AI Mock Interview Tool

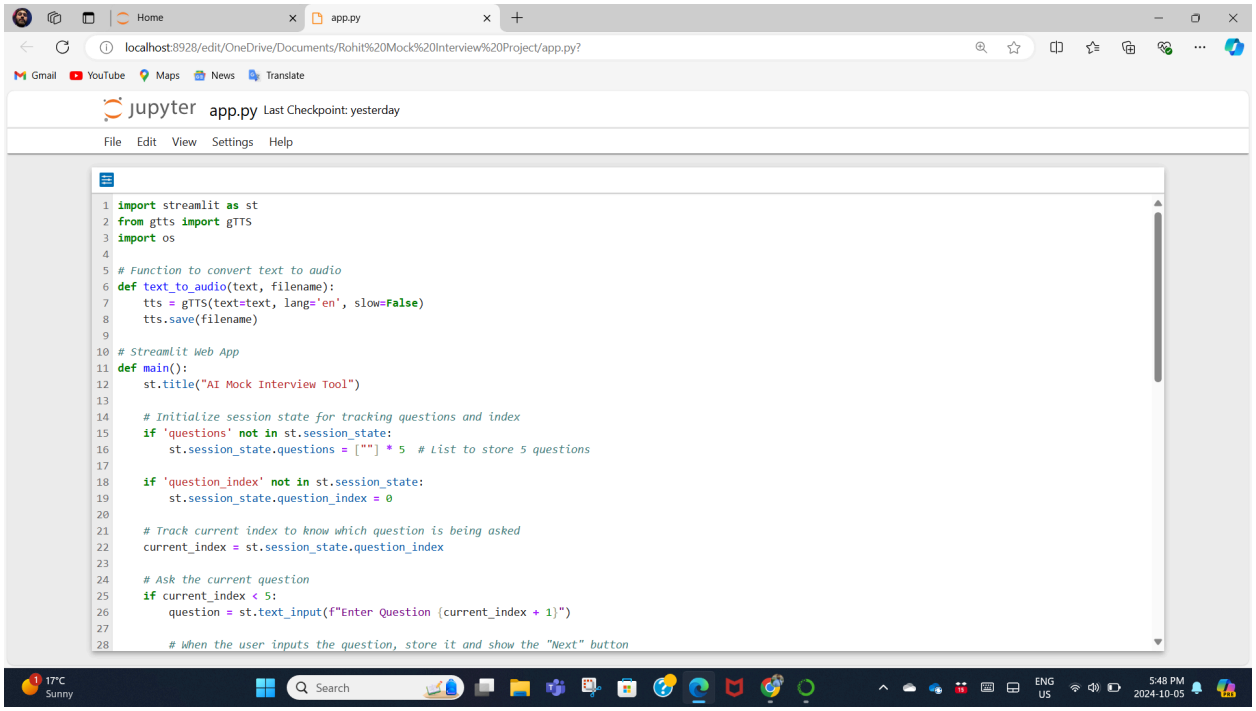
Project Deliverable #3:

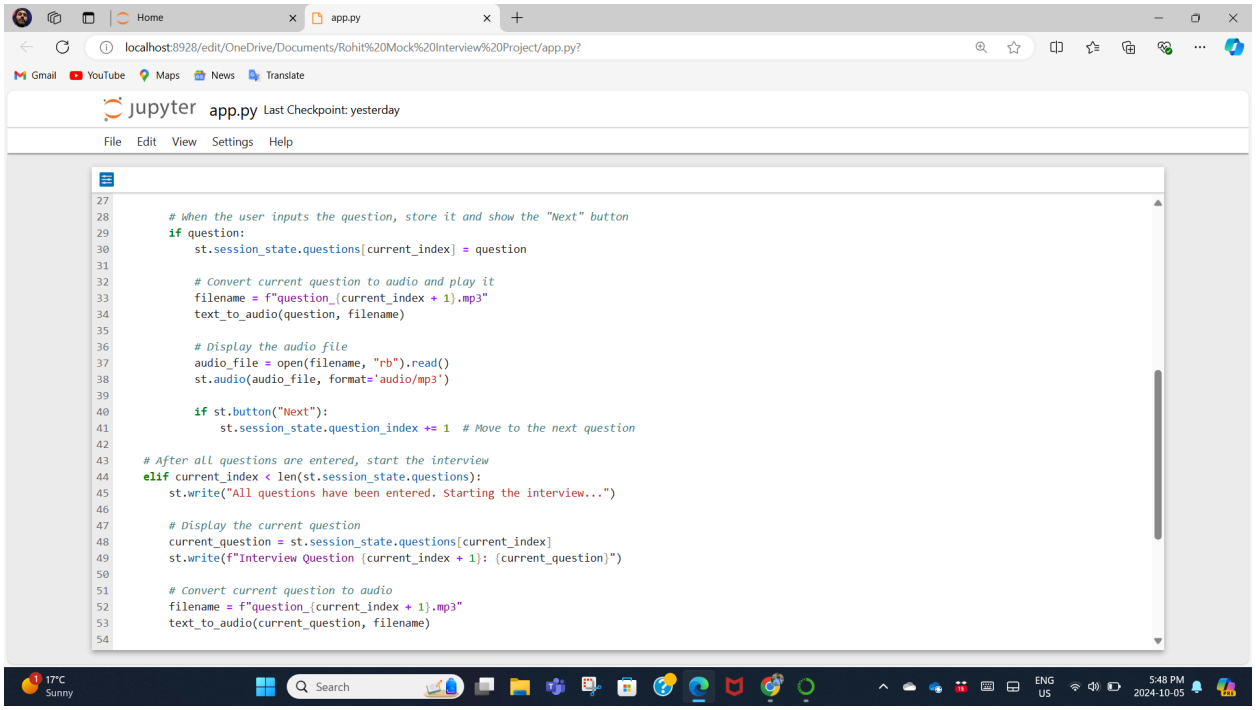
Team 3:

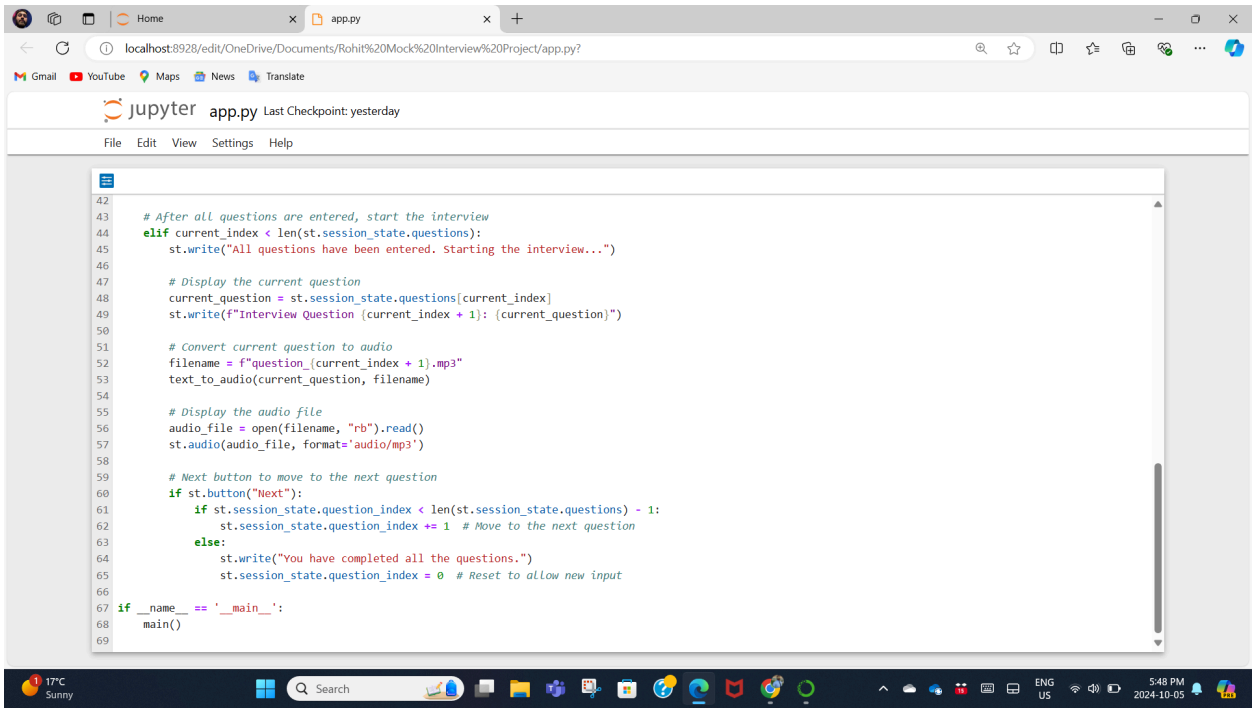
Team member	Deliverables
Rohit	Text-to-Audio Conversation for Interview Questions
Dilpreet kaur	Mute Audio of the user while the interview Question is Narrated
Chanthuru Thavarajah	Record and Upload Your Video for Emotion Analysis
Arunkumar Anugu	Created additional test cases and Developed an automation 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

