

# AI Mock Interview Tool

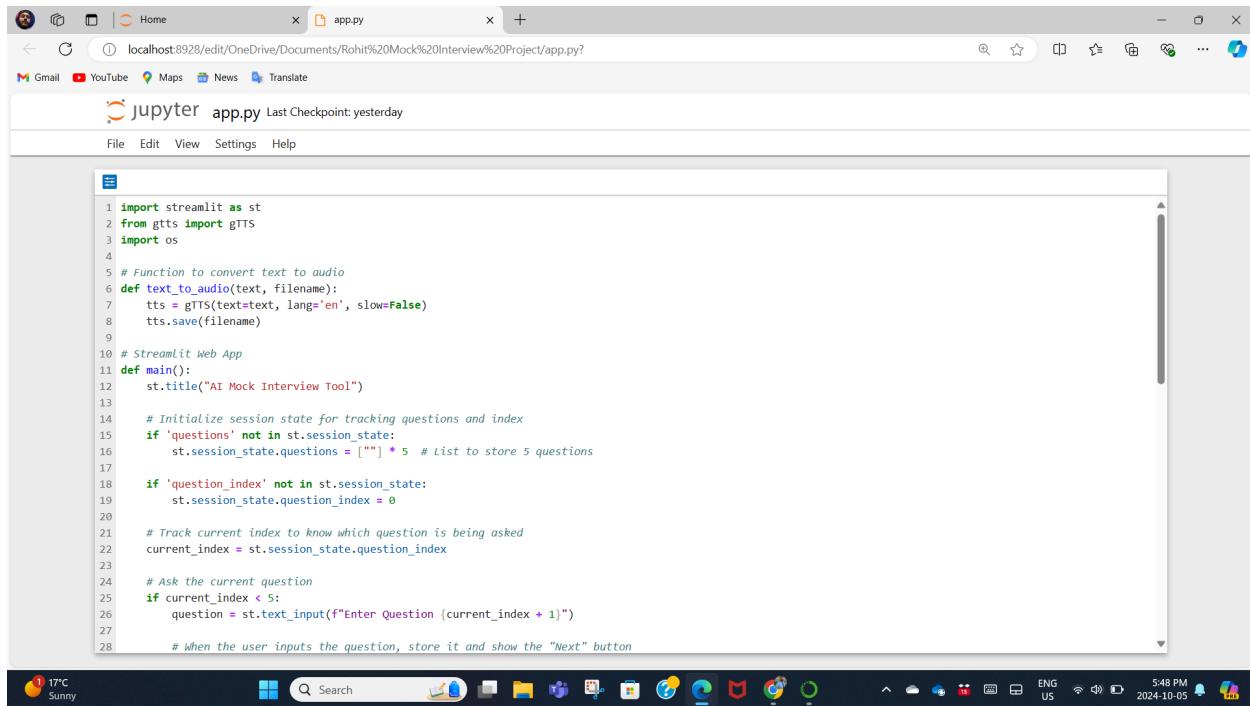
Project Deliverable #3:

Team 3:

Team member	Deliverables
Rohit	Text-to-Audio Conversation for Interview Questions
Dilpreet kaur	Mute Audio of the user while the interview Question is Narrated
Chanthuru Thavarajah	Record and Upload Your Video for Emotion Analysis
Arunkumar Anugu	Created additional test cases and Developed an automation test script for Error Handling silence
Vasu Bejugam	Implementing the GPT based dialogue generation
Arshdeep Kaur	Research on AI mock interview questions
Dhvani Bhavani	App integration for displaying text from speech-to-text
Varunan Gurushev	Developed UI sequential questions, video recording, Convert the video to text and check the keywords.
Jaskaran Singh	Face Presence and Detection to Camera

## Rohit: Text-to-Audio Conversation for Interview Questions

I implemented this Python code which allows users to input interview questions one by one, and then convert each question into audio, which is then played back immediately.



```
1 import streamlit as st
2 from gtts import gTTS
3 import os
4
5 # Function to convert text to audio
6 def text_to_audio(text, filename):
7     tts = gTTS(text=text, lang='en', slow=False)
8     tts.save(filename)
9
10 # Streamlit Web App
11 def main():
12     st.title("AI Mock Interview Tool")
13
14     # Initialize session state for tracking questions and index
15     if 'questions' not in st.session_state:
16         st.session_state.questions = [""] * 5 # List to store 5 questions
17
18     if 'question_index' not in st.session_state:
19         st.session_state.question_index = 0
20
21     # Track current index to know which question is being asked
22     current_index = st.session_state.question_index
23
24     # Ask the current question
25     if current_index < 5:
26         question = st.text_input(f"Enter Question {current_index + 1}")
27
28     # When the user inputs the question, store it and show the "Next" button
```

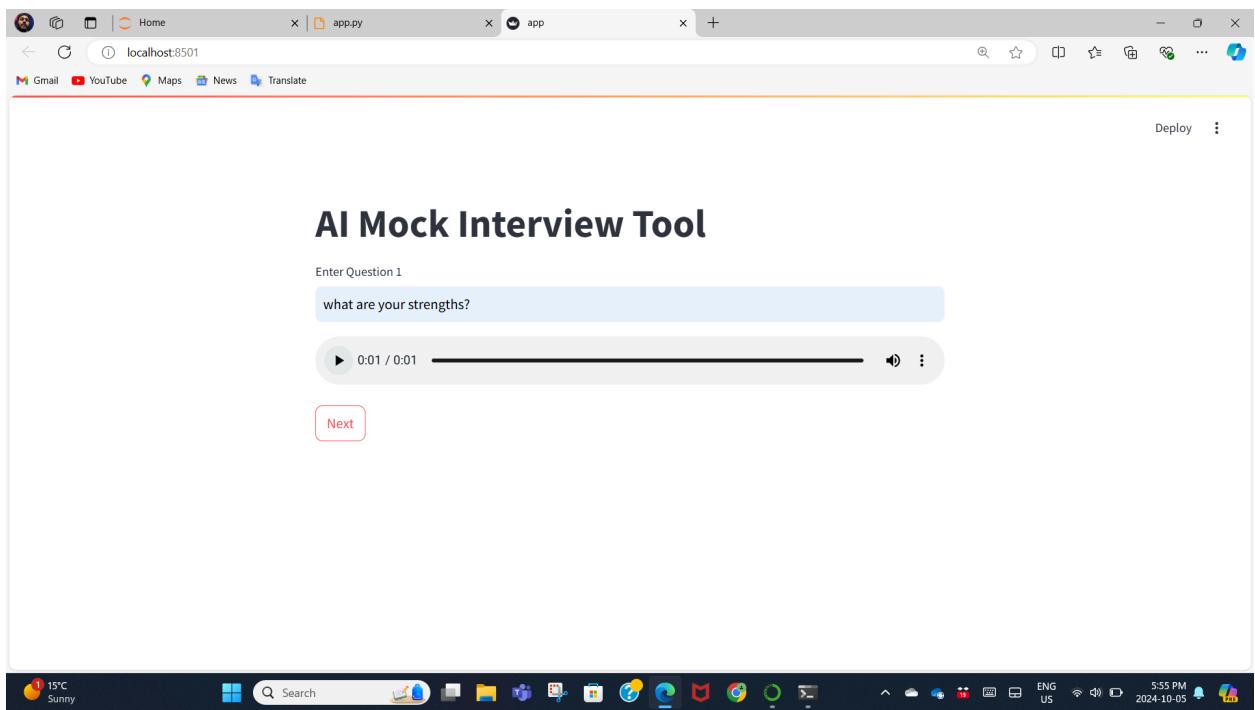
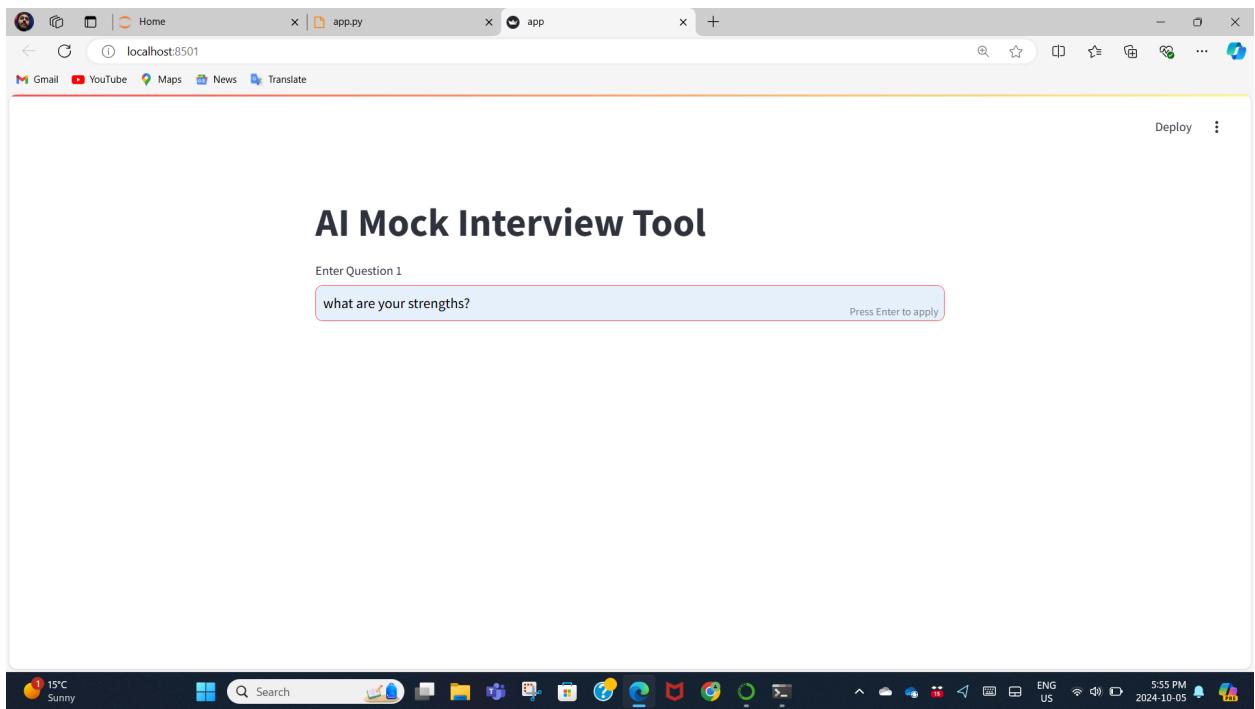
The screenshot shows a Microsoft Edge browser window with the address bar set to `localhost:8928/edit/OneDrive/Documents/Rohit%20Mock%20Interview%20Project/app.py`. The main content area displays a Jupyter notebook cell titled "app.py Last Checkpoint: yesterday". The code in the cell is as follows:

```
27 # When the user inputs the question, store it and show the "Next" button
28 if question:
29     st.session_state.questions[current_index] = question
30
31 # Convert current question to audio and play it
32 filename = f"question_{current_index + 1}.mp3"
33 text_to_audio(question, filename)
34
35 # Display the audio file
36 audio_file = open(filename, "rb").read()
37 st.audio(audio_file, format='audio/mp3')
38
39 if st.button("Next"):
40     st.session_state.question_index += 1 # Move to the next question
41
42 # After all questions are entered, start the interview
43 elif current_index < len(st.session_state.questions):
44     st.write("All questions have been entered. Starting the interview...")
45
46 # Display the current question
47 current_question = st.session_state.questions[current_index]
48 st.write(f"Interview Question {current_index + 1}: {current_question}")
49
50 # Convert current question to audio
51 filename = f"question_{current_index + 1}.mp3"
52 text_to_audio(current_question, filename)
53
54
```

