Computer Communication and Networks(ITCS 6166)

Project Report

Video Call Application(WebRTC) with audio driven talking head generation

Submitted By

Srikar Chamarthy	**ID: 801317299**
Aravind Pabbisetty	**ID: 801274519**
Aneela Gannarapu	**ID: 801312361**
Sai Kiran Reddy Bokka	**ID: 801283586**

INTRODUCTION

The need for anonymous communication arises in various situations, where individuals may wish to protect their identity while communicating with others. For example, in confidential business communication, anonymity may be necessary to protect sensitive information and prevent leaks. Similarly, in support groups, individuals may feel more comfortable sharing their experiences and feelings if they can do so anonymously.

The central idea behind this project is to develop a video call application that allows individuals to remain anonymous while communicating with each other. The application masks the caller's face with a chosen image, creating a "talking head" that replicates the caller's speech patterns. This technology is accomplished using machine learning algorithms that analyze the caller's speech and synchronize the chosen image's mouth movements with the caller's speech.

The “MakeItTalk” model generates a speaker-aware talking-head animation synchronized with the audio. It uses a voice conversion neural network to disentangle the speech content and identity information. The content is speaker-agnostic and captures the general motion of lips and nearby regions. The identity of the speaker determines the specifics of the motions and the rest of the talking-head dynamics.

With the increasing use of video conferencing for work, education, and social interaction, there is a growing need to improve the user experience and make remote communication more engaging and interactive. Additionally, in situations where individuals may wish to protect their identity while communicating with others, there is a need to balance privacy concerns with effective communication. This project aims to address both of these needs by developing a video call application that enhances the user experience while also allowing anonymous communication through a talking head feature.

Requirements:

Hardware Requirements:

- CPU :Multi Core
- GPU(optional)
- RAM: 8 GB
- Webcam

Software Requirements:

- Operating System: Windows 10
- Python 3.10.9

Libraries Used:

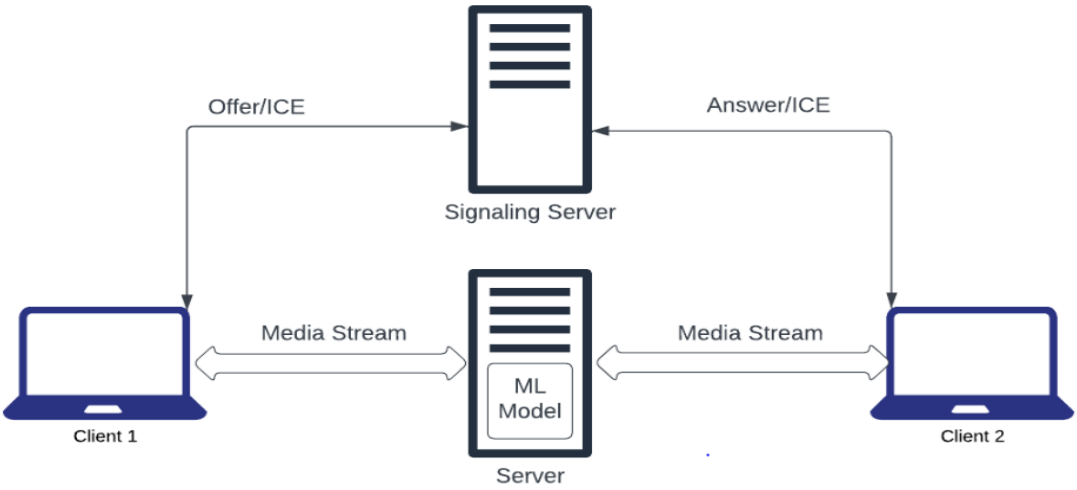
The below are the list of libraries used and their purpose in the this project:

