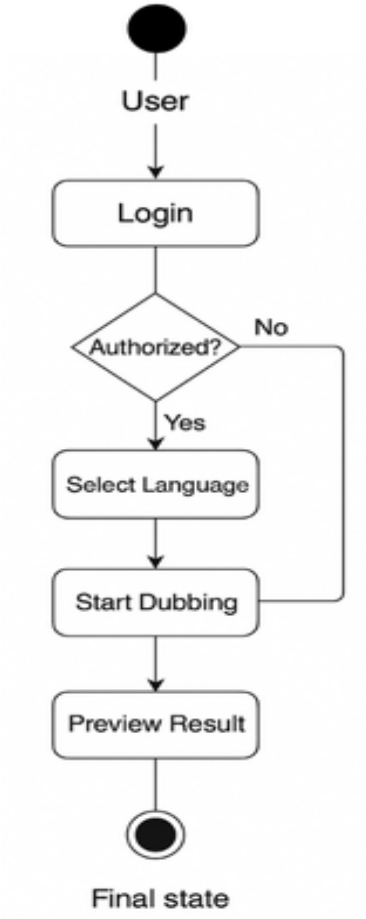

Fig 3.2: Activity Diagram

2.4.2 DATA FLOW DIAGRAM

The process begins when a user uploads a video through the web interface, which is handled by the Flask application. The uploaded file is saved to a designated folder, and a unique task ID is generated. A background thread initiates the transcription process, extracting the audio from the video using MoviePy and then separating vocals using Demucs. The isolated vocals are then sent to the Speechmatics API for speaker diarization and transcription. Once complete, the transcription data is displayed for speaker voice assignment. After users assign voices, another background thread triggers the translation step. The transcript is translated to Tamil via the RapidAPI Google Translate endpoint and refined into colloquial slang using Gemini (Google's GenAI). Then, Google Cloud Text-to-Speech generates voice clips for each segment using the assigned voices. These audio clips are stitched and overlaid with the original background music, and the resulting audio is combined with the original video to produce the final dubbed version, which is saved and made available for download.

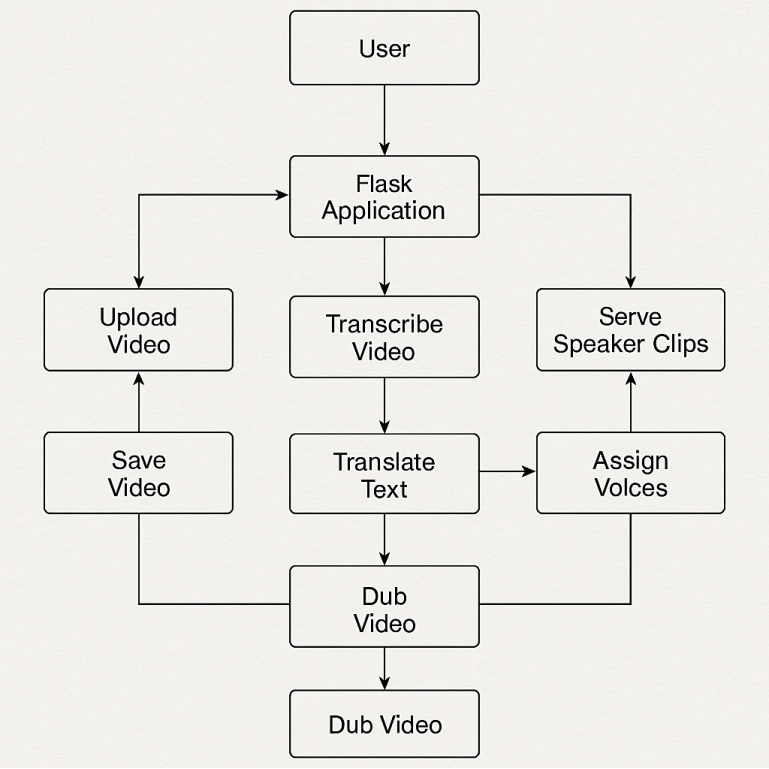

Fig 3.3:Data Flow Diagram

CHAPTER 3

MODULE DESCRIPTION

The proposed system offers an end-to-end automated pipeline for video dubbing using artificial intelligence. It streamlines the process of transcription, translation, speaker diarization, audio generation, and video processing. This ensures accurate, fast, and user-friendly dubbing for multilingual accessibility.