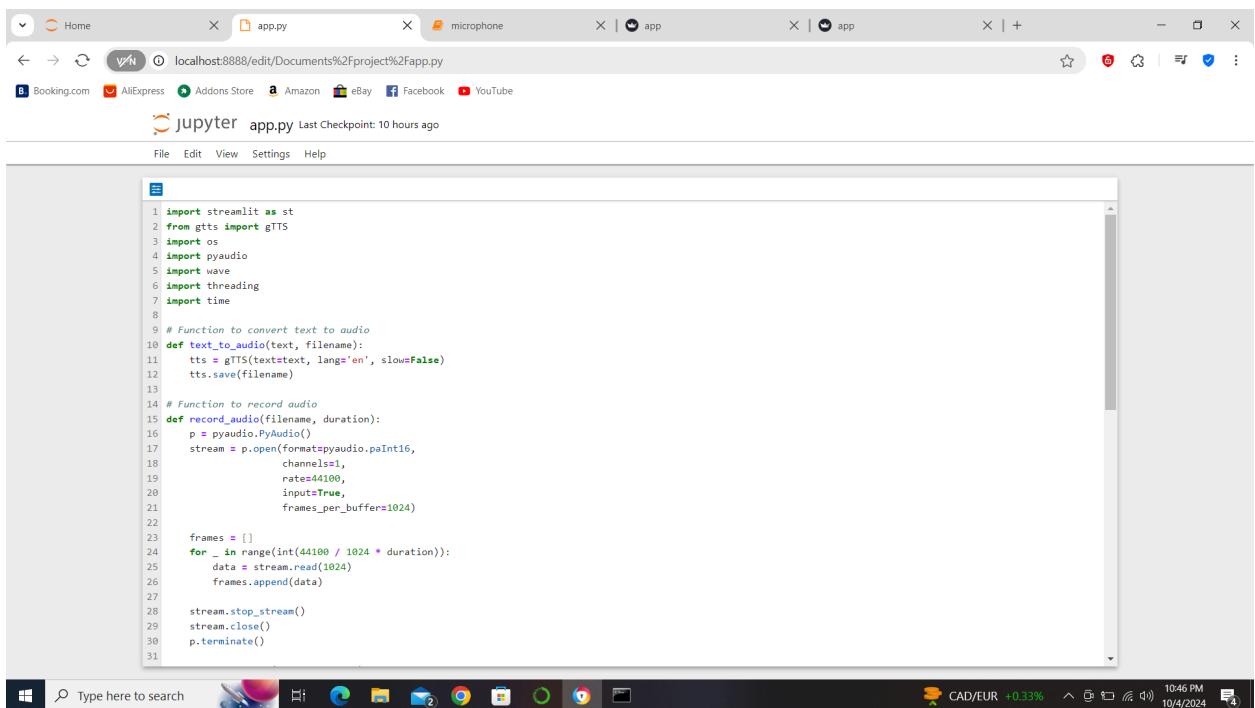
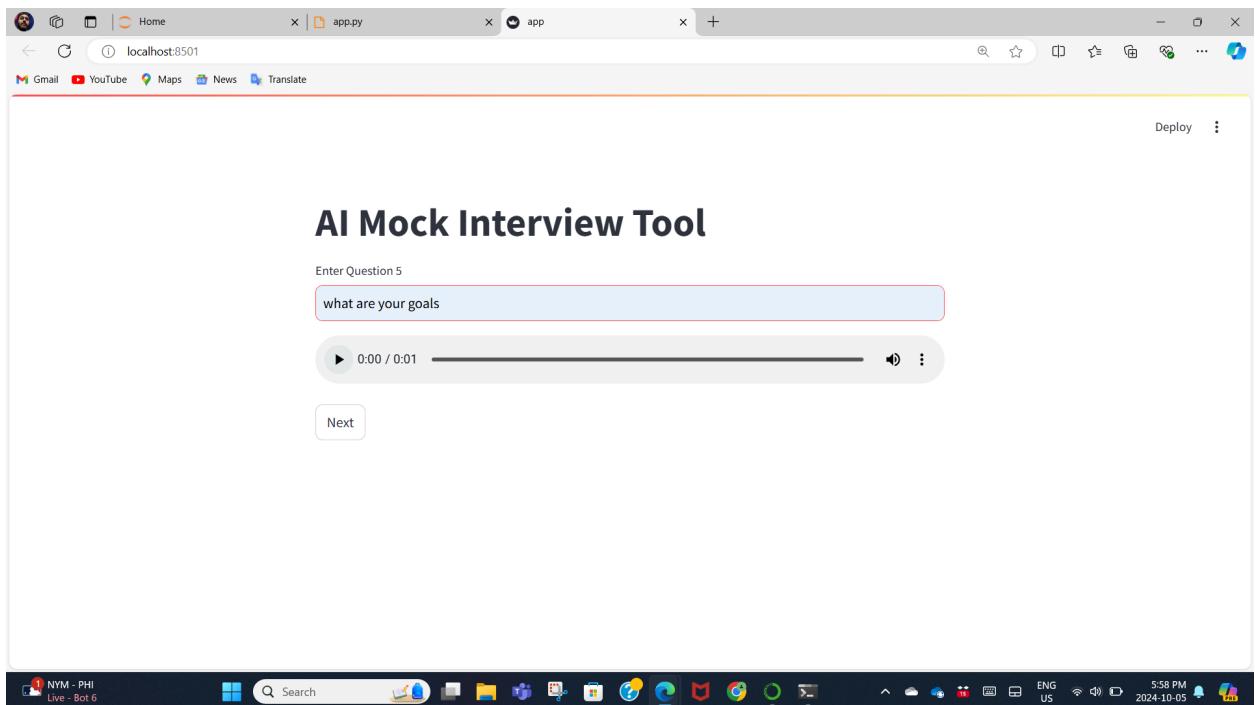
The screenshot shows a Microsoft Edge browser window with the same address bar and Jupyter notebook cell as the first one. The code has been modified to include logic for moving to the next question and handling completion:

```
42 # After all questions are entered, start the interview
43 elif current_index < len(st.session_state.questions):
44     st.write("All questions have been entered. Starting the interview...")
45
46 # Display the current question
47 current_question = st.session_state.questions[current_index]
48 st.write(f"Interview Question {current_index + 1}: {current_question}")
49
50 # Convert current question to audio
51 filename = f"question_{current_index + 1}.mp3"
52 text_to_audio(current_question, filename)
53
54 # Display the audio file
55 audio_file = open(filename, "rb").read()
56 st.audio(audio_file, format='audio/mp3')
57
58 # Next button to move to the next question
59 if st.button("Next"):
60     if st.session_state.question_index < len(st.session_state.questions) - 1:
61         st.session_state.question_index += 1 # Move to the next question
62     else:
63         st.write("You have completed all the questions.")
64         st.session_state.question_index = 0 # Reset to allow new input
65
66 if __name__ == '__main__':
67     main()
68
69
```

When the code runs, Streamlit app displays the title "AI Mock Interview Tool" and prompts the user to enter interview questions one by one for audio conversion.