- ffmpeg-python: This is used to trim, concatenate and overlay the audio files that we record.its command-line interface.
- opencv-python: This can be utilized when dealing with image processing. In our case this library comes handy when we overlap the caller's face with a talking head image.
- face_alignment: It is used to detect 2D and 3D face alignment.
- scikit-learn: library for machine learning in Python. This provides us with a selection of efficient tools for machine learning where we can implement regression, clustering and dimensionality reduction via a consistent interface.
- pydub: This library is used to handle the .wav files which are generated after recording the audio.
- pynormalize: Using this we save the metadata of the original audio file,which can be used for processing.
- soundfile: Read and write operations of audio file are performed using this module.
- librosa(0.8.1): This is used to normalize the audio file and to extract features of the audio file.
- pysptk: It’s a python wrapper library for speech signal processing.

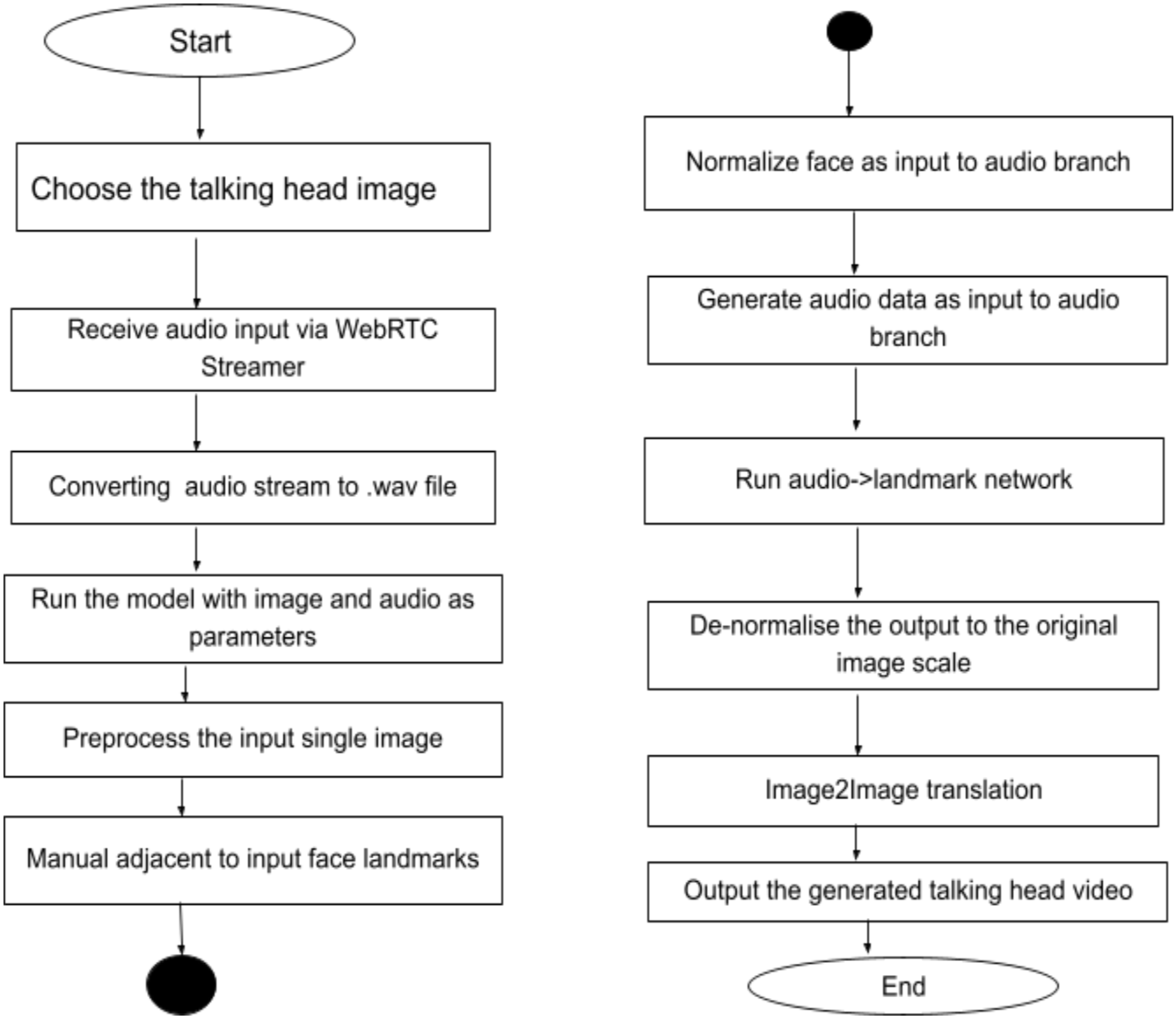
- pyworld: This is used for speech breakdown in our project.
- resemblyzer: It is utilized to implement the deep learning part of our project.
- Streamlit: Webpage application is built using this library.
- Streamlit-WebRTC: This is used for audio-video streaming, enabling real-time transmission.
- Streamlit-Server-State: This is used to maintain separate sessions for each client.
- Mediapipe: This library to used to implement the real time face swap model.