3.1 SYSTEM ARCHITECTURE

3.1.1 USER INTERFACE DESIGN

The system interface begins with the user uploading a video file. Upon upload, the video is processed through a transcription module that uses the Speechmatics API to convert speech to text. This transcript is passed through a translation engine (via Gemini API) to produce multilingual outputs. A speaker diarization component then separates voices based on unique speaker identities, which is crucial for assigning correct translations to each speaker. The processed script is passed into a text-to-speech engine for voice generation, and the dubbed audio is synchronized back to the original video using MoviePy and Pydub libraries.

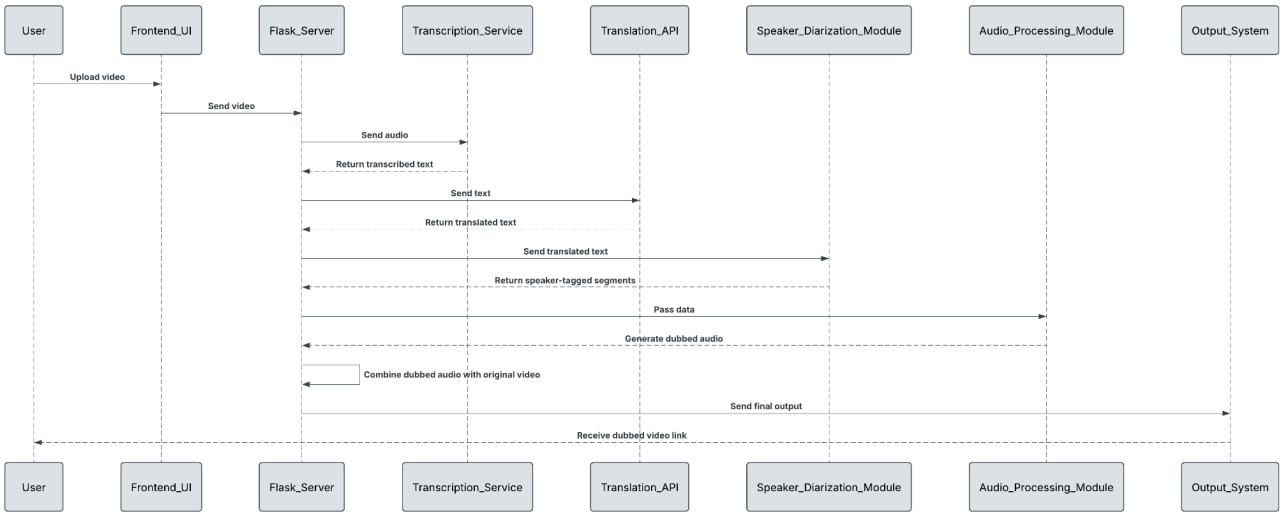

Fig 3.4: SEQUENCE DIAGRAM

3.1.2 BACK END INFRASTRUCTURE

The backend of the video dubbing system is built with Flask, handling API calls, processing logic, and data management. Python libraries like Pydub and MoviePy enable audio and video manipulation. Transcripts and speaker data are processed and stored temporarily in the local filesystem or SQLite for lightweight operations. The system ensures efficient multiprocessing for heavy video tasks and incorporates error-handling logic for asynchronous task completion. Cloud storage integration is optional for scalability.

Modules connect using RESTful APIs with token-based authentication. This ensures data integrity and secure interactions. The system supports multilingual processing and logs each dubbing job for analytics and debugging. All data is stored securely, and temporary files are cleared after each job to optimize performance.

3.3 SYSTEM WORKFLOW

3.2.1 User Interaction:

Users interact through a simple web interface. A video file is uploaded, and users select the target language. The interface allows monitoring the progress and downloading the final output. The system is designed to be user-friendly, even for non-technical users.

