AI Mock Interview Tool

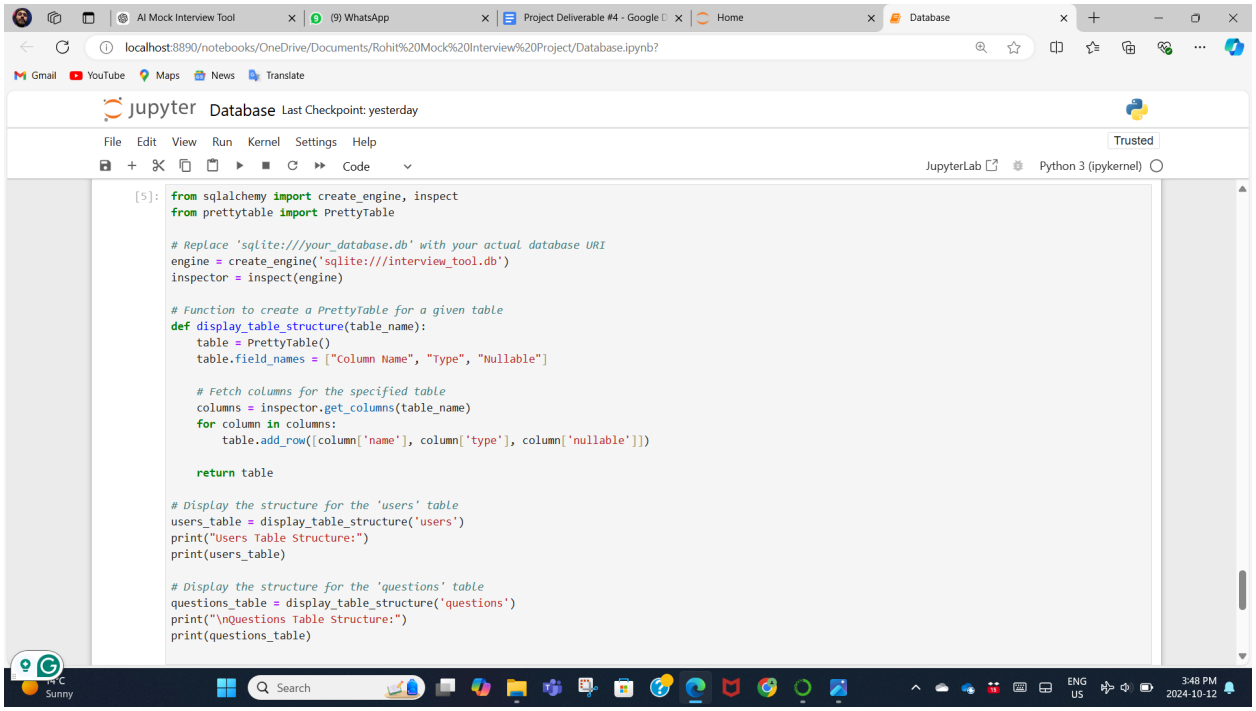
Project Deliverable #4:

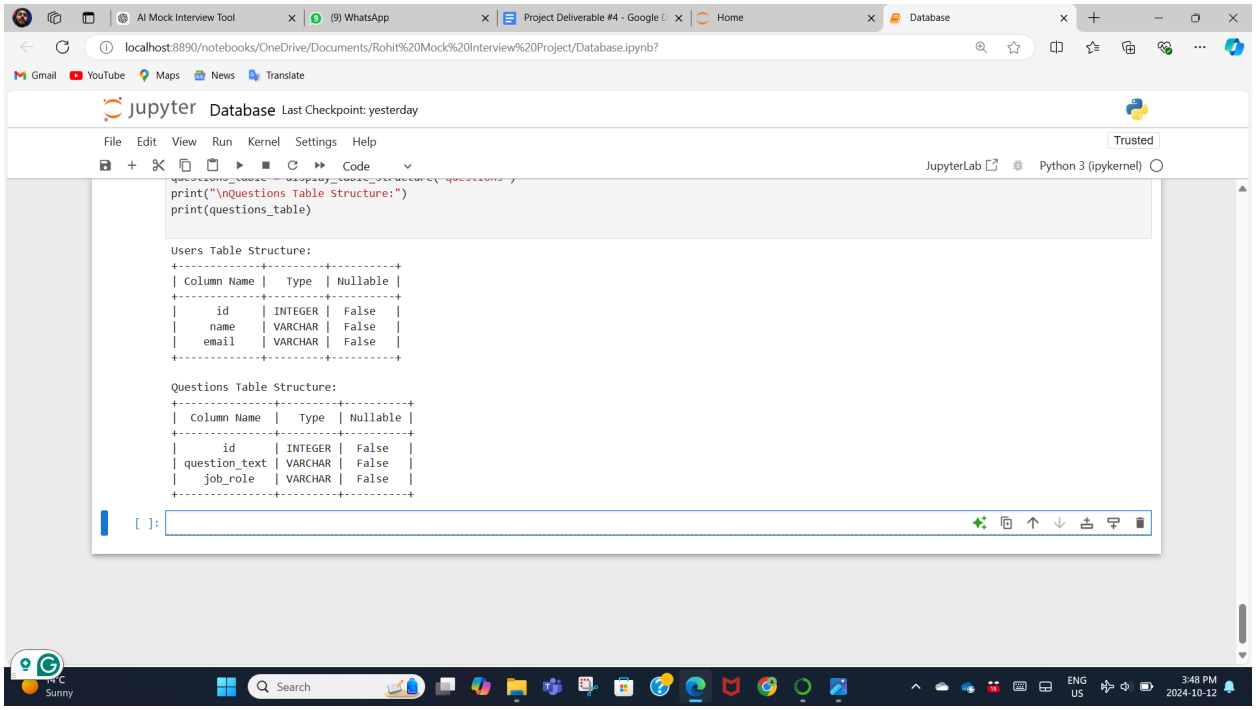
Team 3:

Team member	Deliverables
Rohit	Database Selection and Integration for storing user data
Dilpreet kaur	Filler words detection
Chanthuru Thavarajah	Full app Integration into one cohesive system.
Arunkumar Anugu	Real-time video background optimization
Vasu Bejugam	Live response dialogue generation & Real-time video background check
Arshdeep Kaur	Analyzes the user's spoken English for grammatical errors
Dhvani Bhavani	Live video Face expressions
Varunan Gurushev	External audio background check
Jaskaran Singh	Eye contact analysis

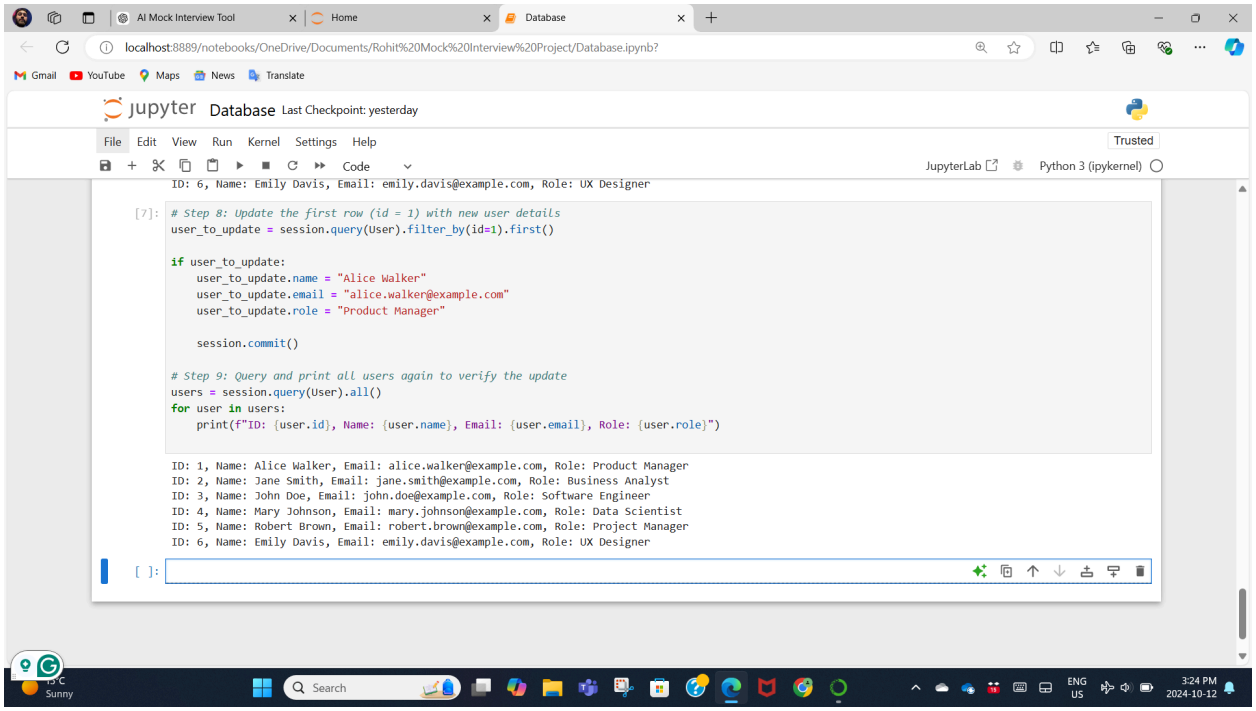
Rohit:

I am working on database integration for the AI Mock Interview Tool to efficiently store and manage interview questions and user data. I chose SQLAlchemy with SQLite for simplicity, ease of use, and no additional cost.

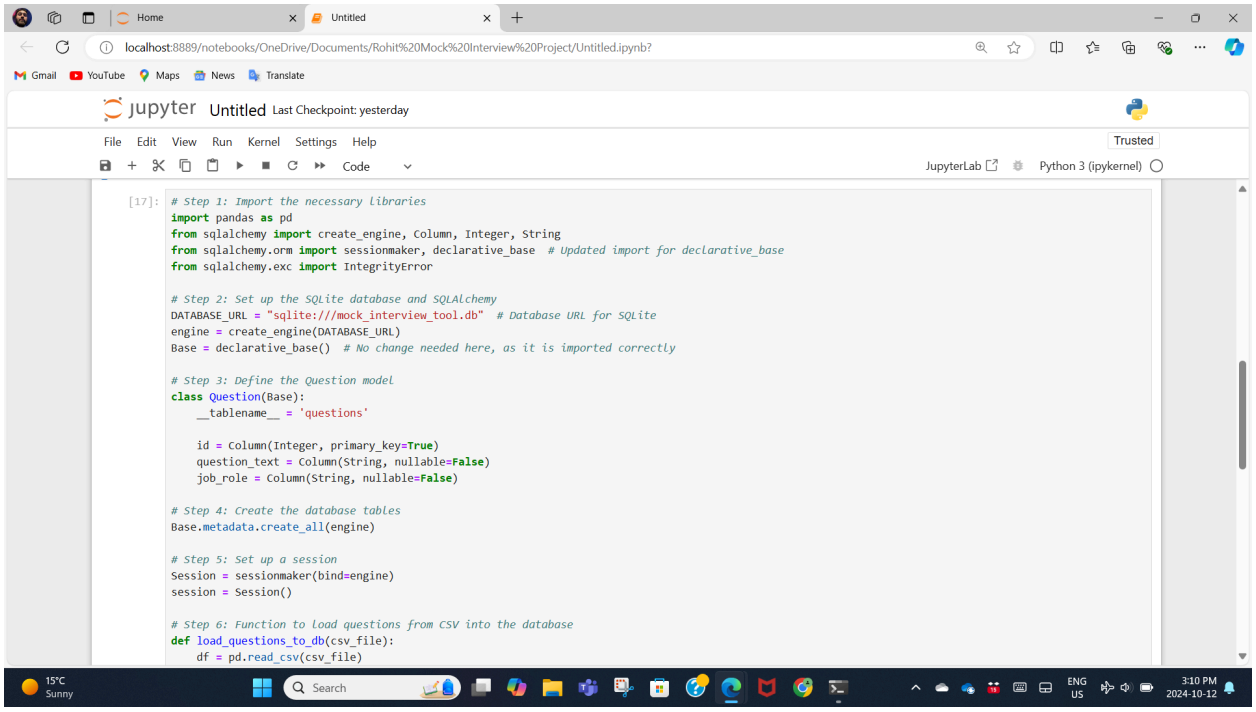


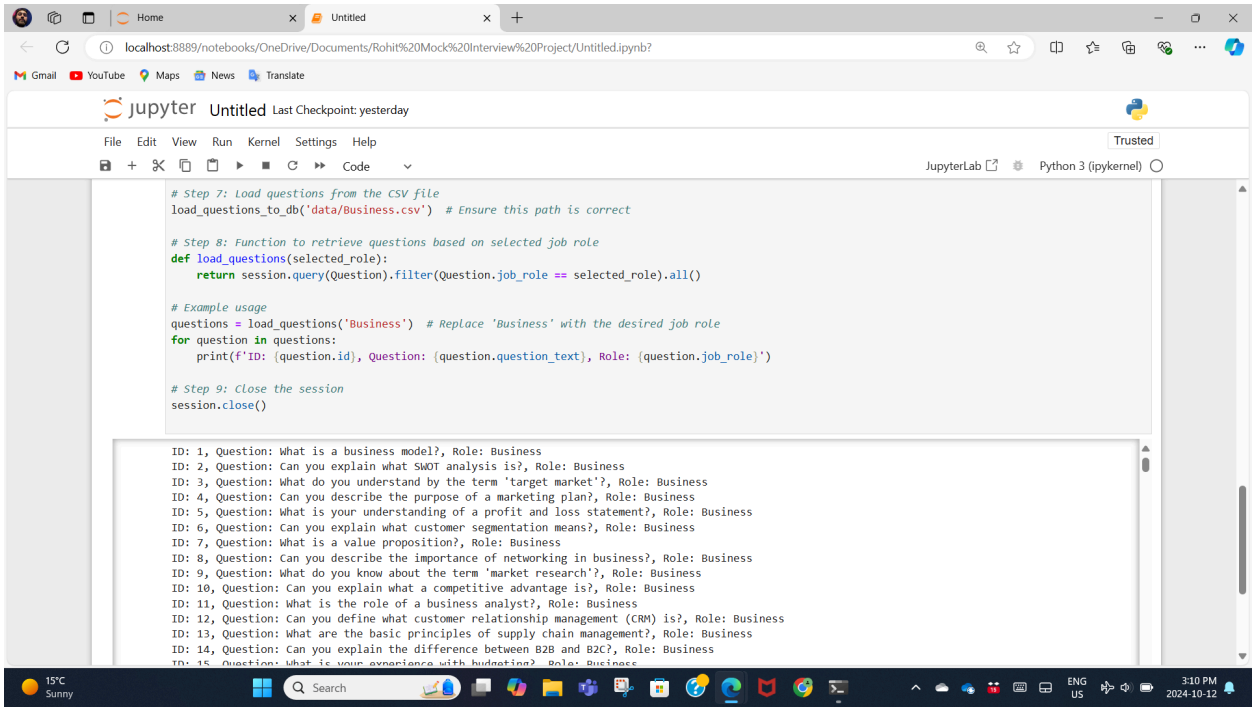


I created a User class to manage user information, including name, email, and role. This class facilitates seamless integration and storage of user data within the database.



I created a question class that connects to the CSV file to fetch interview questions based on different job roles.