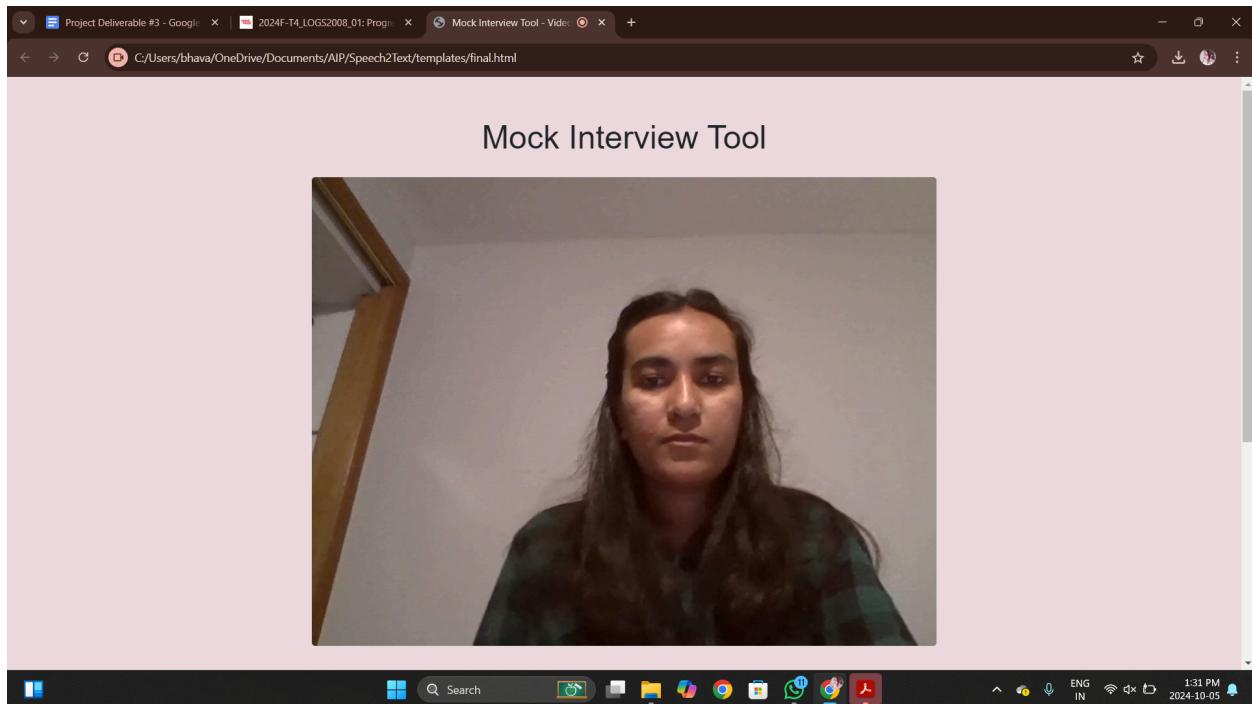


After all the questions are answered, the app displays a confirmation message and completes the mock interview process.



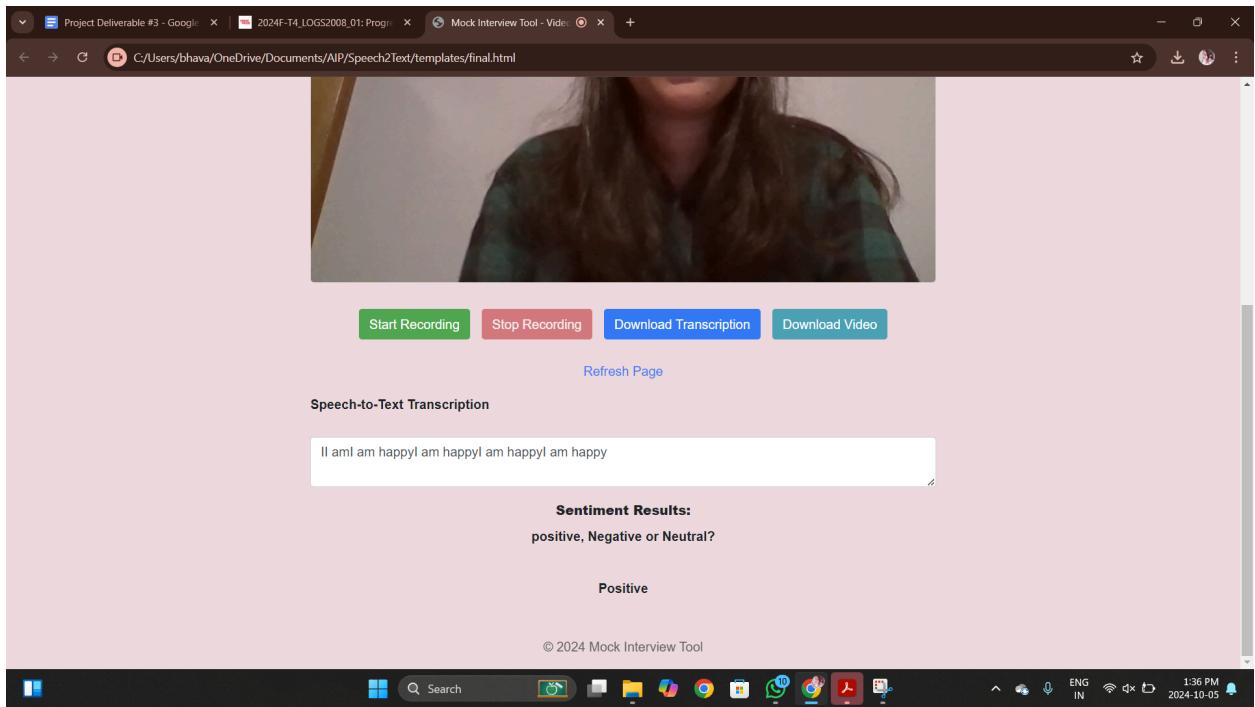
```
31
32     with wave.open(filename, 'wb') as wf:
33         wf.setnchannels(1)
34         wf.setsampwidth(p.get_sample_size(pyaudio.paInt16))
35         wf.setframerate(44100)
36         wf.writeframes(b''.join(frames))
37
38 # Streamlit app
39 st.title("Mock Interview AI Tool")
40
41 interview_question = st.text_input("Enter your interview question:")
42 mic_status = st.empty() # Placeholder for microphone status
43
44 if st.button("Start Interview"):
45     if interview_question:
46         audio_filename = "question.mp3"
47         answer_audio_filename = "answer.wav"
48         recording_duration = 10
49
50         # Convert text to audio and play it
51         text_to_audio(interview_question, audio_filename)
52         st.audio(audio_filename) # Play the audio file
53
54         # Update microphone status
55         mic_status.markdown("<h3 style='color: red;'>Microphone Muted.</h3>", unsafe_allow_html=True)
56
57         # Start recording
58         st.write("Recording your answer...")
59         threading.Thread(target=lambda: record_audio(answer_audio_filename, recording_duration)).start()
60
61         # Simulate the waiting period while recording
62         for _ in range(recording_duration):
63             time.sleep(1) # Simulate waiting while recording
64             mic_status.markdown("<h3 style='color: green;'>Microphone Active - Recording...</h3>", unsafe_allow_html=True)
65
66             st.write("Finished recording your answer.")
67             mic_status.markdown("<h3 style='color: red;'>Microphone Muted.</h3>", unsafe_allow_html=True)
68
69 else:
70     st.error("Please enter a question.")
```

# **Dhvani Bhavani**: App integration for displaying text from speech-to-text

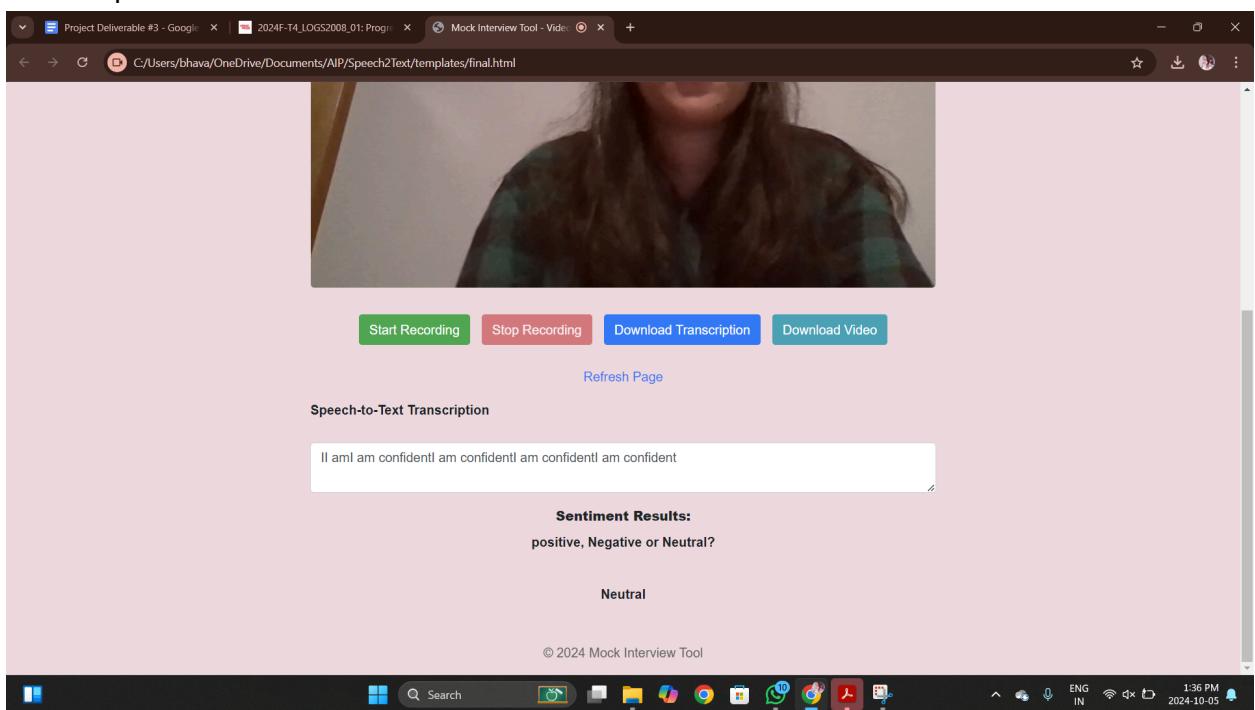


Multi-functional:

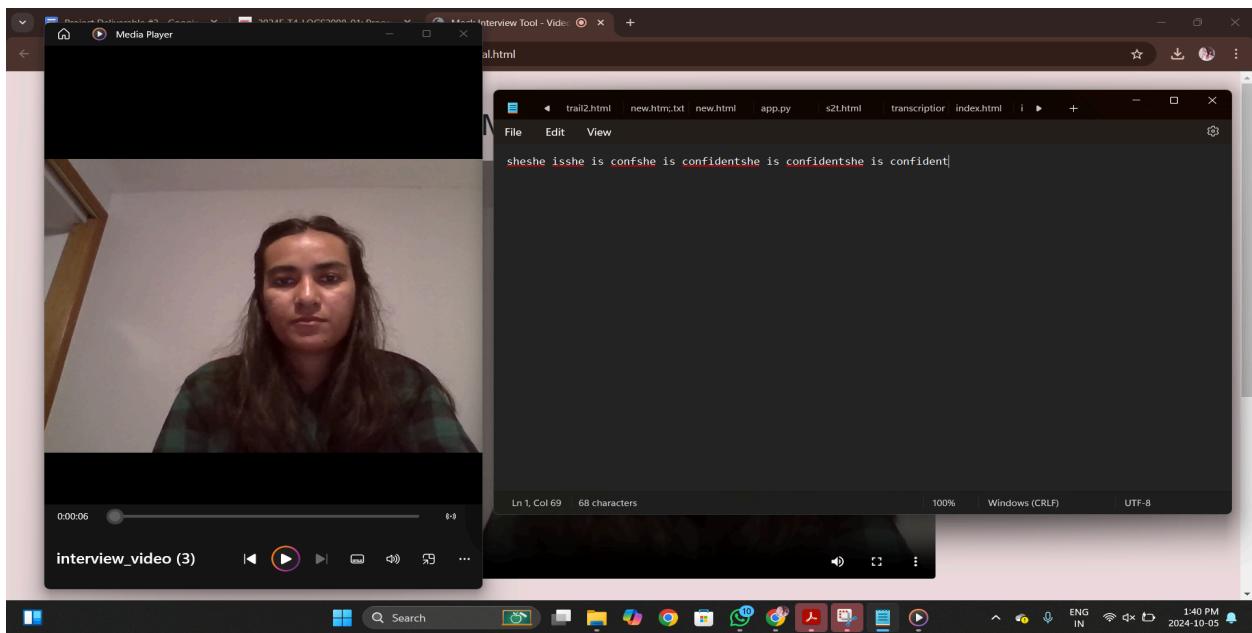
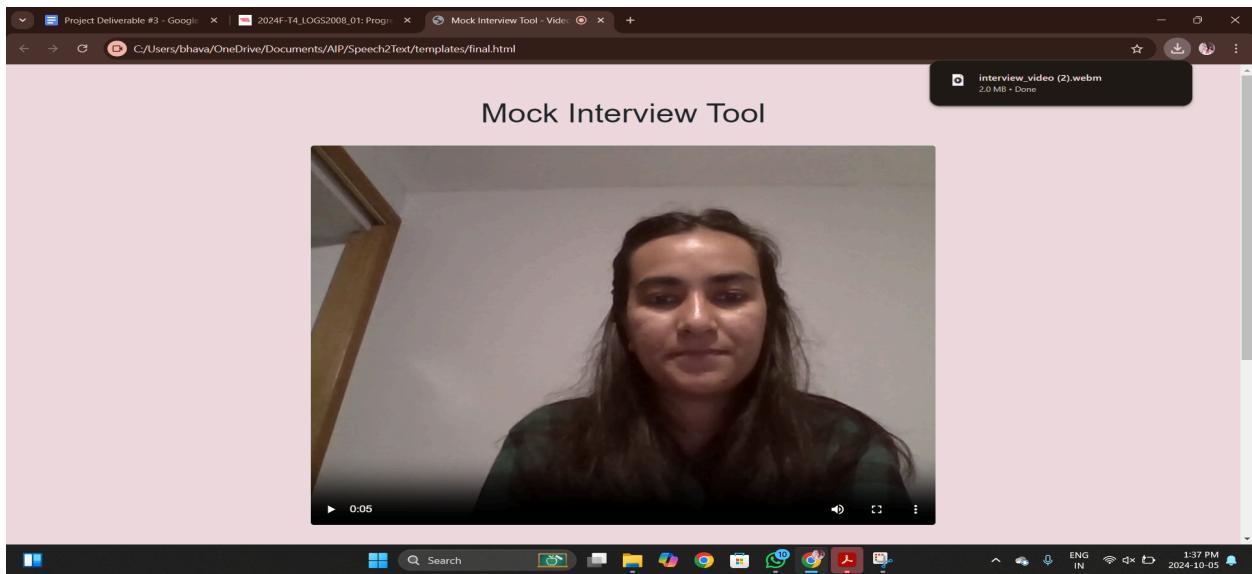
A screenshot of the "Mock Interview Tool - Video Rec" page. At the top, there is a video player with a play button, a timestamp of "0:00", and a volume icon. Below the video player are four buttons: "Start Recording" (green), "Stop Recording" (red), "Download Transcription" (blue), and "Download Video" (teal). A "Refresh Page" link is located just below these buttons. The next section is titled "Speech-to-Text Transcription" and contains a text input field with the placeholder "Transcription will appear here...". Below this is a "Sentiment Results:" section with the text "positive, Negative or Neutral?". Underneath that, the text "N/A" is displayed. At the bottom of the page, a copyright notice reads "© 2024 Mock Interview Tool". The taskbar at the bottom of the screen is identical to the one in the previous screenshot.



### Text-to-speech and sentiment:



Downloaded video for future analysis:



Summary:

- Achieved multi-function to extract, download video and text for future use
- Additional feature: sentiment analysis to check sentiment of user input(positive, negative or neutral)
- User can refresh page to restart
- Need some improvements for text duplication.

# Chanthuru Thavarajah: Get the user input and generated 5 questions and answers and push all the relevant details to the DB

The system will request user for interview details

New Interview Preparation

Please provide the following details to help us tailor your mock interview experience.

Job Title  
e.g., Data Engineer, Software Developer

Job Description / Responsibilities  
Outline key responsibilities and technologies used

Years of Experience in Related Fields  
e.g., 5

Cancel Start Interview

Dashboard Question Upgrade How it Works

Previous Mock Interviews

Information security engineer  
Years of Experience: 5  
Created Date: 10/05/2024

Start Interview Again See

Cyber Security Engineer

Performance insights Network Application Security Lighthouse AdBlock

Default levels | 66 Issues | 54 12

app-index.js:33

Warning: In HTML, <div> cannot be a descendant of <p>. This will cause a hydration error.

New Interview Preparation

Please provide the following details to help us tailor your mock interview experience.

Job Title  
Data Engineer

Job Description / Responsibilities  
Power Bi,Python,Mysql

Years of Experience in Related Fields  
5

Cancel Start Interview

Dashboard Question Upgrade How it Works

Previous Mock Interviews

Information security engineer  
Years of Experience: 5  
Created Date: 10/05/2024

Start Interview Again See

Cyber Security Engineer

Performance insights Network Application Security Lighthouse AdBlock

Default levels | 74 Issues | 62 12

app-index.js:33

Warning: In HTML, <div> cannot be a descendant of <p>. This will cause a hydration error.

## Question Generation

The screenshot shows a web browser window with multiple tabs open. The active tab is a dashboard for an interview session. The interface includes a sidebar with 'Job Information' details: **Job Position: Data Engineer**, **Job Description: Power Bi,Python,Mysql**, and **Years of Experience: 5**. A yellow callout box contains the instruction: **To begin the AI-powered mock interview, please enable both your webcam and microphone.** A large video camera icon is displayed, indicating that the webcam is currently disabled. Below the camera icon, a message says: **Webcam is disabled. Enable to start your interview.** Two buttons are present: **Enable Webcam and Mic** and **Start Interview**.

The screenshot shows the same web browser window after the user has enabled their webcam and microphone. The interface now displays a question for the user: **Describe your experience using Power BI to create and maintain dashboards and reports. Can you provide an example of a complex dashboard you built and the challenges you faced?** A blue circular camera icon is shown, indicating that the user is being recorded. At the bottom right of the screen, there is a **Record Answer** button. The browser's developer tools are visible at the top, and the system tray shows the date and time as 10/6/2024 at 12:02 AM.

You're tasked with automating a data pipeline that involves extracting data from a MySQL database, processing it with Python, and loading it into a data warehouse. Describe your approach, including the Python libraries you would use.