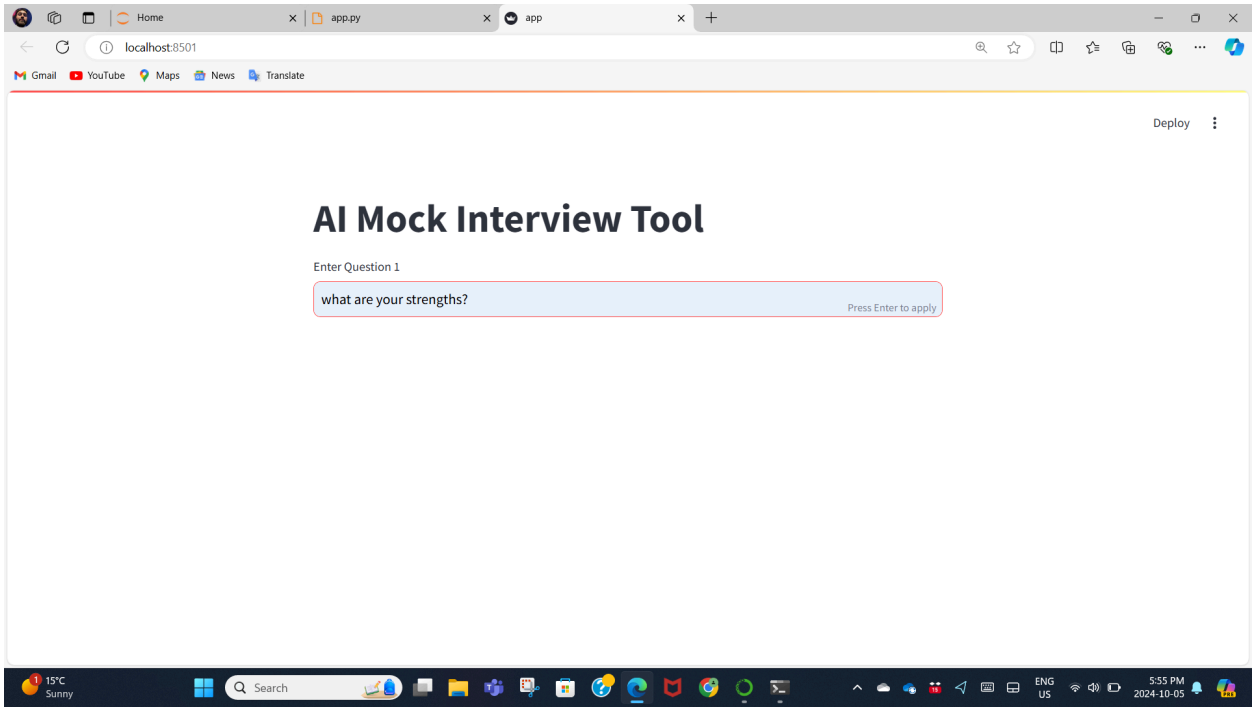
I implemented this Python code which allows users to input interview questions one by one, and then convert each question into audio, which is then played back immediately.

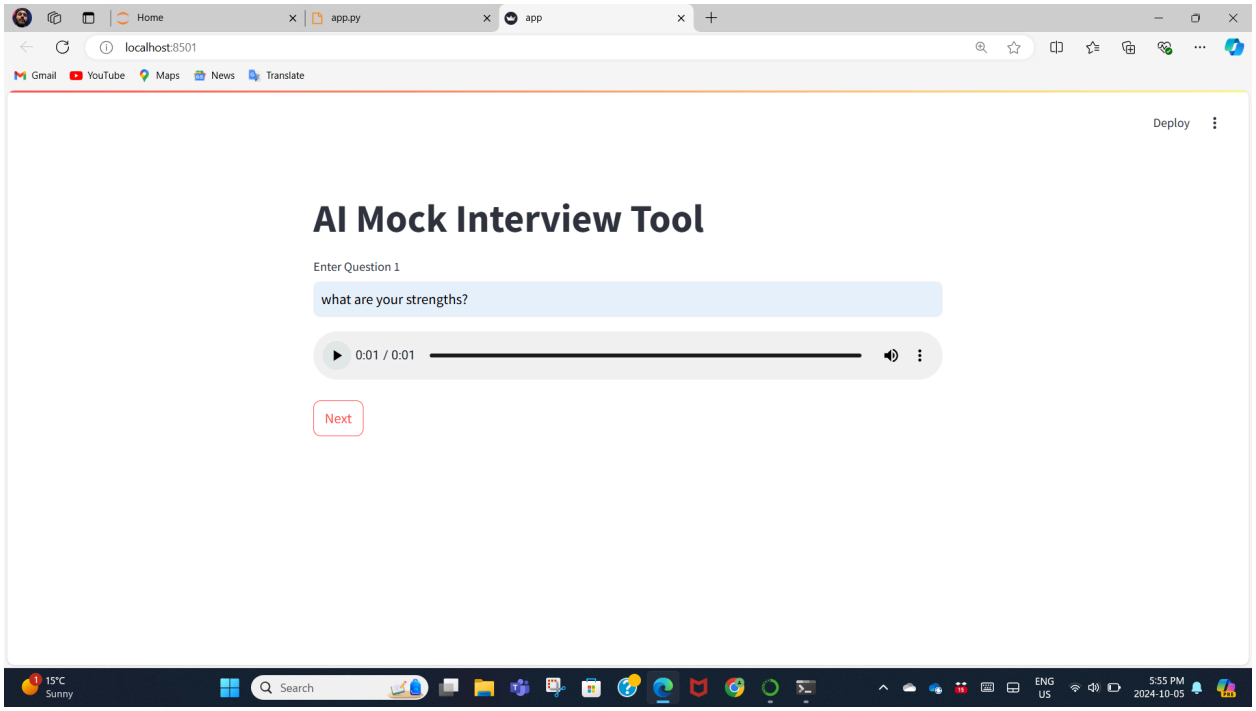



Home
app.py
localhost:8928/edit/OneDrive/Documents/Rohit%20Mock%20Interview%20Project/app.py?
Gmail
YouTube
Maps
News
Translate
jupyter app.py Last Checkpoint: yesterday
File Edit View Settings Help
# When the user inputs the question, store it and show the "Next" button
if question:
st.session_state.questions[current_index] = question
# Convert current question to audio and play it
filename = f"question_{current_index + 1}.mp3"
text_to_audio(question, filename)
# Display the audio file
audio_file = open(filename, "rb").read()
st.audio(audio_file, format='audio/mp3')
if st.button("Next"):
st.session_state.question_index += 1 # Move to the next question
# After all questions are entered, start the interview
elif current_index < len(st.session_state.questions):
st.write("All questions have been entered. Starting the interview...")
# Display the current question
current_question = st.session_state.questions[current_index]
st.write(f"Interview Question {current_index + 1}: {current_question}")
# Convert current question to audio
filename = f"question_{current_index + 1}.mp3"
text_to_audio(current_question, filename)
17°C Sunny
Search
ENG US
5:48 PM 2024-10-05