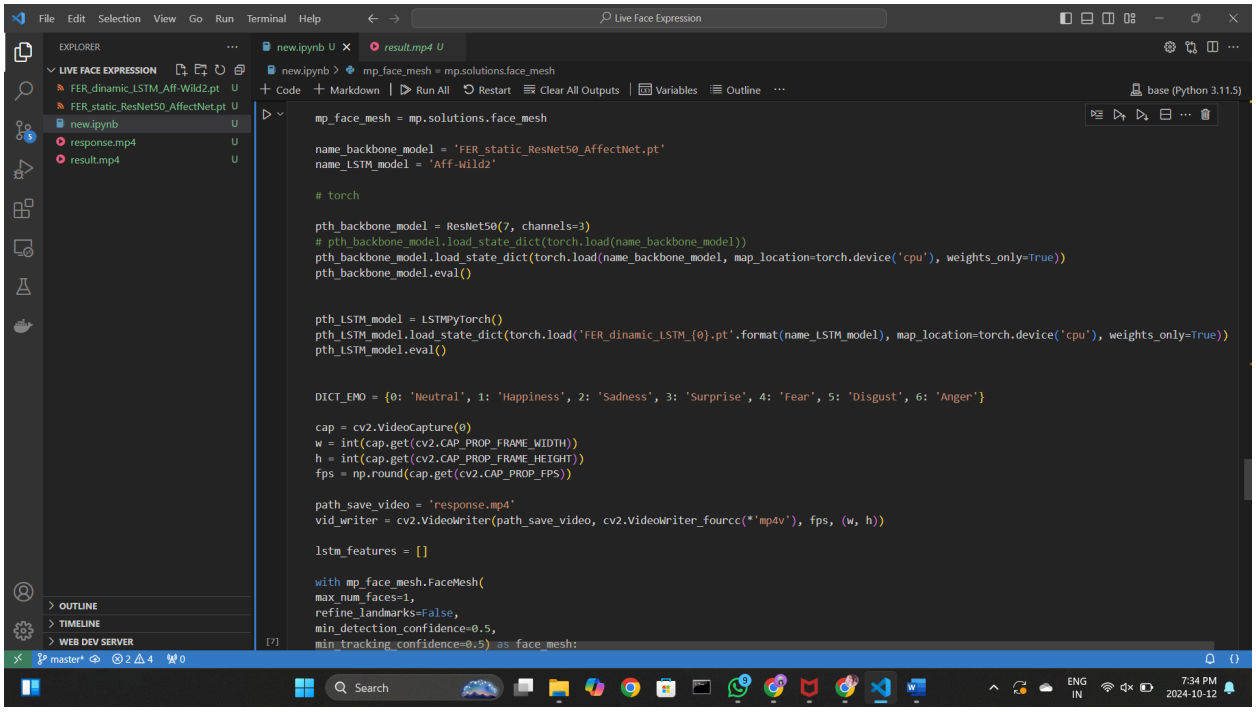


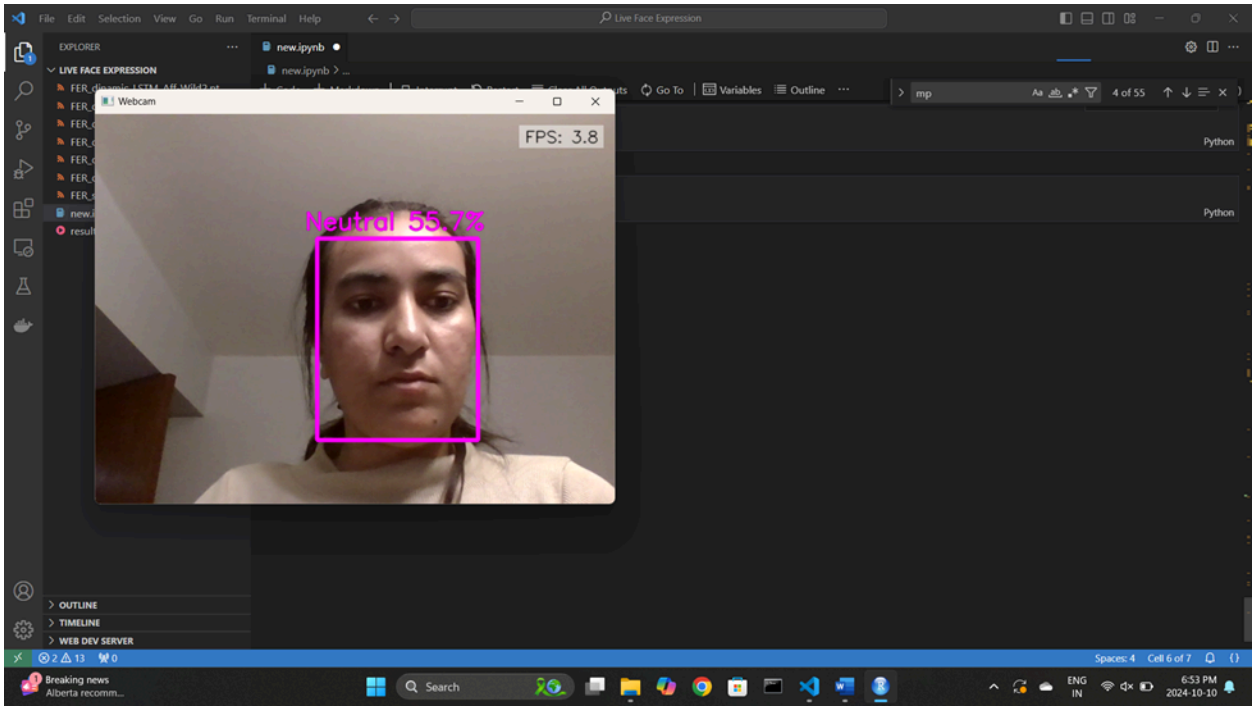


Dhvani Bhavani: Live video Face expressions

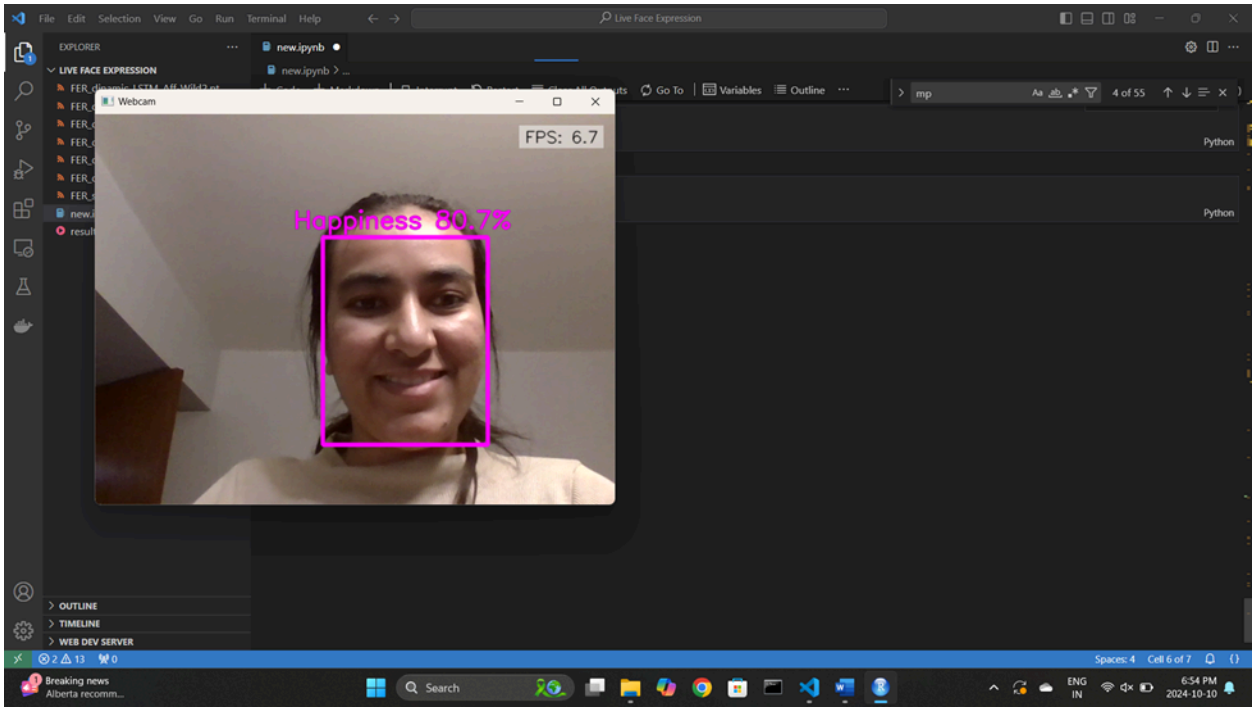
Emo-AffectNet model is designed for robust and real-time facial emotion recognition, whether from single images or live videos, making it versatile for various AI-driven emotional analysis applications.

Emotions can be `'Neutral'`, `'Happiness'`, `'Sadness'`, `'Surprise'`, `'Fear'`, `'Disgust'`, `'Anger'`

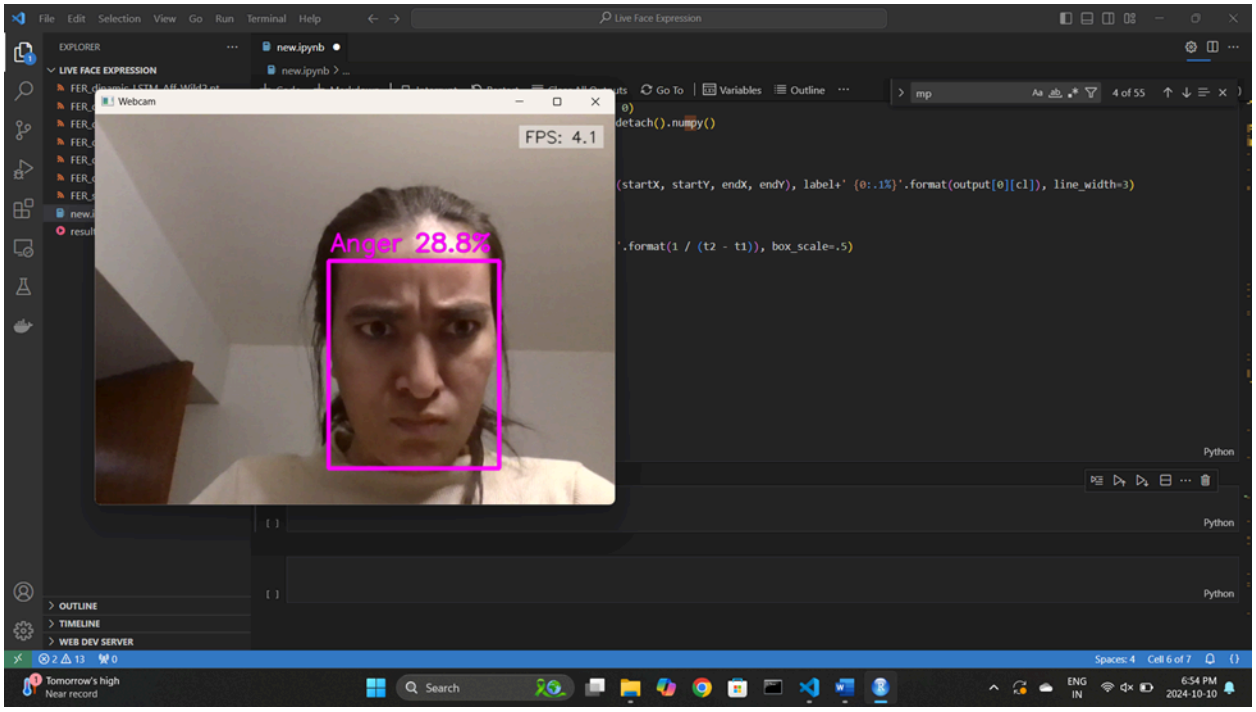




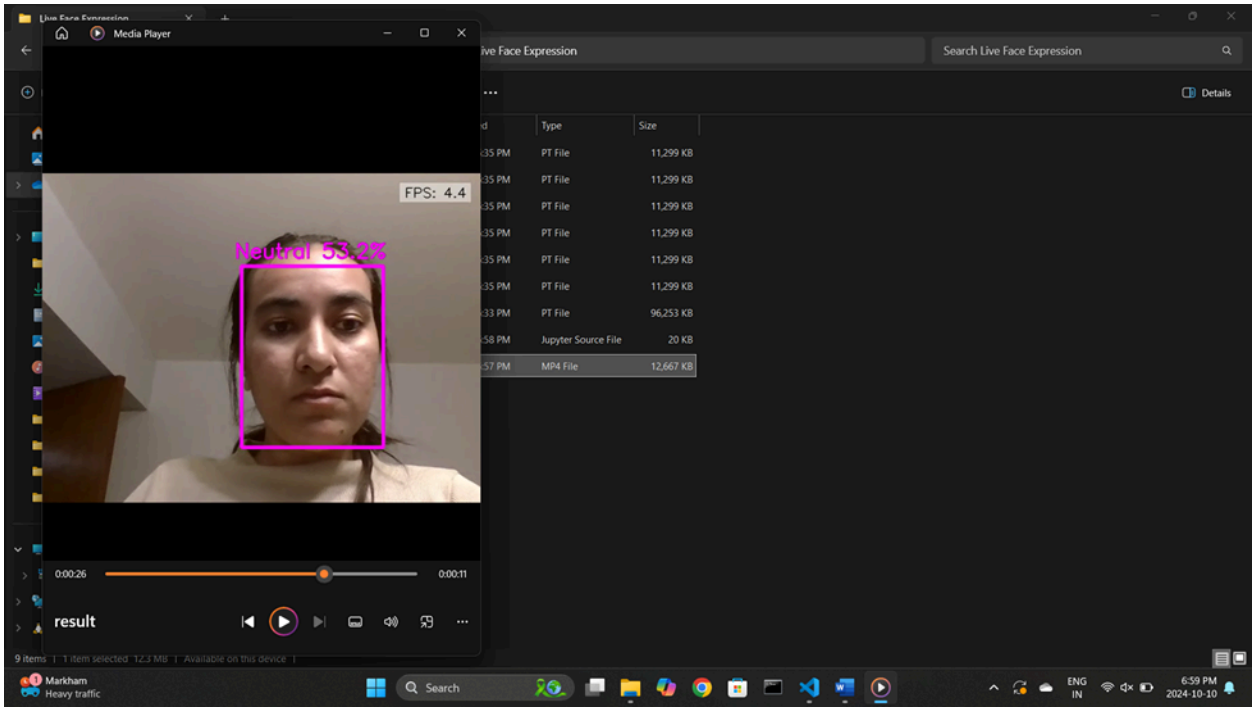
Happy:



Anger:

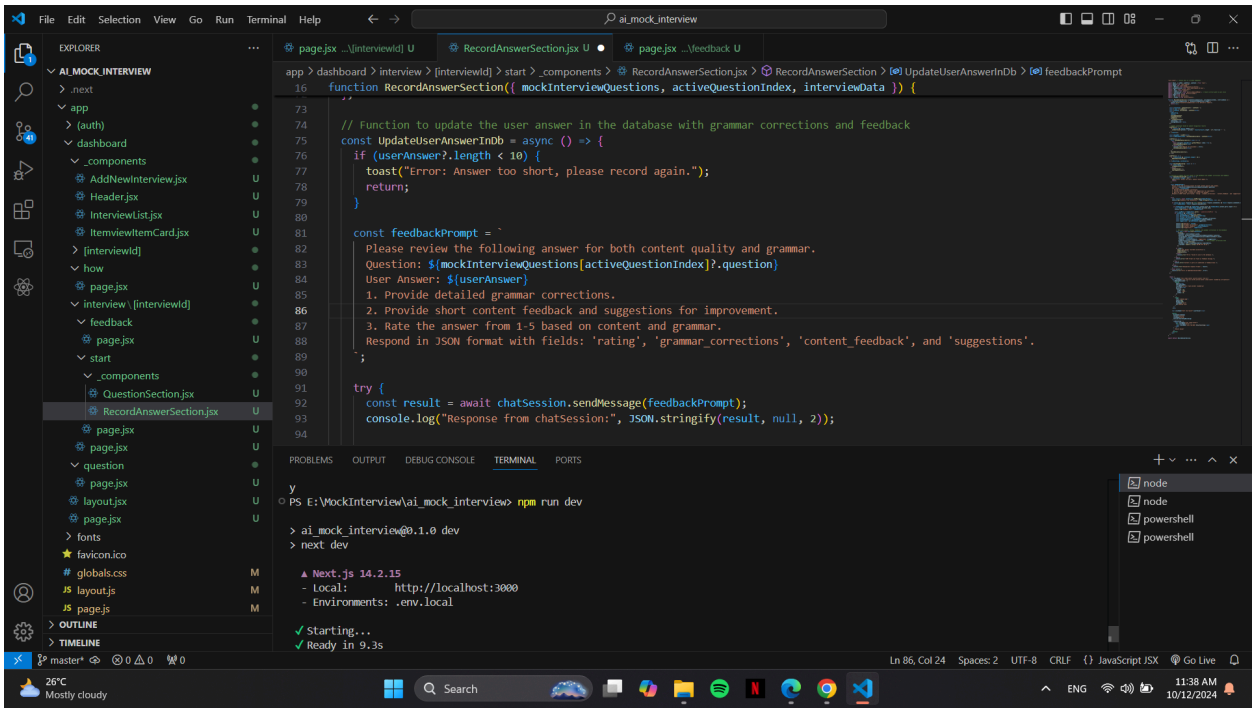


Downloaded/Saved video for response analysis.

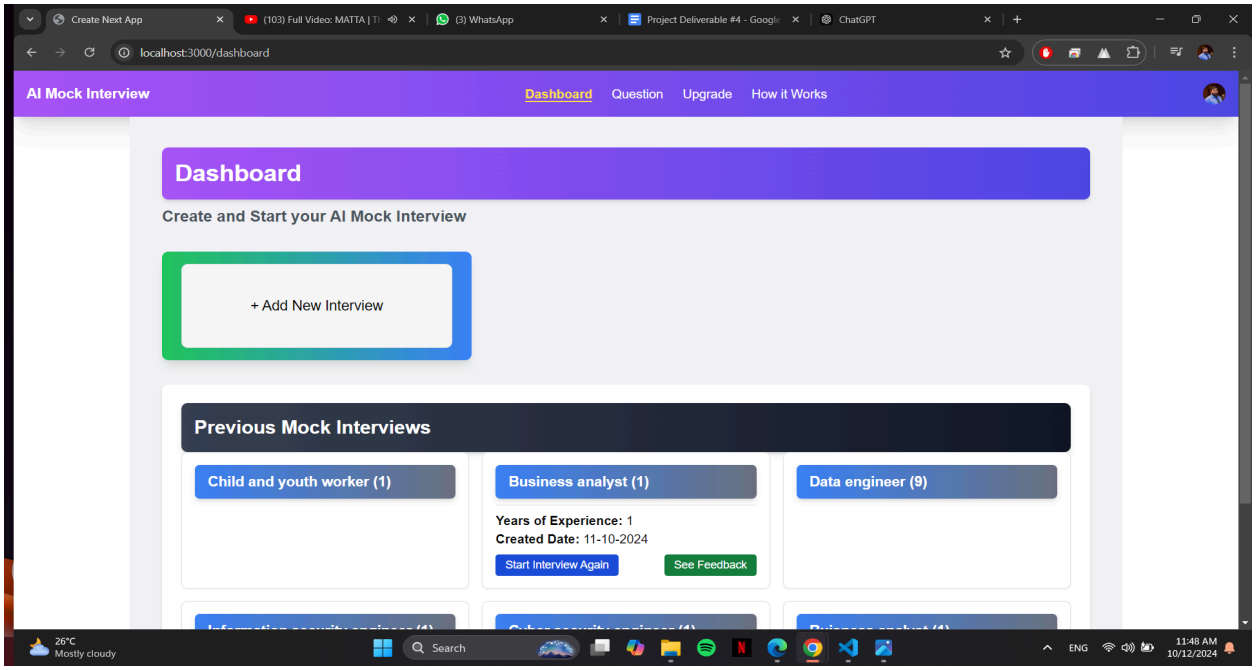


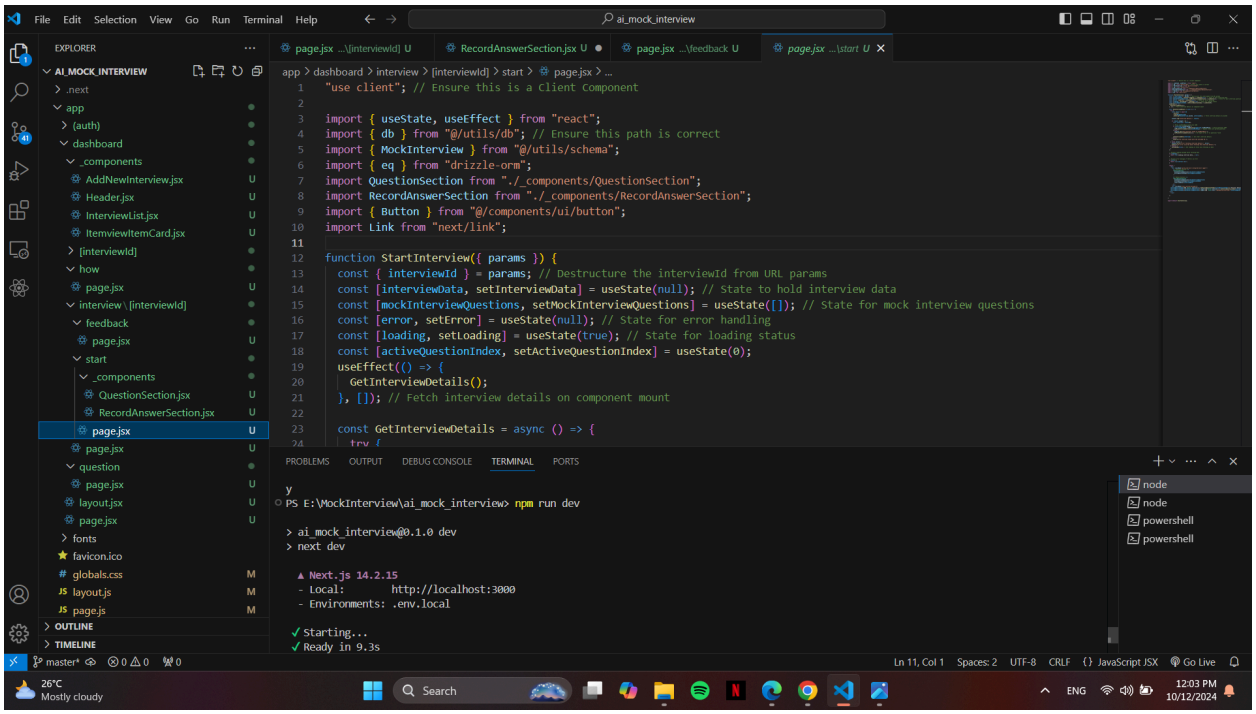
Chanthuru Thavarajah:

Combining all the small models we built together



I developed a user-friendly dashboard that allows users to initiate a new interview and review their past interview history. The dashboard provides the option to either retake a previous interview or view detailed feedback from completed interviews. All relevant data is efficiently retrieved and displayed from our database, ensuring a seamless experience for users managing their interview activities.





I have developed the **New Interview Setup** functionality in our AI-powered mock interview tool, which allows users to create a personalized interview experience.