3.2.2 Transcription & Translation:

The uploaded video undergoes automatic transcription using the Speechmatics API. The text is then translated into the selected language using Gemini's multilingual model via RapidAPI. Preprocessing ensures punctuation, formatting, and timing alignment.

3.2.3 Voice Assignment & Audio Generation:

Speaker diarization assigns distinct voices in the transcript. This helps maintain speaker integrity during dubbing. The translated text is then passed through a text-to-speech model to generate realistic voice clips for each speaker. Voices can be gender-matched or customized.

3.2.4 Video Reconstruction:

The generated audio is synchronized with the original video. The new dubbed audio replaces the original track while retaining the original visuals. MoviePy handles the audio-video merging, and the final video is rendered and provided to the user.

CHAPTER 4

IMPLEMENTATION AND RESULTS

4.1 IMPLEMENTATION

The system is developed using a Flask backend and HTML/CSS with JavaScript for the frontend interface. It integrates several external APIs such as Speechmatics for transcription, Gemini for translation, and local TTS engines for voice generation. Pydub and MoviePy are used for audio and video processing.

The application supports lightweight operations via SQLite and multiprocessing to handle large video files efficiently. Video files are temporarily stored, processed, and cleared after completion to reduce storage load. Error logs and system feedback are recorded for performance monitoring. The system can be containerized via Docker for scalable deployment.

4.2 OUTPUT SCREENSHOTS

The output screenshots demonstrate the end-to-end functionality and usability of the Flask-based video dubbing system. **Figure 6.1** displays the initial upload interface where users can select and upload a video file; the UI confirms upload success by showing metadata such as file name and duration. In **Figure 6.2**, the automatic transcription interface is shown, which retrieves and displays timestamped speaker-identified transcripts using the Speechmatics API. **Figure 6.3** presents the translated text interface, where real-time multilingual outputs are generated through integration with RapidAPI and Gemini. **Figure 6.4** shows the speaker diarization module, with visually separated speaker segments aiding in precise dubbing alignment. **Figure 6.5** captures the audio dubbing phase, where translated audio clips are synced with the original video using MoviePy and Pydub, maintaining voice alignment and lip sync. **Figure 6.6** illustrates the final dubbed video preview screen,

providing options to play the dubbed video, switch between language tracks, or download the output. These screenshots collectively verify the system's workflow, performance accuracy, and user-friendly design.