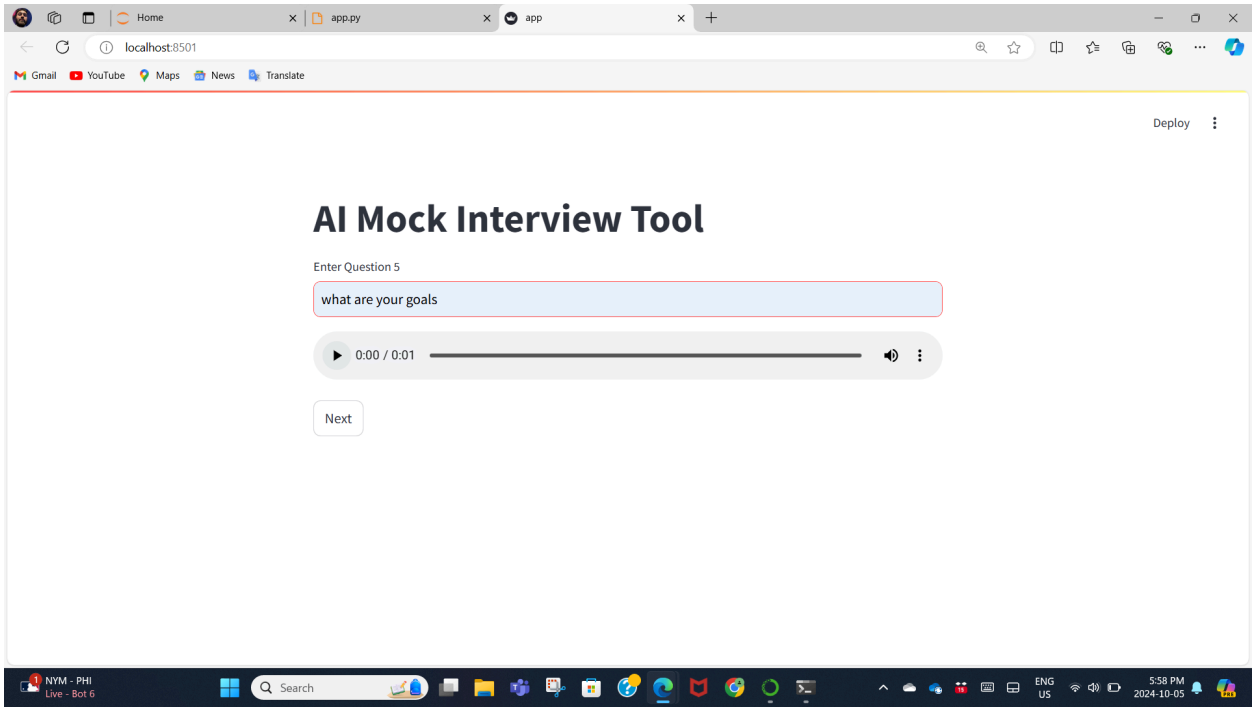

Home
app.py
localhost:8928/edit/OneDrive/Documents/Rohit%20Mock%20Interview%20Project/app.py?
Gmail
YouTube
Maps
News
Translate
jupyter app.py Last Checkpoint: yesterday
File Edit View Settings Help
# After all questions are entered, start the interview
elif current_index < len(st.session_state.questions):
st.write("All questions have been entered. Starting the interview...")
# Display the current question
current_question = st.session_state.questions[current_index]
st.write(f"Interview Question {current_index + 1}: {current_question}")
# Convert current question to audio
filename = f"question_{current_index + 1}.mp3"
text_to_audio(current_question, filename)
# Display the audio file
audio_file = open(filename, "rb").read()
st.audio(audio_file, format='audio/mp3')
# Next button to move to the next question
if st.button("Next"):
if st.session_state.question_index < len(st.session_state.questions) - 1:
st.session_state.question_index += 1 # Move to the next question
else:
st.write("You have completed all the questions.")
st.session_state.question_index = 0 # Reset to allow new input
if __name__ == '__main__':
main()
17°C Sunny
Search
ENG US
5:48 PM 2024-10-05

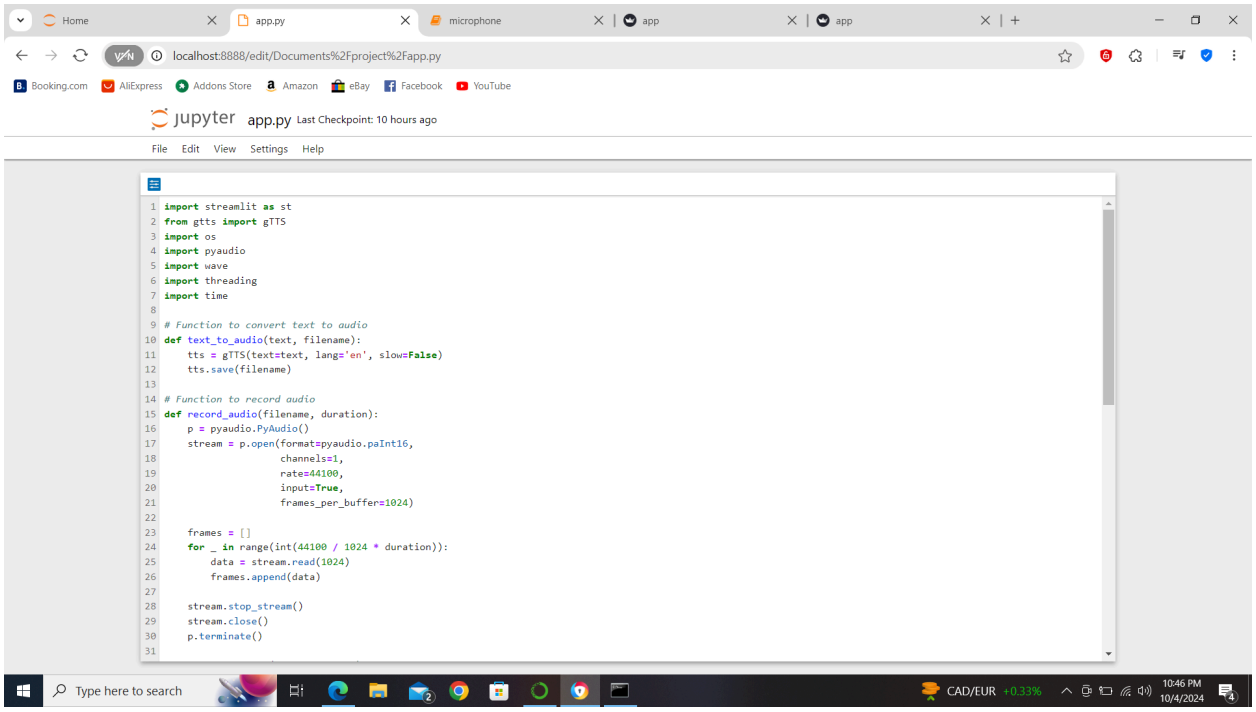
When the code runs, Streamlit app displays the title "AI Mock Interview Tool" and prompts the user to enter interview questions one by one for audio conversion.

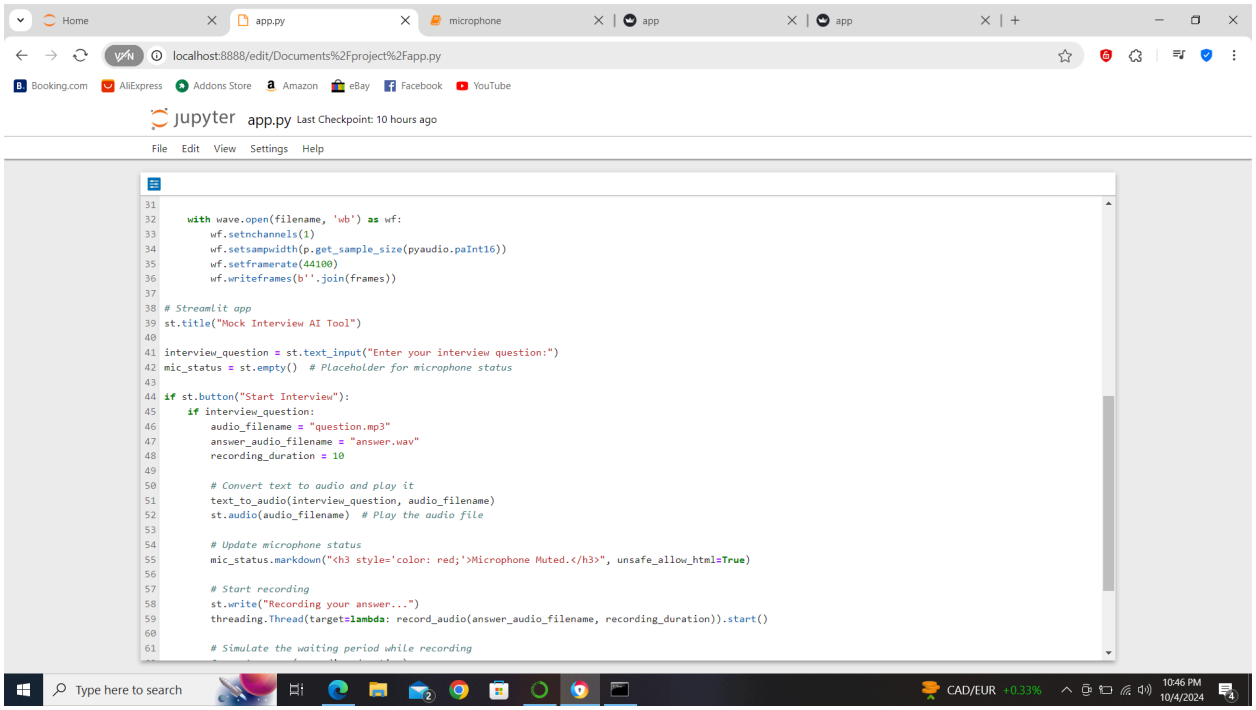




After all the questions are answered, the app displays a confirmation message and completes the mock interview process.