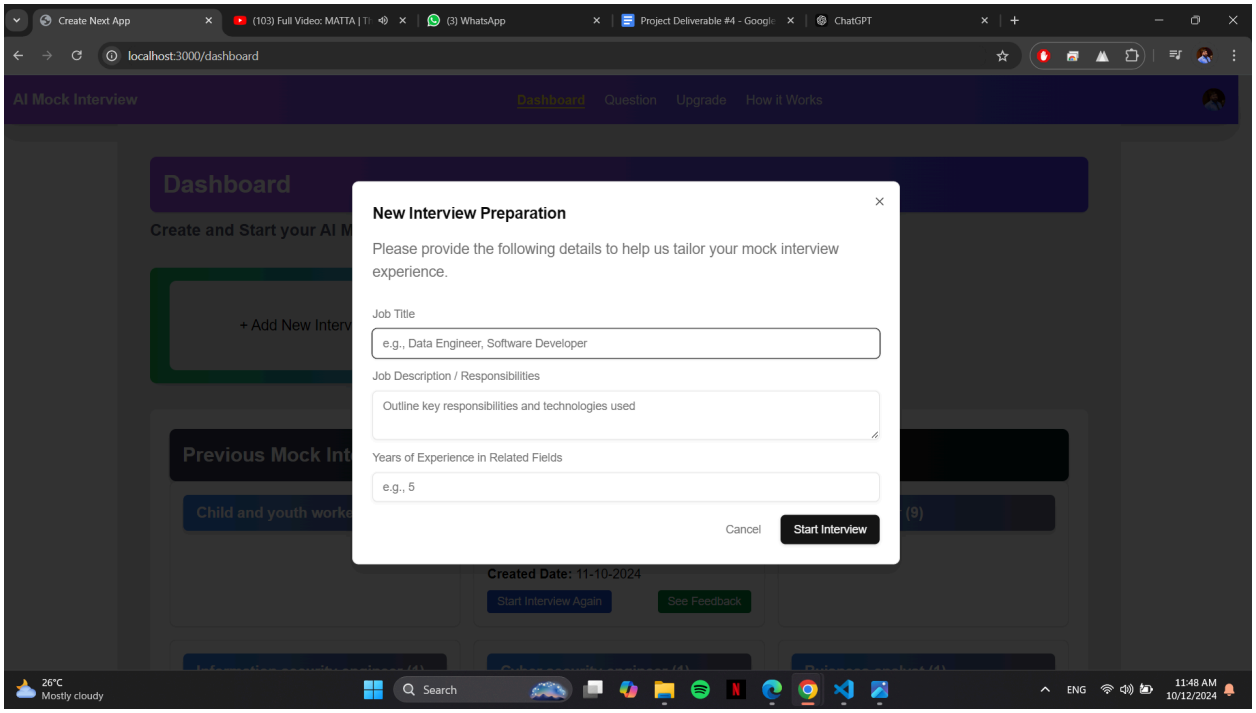
When creating a new interview, users are prompted to fill in the following fields:

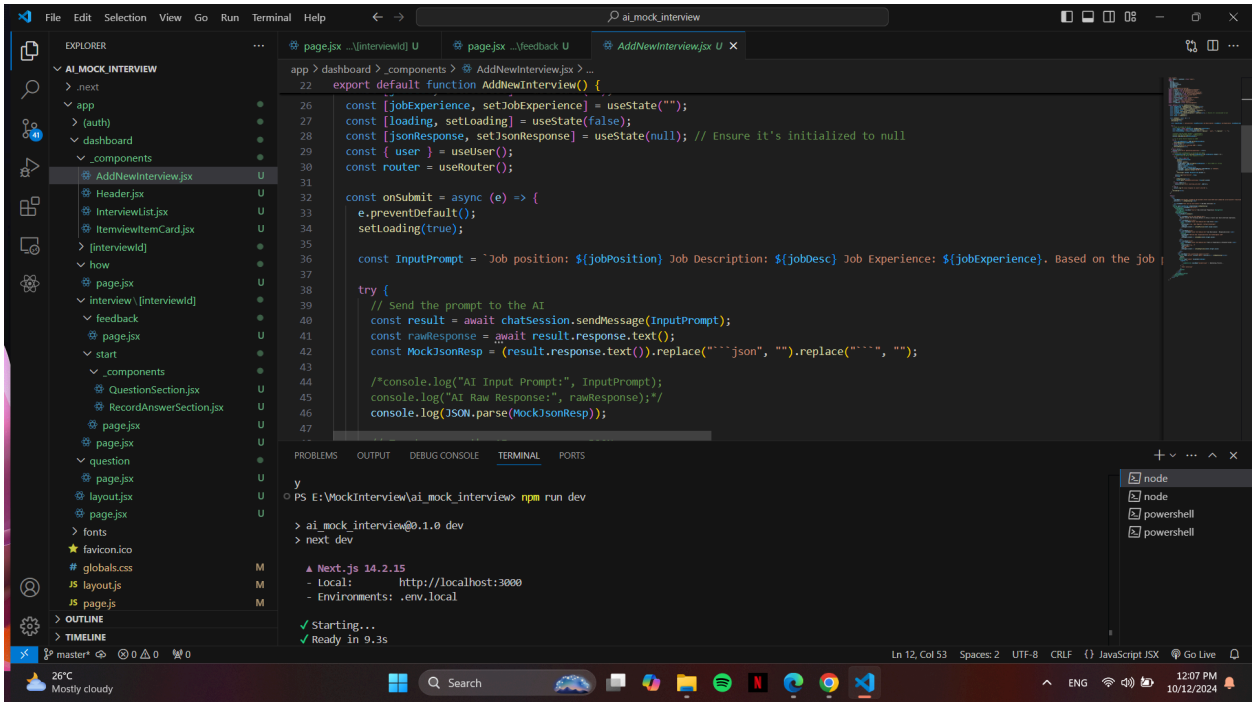
- **Job Title:** Users enter a specific job title, such as "Data Engineer" or "Software Developer."

- **Job Description / Key Responsibilities:** This section allows users to provide a brief overview of the role and its key responsibilities, along with any relevant technologies they may work with.
- **Years of Experience:** Users specify how many years of experience they have in related fields.

Once the user completes these fields, they can proceed to create a tailored mock interview based on the provided information. I'll attach a screenshot of the setup interface for further clarification.

This functionality ensures that the mock interviews are highly relevant to the user's target job role and experience level.

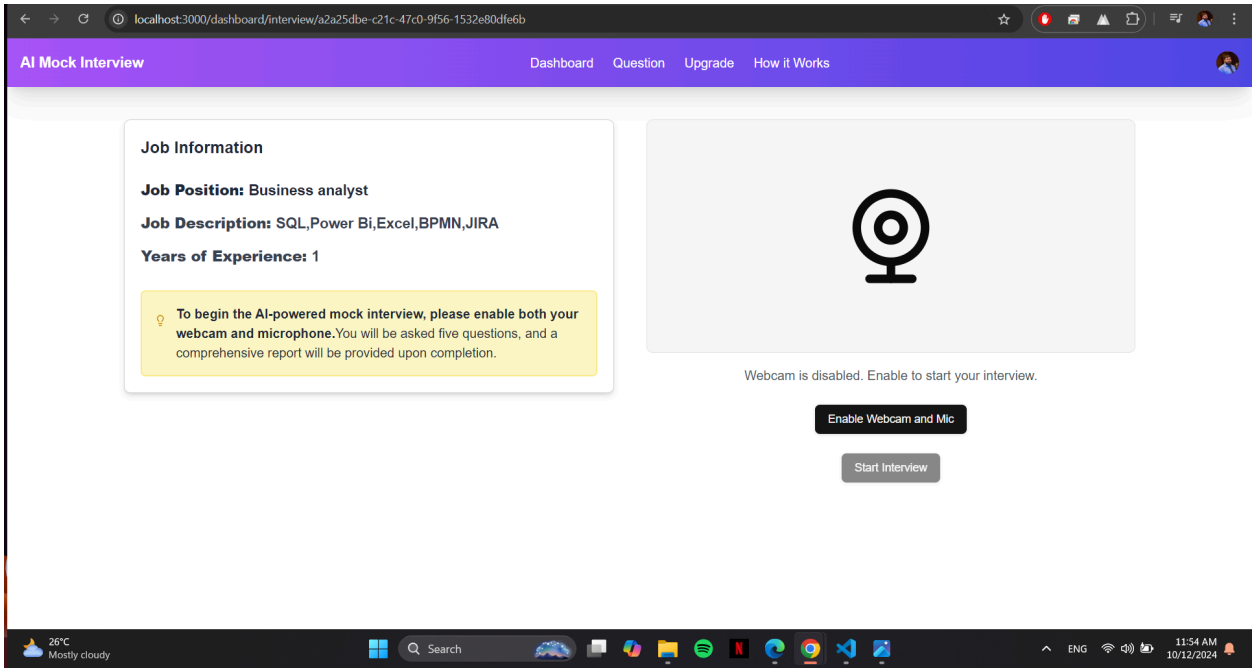


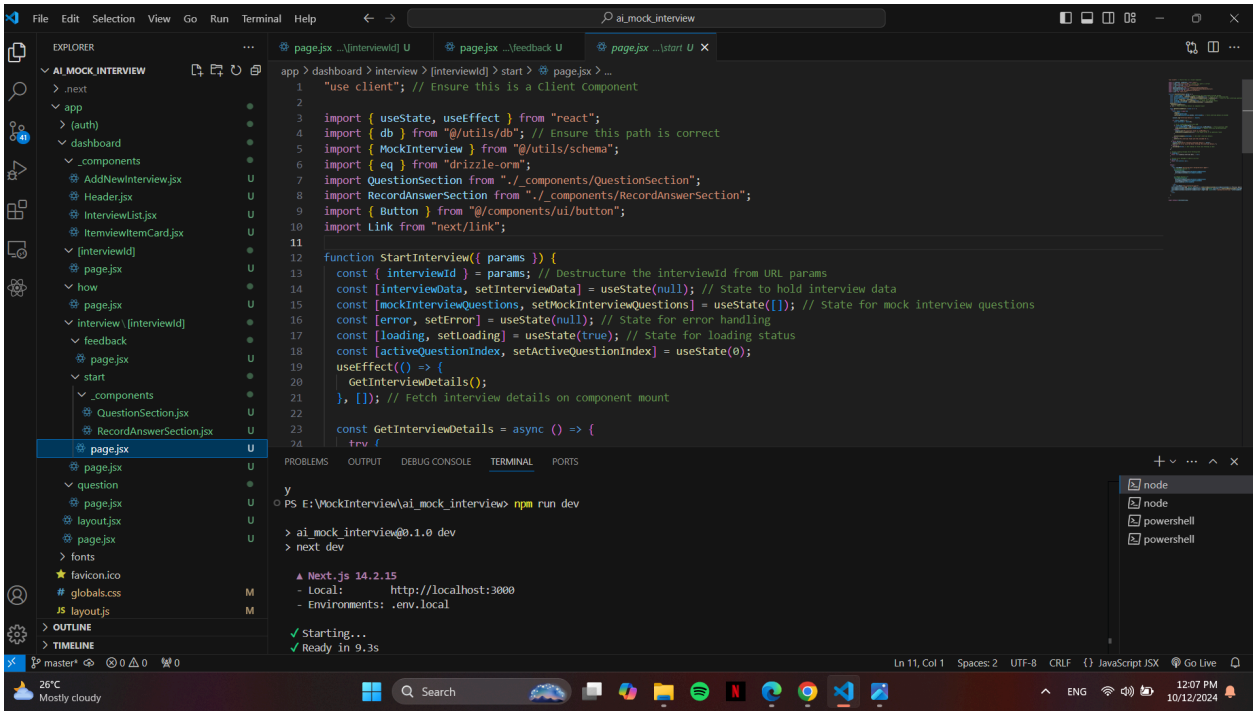


next step in the process after the user provides the job information.

As seen in the attached screenshot, once the user has entered the job position, job description, and years of experience, the page displays a confirmation of the provided job details under "Job Information."

To proceed with the AI-powered mock interview, the user is prompted to enable both the webcam and microphone. After enabling them, they can click "Start Interview" to begin the interview process, during which they will be asked five questions. A comprehensive report based on the interview will be generated once it’s completed. This ensures the user is fully prepared for the mock interview based on their specific job role.





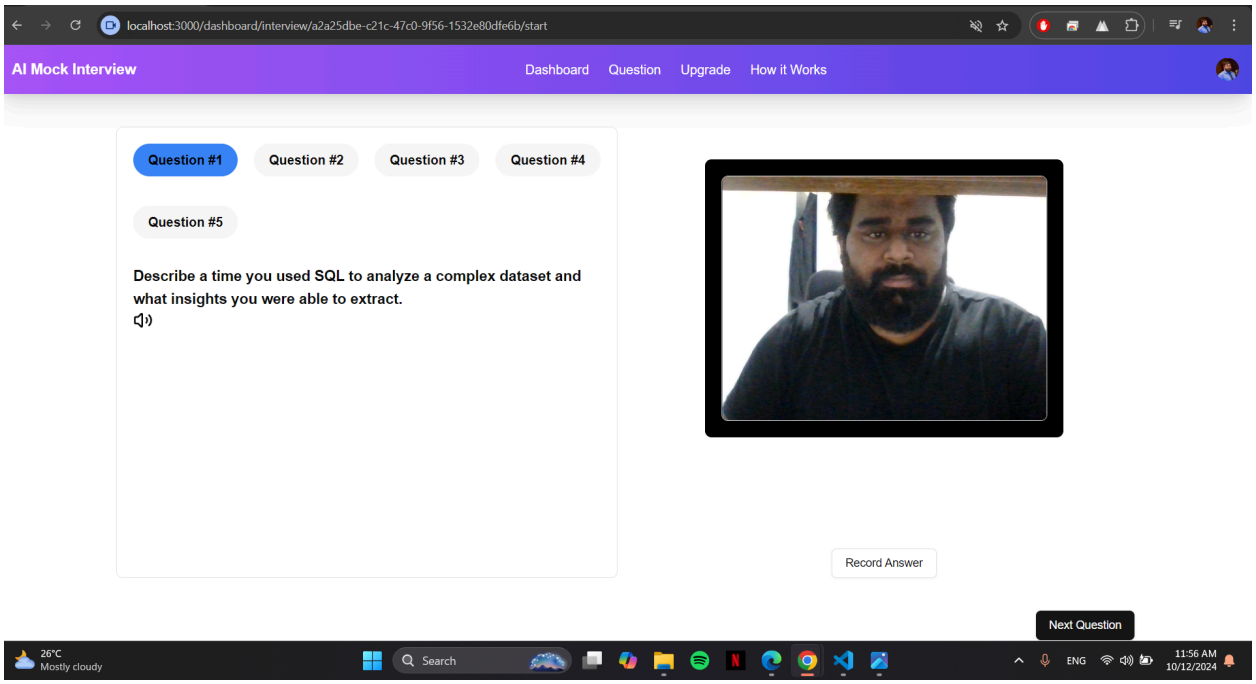
the next screen of the mock interview interface.

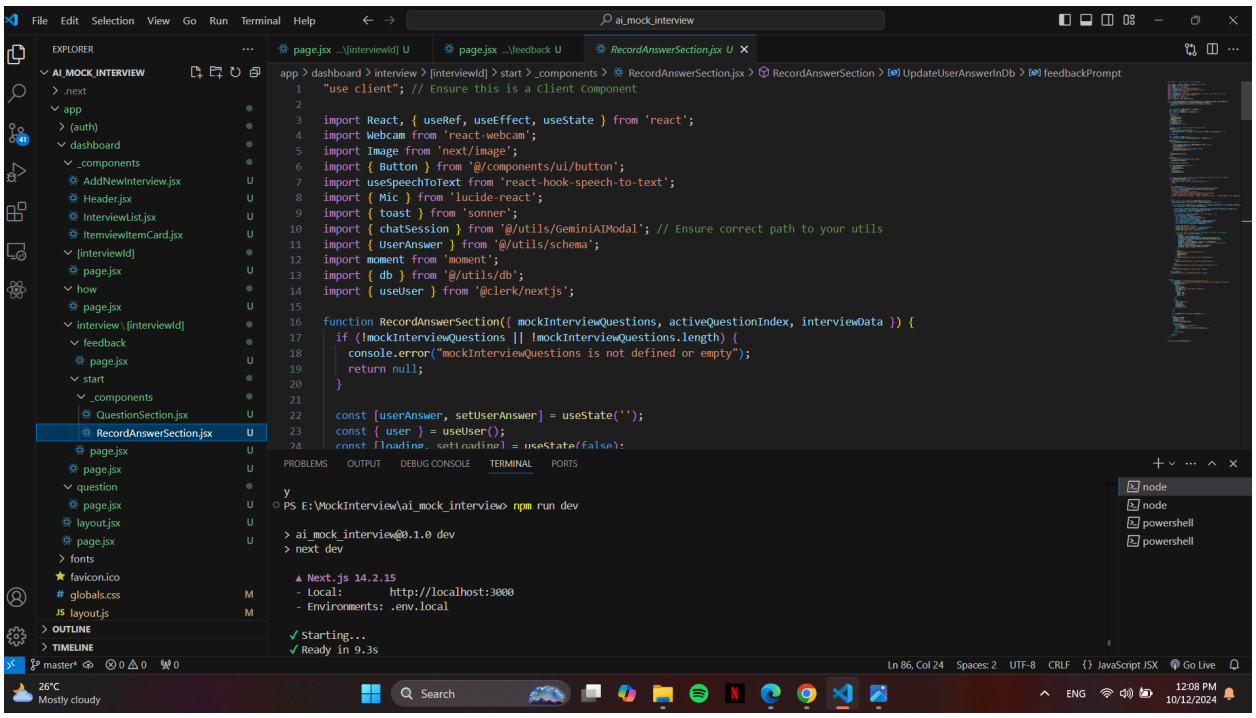
Once the user starts the interview, the system presents the first of five preloaded questions. In this example, the user is prompted with the question:

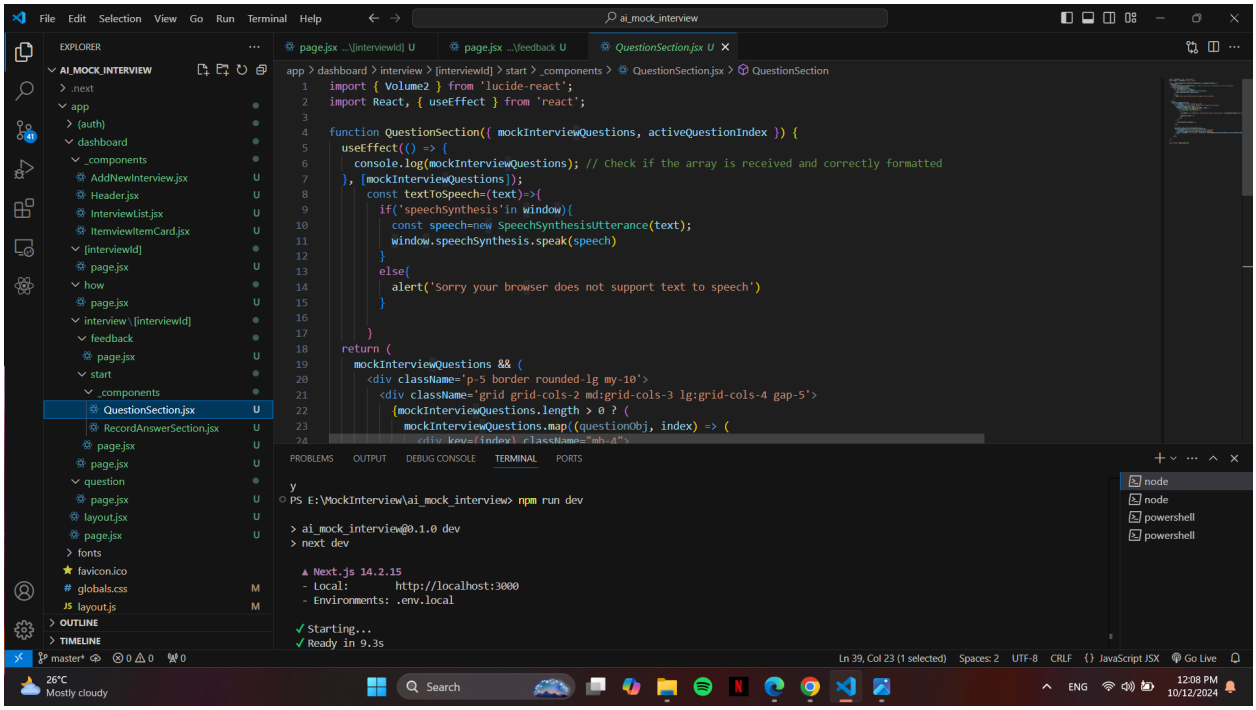
"Describe a time you used SQL to analyze a complex dataset and what insights you were able to extract."

The webcam is active, capturing the user's video response. The user has the option to record their answer by clicking on the **Record Answer** button, followed by the **Next Question** button to proceed to the subsequent question.

This interface ensures that the interview process mimics a real-life scenario, allowing users to respond to each question while being recorded, enabling feedback based on their answers and video interaction.







After the user completes all interview questions, the system provides feedback on their performance. In the screenshot, the overall interview rating is **1.4/5**, indicating areas for improvement.