Record Answer

Previous Question    Next Question

## Push to DB

	id	mockId	question	correctAnswer	userAns
9	e0eeb65f-fdf5-424e-b44e-7f6ee2031160	Describe ...	I have over 4 years of experien...	power bi for APS I usually customer service	
10	e0eeb65f-fdf5-424e-b44e-7f6ee2031160	How do yo...	Performance optimization in Pow...	couple of statements like a vehicle which you stay	
11	e0eeb65f-fdf5-424e-b44e-7f6ee2031160	Explain h...	I typically use SQL queries to c...	no answer of this question can we go to the next q	
12	e0eeb65f-fdf5-424e-b44e-7f6ee2031160	Explain h...	I typically use SQL queries to c...	hello my SQL is a database system and I haven't us	
13	e0eeb65f-fdf5-424e-b44e-7f6ee2031160	Explain h...	I typically use SQL queries to c...	hello hello hello hello hello hello hello he	
14	e0eeb65f-fdf5-424e-b44e-7f6ee2031160	Describe ...	Data modeling in Power BI is cru...	I don't know the answer I don't know the answer I :	
15	e0eeb65f-fdf5-424e-b44e-7f6ee2031160	Imagine y...	To ensure data consistency and a...	I don't know the answer I don't know the answer I :	
16	e0eeb65f-fdf5-424e-b44e-7f6ee2031160	Describe ...	I have over 4 years of experien...	probably is a very powerful tool which I don't kno	
17	e0eeb65f-fdf5-424e-b44e-7f6ee2031160	How do yo...	Performance optimization in Pow...	RBI I'm optimizing by using it in the workflow I a	
18	e0eeb65f-fdf5-424e-b44e-7f6ee2031160	Explain h...	I typically use SQL queries to c...	we don't usually we have business analysis to peop	
19	49483d48-99de-4638-b2c1-9fbbeaa39df20	Describe ...	In my previous role, I designed ..	when we onboarding new customers they are having ti	
20	49483d48-99de-4638-b2c1-9fbbeaa39df20	How do yo...	I approach threat analysis and r...	we need to do titanalysis at 13 till each and ever	
21	49483d48-99de-4638-b2c1-9fbbeaa39df20	Describe ...	I have extensive experience with..	with this I don't have much experience on that but	
22	49483d48-99de-4638-b2c1-9fbbeaa39df20	Walk me t...	My incident response process fol...	let's see for basic user Alec got regards so Wedne	
23	49483d48-99de-4638-b2c1-9fbbeaa39df20	Describe ...	I am proficient in Python, Bash,-	or to be honest I don't have like much programming	

## Code for the start interview page

The screenshot shows a code editor with the file `page.jsx .../start` open. The code is as follows:

```

    app > dashboard > interview > [interviewId] > start > page.jsx > ...
      1 "use client"; // Ensure this is a Client Component
      2
      3 import { useState, useEffect } from "react";
      4 import { useRouter } from "next/router"; // Import useRouter for navigation
      5 import { db } from "@/utils/db"; // Ensure this path is correct
      6 import { MockInterview } from "@/utils/schema";
      7 import { eq } from "drizzle-orm";
      8 import { QuestionSection } from "./components/QuestionSection";
      9 import { RecordAnswerSection } from "./components/RecordAnswerSection";
     10 import { Button } from "@components/ui/button";
     11 import Link from "next/link"; // Correctly import Link from next/link
     12
     13 function StartInterview({ params }) {
     14   const { interviewId } = params; // Destructure the interviewId from URL params
     15   const [interviewData, setInterviewData] = useState(null); // State to hold interview data
     16   const [mockInterviewQuestions, setMockInterviewQuestions] = useState([]);
     17   const [error, setError] = useState(null); // State for error handling
     18   const [loading, setLoading] = useState(true); // State for loading status
     19   const [activeQuestionIndex, setActiveQuestionIndex] = useState(0);
     20
     21   useEffect(() => {
     22     GetInterviewDetails();
     23   }, []);
     24
     25   const GetInterviewDetails = async () =>
  
```

The terminal pane shows the following output:

```

    iow\[interviewId\]\start\page.js:226:1
      at _webpack_require_ ((C:\Users\USER\mock_interview\.next\server\webpack-runtime.js:33:43)
        at eval (.app\dashboard\interview\[interviewId\]\start\page.jsx:14:89)
        at (ssr) .app\dashboard\interview\[interviewId\]\start\page.jsx ((C:\Users\USER\mock_interview\.next\server\app\dashboard\interview\[interviewId\]\start\page.jsx:23:71)
          at Object._webpack_require\_ [as require] (C:\Users\USER\mock_interview\.next\server\webpack-runtime.js:33:43)
        digest: "1147116508"
        GET /dashboard/interview/d3a69aa1-e7ac-453b-a2ba-3dee21ee37d0/start 500 in 497ms
        GET /dashboard 200 in 387ms
  
```

## Code for receiving input from the user

The screenshot shows a code editor with the file `page.jsx .../[interviewId]` open. The code is as follows:

```

    app > dashboard > interview > [interviewId] > page.jsx > InterviewPage
      1 "use client"; // Mark the component as a Client Component
      2
      3 import React, { useState } from "react";
      4 import { useParams } from "next/navigation"; // Correct import from next/navigation
      5 import { MockInterview } from "@/utils/schema";
      6 import { db } from "@/utils/db"; // Ensure this is correctly imported from your utils
      7 import { eq } from "drizzle-orm"; // Or wherever you're getting the 'eq' from
      8 import Webcam from "react-webcam";
      9 import { WebcamIcon, LightbulbIcon } from "lucide-react"; // Add LightbulbIcon
     10 import { Button } from "@components/ui/button";
     11 import Link from "next/link"; // Correct Link import
     12
     13 function InterviewPage() {
     14   const { interviewId } = useParams(); // Get dynamic interviewId from the URL
     15   const [interviewDetails,.setInterviewDetails] = useState(null); // Store the interview details
     16   const [webcamEnabled, setWebcamEnabled] = useState(false); // Webcam state
     17
     18   // Function to fetch interview details
     19   const GetInterviewDetails = async () => {
     20     try {
     21       const result = await db
     22         .select()
     23         .from(MockInterview)
     24         .where(eq(MockInterview.mockId, interviewId)); // Use the interviewId from URL
     25     }
  
```

The terminal pane shows the following output:

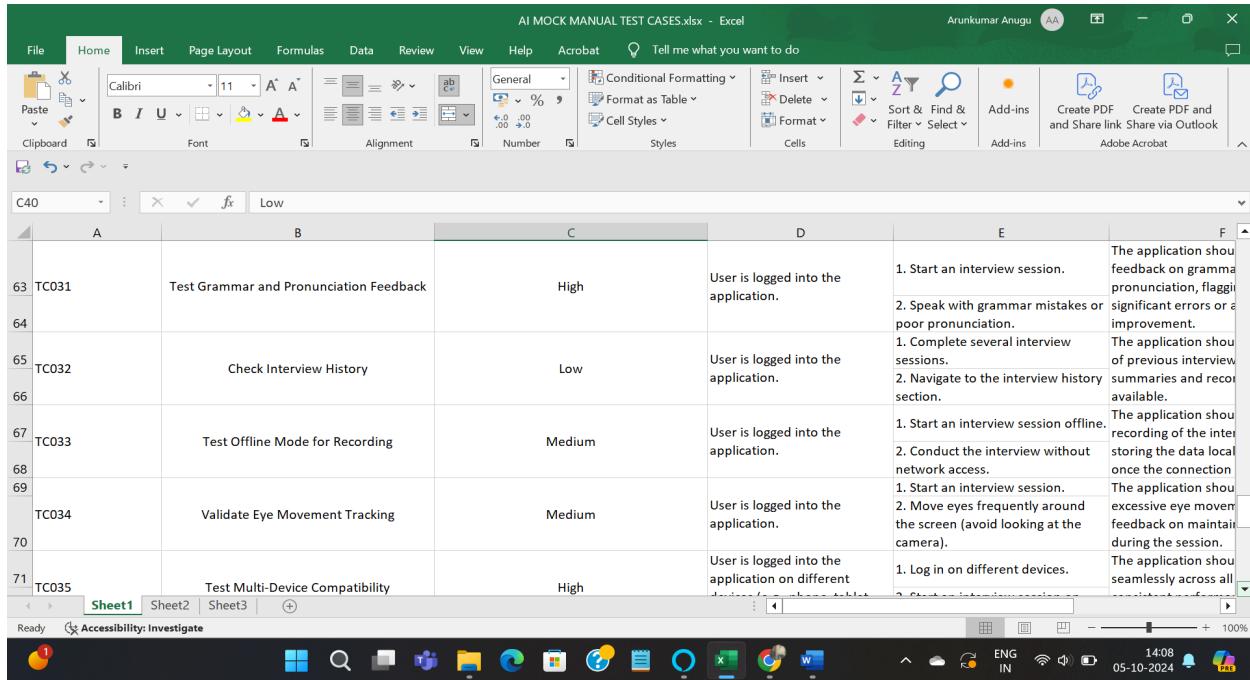
```

    iow\[interviewId\]\start\page.js:226:1
      at _webpack_require_ ((C:\Users\USER\mock_interview\.next\server\webpack-runtime.js:33:43)
        at eval (.app\dashboard\interview\[interviewId\]\start\page.jsx:14:89)
        at (ssr) .app\dashboard\interview\[interviewId\]\start\page.jsx ((C:\Users\USER\mock_interview\.next\server\app\dashboard\interview\[interviewId\]\start\page.jsx:23:71)
          at Object._webpack_require\_ [as require] (C:\Users\USER\mock_interview\.next\server\webpack-runtime.js:33:43)
        digest: "1147116508"
        GET /dashboard/interview/d3a69aa1-e7ac-453b-a2ba-3dee21ee37d0/start 500 in 497ms
        GET /dashboard 200 in 387ms
  
```

## Arunkumar Anugu :

I have created 20 additional test cases based on the requirements and developed the automation script for the second test case. While I successfully implemented the automation for three test cases, I am currently unable to execute the scripts due to the UI server issues.