Architecture of the Project:

The project is a web application and implements a client-server architecture. We will be using WebRTC protocol for real time communication. In this application architecture first the peers will exchange their offer and answer responses via a Signaling server.The offer and answer responses will contain session description, peers media capabilities etc. Once it's done then they exchange ICE(Inter Connectivity Establishment) candidates to identify and exchange network addresses, ports via the same signaling server. In between the peers there will be a relay server which hosts the ML model used to convert the audio stream from the first peer using the face selected into the audio driven face stream. The relay server will send the converted stream to the other peer in the WebRTC session. The signaling server and relay server can be implemented using the same server but for the architecture purpose we have shown differently.



Flowchart:


Start
Choose the talking head image
Receive audio input via WebRTC Streamer
Converting audio stream to .wav file
Run the model with image and audio as parameters
Preprocess the input single image
Manual adjacent to input face landmarks
Normalize face as input to audio branch
Generate audio data as input to audio branch
Run audio->landmark network
De-normalise the output to the original image scale
Image2Image translation
Output the generated talking head video
End

Implementation Details:

Talking Head Generation:

We have created a WebRTC streamer using streamlit-webrtc in streamlit. When the user starts the streamer, the streamer starts to accept audio streams from the user. After the user stops the streamer the Media Recorder component in the recorder_factory function of streamlit-webrtc library is used to convert the recorded audio stream to a .wav audio file. Using the streamlit-image-select library user is given the choice to select an image for the talking head. The chosen image and the audio file are transmitted to the server where the model is present. The model accepts these parameters as inputs and processes them using various image and audio processing techniques. Using audio to landmark and image2image translation techniques a scaled video is generated and the image is transformed as per the scaled video. The model returns the talking head video to the video streamer and the video is streamed to the user. We tried to integrate this in real time video call but were unable to implement it.

Real Time Face Swap:

We have implemented a video call application where two clients can mutually communicate with each other and can be able to swap their faces with the selected image. We have created it mainly using streamlit and streamlit-webrtc. The video call user interface is developed using streamlit and it consists of two webrtc streamers one for sending and other for receiving. The connection between two clients is established with the help of WebRTC protocol, streamlit_server_state, Stun server and Ice candidates are generated and exchanged. A streamlit session is created and once the users click on the start button of the webrtc streamer takes the video and audio of the users from the webcam and transmits to the receiver at the other end and vice versa. Users can do all other basic functionalities like turning off their webcam or muting the audio if they wish to.Users can select an image and make their face swap with the selected image as well in real time. Once the user stops the webrtc streamer the transmission is stopped and the call is terminated.

Iteration Plan

Iteration 1:

- Installing Python3.10
- Installing Conda and setting up the Conda environment.
- Researching provided ML models for the audio driven approach and deciding which model to implement.
- Learning about streamlit and WebRTC concepts.

Iteration 2:

- Creating a UI design for the application.
- Developing a video call application using WebRTC.
- Implementing code to send the audio stream and the image selected in the UI to the relay server.