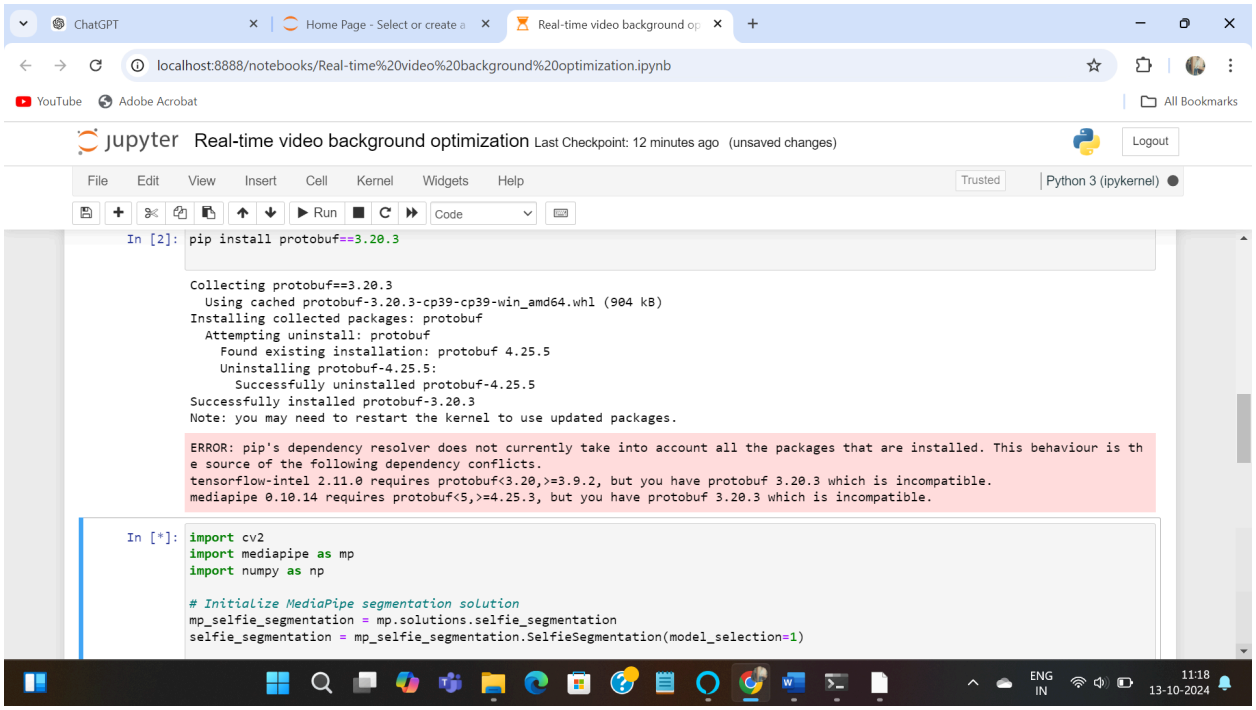
The feedback section displays each question from the mock interview along with the user's recorded answers and specific feedback for improvement, though the answers are currently not shown in the image. This allows users to review their responses to each question, such as:

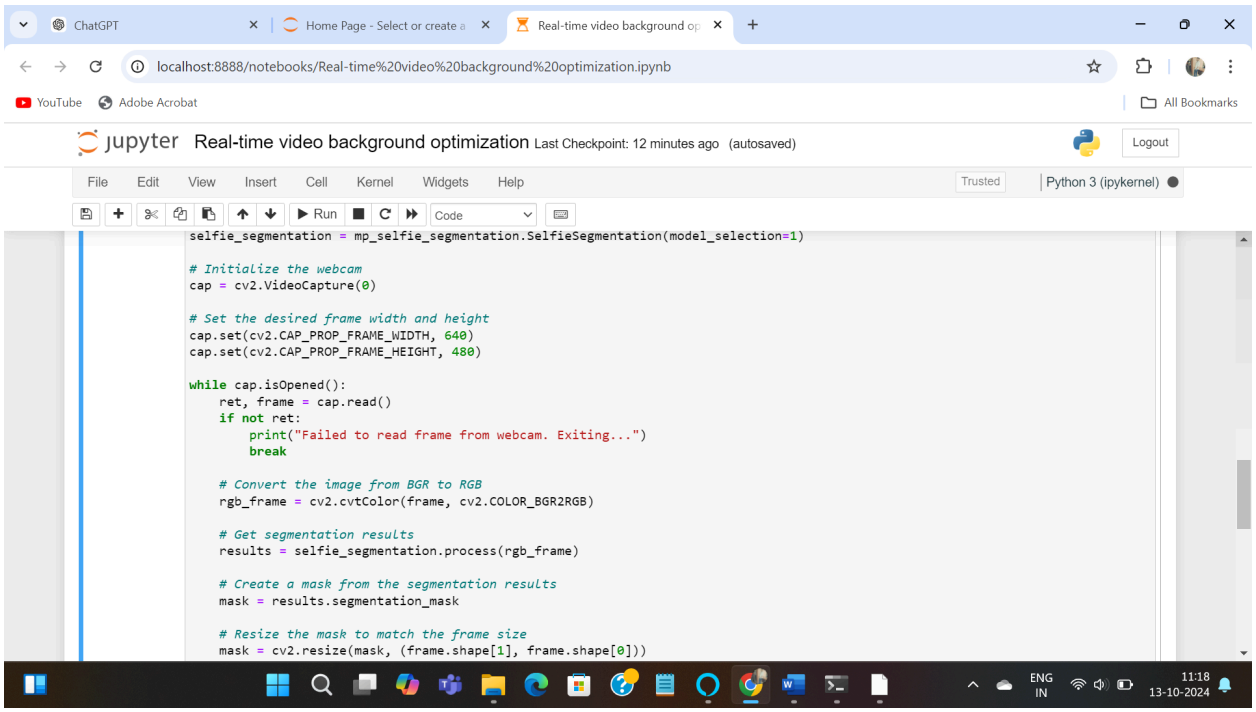
- **Question 1:** "Describe a time you used SQL to analyze a complex dataset and what insights you were able to extract."
- **Question 2:** "Explain the concept of BPMN and how you would use it to model a business process."
- **Question 3:** "Describe your experience using Jira and how you have used it to manage projects and track progress."
- **Question 4:** "You are given a large Excel spreadsheet with data from multiple departments. How would you clean and prepare this data for analysis?"

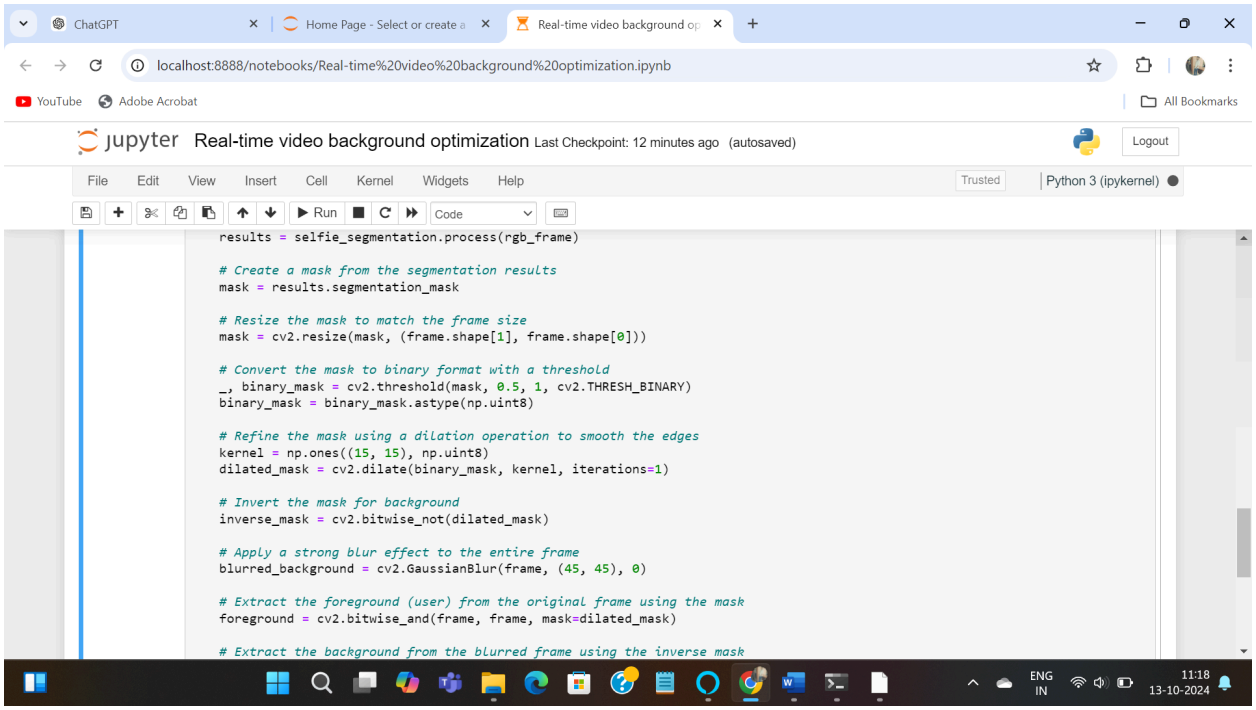
This structured feedback helps users understand their strengths and areas for improvement in specific topics related to the job they are interviewing for, enhancing their preparation for real interviews.

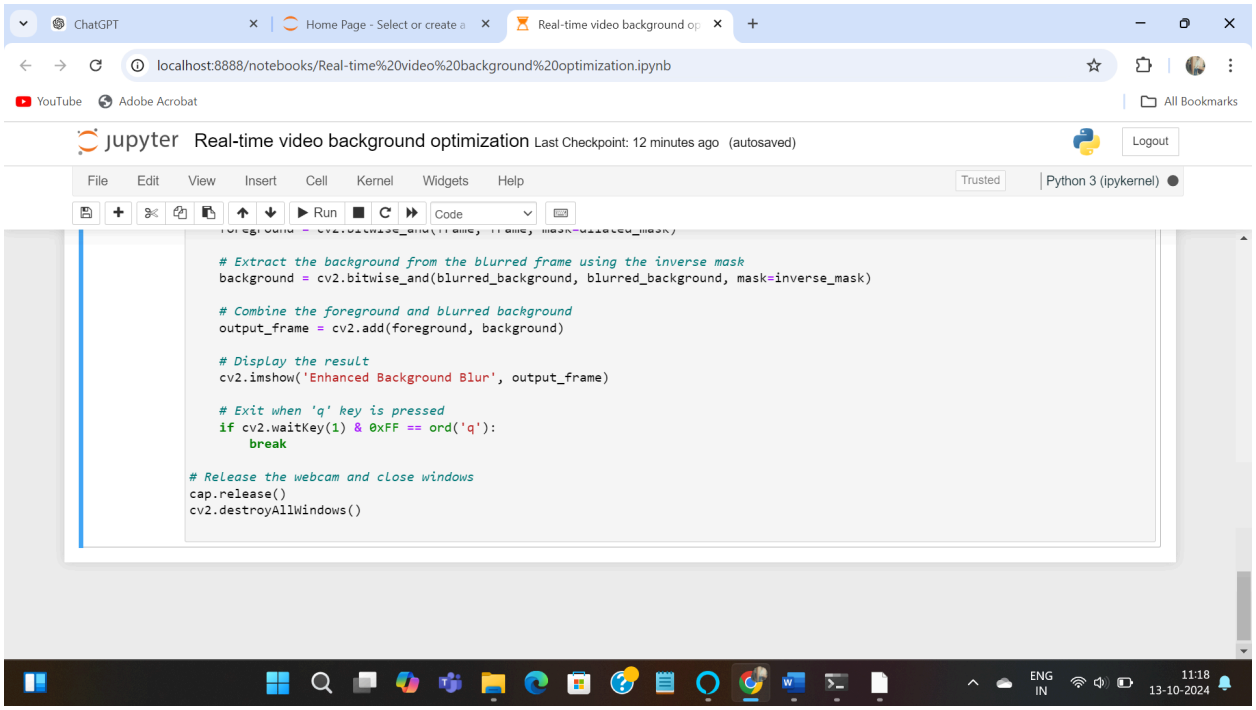
Arunkumar Anugu :

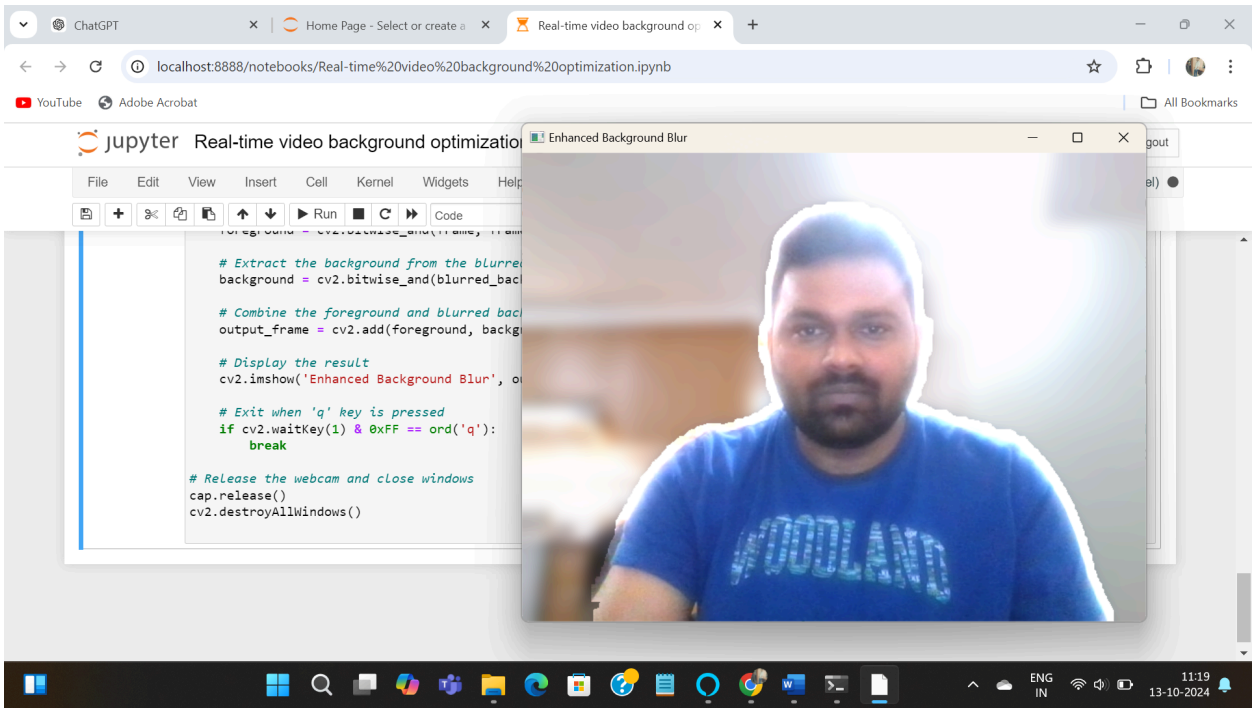
The code uses OpenCV and MediaPipe to access the webcam and capture live video. MediaPipe’s SelfieSegmentation creates a mask to separate the user from the background, which is refined for smooth edges. The background is blurred using a Gaussian filter, and the foreground (user) is combined with the blurred background for optimization. The video feed displays the result in real-time until 'q' is pressed, and resources are released afterward.


ChatGPT
Home Page - Select or create a
Real-time video background op
localhost:8888/notebooks/Real-time%20video%20background%20optimization.ipynb
YouTube
Adobe Acrobat
All Bookmarks
jupyter Real-time video background optimization Last Checkpoint: 12 minutes ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [2]: pip install protobuf==3.20.3
Collecting protobuf==3.20.3
Using cached protobuf-3.20.3-cp39-cp39-win_amd64.whl (904 kB)
Installing collected packages: protobuf
Attempting uninstall: protobuf
Found existing installation: protobuf 4.25.5
Uninstalling protobuf-4.25.5:
Successfully uninstalled protobuf-4.25.5
Successfully installed protobuf-3.20.3
Note: you may need to restart the kernel to use updated packages.
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is th
e source of the following dependency conflicts.
tensorflow-intel 2.11.0 requires protobuf<3.20,>=3.9.2, but you have protobuf 3.20.3 which is incompatible.
mediapipe 0.10.14 requires protobuf<5,>=4.25.3, but you have protobuf 3.20.3 which is incompatible.
In [*]: import cv2
import mediapipe as mp
import numpy as np
# Initialize MediaPipe segmentation solution
mp_selfie_segmentation = mp.solutions.selfie_segmentation
selfie_segmentation = mp_selfie_segmentation.SelfieSegmentation(model_selection=1)
ENG
IN
11:18
13-10-2024


ChatGPT
Home Page - Select or create a
Real-time video background op
localhost:8888/notebooks/Real-time%20video%20background%20optimization.ipynb
YouTube
Adobe Acrobat
All Bookmarks
jupyter Real-time video background optimization Last Checkpoint: 12 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
selfie_segmentation = mp_selfie_segmentation.SelfieSegmentation(model_selection=1)
# Initialize the webcam
cap = cv2.VideoCapture(0)
# Set the desired frame width and height
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
while cap.isOpened():
ret, frame = cap.read()
if not ret:
print("Failed to read frame from webcam. Exiting...")
break
# Convert the image from BGR to RGB
rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
# Get segmentation results
results = selfie_segmentation.process(rgb_frame)
# Create a mask from the segmentation results
mask = results.segmentation_mask
# Resize the mask to match the frame size
mask = cv2.resize(mask, (frame.shape[1], frame.shape[0]))
ENG
IN
11:18
13-10-2024


ChatGPT
Home Page - Select or create a
Real-time video background op
localhost:8888/notebooks/Real-time%20video%20background%20optimization.ipynb
YouTube
Adobe Acrobat
All Bookmarks
jupyter
Real-time video background optimization
Last Checkpoint: 12 minutes ago
(autosaved)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ipykernel)
Run
Code
results = selfie_segmentation.process(rgb_frame)
# Create a mask from the segmentation results
mask = results.segmentation_mask
# Resize the mask to match the frame size
mask = cv2.resize(mask, (frame.shape[1], frame.shape[0]))
# Convert the mask to binary format with a threshold
_, binary_mask = cv2.threshold(mask, 0.5, 1, cv2.THRESH_BINARY)
binary_mask = binary_mask.astype(np.uint8)
# Refine the mask using a dilation operation to smooth the edges
kernel = np.ones((15, 15), np.uint8)
dilated_mask = cv2.dilate(binary_mask, kernel, iterations=1)
# Invert the mask for background
inverse_mask = cv2.bitwise_not(dilated_mask)
# Apply a strong blur effect to the entire frame
blurred_background = cv2.GaussianBlur(frame, (45, 45), 0)
# Extract the foreground (user) from the original frame using the mask
foreground = cv2.bitwise_and(frame, frame, mask=dilated_mask)
# Extract the background from the blurred frame using the inverse mask
ENG
IN
11:18
13-10-2024


ChatGPT
Home Page - Select or create a
Real-time video background op
localhost:8888/notebooks/Real-time%20video%20background%20optimization.ipynb
YouTube
Adobe Acrobat
All Bookmarks
jupyter Real-time video background optimization Last Checkpoint: 12 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
# Extract the background from the blurred frame using the inverse mask
background = cv2.bitwise_and(blurred_background, blurred_background, mask=inverse_mask)
# Combine the foreground and blurred background
output_frame = cv2.add(foreground, background)
# Display the result
cv2.imshow('Enhanced Background Blur', output_frame)
# Exit when 'q' key is pressed
if cv2.waitKey(1) & 0xFF == ord('q'):
break
# Release the webcam and close windows
cap.release()
cv2.destroyAllWindows()
ENG
IN
11:18
13-10-2024