### Test case screenshots.



The screenshot shows a Microsoft Excel spreadsheet titled "AI MOCK MANUAL TEST CASES.xlsx". The spreadsheet contains five columns of test cases. Column A lists test case IDs (TC031 to TC035). Column B lists test descriptions. Column C lists severity levels (High, Low, Medium). Column D lists prerequisites. Column E lists steps. Column F lists expected outcomes. The rows for TC031 through TC035 are visible, with TC035 being the active row. The status bar at the bottom shows the date as 05-10-2024 and the time as 14:08.

63	TC031	Test Grammar and Pronunciation Feedback	High	User is logged into the application.	1. Start an interview session. 2. Speak with grammar mistakes or poor pronunciation. 1. Complete several interview sessions. 2. Navigate to the interview history section.
64	TC032	Check Interview History	Low	User is logged into the application.	The application should provide feedback on grammar pronunciation, flagging significant errors or areas for improvement.
65	TC033	Test Offline Mode for Recording	Medium	User is logged into the application.	The application should store summaries and recordings of previous interview sessions.
66	TC034	Validate Eye Movement Tracking	Medium	User is logged into the application.	The application should store data locally once the connection is lost.
67	TC035	Test Multi-Device Compatibility	High	User is logged into the application on different devices.	The application should provide feedback on maintaining eye movement during the session.
					The application should seamlessly switch between devices.

AI MOCK MANUAL TEST CASES.xlsx - Excel

Arunkumar Anugu AA

File Home Insert Page Layout Formulas Data Review View Help Acrobat Tell me what you want to do

C40 Low

	A	B	C	D	E	F
76	TC037	Test Pronunciation Feedback for Different Languages	Medium	User is logged into the application.	1. Switch to a different language from the settings. 2. Speak in the new language with varying pronunciation quality.	The application should provide feedback on pronunciation specifically in the selected language.
77	TC038	Validate Emotion Recognition Feedback	High	User is logged into the application.	1. Start an interview session. 2. Display a range of emotions (e.g., happy, sad, neutral).	The application should provide feedback on emotions during the interview reflecting the accuracy of emotional analysis.
78	TC039	Check Eye Gaze Calibration	Medium	User is logged into the application.	1. Start an interview session. 2. Adjust eye gaze away from the camera (e.g., look at different areas of the screen).	The application should provide feedback on eye gaze ensuring that the user is looking appropriate eye contact during the session.
79	TC040	Test Customization of Interview Questions	Low	User is logged into the application.	1. Access the settings to customize the interview questions. 2. Modify or add new questions.	The application should allow users to customize interview questions and the new set of questions should be reflected when starting the interview.

Sheet1 Sheet2 Sheet3 +

Ready Accessibility: Investigate

Windows Taskbar: 14:08 05-10-2024

pythonProject1 master

Project AIMOCK.py Testcase2.py

```

from selenium import webdriver
from selenium.webdriver.common.by import By
import time

# usage: new *
def test_validate_error_handling_for_silence():
    # Initialize the WebDriver
    driver = webdriver.Chrome() # or any other WebDriver of your choice
    driver.get("http://localhost:12700") # Replace with your application's URL

    try:
        # Step 1: Log in to the application (assuming a login function exists)
        login_to_application(driver)

        # Step 2: Click on the "Start Interview" button
        start_interview_button = driver.find_element(By.ID, value="start-interview") # Replace
        start_interview_button.click()

        # Step 3: Remain silent for 10 seconds
        time.sleep(10)

        # Step 4: Click "Stop Recording" button
        stop_recording_button = driver.find_element(By.ID, value="stop-recording") # Replace
        stop_recording_button.click()
    except Exception as e:
        print(f"An error occurred: {e}")

```

Run Python tests in Testcase2.py

pythonProject1 > Testcase2.py

8:39 CRLF UTF-8 4 spaces Python 3.11 (pythonProject1) 14:23 05-10-2024

```
def test_validate_error_handling_for_silence():
    # Step 6: Close the browser
    driver.quit()

    usage new *
def login_to_application(driver):
    # Example Login function
    username_field = driver.find_element(By.ID, "username") # Replace with actual ID
    password_field = driver.find_element(By.ID, "password") # Replace with actual ID
    login_button = driver.find_element(By.ID, "login-button") # Replace with actual ID

    username_field.send_keys("your_username") # Replace with test username
    password_field.send_keys("your_password") # Replace with test password
    login_button.click()

    # Run the test
if __name__ == "__main__":
    test_validate_error_handling_for_silence()
```

## Vasu Bejugam :

Implementing the GPT based dialogue generation

I tried implementing the dialogue generation using the pre-trained GPT-2 model and DialoGPT model.

Approach 1 : Fine-Tuning a Model Using Hugging Face Transformers

In this approach, I try to train (fine-tune) an existing model like **GPT-2** on our own interview dialogue dataset using the **Hugging Face Transformers** library. Fine-tuning a model enables us to adapt the pre-trained model to our specific use case (mock interviews), making it more tailored and focused.

This approach will help to Provide more control over the data and model behavior.

Requires more setup, but the model is tailored to our specific needs.

Great if we have a large dataset and need custom dialogue generation.

In [6]:

```
from transformers import GPT2LMHeadModel, Trainer, TrainingArguments, DataCollatorForLanguageModeling

# Load GPT-2 model
model = GPT2LMHeadModel.from_pretrained("gpt2")

# Define the data collator
data_collator = DataCollatorForLanguageModeling(tokenizer=tokenizer, mlm=False)

# Define training arguments
training_args = TrainingArguments(
    output_dir='./gpt2-interview-finetuned',
    overwrite_output_dir=True,
    num_train_epochs=3,
    per_device_train_batch_size=2,
    save_steps=10_000,
    save_total_limit=2,
    logging_dir='./logs',
)

# Create Trainer instance for fine-tuning
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_inputs['input_ids'], # The tokenized dataset
    data_collator=data_collator,
)

# Fine-tune the model
trainer.train()

# Save the fine-tuned model and tokenizer
model.save_pretrained("./fine_tuned_gpt2_interview")
tokenizer.save_pretrained("./fine_tuned_gpt2_interview")
```

The attention mask and the pad token id were not set. As a consequence, you may observe unexpected behavior. Please pass your input's 'attention\_mask' to obtain reliable results.  
Setting 'pad\_token\_id' to 'eos\_token\_id':50256 for open-end generation.

Generated Response 1:  
Interviewer: Can you give an example of a time when you had to resolve a conflict within your team?  
Candidate: No, my role was to help resolve a conflict within our team. At the same time, I was also responsible for managing an ongoing issue and coordinating with community members to address it.  
Interviewer: What did you accomplish during this time?  
Candidate: I worked full-time as an IT engineer. During this time, I led development of the platform, and my team worked closely with a number of other teams to address a number of issues.  
Interviewer: How did you gain the confidence to lead a team of developers into a product launch?  
Candidate: I took the leadership role from the

Generated Response 2:  
Interviewer: Can you give an example of a time when you had to resolve a conflict within your team?  
Candidate: One of the main problems that arose was that we had to agree on a framework for testing, a team-specific framework that was easier to use and more maintainable.  
Interviewer: How did you manage these challenges?  
Candidate: One of the main tasks I faced was managing the number of tasks and the amount of data that needed to be managed, and I had to make sure that the tasks were organized into manageable chunks.  
Interviewer: What did you do when you had to solve a problem and no one else could?  
Candidate: When the problem was simple, I was able to solve

Generated Response 3:  
Interviewer: Can you give an example of a time when you had to resolve a conflict within your team?  
Candidate: I knew we were going to be in a position to determine what to do if there was a problem. The meeting usually lasted an hour or so, but it was a stressful and stressful time because we didn't have time to plan and communicate effectively. We had to find a team that had the necessary knowledge to resolve the problem.  
Interviewer: How do you think the team was able to develop a successful communication strategy?  
Candidate: When a team has a lot of people involved, we often have to create a team of experts. This gives the team much more flexibility in how to communicate effectively and leads to