Iteration 3:

- Installing all the required libraries for the chosen ML model in the Conda environment.
- Integrating ML model to transform audio and image to generate a video.
- Transmitting the generated video to the second client via the relay server.

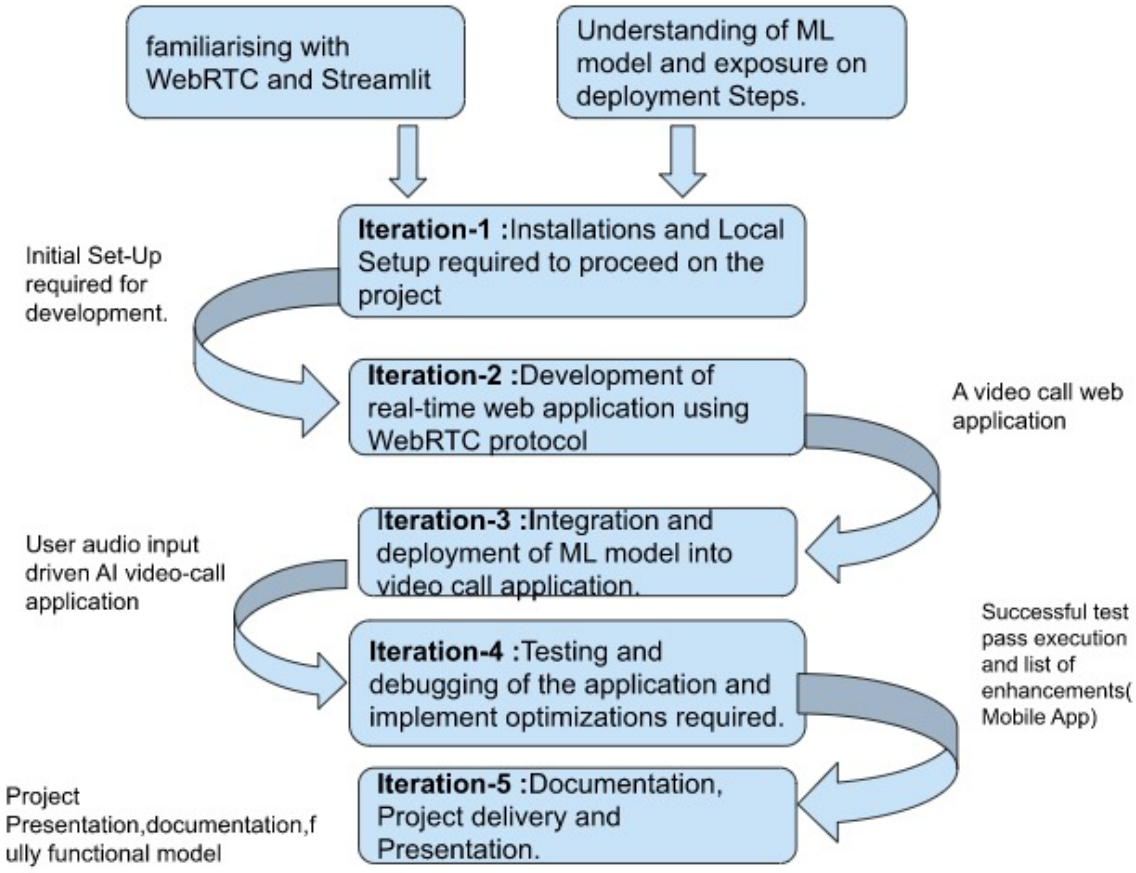
Iteration 4:

- Developed a Real Time Face Swap application using the media pipe model.
- Testing the application end-to-end and checking the performance/latency of the application.
- Improve the code to optimize and improve latency if needed.