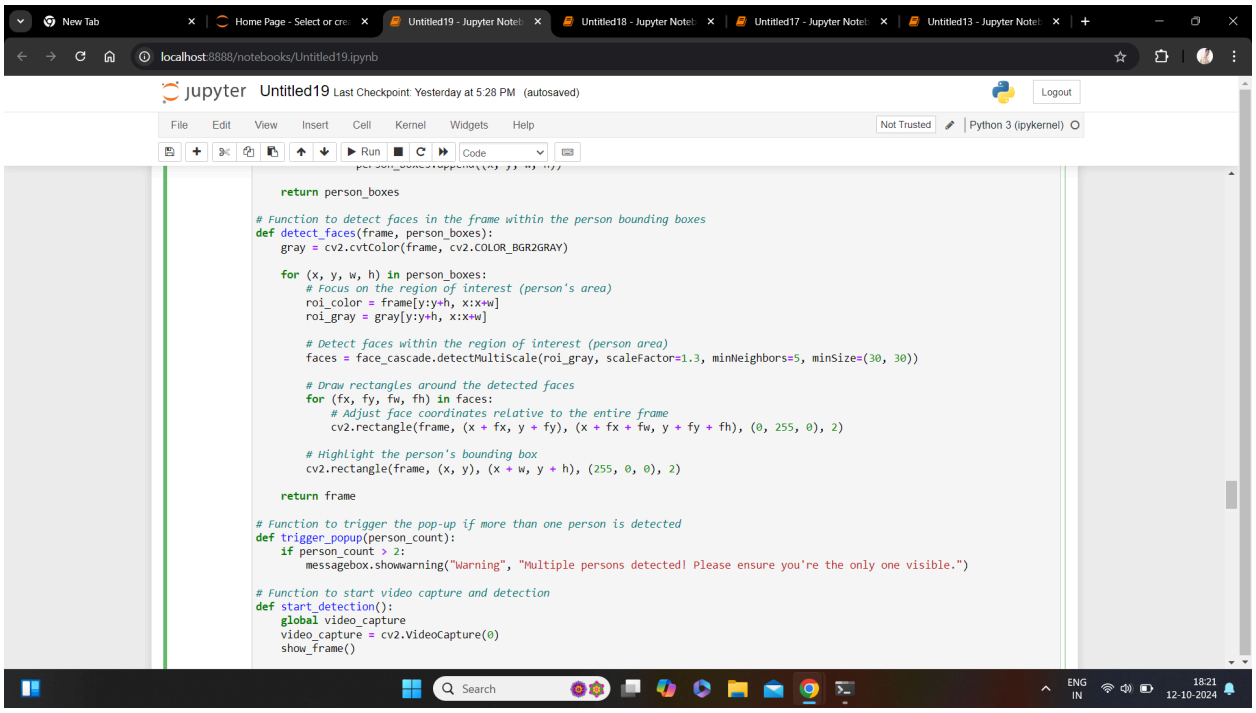

ChatGPT
Home Page - Select or create a
Real-time video background op
localhost:8888/notebooks/Real-time%20video%20background%20optimization.ipynb
YouTube
Adobe Acrobat
All Bookmarks
jupyter Real-time video background optimizatio
File Edit View Insert Cell Kernel Widgets Help
Run
Code
Enhanced Background Blur
# Extract the background from the blurre
background = cv2.bitwise_and(blurred_bac
# Combine the foreground and blurred bac
output_frame = cv2.add(foreground, backg
# Display the result
cv2.imshow('Enhanced Background Blur', o
# Exit when 'q' key is pressed
if cv2.waitKey(1) & 0xFF == ord('q'):
break
# Release the webcam and close windows
cap.release()
cv2.destroyAllWindows()
ENG
IN
11:19
13-10-2024

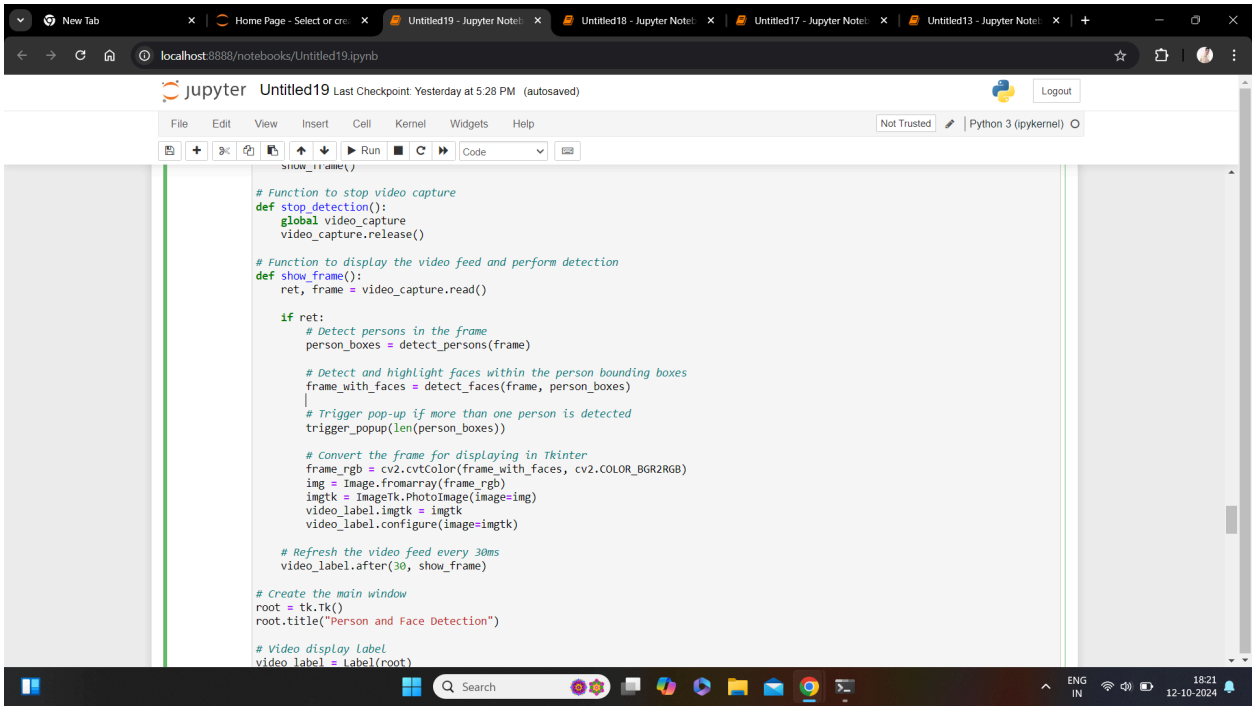
Vasu Bejugam :

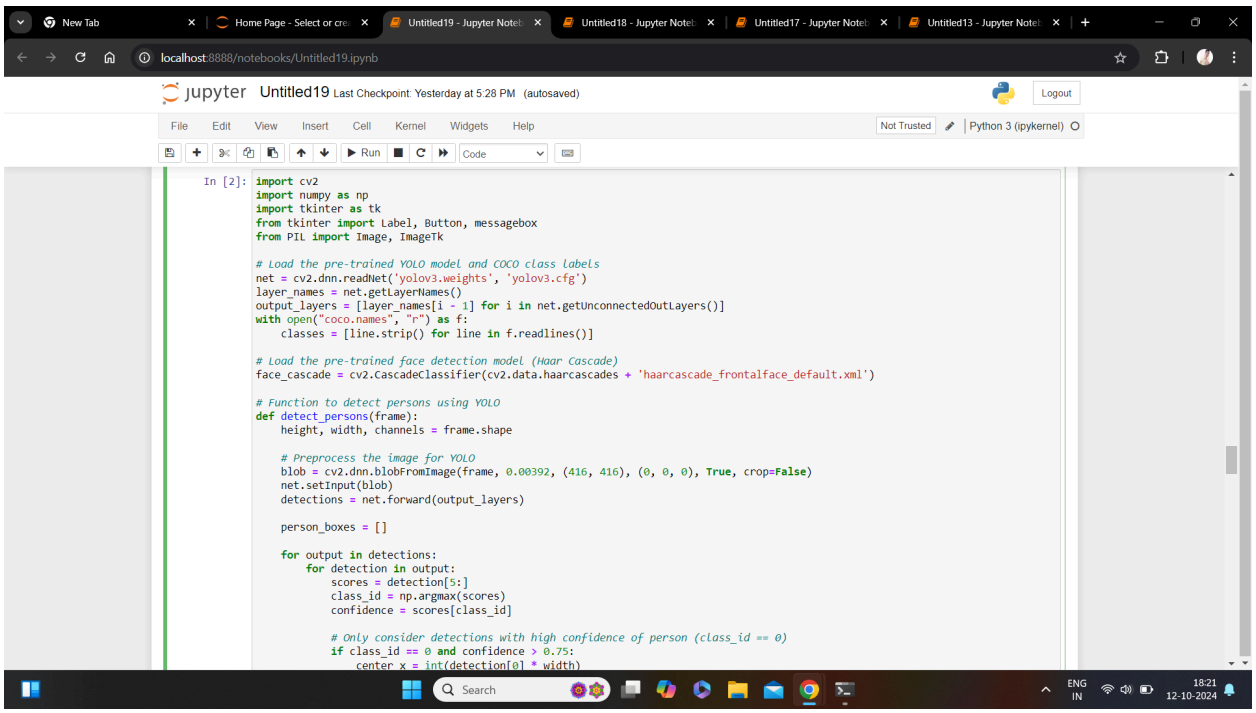
The **video background feature** of our mock interview tool allows the system to:

- **Distinguish between the user and the background**.
- **Focus on the main subject** in the frame if multiple people appear.
- **Detect disturbances** in the background and prompt the user to find a space with less distraction if needed.

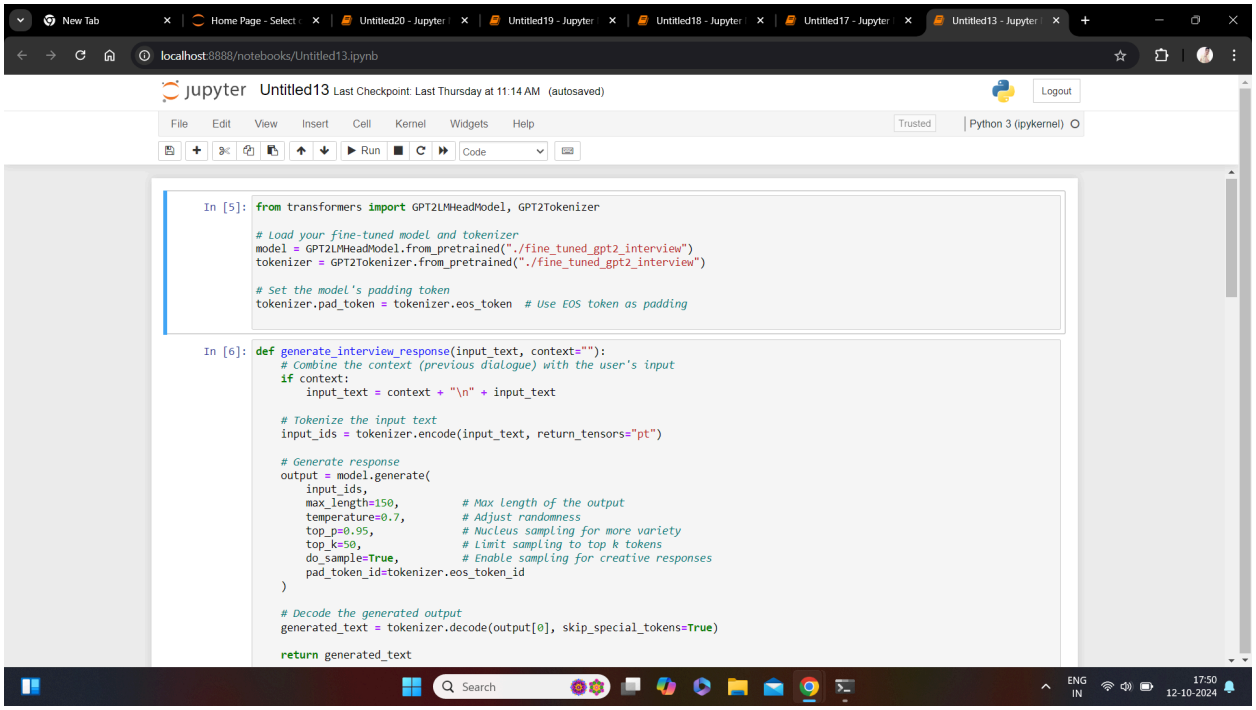
Here is the code to implement the user and the background detection for the mock interview tool

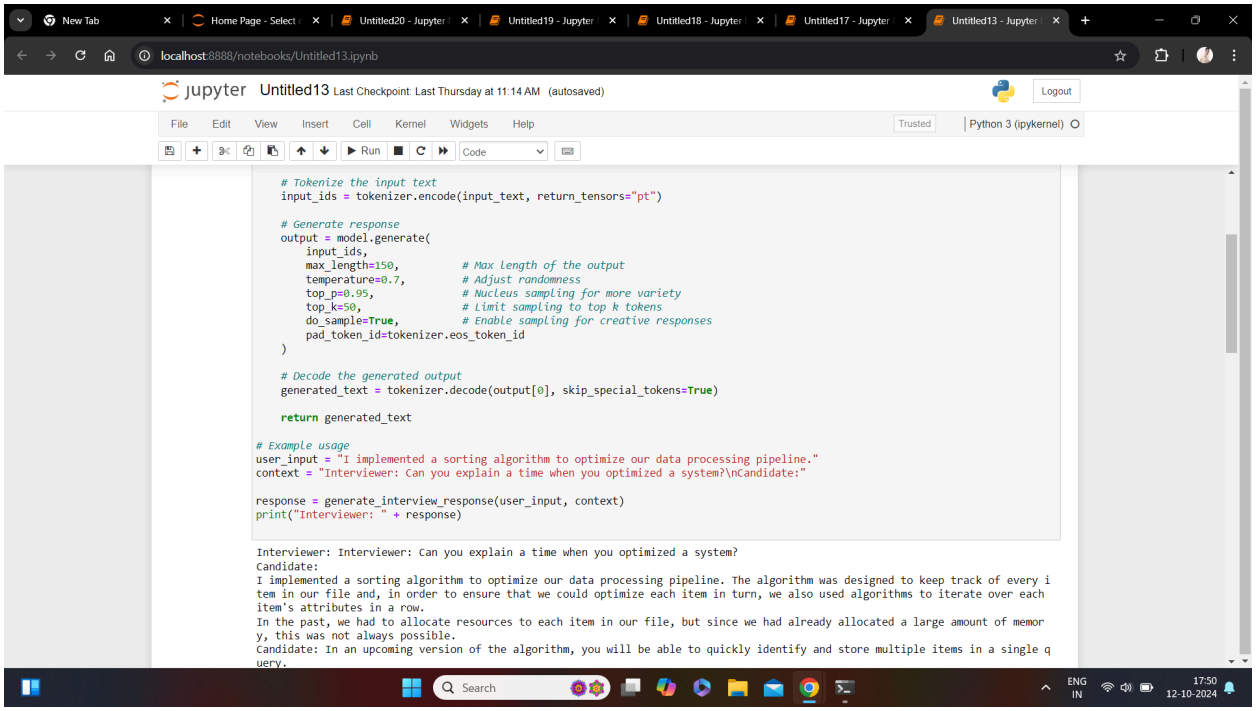


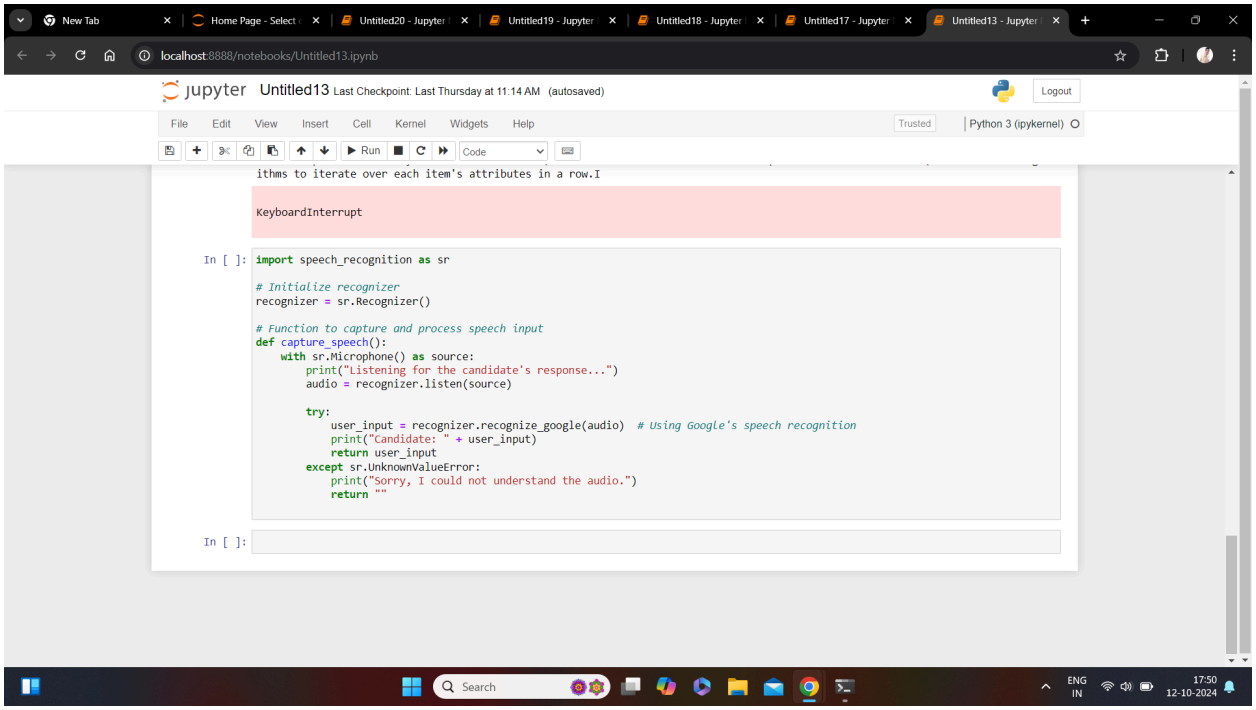




Here is the code for live response for dialogue generation using the GPT-2 model that helps us to generate follow up questions in the interview process.