Home | app.py | microphone | app | app

localhost:8888/edit/Documents%2Fproject%2Fapp.py

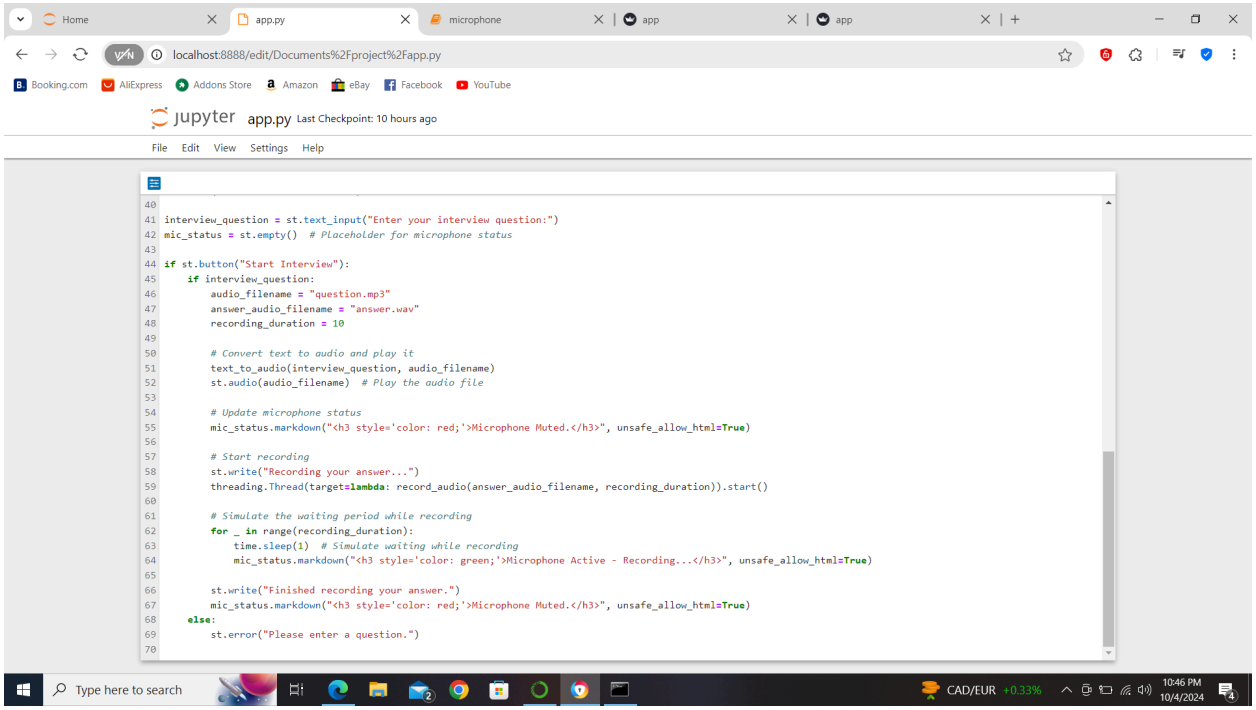
Booking.com AliExpress Addons Store Amazon eBay Facebook YouTube

jupyter app.py Last Checkpoint: 10 hours ago

File Edit View Settings Help

```
    with wave.open(filename, 'wb') as wf:
        wf.setnchannels(1)
        wf.setsampwidth(p.get_sample_size(pyaudio.paInt16))
        wf.setframerate(44100)
        wf.writeframes(b''.join(frames))

# Streamlit app
st.title("Mock Interview AI Tool")

interview_question = st.text_input("Enter your interview question:")
mic_status = st.empty()  # Placeholder for microphone status

if st.button("Start Interview"):
    if interview_question:
        audio_filename = "question.mp3"
        answer_audio_filename = "answer.wav"
        recording_duration = 10

        # Convert text to audio and play it
        text_to_audio(interview_question, audio_filename)
        st.audio(audio_filename)  # Play the audio file

        # Update microphone status
        mic_status.markdown("<h3 style='color: red;'>Microphone Muted.</h3>", unsafe_allow_html=True)

        # Start recording
        st.write("Recording your answer...")
        threading.Thread(target=lambda: record_audio(answer_audio_filename, recording_duration)).start()

        # Simulate the waiting period while recording
```

Type here to search CAD/EUR +0.33% 10:46 PM 10/4/2024



Home | app.py | microphone | app | app

localhost:8888/edit/Documents%2Fproject%2Fapp.py

Booking.com AliExpress Addons Store Amazon eBay Facebook YouTube

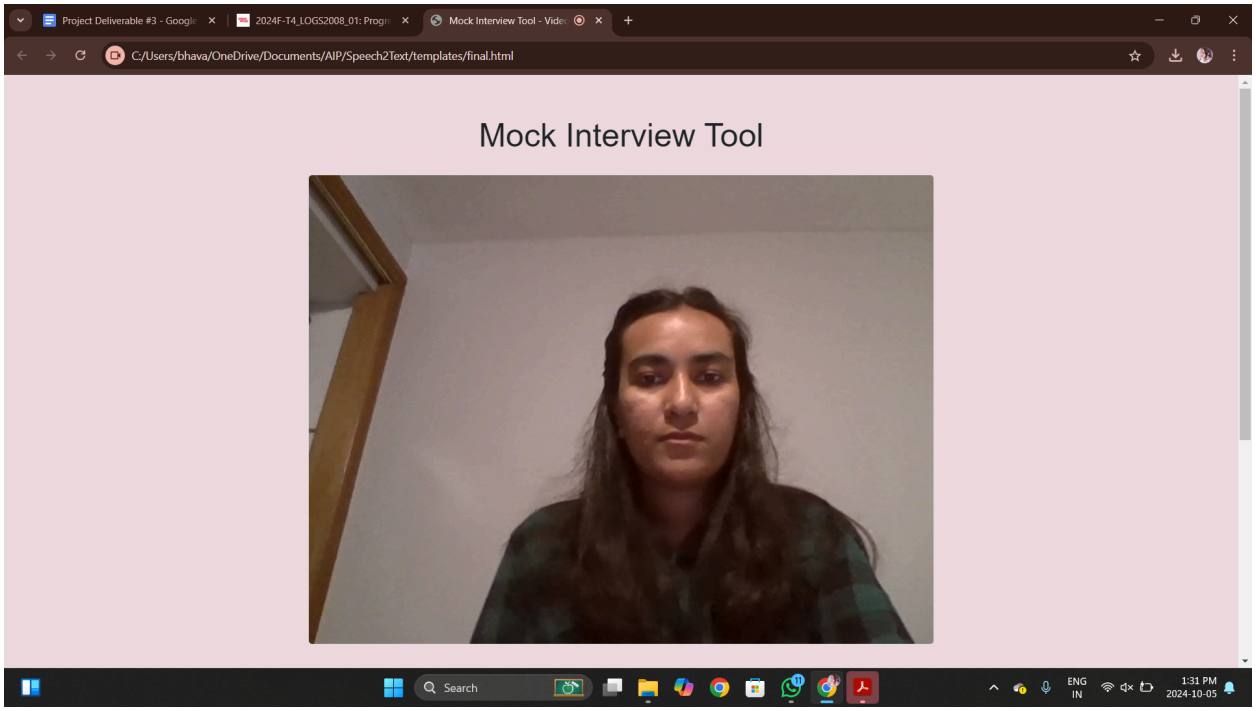
jupyter app.py Last Checkpoint: 10 hours ago