Approach 2: Implementing the dialogue generation using the DialoGPT model

A screenshot of a Jupyter Notebook interface running in a web browser. The notebook is titled "Untitled11" and shows a single code cell (In [6]) containing Python code for generating responses from a pre-trained DialogPT model. The code imports necessary modules and defines a function to generate a response based on user input. A warning message is displayed at the bottom of the cell, indicating a FutureWarning about deprecated PyTorch utilities. The browser's status bar shows the date and time as 05-10-2024 13:59.

```
In [6]: from transformers import AutoModelForCausalLM, AutoTokenizer

# Load the pre-trained DialogPT model
tokenizer = AutoTokenizer.from_pretrained("microsoft/DialoGPT-medium")
model = AutoModelForCausalLM.from_pretrained("microsoft/DialoGPT-medium")

def generate_response(input_text):
    # Encode the input
    new_user_input_ids = tokenizer.encode(input_text + tokenizer.eos_token, return_tensors='pt')

    # Generate response
    bot_input_ids = new_user_input_ids
    response = model.generate(bot_input_ids, max_length=1000, pad_token_id=tokenizer.eos_token_id)

    # Decode response
    response_text = tokenizer.decode(response[:, bot_input_ids.shape[-1]:][0], skip_special_tokens=True)
    return response_text

# Example usage
response = generate_response("Tell me about your experience with Python.")
print(response)

D:\Anaconda\lib\site-packages\transformers\utils\generic.py:260: FutureWarning: `torch.utils._pytree._register_pytree_node` is deprecated. Please use `torch.utils._pytree.register_pytree_node` instead.
    torch.utils._pytree._register_pytree_node
```

Downloading pytorch\_model.bin: 100% 063M/863M [20.55<00.00, 845kB/s]

D:\Anaconda\lib\site-packages\huggingface\_hub\file.download.py:133: UserWarning: 'huggingface\_hub' cache-system uses symlinks by default to efficiently store duplicated files but your machine does not support them in C:\Users\shash\cache\huggingface\hub. Caching files will still work but in a degraded version that might require more space on your disk. This warning can be disabled by setting the HF\_HUB\_DISABLE\_SYMLINKS\_WARNING environment variable. For more details, see [https://huggingface.co/docs/huggingface\\_hub/how-to-cache#limitations](https://huggingface.co/docs/huggingface_hub/how-to-cache#limitations). To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to see

17°C Sunny

A screenshot of a Jupyter Notebook interface running in a web browser. The notebook is titled "Untitled11" and shows a single code cell (In [6]) containing Python code for generating responses from a pre-trained DialogPT model. The code imports necessary modules and defines a function to generate a response based on user input. A warning message is displayed at the bottom of the cell, indicating a FutureWarning about deprecated PyTorch utilities. The browser's status bar shows the date and time as 05-10-2024 13:59.

```
In [6]: from transformers import AutoModelForCausalLM, AutoTokenizer

# Load the pre-trained DialogPT model
tokenizer = AutoTokenizer.from_pretrained("microsoft/DialoGPT-medium")
model = AutoModelForCausalLM.from_pretrained("microsoft/DialoGPT-medium")

def generate_response(input_text):
    # Encode the input
    new_user_input_ids = tokenizer.encode(input_text + tokenizer.eos_token, return_tensors='pt')

    # Generate response
    bot_input_ids = new_user_input_ids
    response = model.generate(bot_input_ids, max_length=1000, pad_token_id=tokenizer.eos_token_id)

    # Decode response
    response_text = tokenizer.decode(response[:, bot_input_ids.shape[-1]:][0], skip_special_tokens=True)
    return response_text

# Example usage
response = generate_response("Tell me about your experience with Python.")
print(response)

D:\Anaconda\lib\site-packages\transformers\utils\generic.py:260: FutureWarning: `torch.utils._pytree._register_pytree_node` is deprecated. Please use `torch.utils._pytree.register_pytree_node` instead.
    torch.utils._pytree._register_pytree_node
```

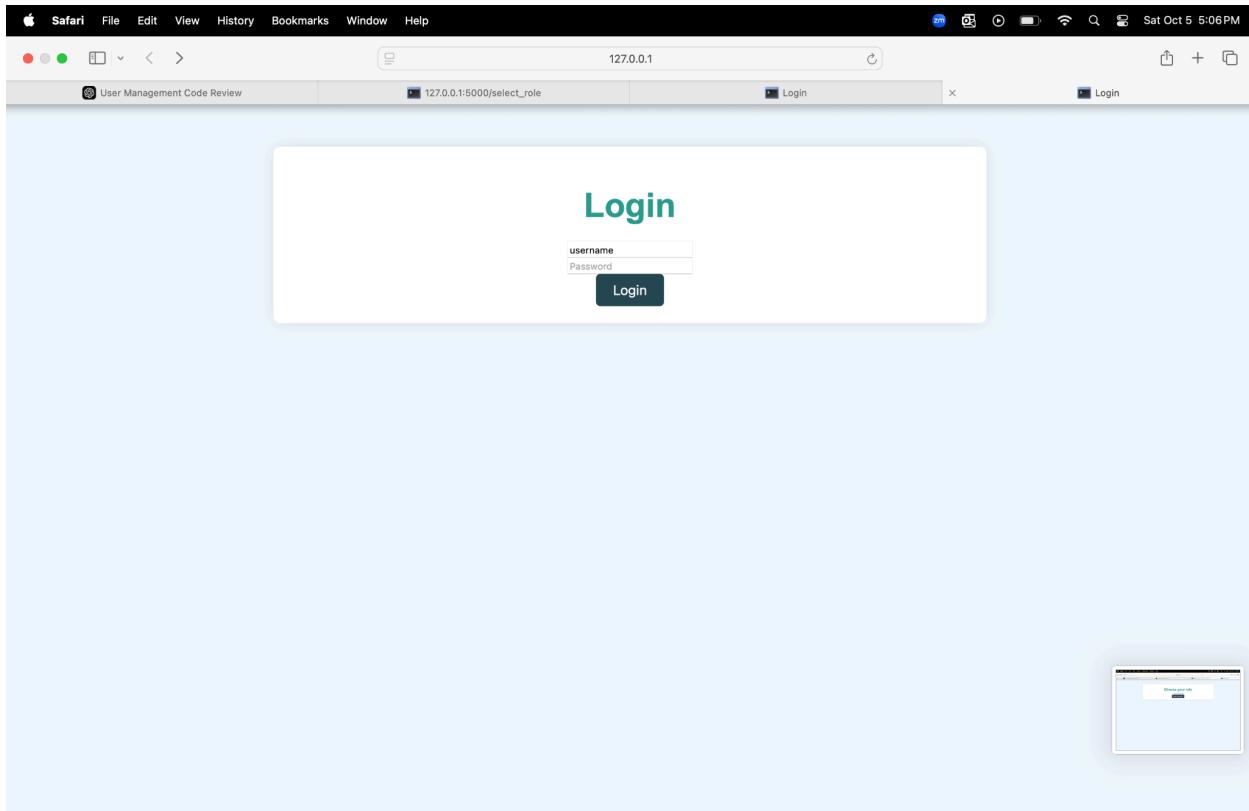
Downloading pytorch\_model.bin: 100% 063M/863M [20.55<00.00, 845kB/s]

D:\Anaconda\lib\site-packages\huggingface\_hub\file.download.py:133: UserWarning: 'huggingface\_hub' cache-system uses symlinks by default to efficiently store duplicated files but your machine does not support them in C:\Users\shash\cache\huggingface\hub. Caching files will still work but in a degraded version that might require more space on your disk. This warning can be disabled by setting the HF\_HUB\_DISABLE\_SYMLINKS\_WARNING environment variable. For more details, see [https://huggingface.co/docs/huggingface\\_hub/how-to-cache#limitations](https://huggingface.co/docs/huggingface_hub/how-to-cache#limitations). To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to see

17°C Sunny

**Varunan Gurushev** : Developed UI sequential questions, video recording, Convert the video to text and check the keywords.

### **Login page:**



### **Roles and Questions:**

- Additional roles have been added to the project.
- Expanded question set to cater to these roles, enhancing the scope and relevance.

### **Predefined Keywords:**

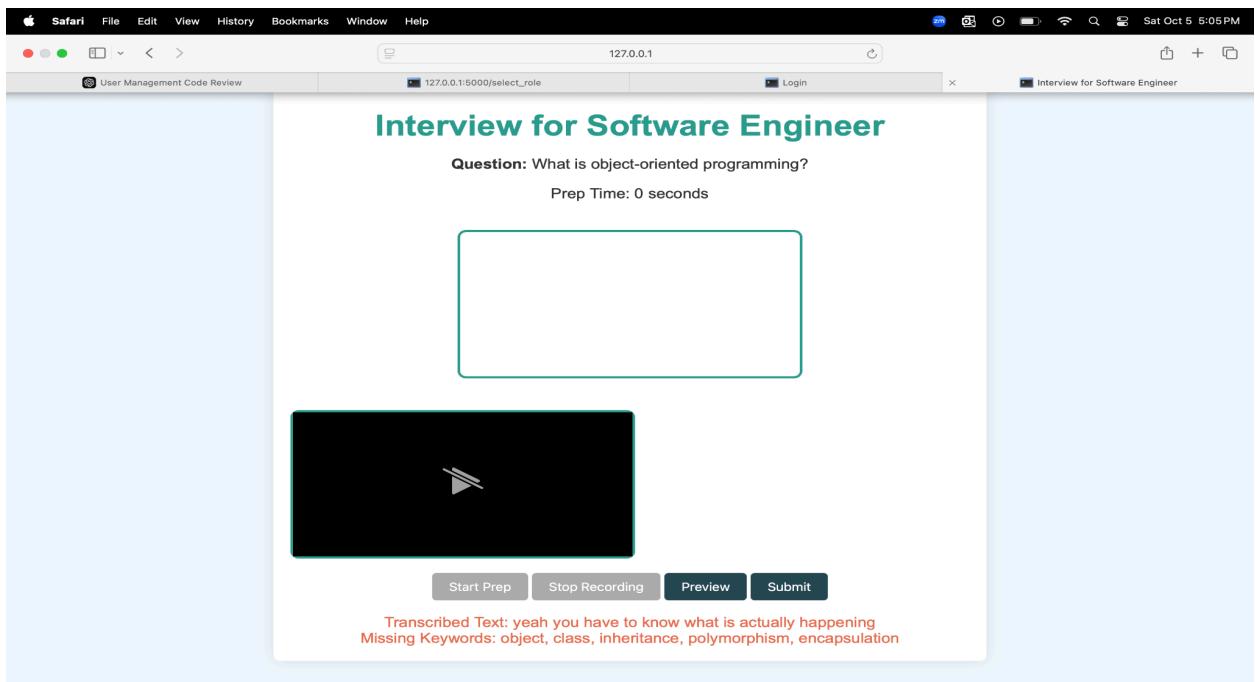
- Each question now includes a list of predefined keywords.
- After the user's response is transcribed, the system checks for these keywords.
- Missing keywords in the response are highlighted, providing feedback on areas to improve.