Iteration 5:

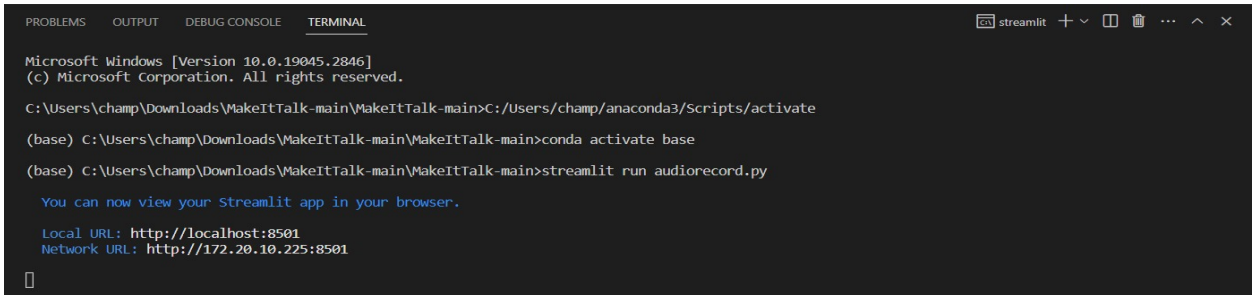
- Perform final tests to ensure the video transmission is in place and work on fixing the bugs if any.
- Focus will be on the project documentation that includes step-by-step instructions to execute the project and presentations that include test-case

results.

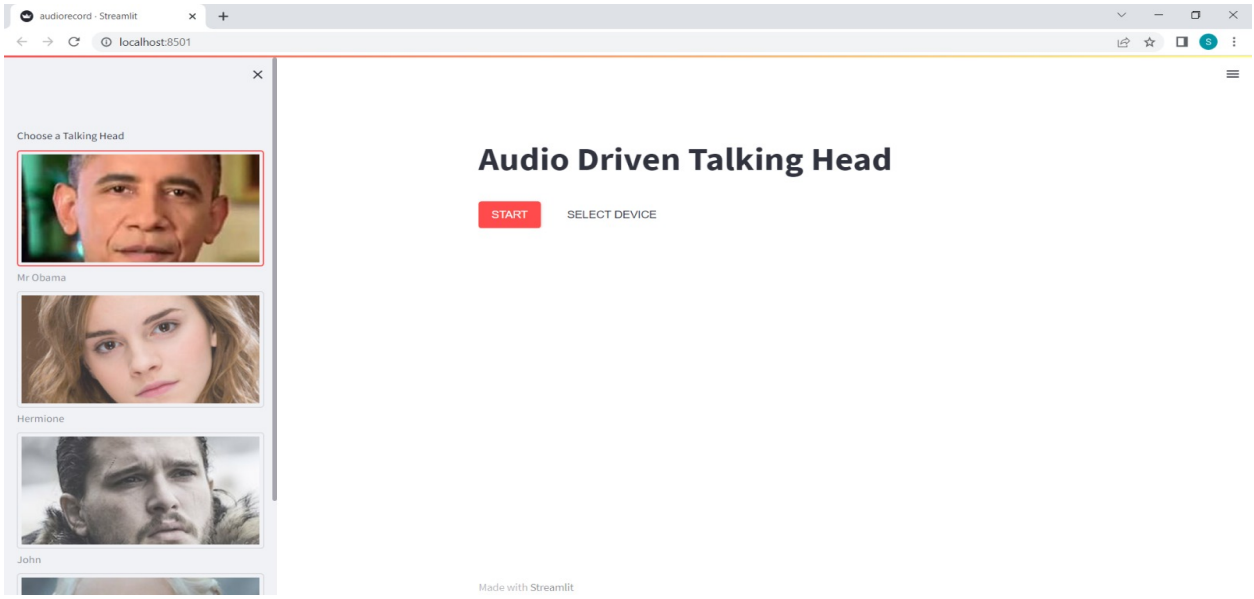


Output Screenshots:

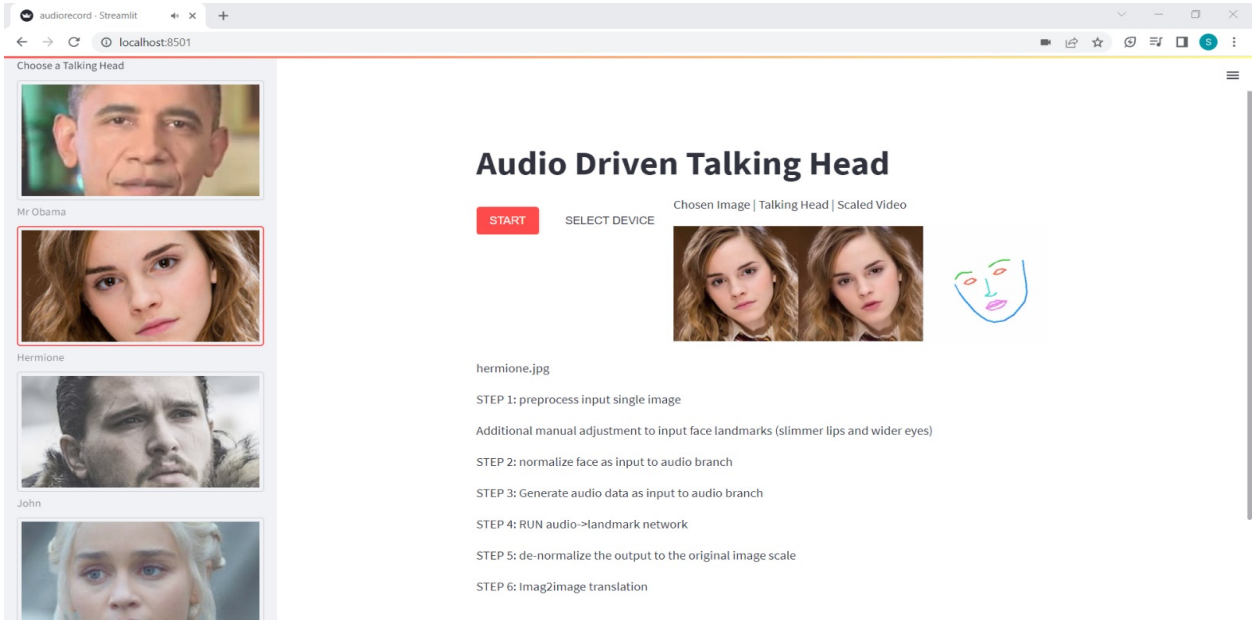
Running the Code:



Recording audio and selecting an image for Talking Head:

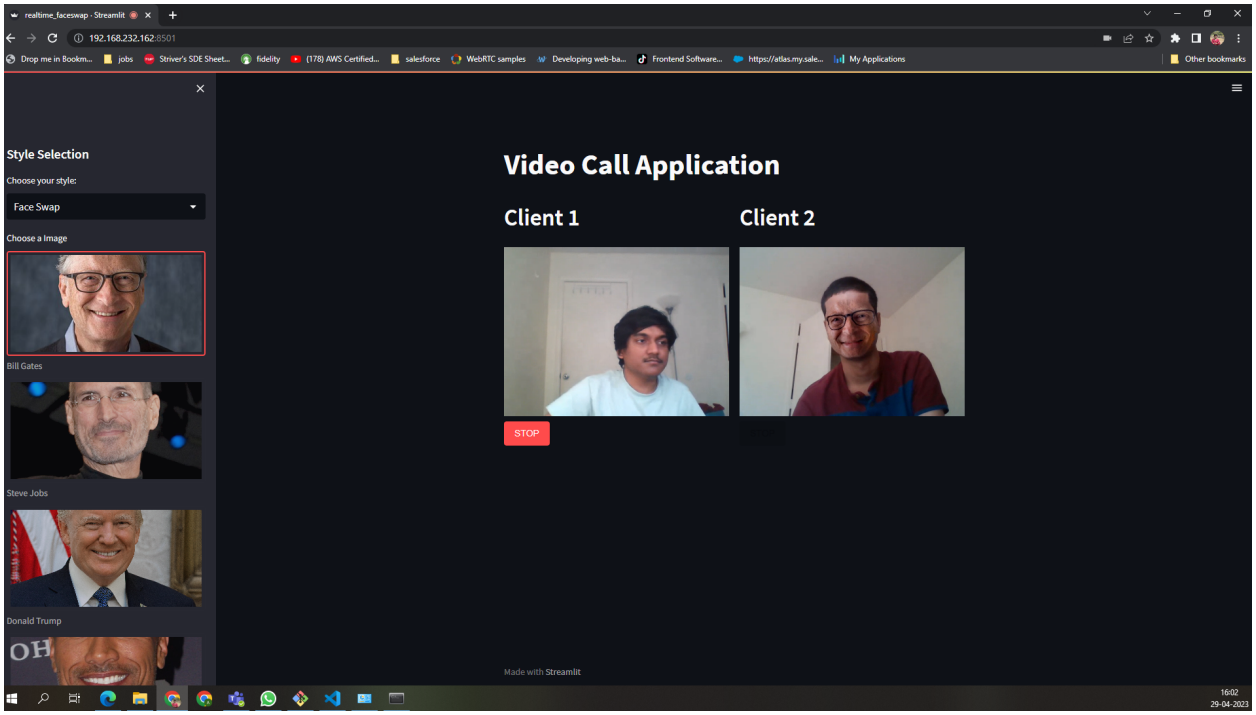


Output a Talking Head Video for the Recorded Audio and Selected Image:

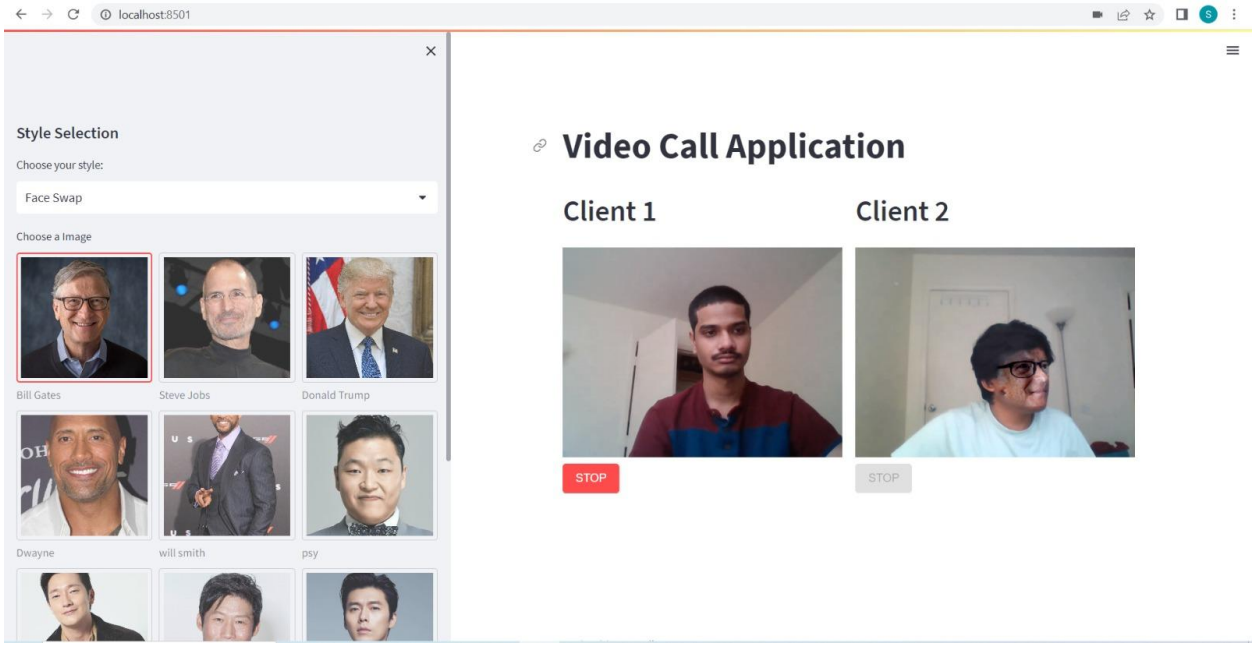


Real Time Face Swap:

Client1:



Client2:



Challenges faced:

The first challenge we faced was related to video call transmission on both clients during the second iteration. We tried out different options with webrtc streamers, stun servers, and ICE candidates to resolve the issue. However, it was not easy to pinpoint the exact problem and it required trying all the possible options to find a solution. Finally, we used Streamlit-server-state to create a separate session and establish a connection between the clients, which helped us to overcome the issue.

The second challenge occurred to us while trying to run the model.This was encountered because the model is developed to run on GPU using CUDA, which uses the GPUs for general-purpose processing. However,we encountered issues with the CUDA configuration, which prevented the model from running on the GPU.To address this issue, we decided to switch to running the model on a CPU instead of a GPU. While running the model on a CPU may be slower than running it on a GPU, it is still a viable option that can provide accurate results.This switch of running the model on a CPU required a thorough understanding and research of model code and the limitations of the system.

The third challenge was related to integrating the model with the video call application, which brought many challenges, including input reception and module dependencies. We had to try out multiple Windows versions and other operating systems to locate the file dependency and continue with the development. This indicates that the issue was related to compatibility issues between different libraries and operating systems.There were multiple build failures along the way and we resolved each one step by step with help of official documentation and forums.

The fourth challenge was for real-time transmission that involved integration of the audio stream with the ML model. The MakeITTalk ML model only accepts a complete audio file in .wav format as input, which means that it cannot handle a continuous stream of audio frames. This posed a challenge as we needed to modify the ML model to accept the continuous stream of audio frames in accordance with the input stream.We struggled here a lot as it required expertise in machine

learning as well as the model architecture.To overcome this challenge we considered several factors. Firstly, we tried to implement the sliding window concept, which involves grouping of multiple frames.To try this we have used queued_frame_callback function from the streamlit_webrtc module to group 100 frames together and passed it as a single file to the model. This did not help as the output we got is video with zero seconds length.. Additionally, we tried to explore other models and tried to implement deep fake models for face swap. We have implemented a real time video call application (video_call.py) integrated with a mediapipe face swap model.

Github link for the project:

https://github.com/sbokka1/ITCS6166_Spring23_Group15_Project

Info about Github repo:

- The project is hosted on git and multiple branches are created for each iteration.
- The file IterationX_Progress.pdf will contain all the information about our development status,requirements for running the code files in each iteration along with the output screenshots.
- Requirements.txt has the information about all the pre-required libraries or modules i.e software dependencies required to run the code files in each iteration.
- The folder examples contain all the images used for training the ML model.
- The file audiorecord.py has the initial version of the code integration i.e. video call application along with the ML model.
- The ML model code can be found in the src folder.

References:

- Lele Chen, Zhiheng Li, Ross K Maddox, Zhiyao Duan, and Chenliang Xu. 2018. Lip movements generation at a glance. In Proc. ECCV.

- Sefik Emre Eskimez, Ross K Maddox, Chenliang Xu, and Zhiyao Duan. 2019. NoiseResilient Training Method for Face Landmark Generation From Speech. IEEE/ACM Trans. Audio, Speech, and Language Processing (2019).
- Yilong Liu, Feng Xu, Jinxiang Chai, Xin Tong, Lijuan Wang, and Qiang Huo. 2015. Video-audio driven real-time facial animation. ACM Trans. Graphics (2015).
- https://www.sciencedirect.com/science/article/abs/pii/S0925231222012334
- https://www.researchgate.net/publication/358907100_Review_on_Frameworks_Used_for_Deployment_of_Machine_Learning_Model
- https://github.com/hermes7308/Realtime-Face-Swap

THANK YOU