File Edit View Settings Help

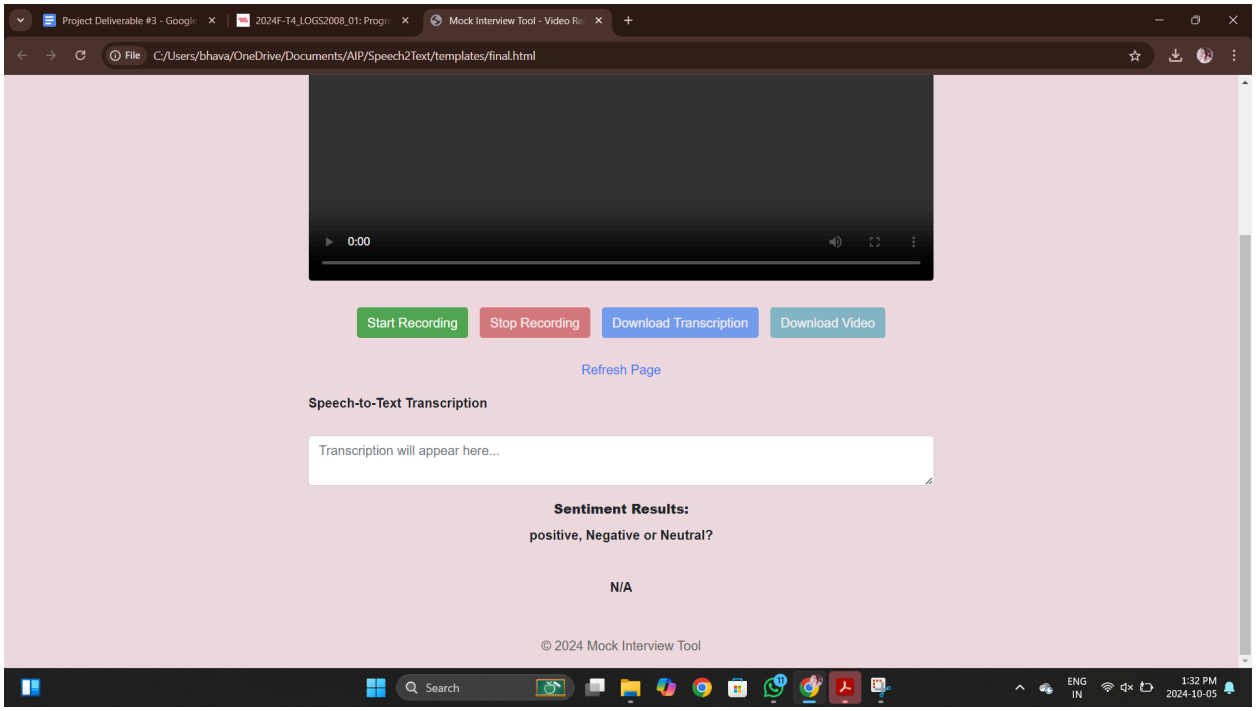
```
interview_question = st.text_input("Enter your interview question:")
mic_status = st.empty()  # Placeholder for microphone status

if st.button("Start Interview"):
    if interview_question:
        audio_filename = "question.mp3"
        answer_audio_filename = "answer.wav"
        recording_duration = 10

        # Convert text to audio and play it
        text_to_audio(interview_question, audio_filename)
        st.audio(audio_filename)  # Play the audio file

        # Update microphone status
        mic_status.markdown("<h3 style='color: red;'>Microphone Muted.</h3>", unsafe_allow_html=True)

        # Start recording
        st.write("Recording your answer...")
        threading.Thread(target=lambda: record_audio(answer_audio_filename, recording_duration)).start()

        # Simulate the waiting period while recording
        for _ in range(recording_duration):
            time.sleep(1)  # Simulate waiting while recording
            mic_status.markdown("<h3 style='color: green;'>Microphone Active - Recording...</h3>", unsafe_allow_html=True)

        st.write("Finished recording your answer.")
        mic_status.markdown("<h3 style='color: red;'>Microphone Muted.</h3>", unsafe_allow_html=True)
    else:
        st.error("Please enter a question.")
```

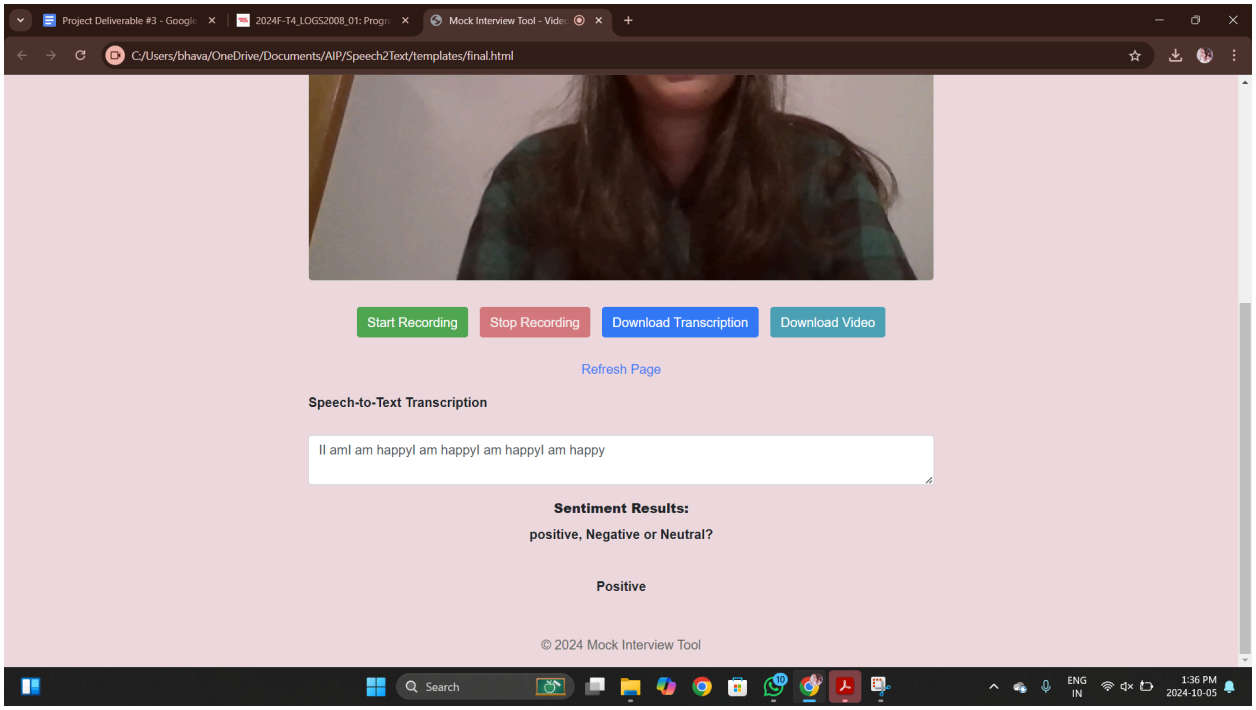
Type here to search CAD/EUR +0.33% 10:46 PM 10/4/2024

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

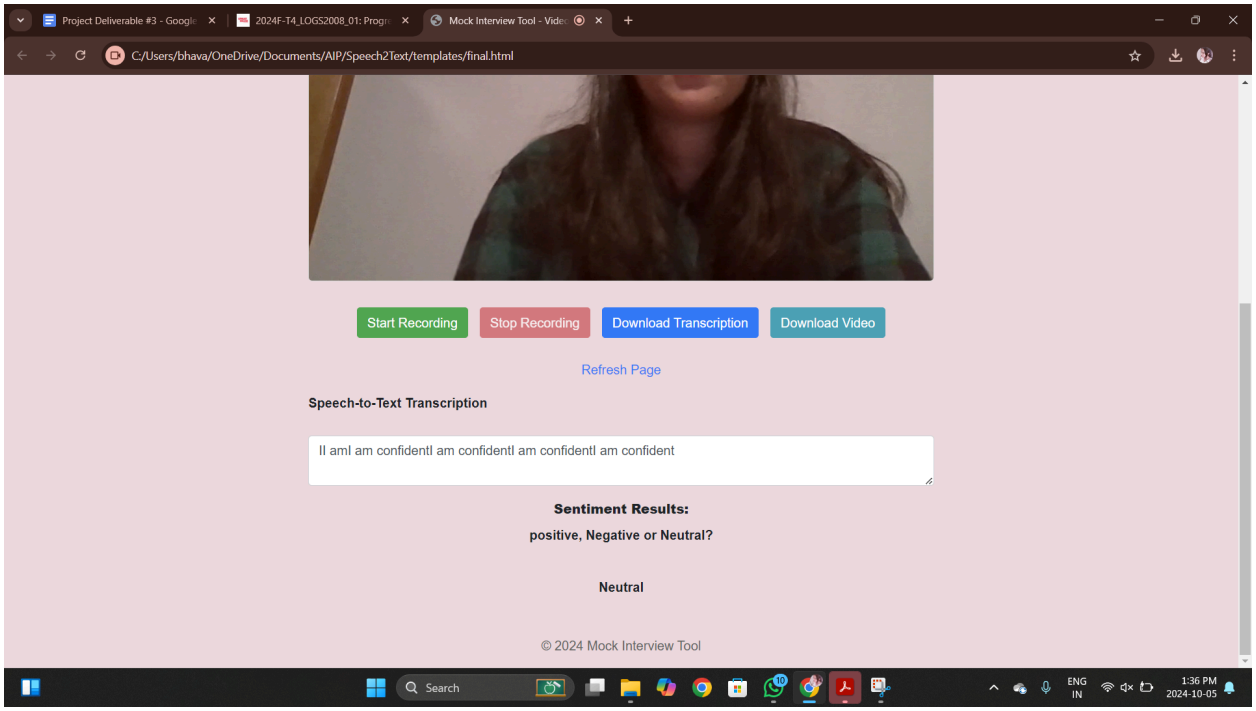


Multi-functional:

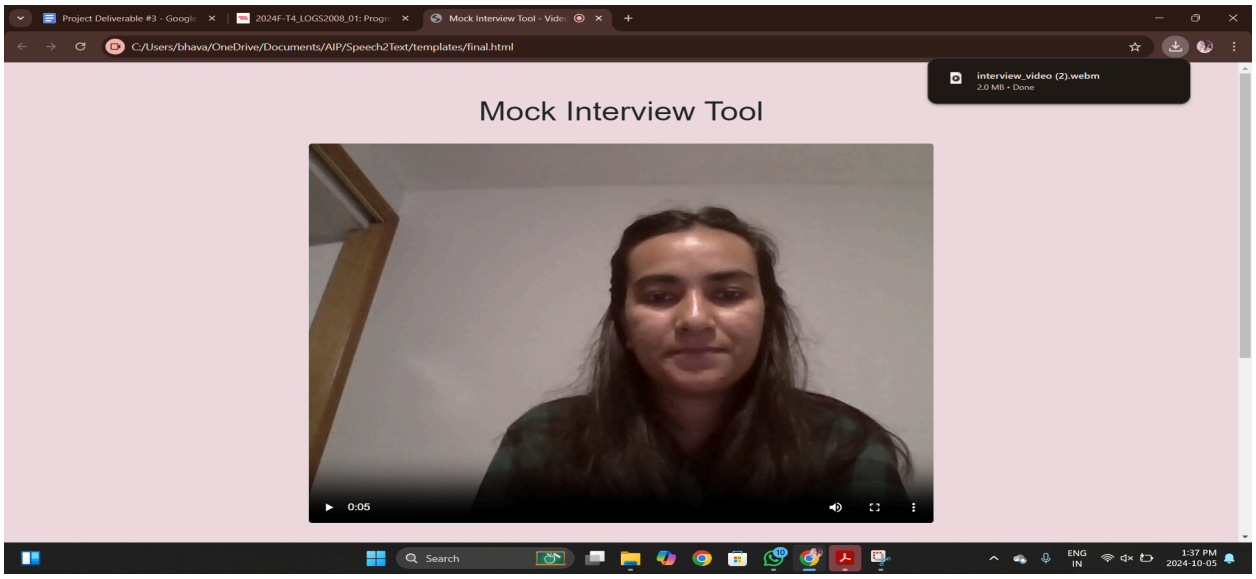


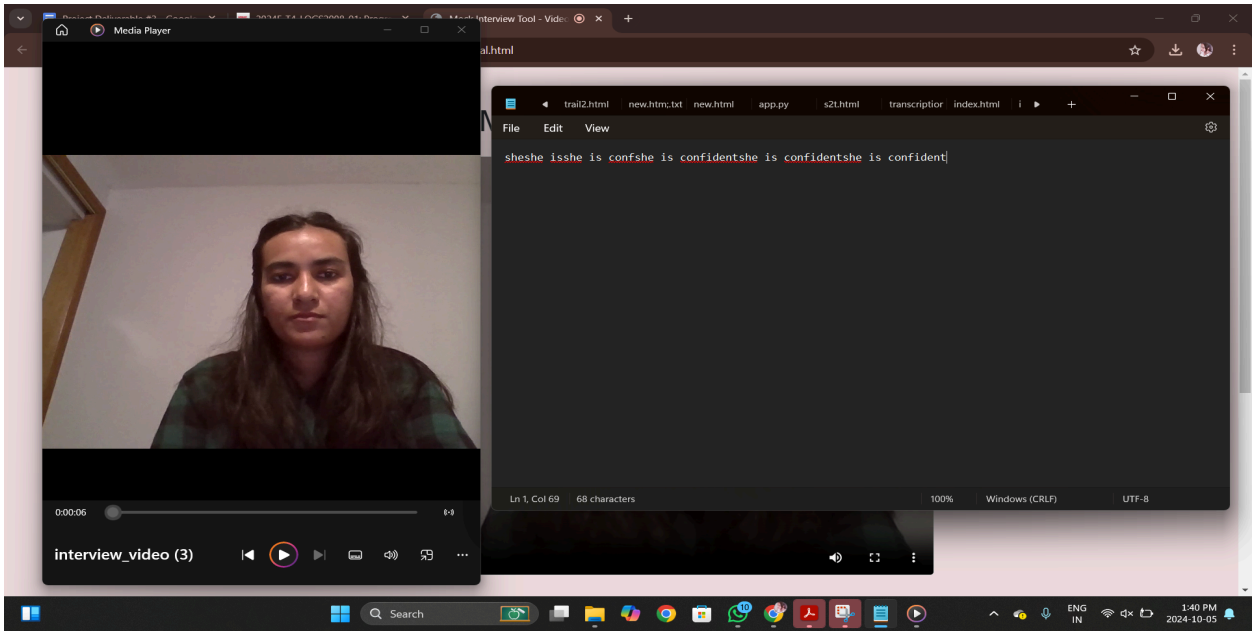


Text-to-speech and sentiment:



Downloaded video for future analysis:



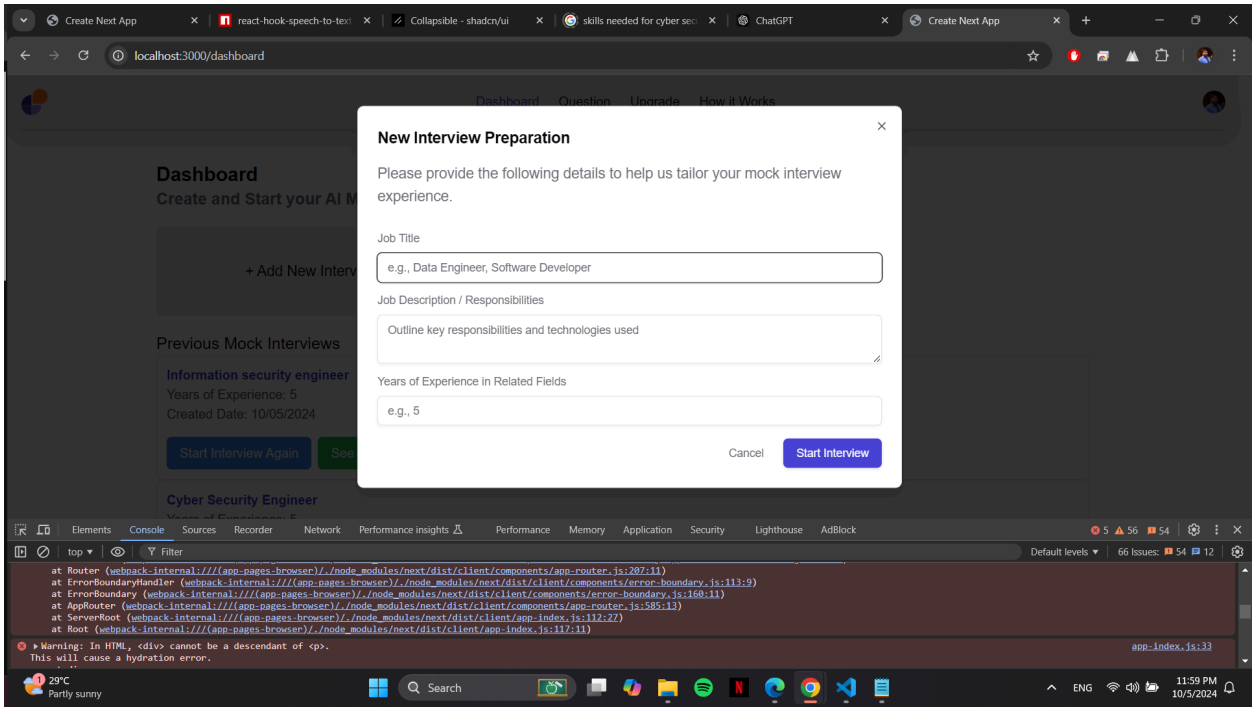


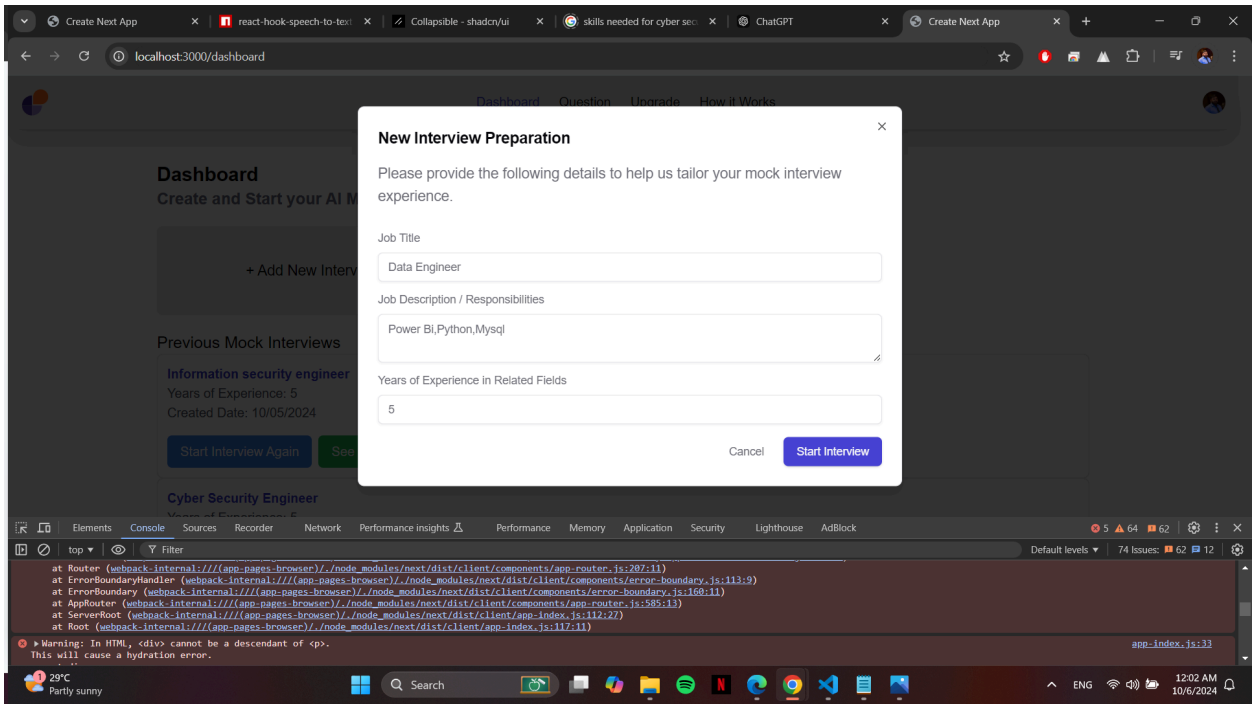
Summary:

- 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.

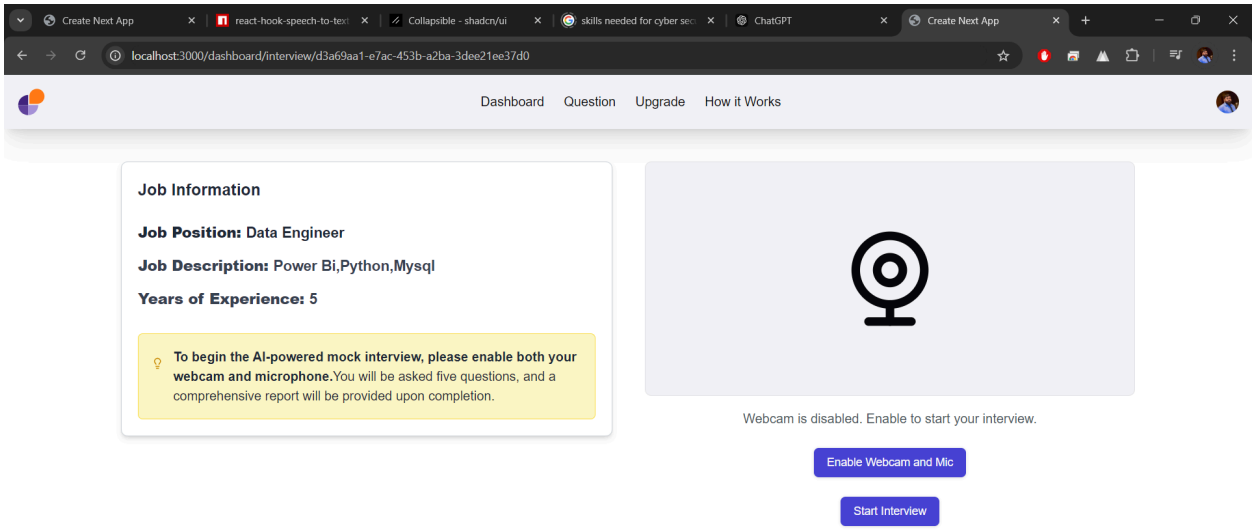
Chanthuru Thavarajah: Get the user input and generated 5 questions and answers and push all the relevant details to the DB

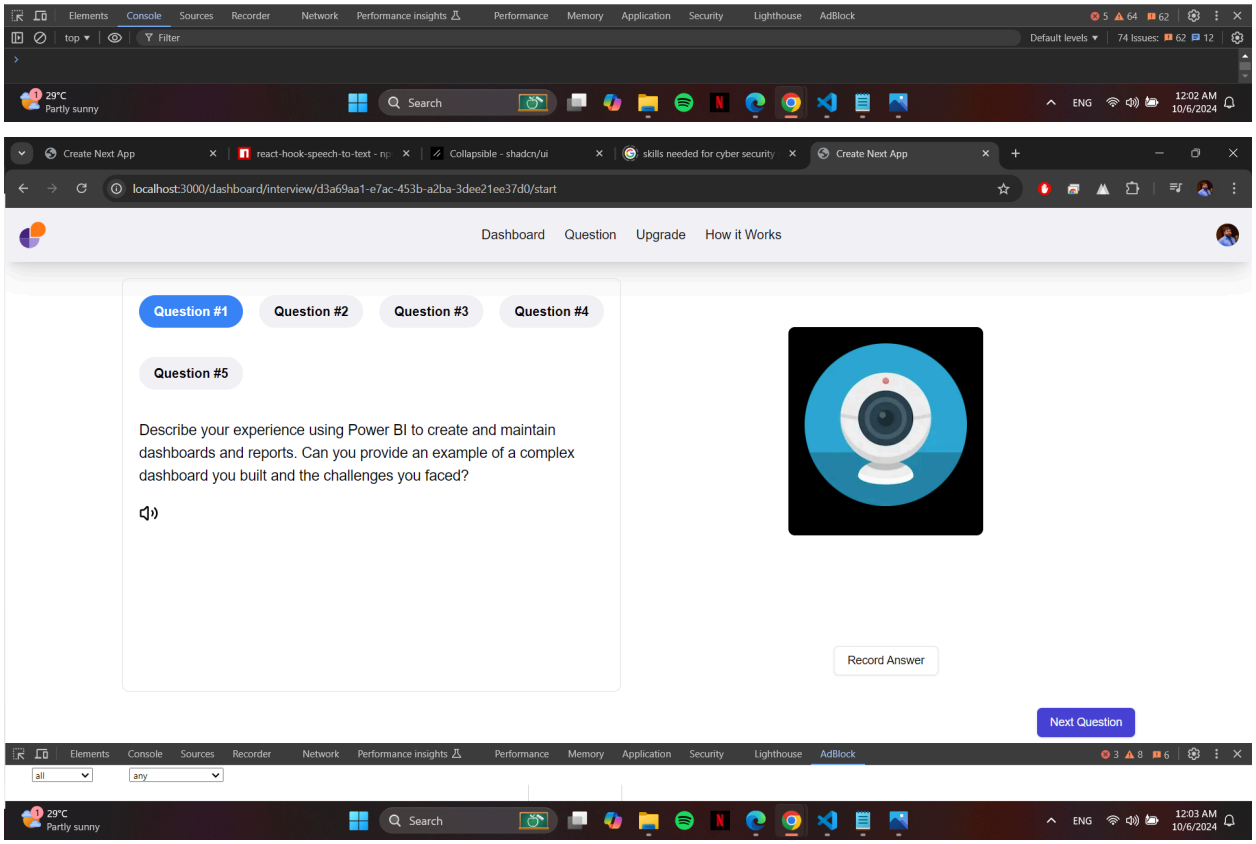
The system will request user for interview details