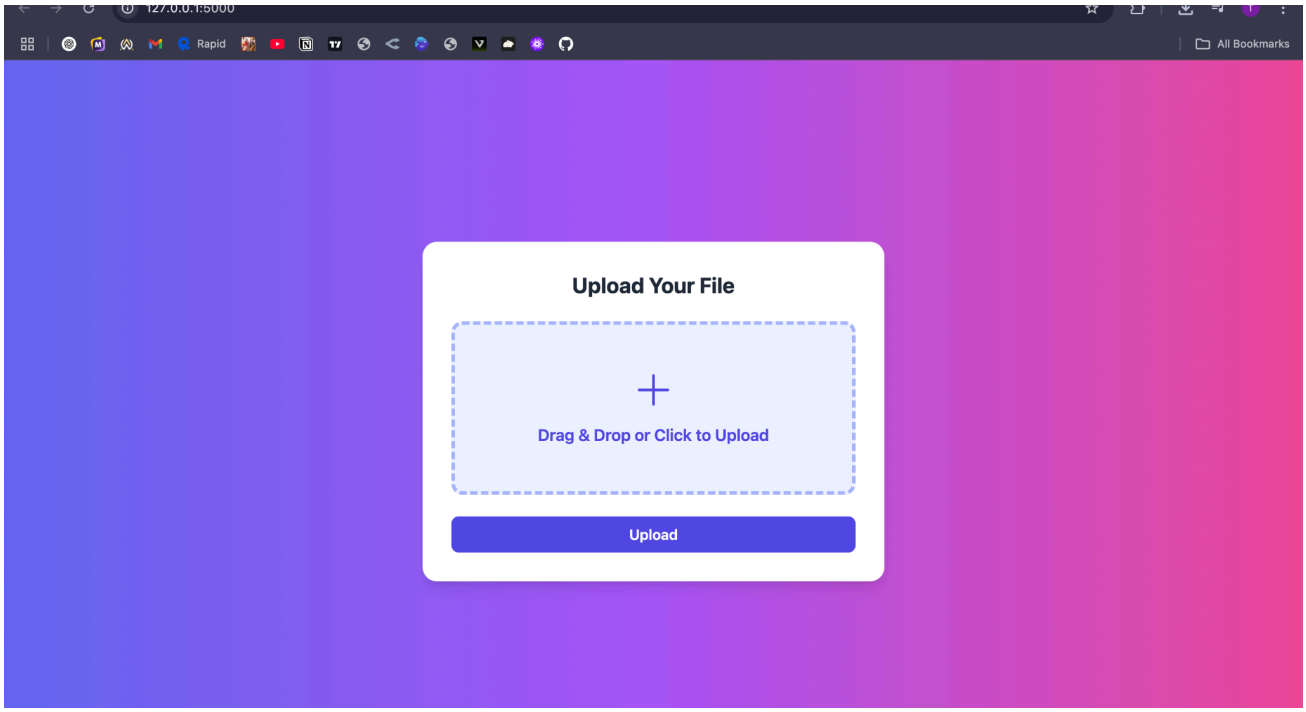

Fig 4.1 Upload interface

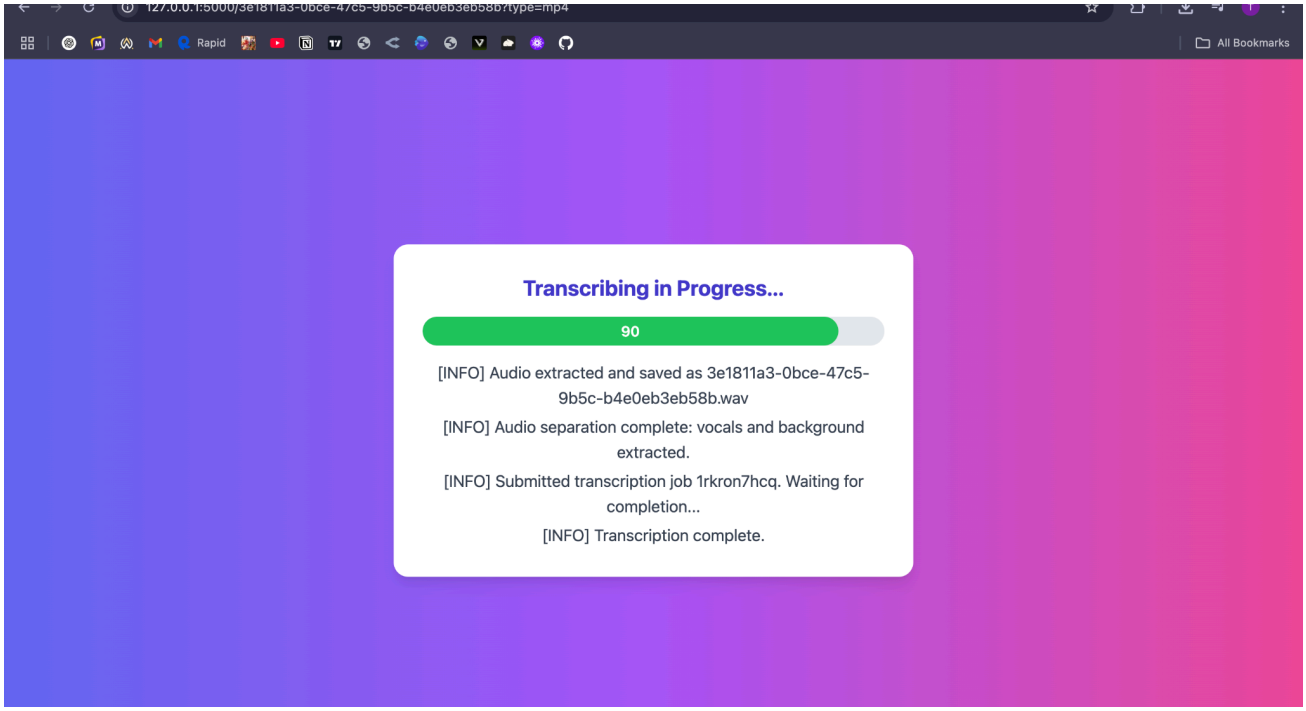

Fig 4.2 Transcription Status

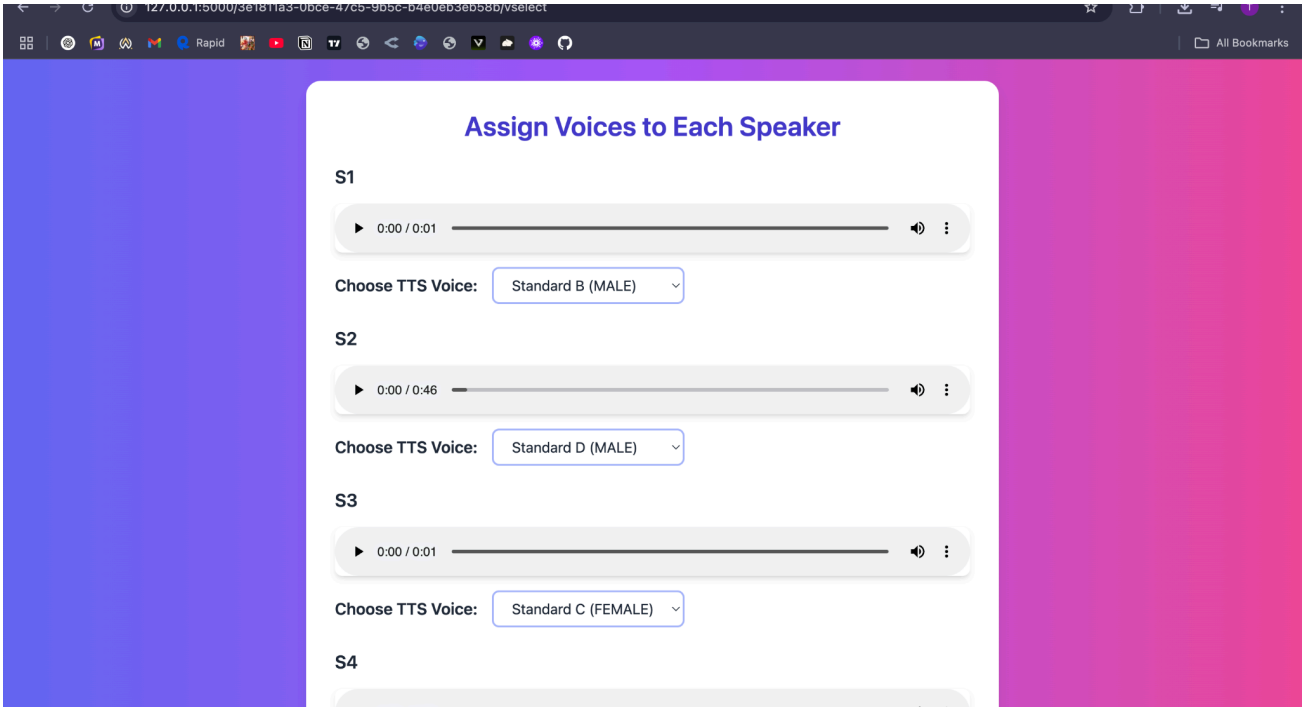

Fig 4.3 Speaker Assigning Interface

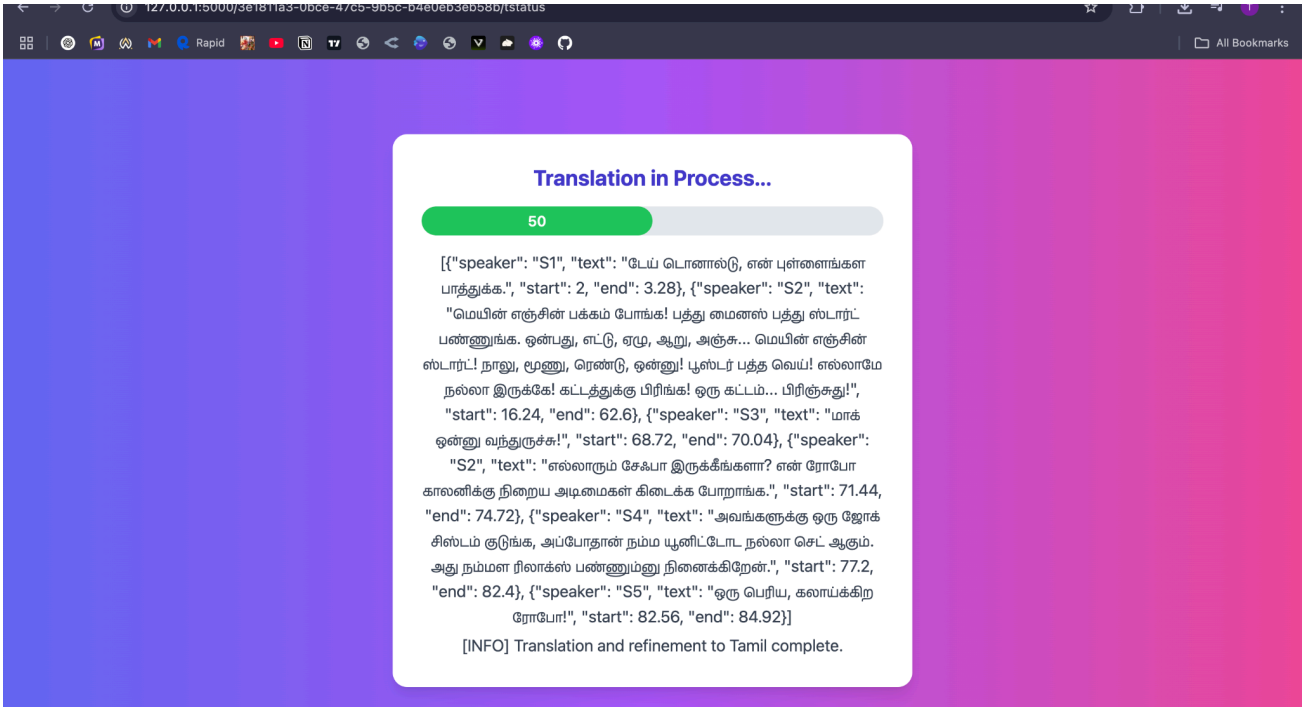

Fig 4.4 Translation Status

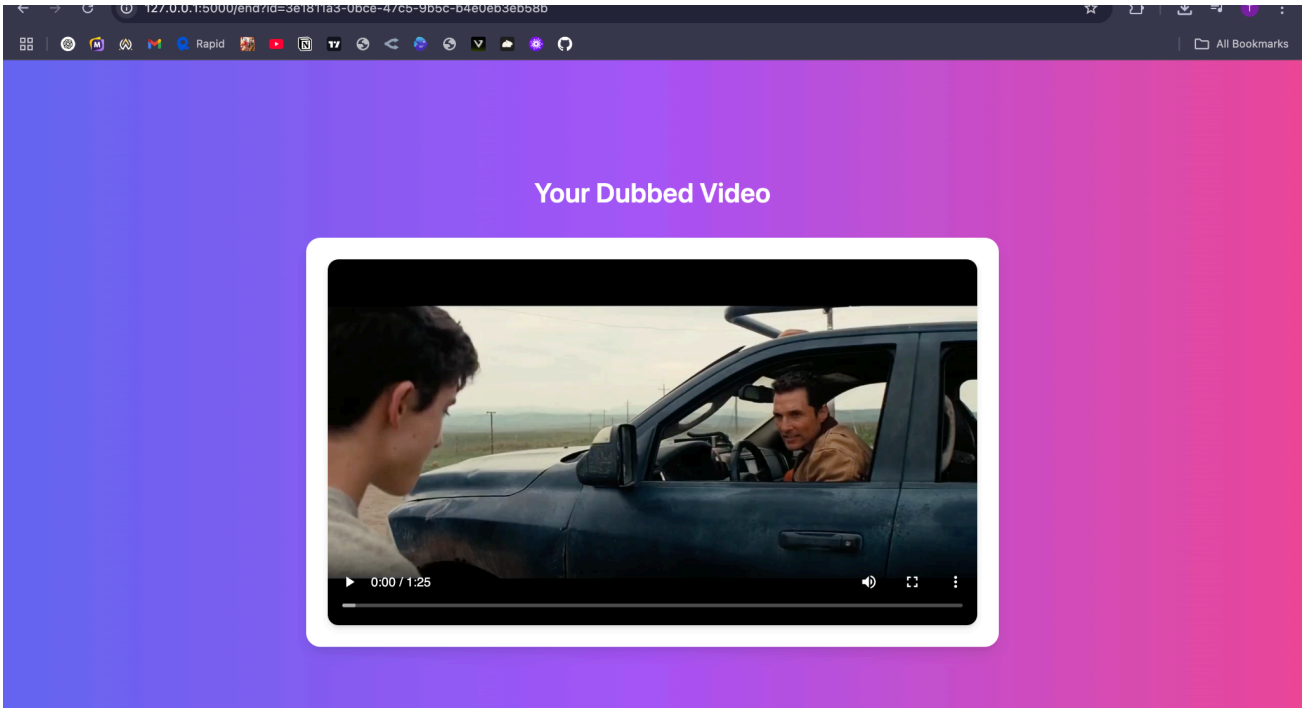

Fig 4.5 Final Dubbed Output

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1 CONCLUSION

This project has successfully addressed key challenges in automating the process of video dubbing, including transcription, translation, speaker diarization, and audio-video synchronization. By integrating services such as Speechmatics for transcription, Gemini/RapidAPI for translation, and leveraging