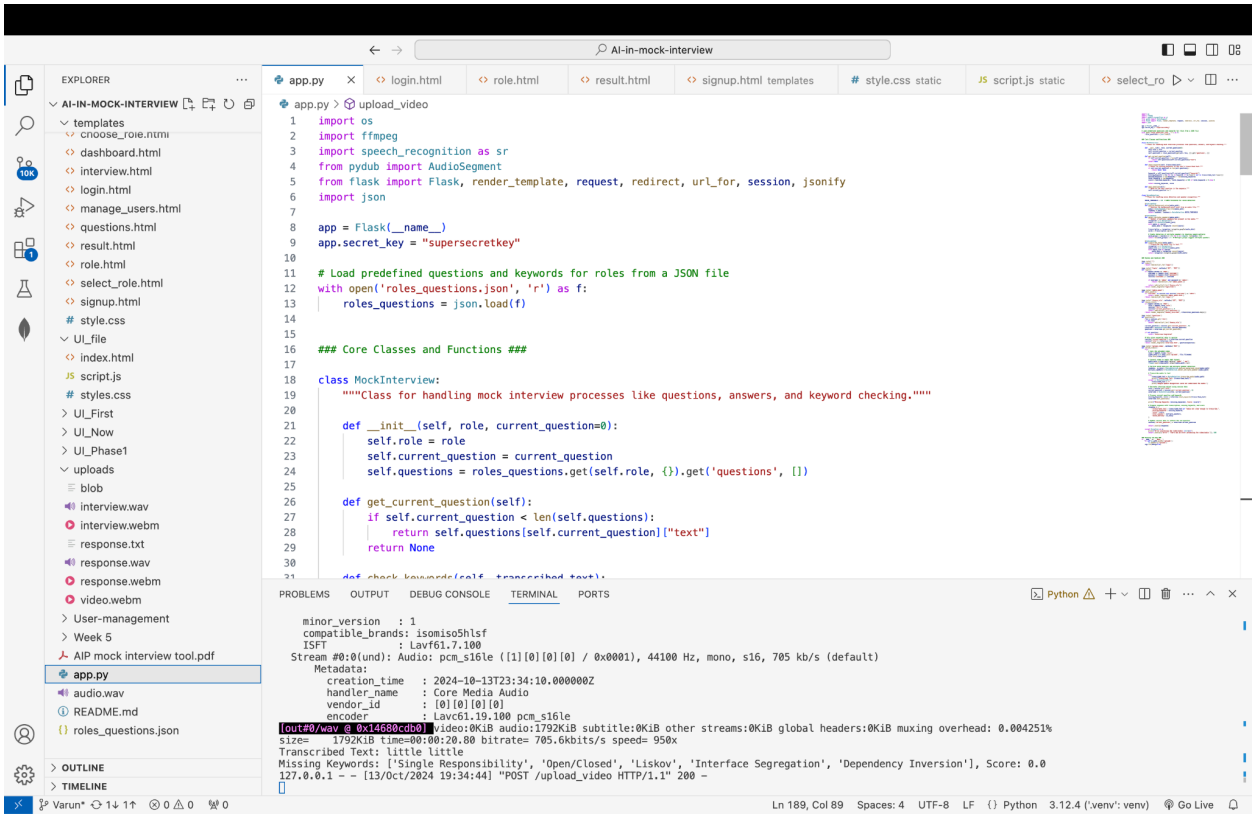

New Tab
Home Page - Select
Untitled20 - Jupyter
Untitled19 - Jupyter
Untitled18 - Jupyter
Untitled17 - Jupyter
Untitled13 - Jupyter
localhost:8888/notebooks/Untitled13.ipynb
jupyter Untitled13 Last Checkpoint: Last Thursday at 11:14 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [5]:
from transformers import GPT2LMHeadModel, GPT2Tokenizer
# Load your fine-tuned model and tokenizer
model = GPT2LMHeadModel.from_pretrained("./fine_tuned_gpt2_interview")
tokenizer = GPT2Tokenizer.from_pretrained("./fine_tuned_gpt2_interview")
# Set the model's padding token
tokenizer.pad_token = tokenizer.eos_token # Use EOS token as padding
In [6]:
def generate_interview_response(input_text, context=""):
# Combine the context (previous dialogue) with the user's input
if context:
input_text = context + "\n" + input_text
# Tokenize the input text
input_ids = tokenizer.encode(input_text, return_tensors="pt")
# Generate response
output = model.generate(
input_ids,
max_length=150, # Max length of the output
temperature=0.7, # Adjust randomness
top_p=0.95, # Nucleus sampling for more variety
top_k=50, # Limit sampling to top k tokens
do_sample=True, # Enable sampling for creative responses
pad_token_id=tokenizer.eos_token_id
)
# Decode the generated output
generated_text = tokenizer.decode(output[0], skip_special_tokens=True)
return generated_text
Search
ENG
IN
17:50
12-10-2024

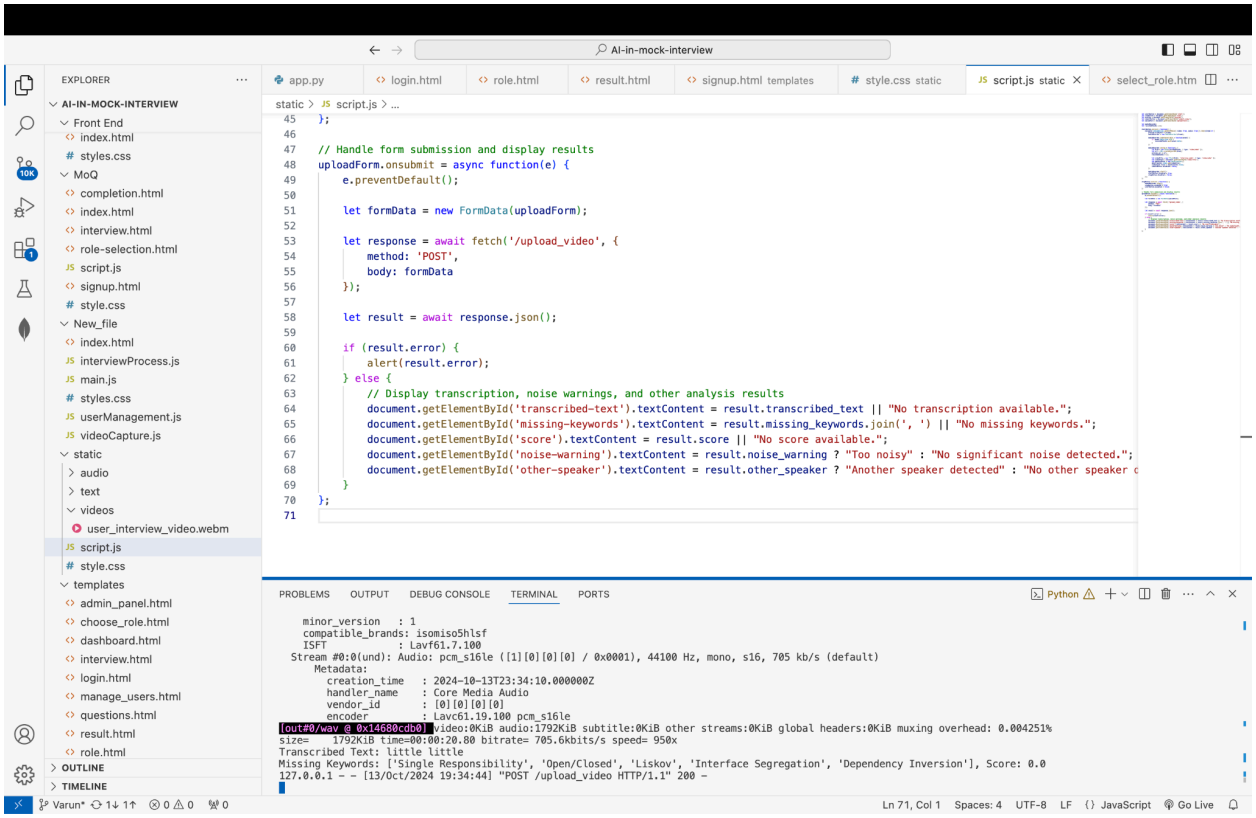

New Tab
Home Page - Select
Untitled20 - Jupyter
Untitled19 - Jupyter
Untitled18 - Jupyter
Untitled17 - Jupyter
Untitled13 - Jupyter
localhost:8888/notebooks/Untitled13.ipynb
jupyter Untitled13 Last Checkpoint: Last Thursday at 11:14 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
# Tokenize the input text
input_ids = tokenizer.encode(input_text, return_tensors="pt")
# Generate response
output = model.generate(
input_ids,
max_length=150, # Max length of the output
temperature=0.7, # Adjust randomness
top_p=0.95, # Nucleus sampling for more variety
top_k=50, # Limit sampling to top k tokens
do_sample=True, # Enable sampling for creative responses
pad_token_id=tokenizer.eos_token_id
)
# Decode the generated output
generated_text = tokenizer.decode(output[0], skip_special_tokens=True)
return generated_text
# Example usage
user_input = "I implemented a sorting algorithm to optimize our data processing pipeline."
context = "Interviewer: Can you explain a time when you optimized a system?\nCandidate:"
response = generate_interview_response(user_input, context)
print("Interviewer: " + response)
Interviewer: Interviewer: Can you explain a time when you optimized a system?
Candidate:
I implemented a sorting algorithm to optimize our data processing pipeline. The algorithm was designed to keep track of every i
tem in our file and, in order to ensure that we could optimize each item in turn, we also used algorithms to iterate over each
item's attributes in a row.
In the past, we had to allocate resources to each item in our file, but since we had already allocated a large amount of memor
y, this was not always possible.
Candidate: In an upcoming version of the algorithm, you will be able to quickly identify and store multiple items in a single q
uery.
Search
ENG
IN
17:50
12-10-2024


New Tab
Home Page - Select
Untitled20 - Jupyter
Untitled19 - Jupyter
Untitled18 - Jupyter
Untitled17 - Jupyter
Untitled13 - Jupyter
localhost:8888/notebooks/Untitled13.ipynb
jupyter Untitled13 Last Checkpoint: Last Thursday at 11:14 AM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
ithms to iterate over each item's attributes in a row.I
KeyboardInterrupt
In [ ]:
import speech_recognition as sr
# Initialize recognizer
recognizer = sr.Recognizer()
# Function to capture and process speech input
def capture_speech():
with sr.Microphone() as source:
print("Listening for the candidate's response...")
audio = recognizer.listen(source)
try:
user_input = recognizer.recognize_google(audio) # Using Google's speech recognition
print("Candidate: " + user_input)
return user_input
except sr.UnknownValueError:
print("Sorry, I could not understand the audio.")
return ""
In [ ]:
Search
ENG
IN
17:50
12-10-2024

Varunan Gurushev :

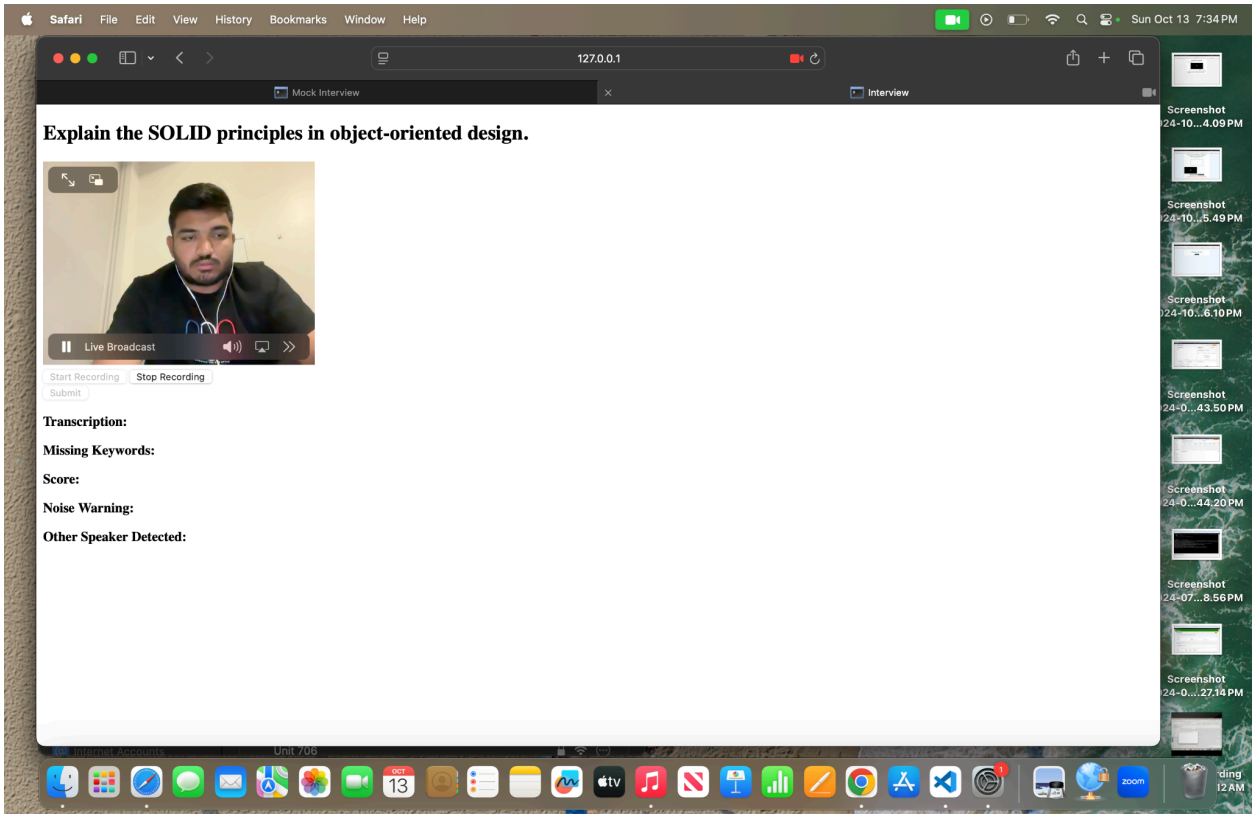
- Each question now includes predefined keywords; missing keywords in transcriptions are highlighted for feedback.

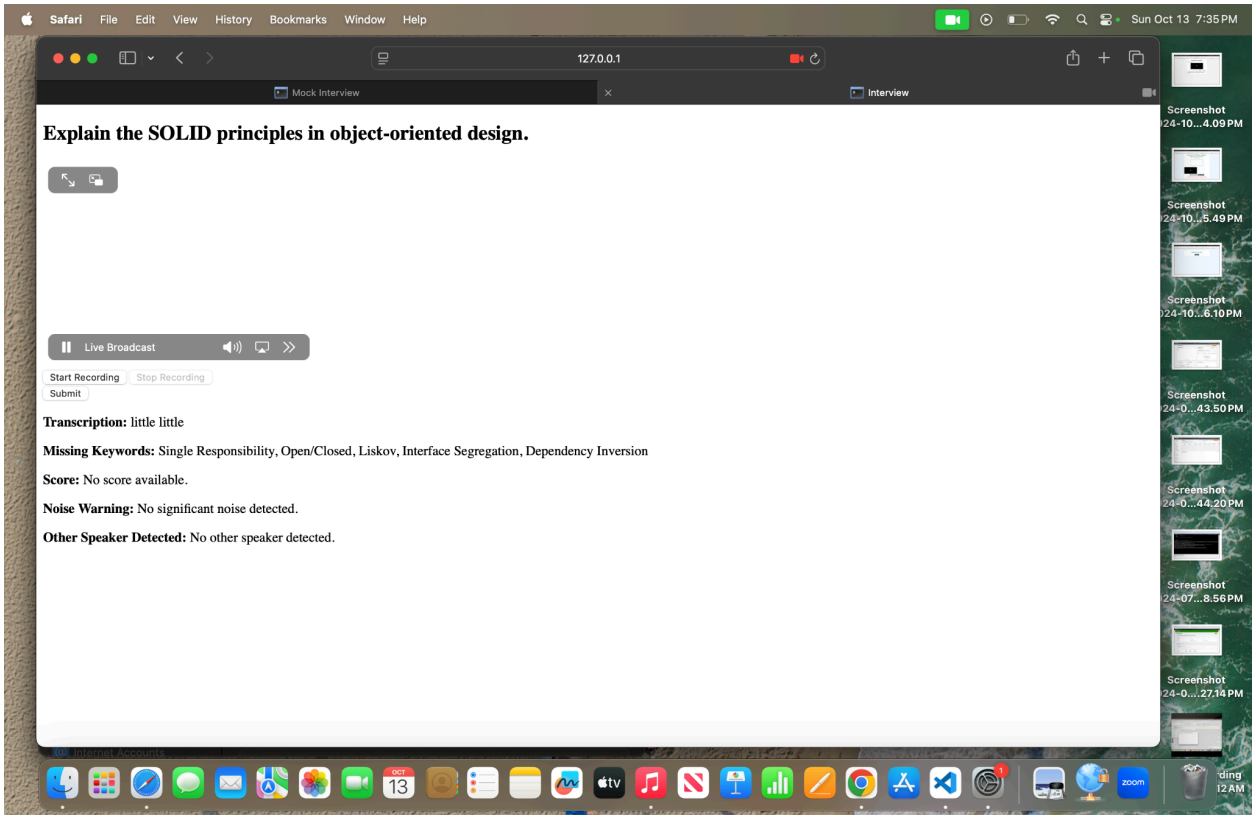




Audio Background Detection:

- Developed a model to detect the surrounding environment during audio input.
- The model also identifies if other individuals are present, ensuring focused and controlled input.

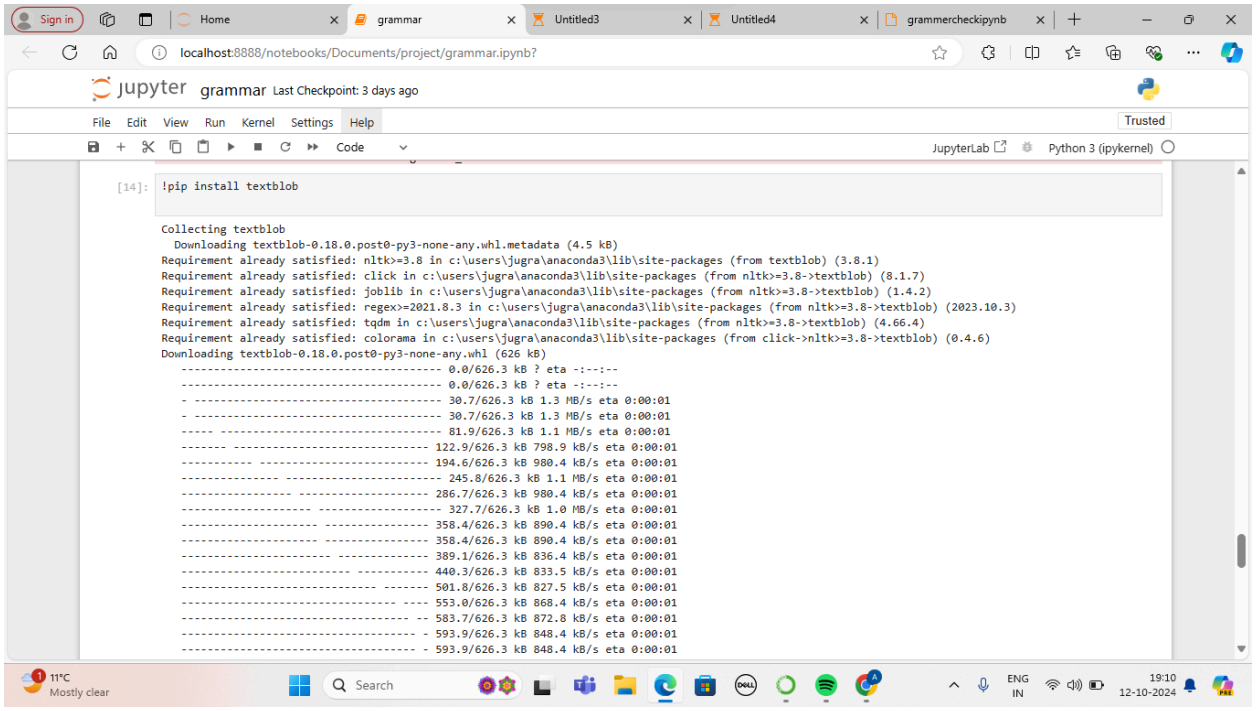

Safari File Edit View History Bookmarks Window Help
Sun Oct 13 7:34 PM
127.0.0.1
Mock Interview
Interview
Explain the SOLID principles in object-oriented design.
Live Broadcast
Start Recording
Stop Recording
Submit
Transcription:
Missing Keywords:
Score:
Noise Warning:
Other Speaker Detected:


Safari File Edit View History Bookmarks Window Help
Sun Oct 13 7:35 PM
127.0.0.1
Mock Interview
Interview
Explain the SOLID principles in object-oriented design.
Live Broadcast
Start Recording
Stop Recording
Submit
Transcription: little little
Missing Keywords: Single Responsibility, Open/Closed, Liskov, Interface Segregation, Dependency Inversion
Score: No score available.
Noise Warning: No significant noise detected.
Other Speaker Detected: No other speaker detected.

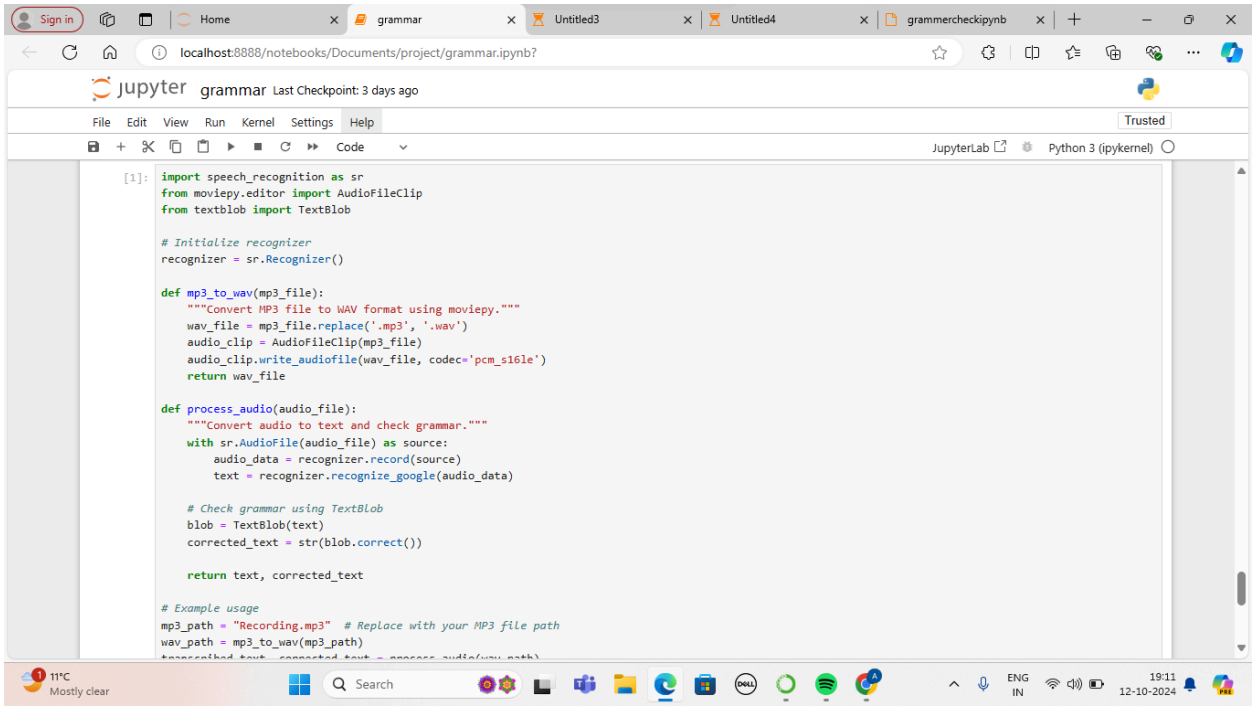
Arshdeep Kaur:

I am analyzing user's grammatical errors in an interview by converting audio to text ,after that checking the basic grammatical errors .