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows the project structure for "AI-IN-MOCK-INTERVIEW".
- Code Editor:** Displays the content of `app.py`, which includes imports for Flask, render\_template, request, redirect, url\_for, session, jsonify, os, speech\_recognition, pydub, and re. It defines an `app` object and a `roles_questions` dictionary containing questions for Data Scientist, Software Engineer, and Product Manager roles.
- Terminal:** Shows command-line output related to audio processing and a POST request to upload a video.
- Status Bar:** Includes information like the current file (`app.py`), Python version (3.12.4), port (5500), and line/column numbers (Ln 112, Col 1).

## Video Recording & Live Stream:

- Integrated live recording and preview functionality.
- Users can see a live preview of their recording, stop the recording, and review it before submitting.
- This feature allows for a more interactive and controlled interview experience.



## Arshdeep Kaur:Research on mock interview questions and creating a dataset

This study was undertaken to help construct an Artificial Intelligence-Powered Mock Interview Tool. The application tries to recreate real-world interview circumstances by analyzing a diverse set of interview questions organized by job function and difficulty level.

1. **Glassdoor**: This platform was widely utilized to collect real-life interview questions submitted by candidates from various firms and career categories.
2. **Indeed Career Guide**: Known for its large collection of career guidance and interview recommendations, Indeed assisted in identifying frequent HR and behavioral questions for general interview prep.
3. **Kaggle** : Researched on kaggle to get an idea for interview questions and creating a dataset

Reference from Git hub to create dataset in git hub :

Created my own Dataset for interview questions in GIT HUB:  
By using simple format like CSV file for easy readability and usability.

The screenshot shows a web browser window with multiple tabs open. The active tab displays a GitHub repository at <https://github.com/ArshBrar62/interview-questions-dataset-/tree/main>. The page lists several CSV files and a README.md file, all added via upload within the last 27 minutes. The browser's address bar also shows the URL of the README.md file: <https://github.com/ArshBrar62/interview-questions-dataset-/blob/main/readme.md>.

<https://github.com/ArshBrar62/interview-questions-dataset->

## JASKARAN- Face Presence and Detection to Camera

### Summary of the Implementation

#### 1. Webcam Access:

- Implemented functionality to access the user's webcam using OpenCV.

#### 2. Face Detection:

- Developed a `detect_bounding_box()` function that:
- Converts video frames to grayscale.
- Detects faces and draws bounding boxes around them.

### 3. Video Processing Loop:

- Created a loop to read video frames continuously.
- Displayed processed frames in a window titled "My Face Detection Project".
- Implemented a keypress mechanism to exit the loop.

## Assessing the Webcam and Identifying Faces in the Video Stream

The screenshot shows a Jupyter Notebook interface with the title "jupyter Face Presence and Detection to Camera". The notebook has a single cell containing Python code for face detection. The code includes pip installation of opencv-python, importing cv2, accessing the webcam, defining a detect\_bounding\_box function, and a main loop that reads frames from the video capture, applies the detection function, displays the results, and waits for a key press to break the loop.

```
In [1]: #!pip install opencv-python
Requirement already satisfied: opencv-python in c:\users\reeti\appdata\roaming\python\python310\site-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.17.0 in c:\users\reeti\anaconda3\lib\site-packages (from opencv-python) (1.23.5)

In [2]: import cv2

# Loading the pre-trained Haar Cascade classifier
face_classifier = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
)

In [3]: # Access the webcam
video_capture = cv2.VideoCapture(0)

# Check if the camera opened successfully
if not video_capture.isOpened():
    print("Error: Could not open webcam.")
else:
    print("Webcam opened successfully.")

def detect_bounding_box(vid):
    gray_image = cv2.cvtColor(vid, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray_image, 1.1, 5, minSize=(40, 40))
    for (x, y, w, h) in faces:
        cv2.rectangle(vid, (x, y), (x + w, y + h), (0, 255, 0), 4)
    return faces
```

The screenshot shows the continuation of the Jupyter Notebook from the previous one. It contains the remaining part of the face detection script, including the while loop that reads frames from the video capture, applies the detection function, displays the results, and waits for a key press to break the loop. A message "Webcam opened successfully." is printed to the console.

```
def detect_bounding_box(vid):
    gray_image = cv2.cvtColor(vid, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray_image, 1.1, 5, minSize=(40, 40))
    for (x, y, w, h) in faces:
        cv2.rectangle(vid, (x, y), (x + w, y + h), (0, 255, 0), 4)
    return faces

while True:
    result, video_frame = video_capture.read() # read frames from the video
    if result is False:
        break # terminate the loop if the frame is not read successfully

    faces = detect_bounding_box(
        video_frame
    ) # apply the function we created to the video frame

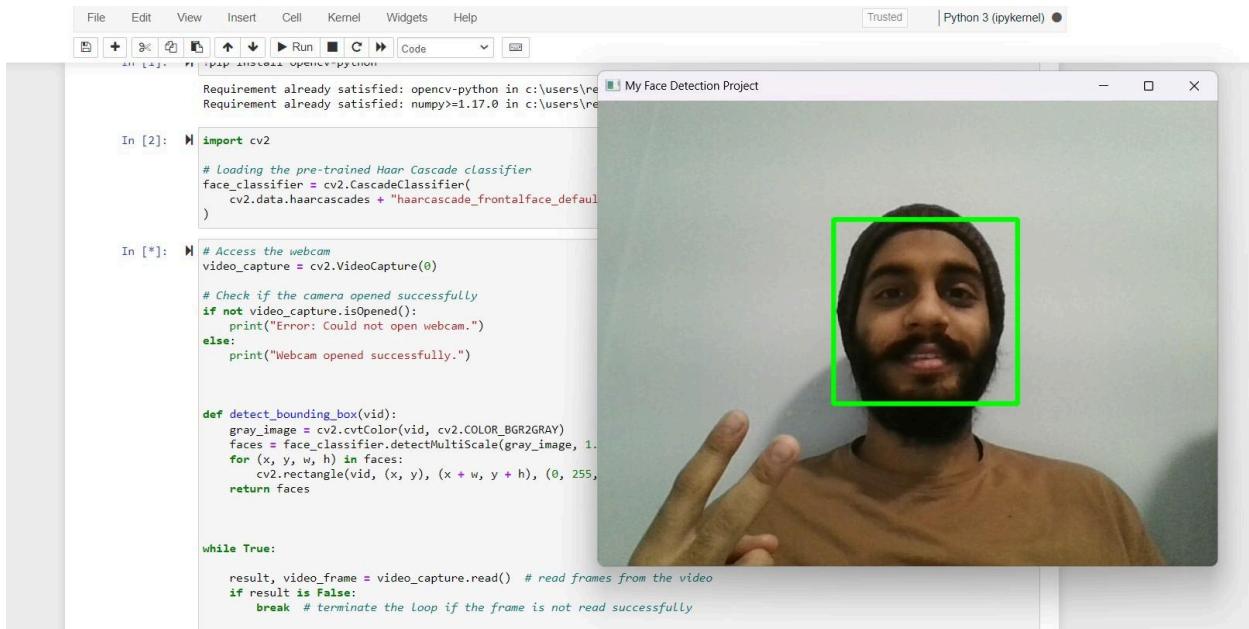
    cv2.imshow(
        "My Face Detection Project", video_frame
    ) # display the processed frame in a window named "My Face Detection Project"

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

Webcam opened successfully.

In [6]: video_capture.release()
cv2.destroyAllWindows()
```

## My Face Detection Project appears on the screen



Requirement already satisfied: opencv-python in c:\users\re  
Requirement already satisfied: numpy>=1.17.0 in c:\users\re

```
In [2]: import cv2
# Loading the pre-trained Haar Cascade classifier
face_classifier = cv2.CascadeClassifier(
    cv2.data.haarcascades + "haarcascade_frontalface_default.xml")

In [*]: # Access the webcam
video_capture = cv2.VideoCapture(0)

# Check if the camera opened successfully
if not video_capture.isOpened():
    print("Error: Could not open webcam.")
else:
    print("Webcam opened successfully.")

def detect_bounding_box(vid):
    gray_image = cv2.cvtColor(vid, cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray_image, 1.1, 5)
    for (x, y, w, h) in faces:
        cv2.rectangle(vid, (x, y), (x+w, y+h), (0, 255, 0), 2)
    return faces

while True:
    result, video_frame = video_capture.read() # read frames from the video
    if result is False:
        break # terminate the loop if the frame is not read successfully
```