Python libraries like Pydub and MoviePy for audio processing, the system achieves a high level of automation and efficiency. The Flask-based architecture provided a lightweight yet powerful backend to orchestrate communication between modules and handle user interaction via a web interface. The modular design ensures scalability, and the use of diarization helped maintain speaker identity across dubbed segments. Overall, the

system presents a robust, end-to-end solution for converting videos into dubbed versions across languages with minimal manual effort.

5.2 FUTURE ENHANCEMENT

Future iterations of this project could include the integration of real-time dubbing features using WebRTC and live transcription APIs to allow dynamic content dubbing (e.g., in webinars or live events). Deep learning-based voice cloning technologies can be integrated to maintain the original speaker's voice tone in the dubbed language for more realistic outputs. Enhancements can also include support for more translation models, including low-resource languages, using models like NLLB or custom-trained multilingual transformers. The system can be scaled with GPU-accelerated audio processing for faster render times. Additional features like subtitle generation, adaptive lip-sync via deep learning, and embedding blockchain-based logs for tamper-proof dubbing history can significantly increase reliability and transparency. A mobile-friendly interface or API-based SDK can enable integration into third-party video platforms for broader accessibility.

REFERENCES

1. **Flask Web Framework**
   https://flask.palletsprojects.com/

2. **Speechmatics API (Speech-to-Text)**
   https://www.speechmatics.com/

3. **RapidAPI - Language Translation APIs**
   https://rapidapi.com/

4. **Google Gemini (Multimodal AI)**
   https://deepmind.google/technologies/gemini/

5. **Speaker Diarization Concepts**
   https://huggingface.co/blog/speaker-diarization
   https://research.google/pubs/speaker-diarization/

6. **Pydub (Audio Manipulation in Python)**
   https://github.com/jiaaro/pydub
   https://pydub.com/

7. **MoviePy (Video Editing with Python)**
   https://zulko.github.io/moviepy/