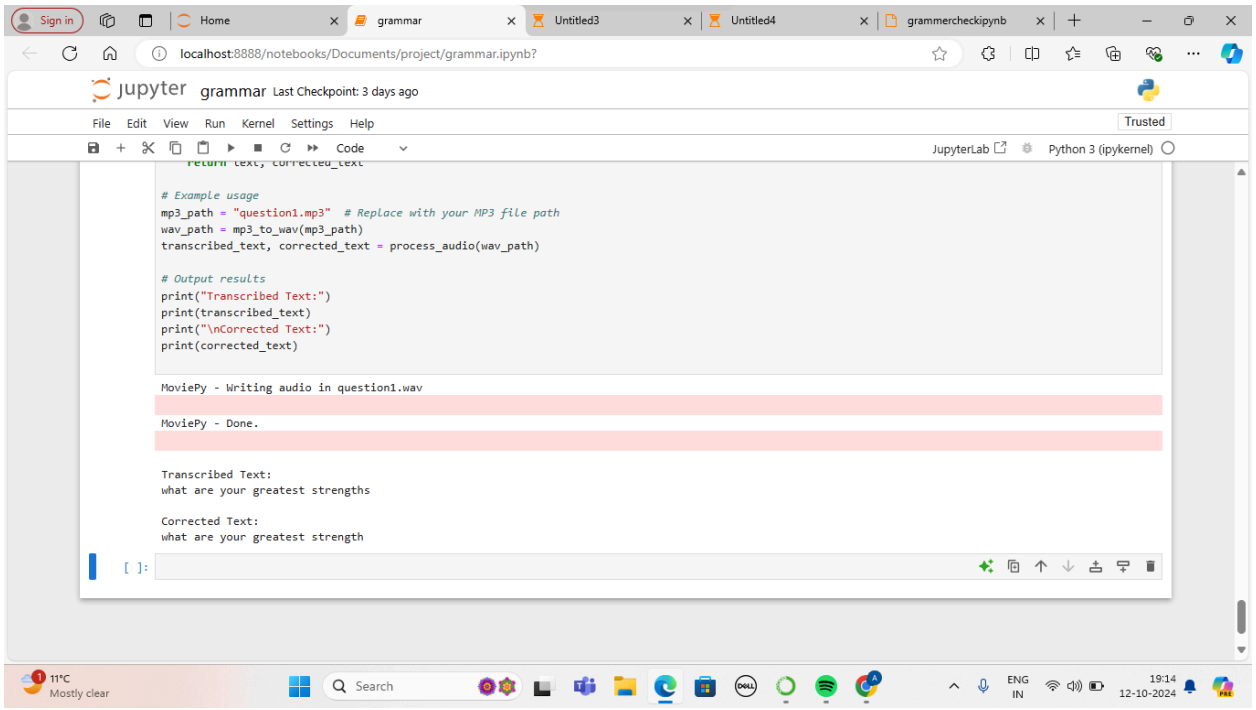
Analyzes the transcribed text for grammatical errors using a grammar-checking tool that is textblob



Converts audio into text using a speech recognition library and checking the error



This is the output



JASKARAN SINGH-

Overview

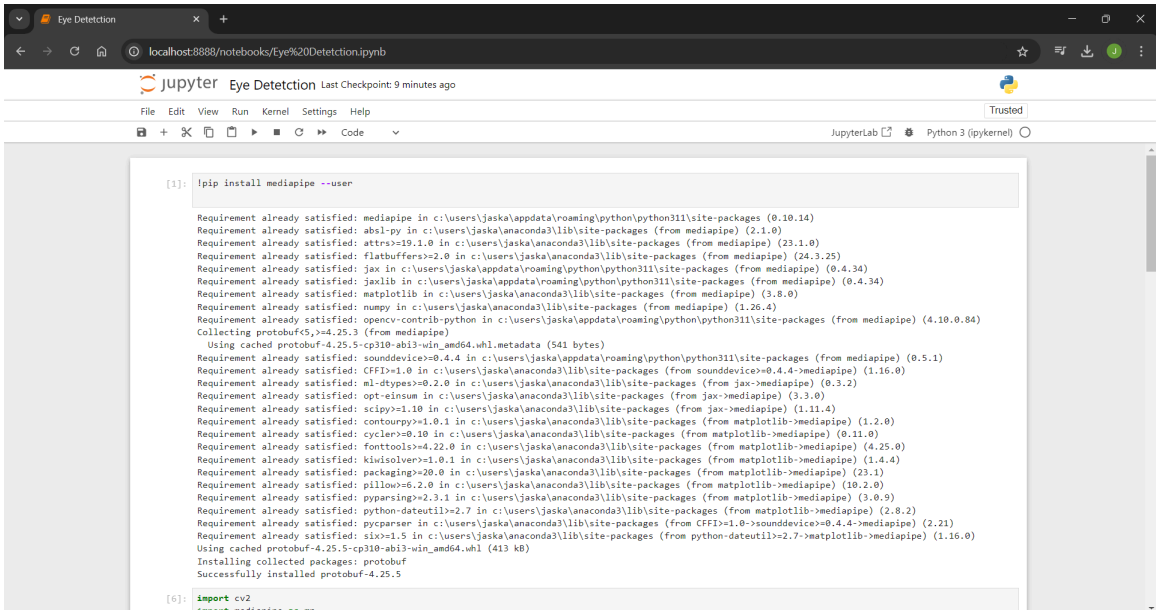
This application uses your computer's webcam to detect where your eyes are looking. It can tell if you are looking straight at the camera or if your gaze is directed elsewhere. The app aims to help users focus on the camera during Mock Interview.

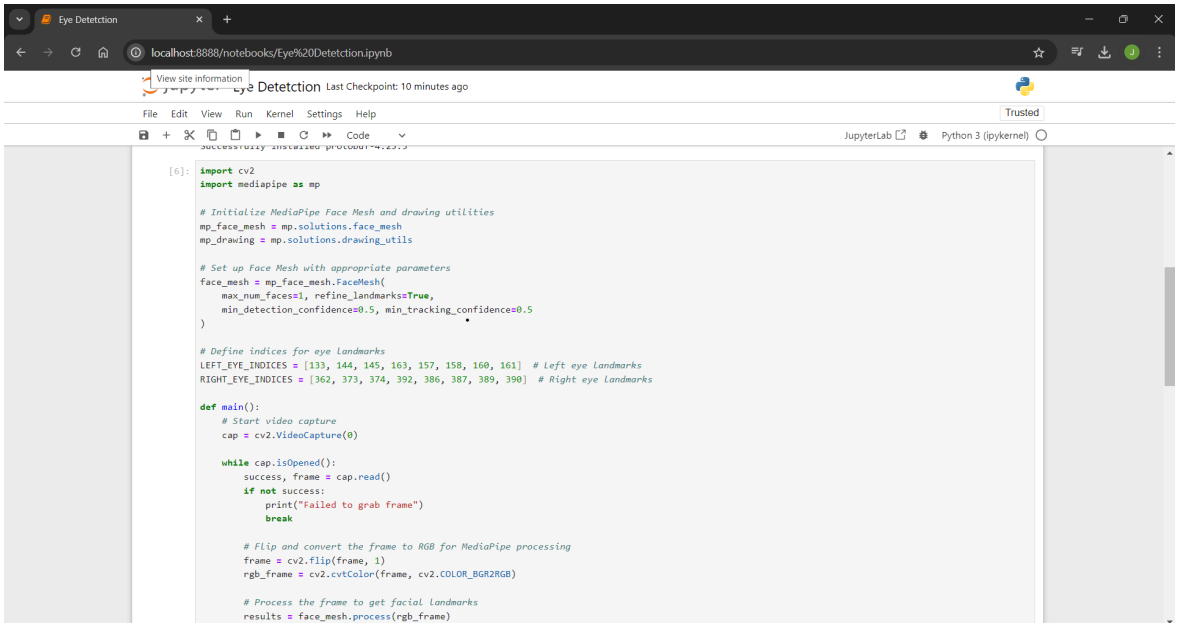
Key Features

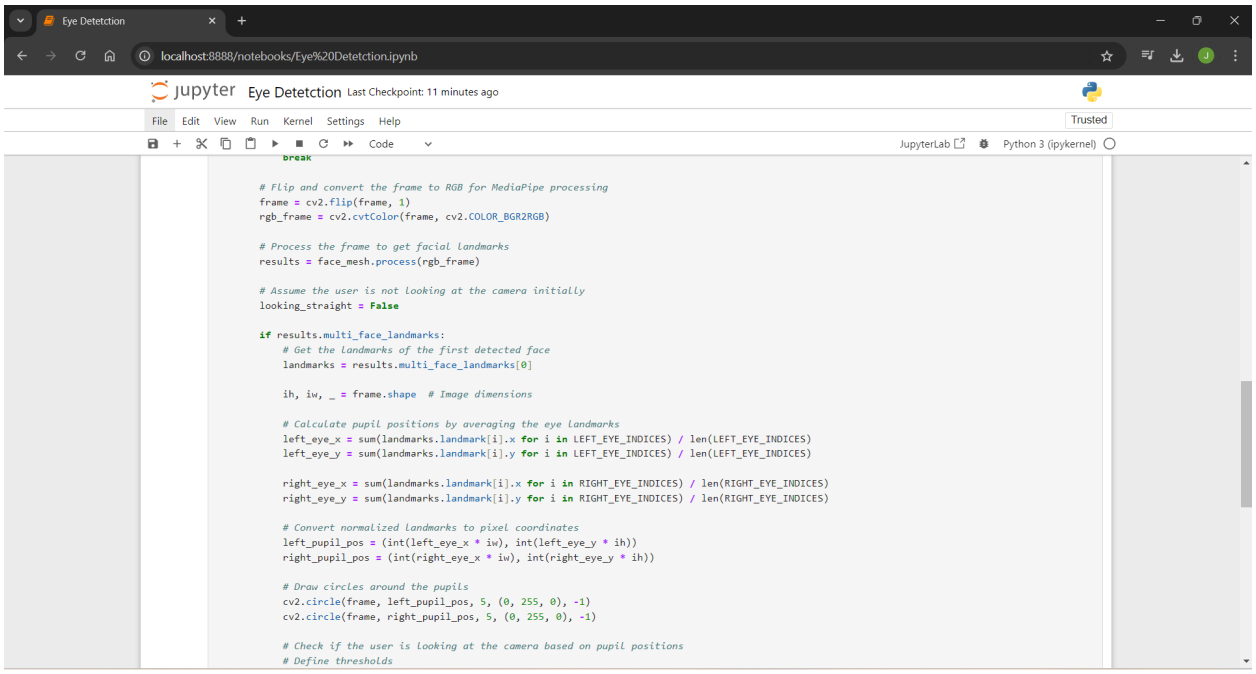
- **Live Video Feed:** The app shows a live video feed from your webcam.
- **Eye Detection:** It detects your eyes and calculates their position.
- **Gaze Feedback:** It provides real-time feedback on whether you are looking at the camera.

How It Works

1. **Camera Access:** When you start the application, it will access your webcam to capture video.
2. **Eye Tracking:** The application identifies the position of your eyes in the video feed.
3. **Gaze Evaluation:** It checks if your eyes are looking straight ahead. If you look away, it will alert you.
4. **Visual Cues:** Circles are drawn around your pupils to indicate their positions, and helpful messages are displayed on the screen.


Eye Detetction
localhost:8888/notebooks/Eye%20Detetction.ipynb
jupyter Eye Detetction Last Checkpoint: 9 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[1]: !pip install mediapipe --user
Requirement already satisfied: mediapipe in c:\users\jaska\appdata\roaming\python\python311\site-packages (0.10.14)
Requirement already satisfied: absl-py in c:\users\jaska\anaconda3\lib\site-packages (from mediapipe) (2.1.0)
Requirement already satisfied: attrs>=19.1.0 in c:\users\jaska\anaconda3\lib\site-packages (from mediapipe) (23.1.0)
Requirement already satisfied: flatbuffers>=2.0 in c:\users\jaska\anaconda3\lib\site-packages (from mediapipe) (24.3.25)
Requirement already satisfied: jax in c:\users\jaska\appdata\roaming\python\python311\site-packages (from mediapipe) (0.4.34)
Requirement already satisfied: jaxlib in c:\users\jaska\appdata\roaming\python\python311\site-packages (from mediapipe) (0.4.34)
Requirement already satisfied: matplotlib in c:\users\jaska\anaconda3\lib\site-packages (from mediapipe) (3.8.0)
Requirement already satisfied: numpy in c:\users\jaska\anaconda3\lib\site-packages (from mediapipe) (1.26.4)
Requirement already satisfied: opencv-contrib-python in c:\users\jaska\appdata\roaming\python\python311\site-packages (from mediapipe) (4.10.0.84)
Collecting protobuf<5,>=4.25.3 (from mediapipe)
Using cached protobuf-4.25.5-cp310-abi3-win_amd64.whl.metadata (541 bytes)
Requirement already satisfied: sounddevice>=0.4.4 in c:\users\jaska\appdata\roaming\python\python311\site-packages (from mediapipe) (0.5.1)
Requirement already satisfied: CFFI>=1.0 in c:\users\jaska\anaconda3\lib\site-packages (from sounddevice>=0.4.4->mediapipe) (1.16.0)
Requirement already satisfied: ml-dtypes>=0.2.0 in c:\users\jaska\anaconda3\lib\site-packages (from jax->mediapipe) (0.3.2)
Requirement already satisfied: opt-einsum in c:\users\jaska\anaconda3\lib\site-packages (from jax->mediapipe) (3.3.0)
Requirement already satisfied: scipy>=1.10 in c:\users\jaska\anaconda3\lib\site-packages (from jax->mediapipe) (1.11.4)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\jaska\anaconda3\lib\site-packages (from matplotlib->mediapipe) (1.2.0)
Requirement already satisfied: cycler>=0.10 in c:\users\jaska\anaconda3\lib\site-packages (from matplotlib->mediapipe) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\jaska\anaconda3\lib\site-packages (from matplotlib->mediapipe) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\jaska\anaconda3\lib\site-packages (from matplotlib->mediapipe) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\jaska\anaconda3\lib\site-packages (from matplotlib->mediapipe) (23.1)
Requirement already satisfied: pillow>=6.2.0 in c:\users\jaska\anaconda3\lib\site-packages (from matplotlib->mediapipe) (10.2.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\jaska\anaconda3\lib\site-packages (from matplotlib->mediapipe) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\jaska\anaconda3\lib\site-packages (from matplotlib->mediapipe) (2.8.2)
Requirement already satisfied: pycparser in c:\users\jaska\anaconda3\lib\site-packages (from CFFI>=1.0->sounddevice>=0.4.4->mediapipe) (2.21)
Requirement already satisfied: six>=1.5 in c:\users\jaska\anaconda3\lib\site-packages (from python-dateutil>=2.7->matplotlib->mediapipe) (1.16.0)
Using cached protobuf-4.25.5-cp310-abi3-win_amd64.whl (413 kB)
Installing collected packages: protobuf
Successfully installed protobuf-4.25.5
[6]: import cv2

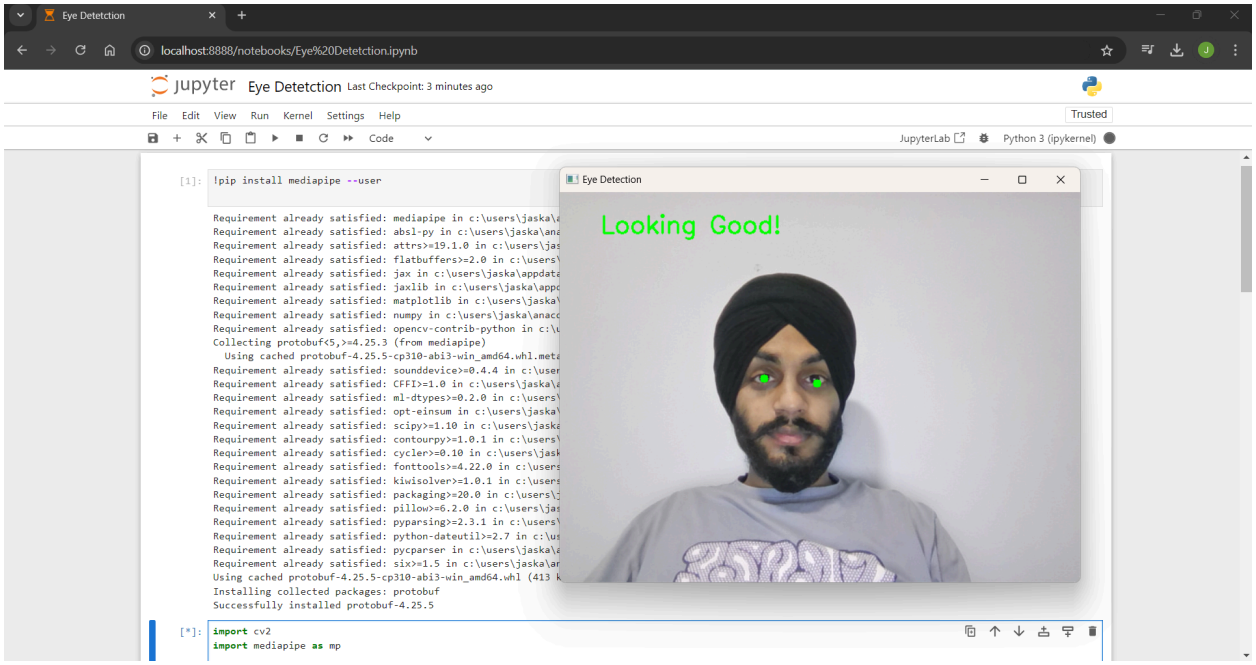

Eye Detetction
localhost:8888/notebooks/Eye%20Detetction.ipynb
View site information
Detetction Last Checkpoint: 10 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
[6]:
import cv2
import mediapipe as mp
# Initialize MediaPipe Face Mesh and drawing utilities
mp_face_mesh = mp.solutions.face_mesh
mp_drawing = mp.solutions.drawing_utils
# Set up Face Mesh with appropriate parameters
face_mesh = mp_face_mesh.FaceMesh(
max_num_faces=1, refine_landmarks=True,
min_detection_confidence=0.5, min_tracking_confidence=0.5
)
# Define indices for eye landmarks
LEFT_EYE_INDICES = [133, 144, 145, 163, 157, 158, 160, 161] # Left eye landmarks
RIGHT_EYE_INDICES = [362, 373, 374, 392, 386, 387, 389, 390] # Right eye landmarks
def main():
# Start video capture
cap = cv2.VideoCapture(0)
while cap.isOpened():
success, frame = cap.read()
if not success:
print("Failed to grab frame")
break
# Flip and convert the frame to RGB for MediaPipe processing
frame = cv2.flip(frame, 1)
rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
# Process the frame to get facial landmarks
results = face_mesh.process(rgb_frame)



Output

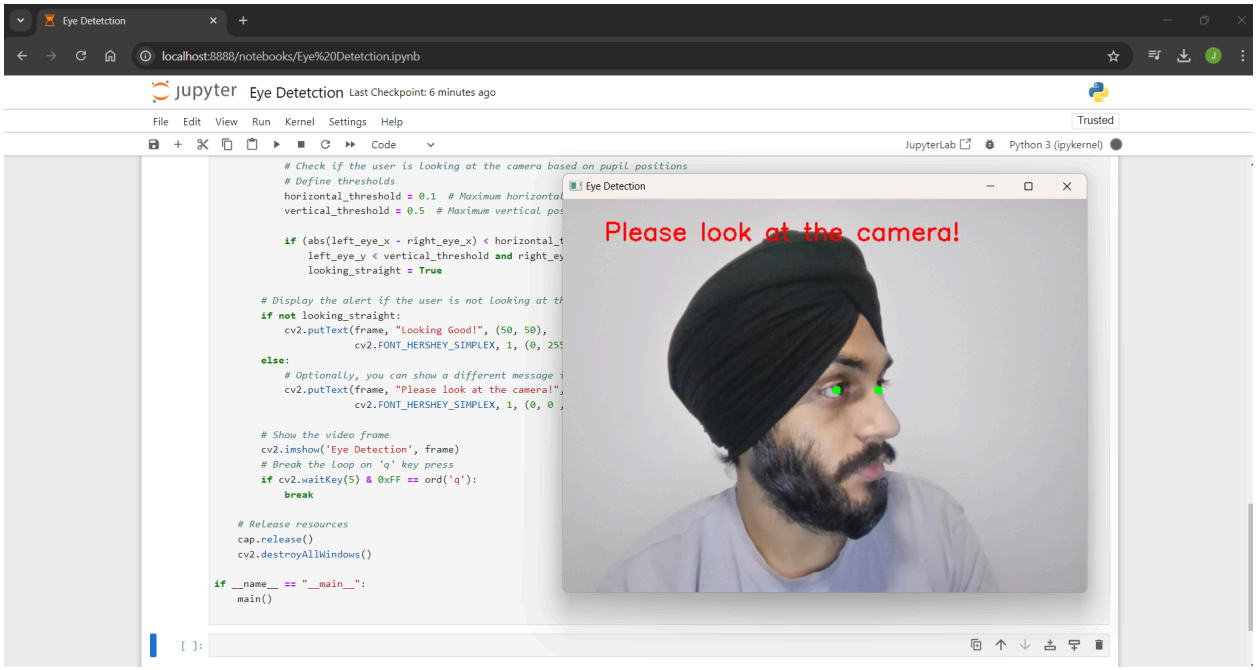
Looking in the Camera

When the eyes are detected looking in the camera, a message appears **“Looking Good!”**



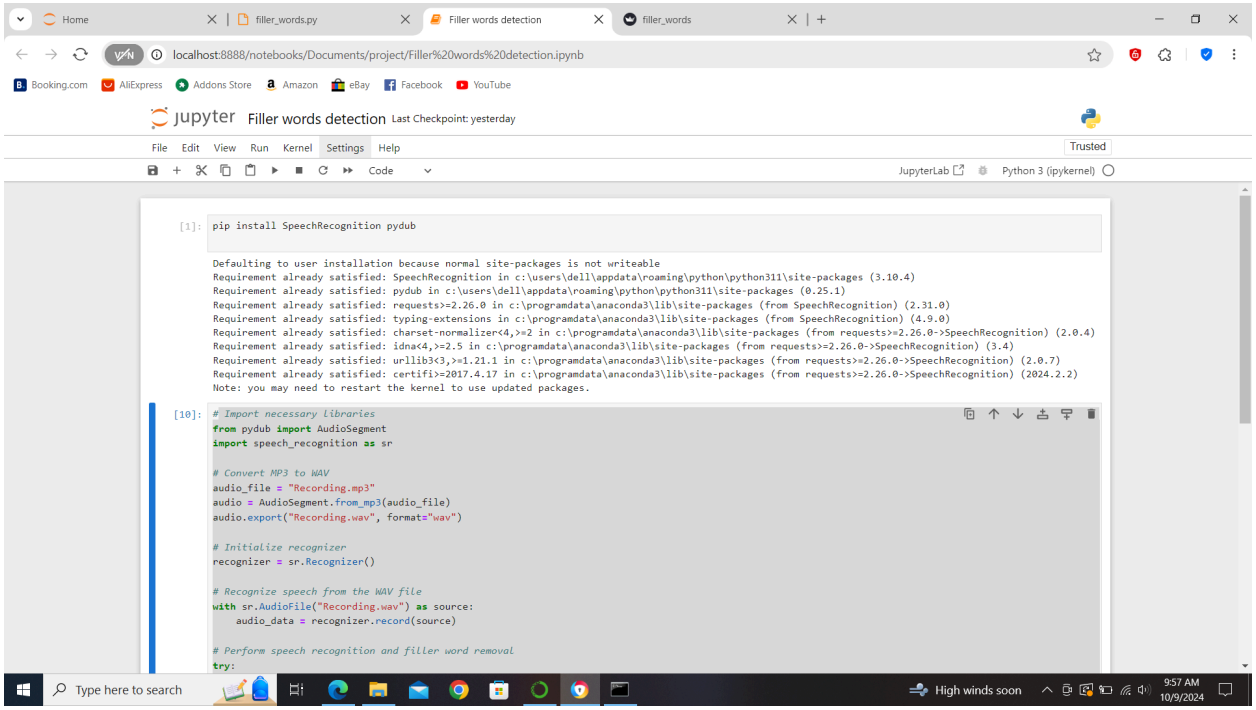
Looking Away from the Camera

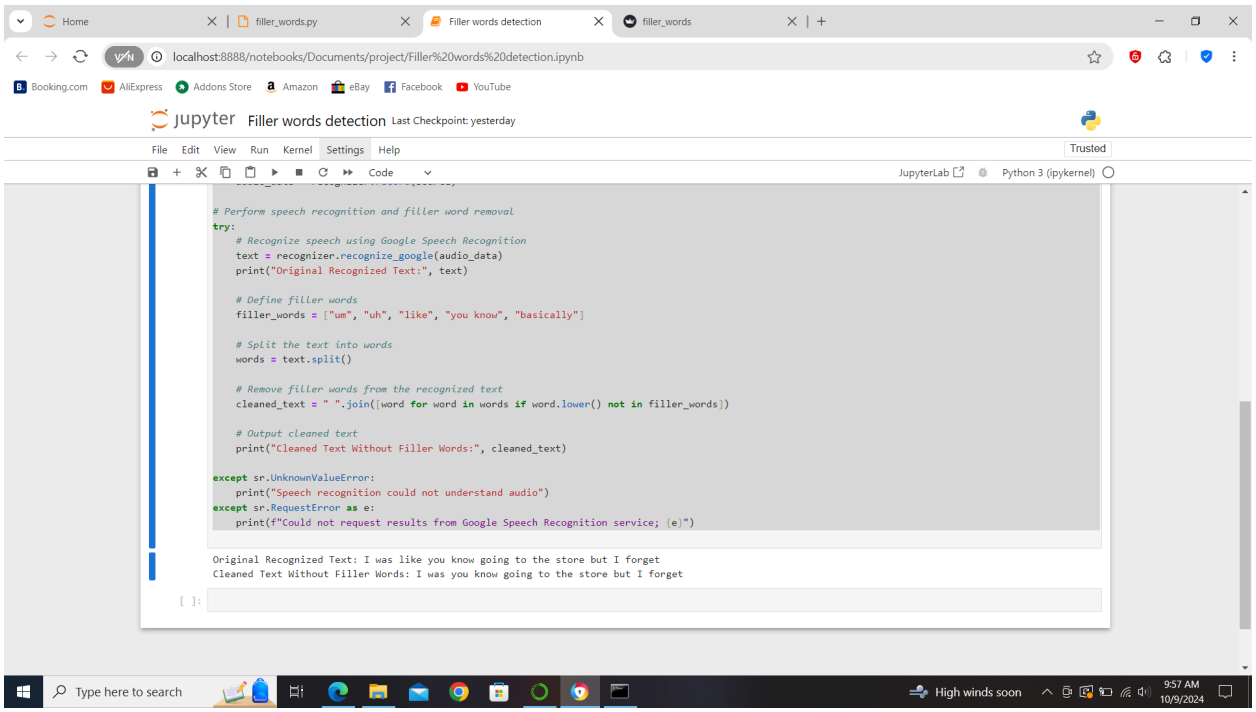
When the eyes are detected looking away from the camera, a message appears **“Please look at the camera!”**

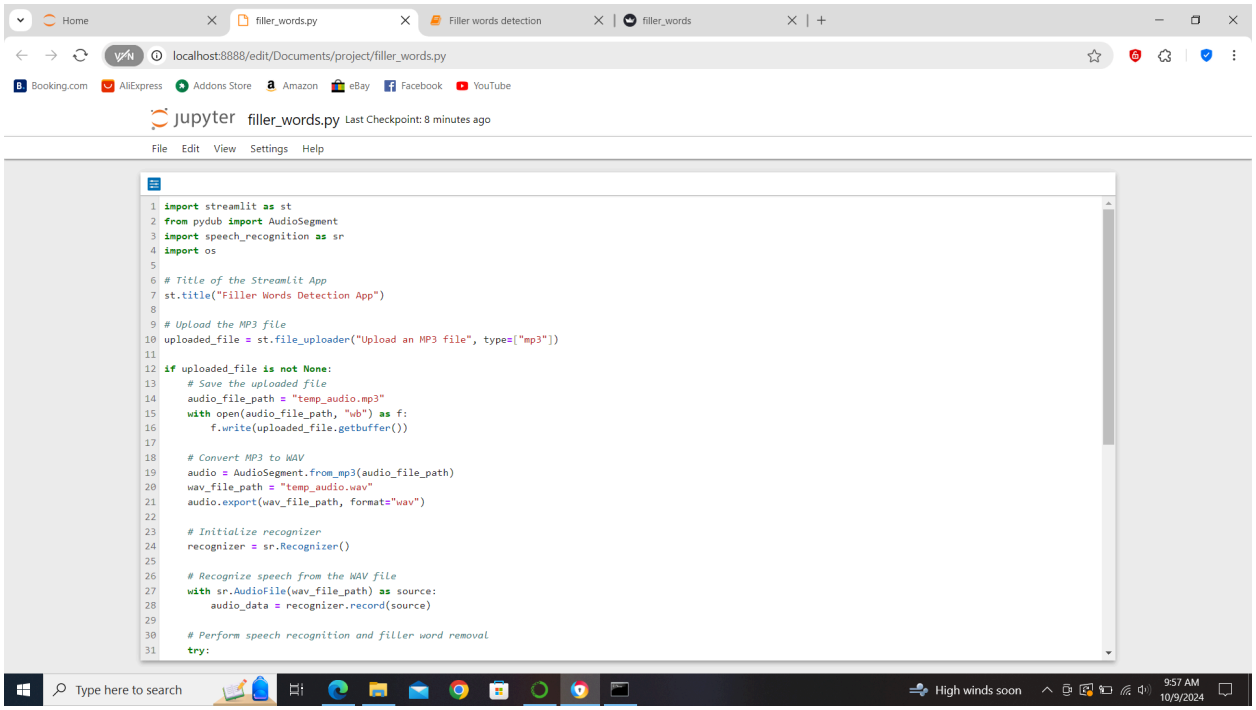


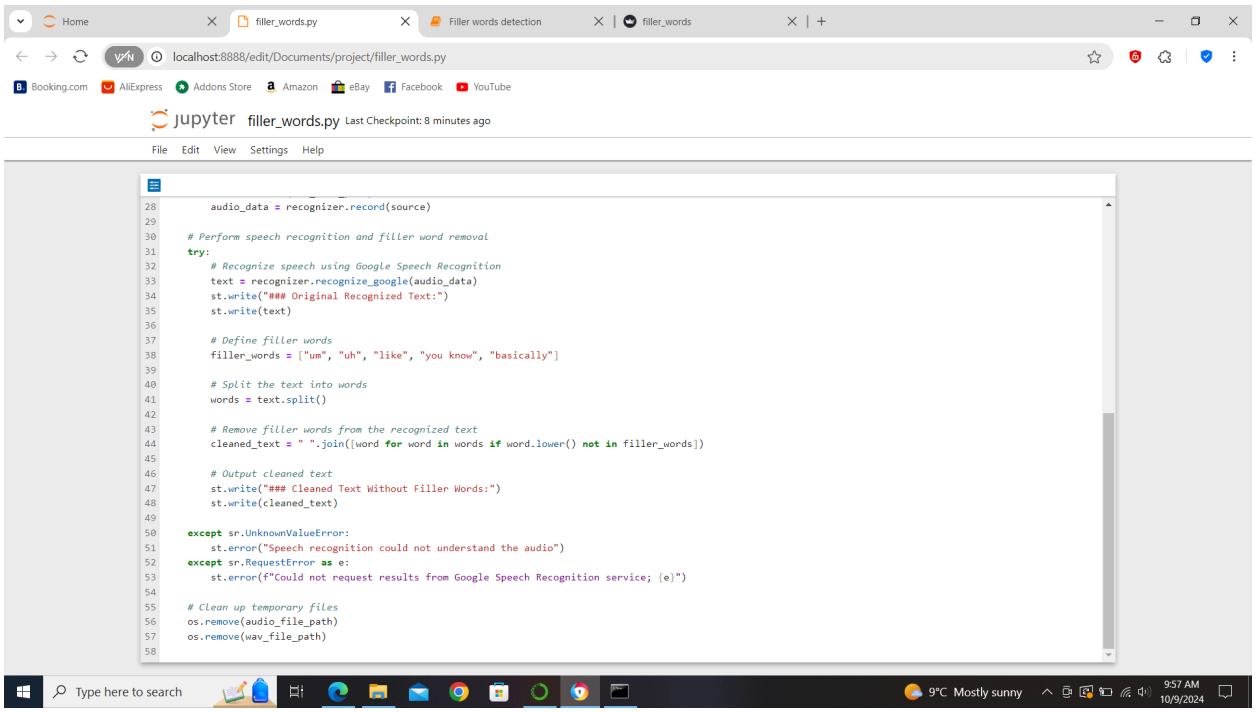
Dilpreet Kaur

I am working on Filler words Detection.

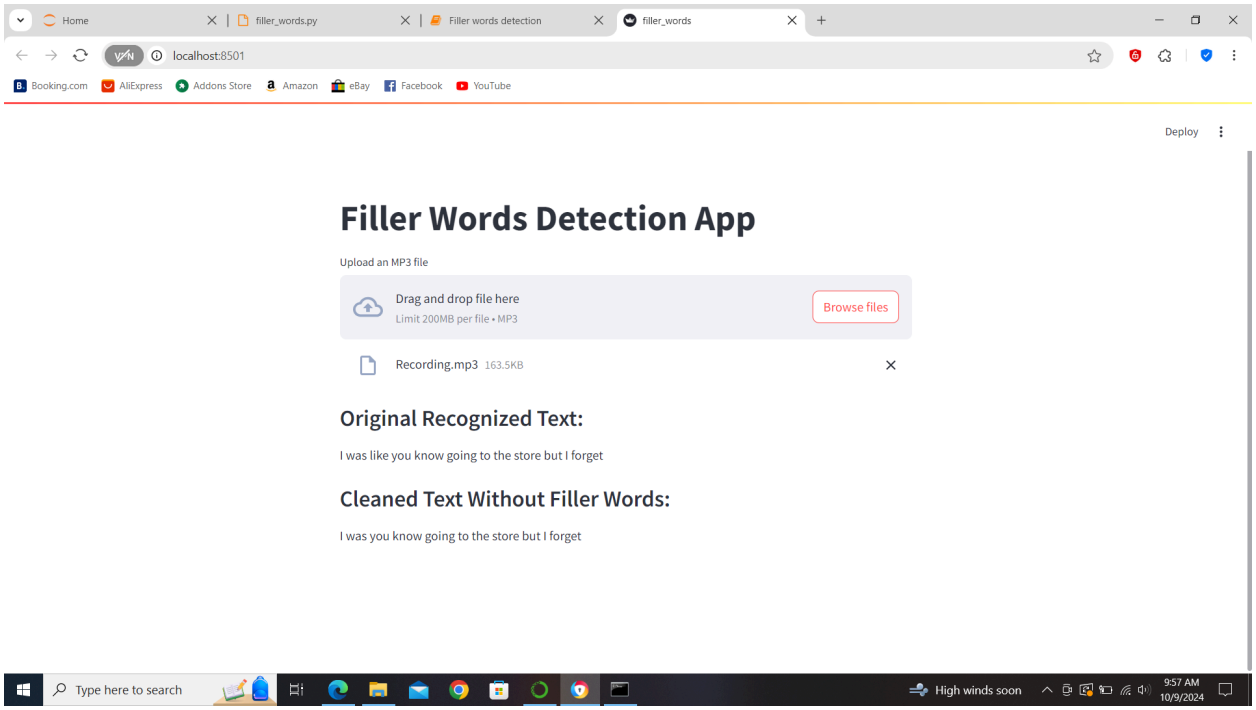





Home
filler_words.py
Filler words detection
filler_words
localhost:8888/edit/Documents/project/filler_words.py
Booking.com
AliExpress
Addons Store
Amazon
eBay
Facebook
YouTube
jupyter
filler_words.py
Last Checkpoint: 8 minutes ago
File
Edit
View
Settings
Help
import streamlit as st
from pydub import AudioSegment
import speech_recognition as sr
import os
# Title of the Streamlit App
st.title("Filler Words Detection App")
# Upload the MP3 file
uploaded_file = st.file_uploader("Upload an MP3 file", type=["mp3"])
if uploaded_file is not None:
# Save the uploaded file
audio_file_path = "temp_audio.mp3"
with open(audio_file_path, "wb") as f:
f.write(uploaded_file.getbuffer())
# Convert MP3 to WAV
audio = AudioSegment.from_mp3(audio_file_path)
wav_file_path = "temp_audio.wav"
audio.export(wav_file_path, format="wav")
# Initialize recognizer
recognizer = sr.Recognizer()
# Recognize speech from the WAV file
with sr.AudioFile(wav_file_path) as source:
audio_data = recognizer.record(source)
# Perform speech recognition and filler word removal
try:
Type here to search
High winds soon
9:57 AM
10/9/2024


Home
filler_words.py
Filler words detection
filler_words
localhost:8888/edit/Documents/project/filler_words.py
Booking.com
AliExpress
Addons Store
Amazon
eBay
Facebook
YouTube
jupyter
filler_words.py
Last Checkpoint: 8 minutes ago
File
Edit
View
Settings
Help
audio_data = recognizer.record(source)
# Perform speech recognition and filler word removal
try:
# Recognize speech using Google Speech Recognition
text = recognizer.recognize_google(audio_data)
st.write("### Original Recognized Text:")
st.write(text)
# Define filler words
filler_words = ["um", "uh", "like", "you know", "basically"]
# Split the text into words
words = text.split()
# Remove filler words from the recognized text
cleaned_text = " ".join([word for word in words if word.lower() not in filler_words])
# Output cleaned text
st.write("### Cleaned Text Without Filler Words:")
st.write(cleaned_text)
except sr.UnknownValueError:
st.error("Speech recognition could not understand the audio")
except sr.RequestError as e:
st.error(f"Could not request results from Google Speech Recognition service; {e}")
# Clean up temporary files
os.remove(audio_file_path)
os.remove(wav_file_path)
Type here to search
9°C Mostly sunny
9:57 AM
10/9/2024

Deployment


Home
filler_words.py
Filler words detection
filler_words
localhost:8501
Booking.com
AliExpress
Addons Store
Amazon
eBay
Facebook
YouTube
Deploy
Filler Words Detection App
Upload an MP3 file
Drag and drop file here
Limit 200MB per file • MP3
Browse files
Recording.mp3 163.5KB
Original Recognized Text:
I was like you know going to the store but I forget
Cleaned Text Without Filler Words:
I was you know going to the store but I forget
Type here to search
High winds soon
9:57 AM
10/9/2024