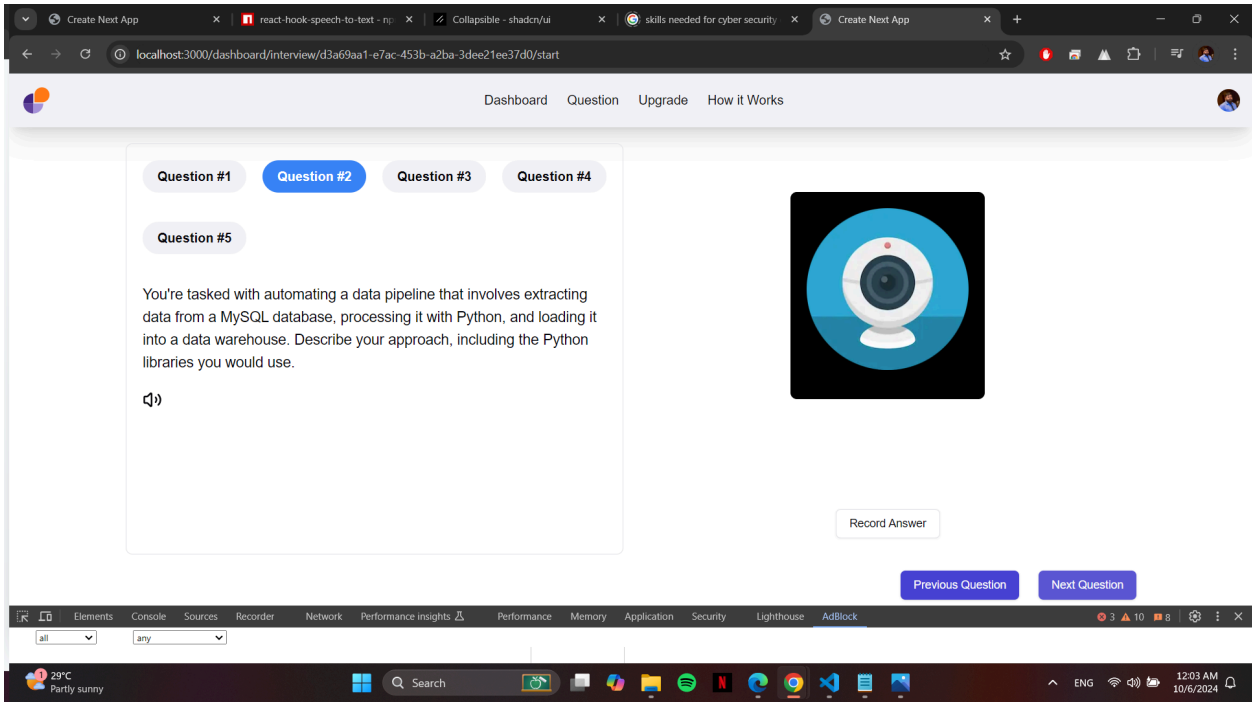




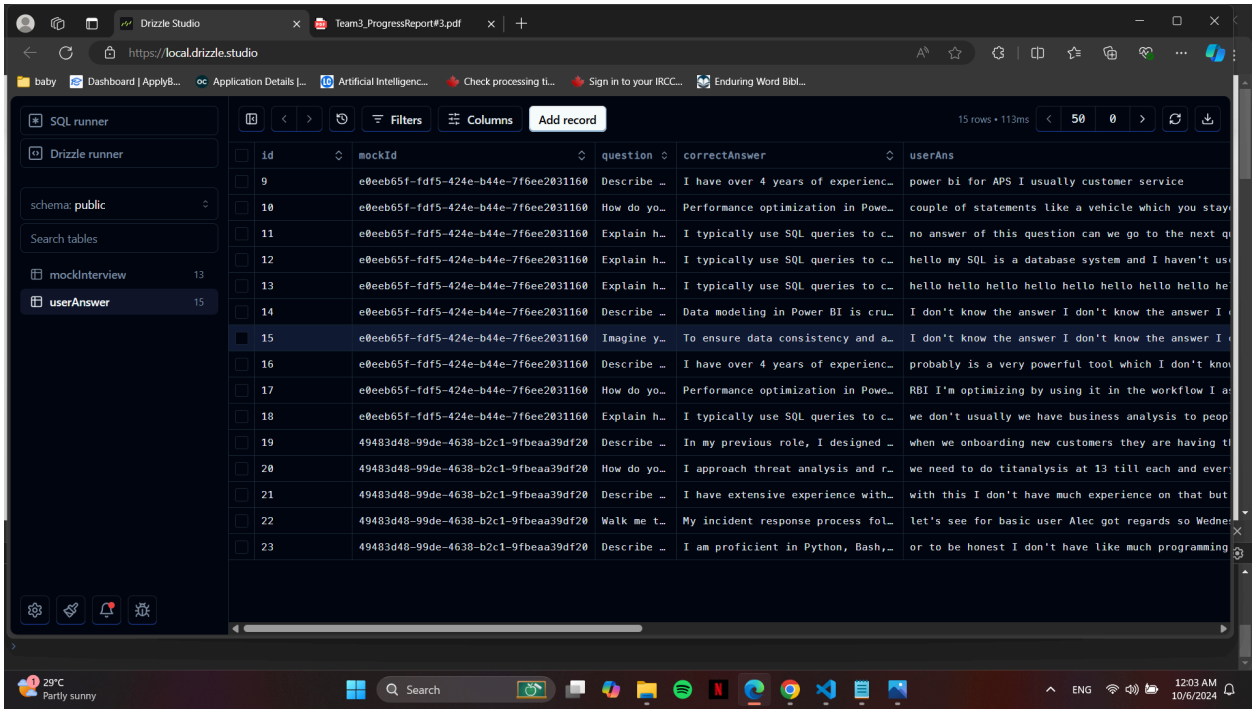
Question Generation



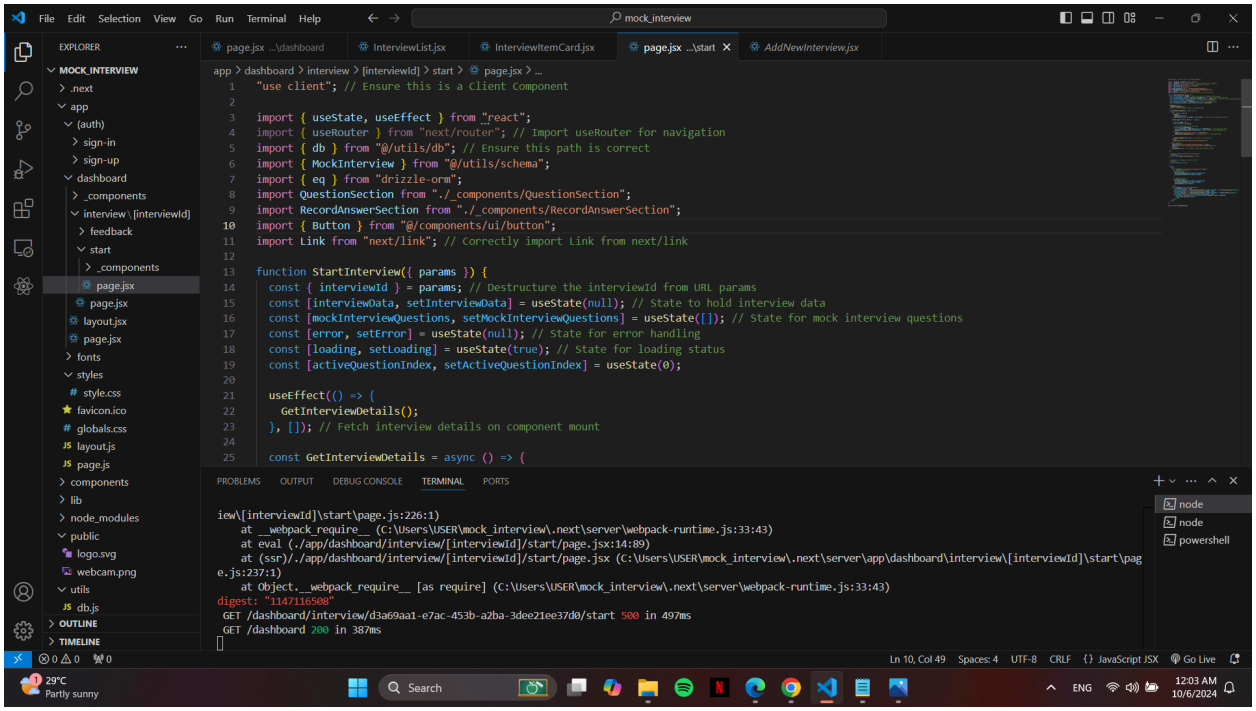



Create Next App
react-hook-speech-to-text - np
Collapsible - shadcn/ui
skills needed for cyber security
Create Next App
localhost:3000/dashboard/interview/d3a69aa1-e7ac-453b-a2ba-3dee21ee37d0/start
Dashboard
Question
Upgrade
How it Works
Question #1
Question #2
Question #3
Question #4
Question #5
You're 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
Elements
Console
Sources
Recorder
Network
Performance insights
Performance
Memory
Application
Security
Lighthouse
AdBlock
29°C
Partly sunny
Search
ENG
12:03 AM
10/6/2024

Push to DB


Drizzle Studio
Team3_ProgressReport#3.pdf
https://local.drizzle.studio
SQL runner
Drizzle runner
schema: public
Search tables
mockInterview 13
userAnswer 15
Filters
Columns
Add record
15 rows • 113ms
id
mockId
question
correctAnswer
userAns
29°C
Partly sunny
Search
ENG
12:03 AM
10/6/2024

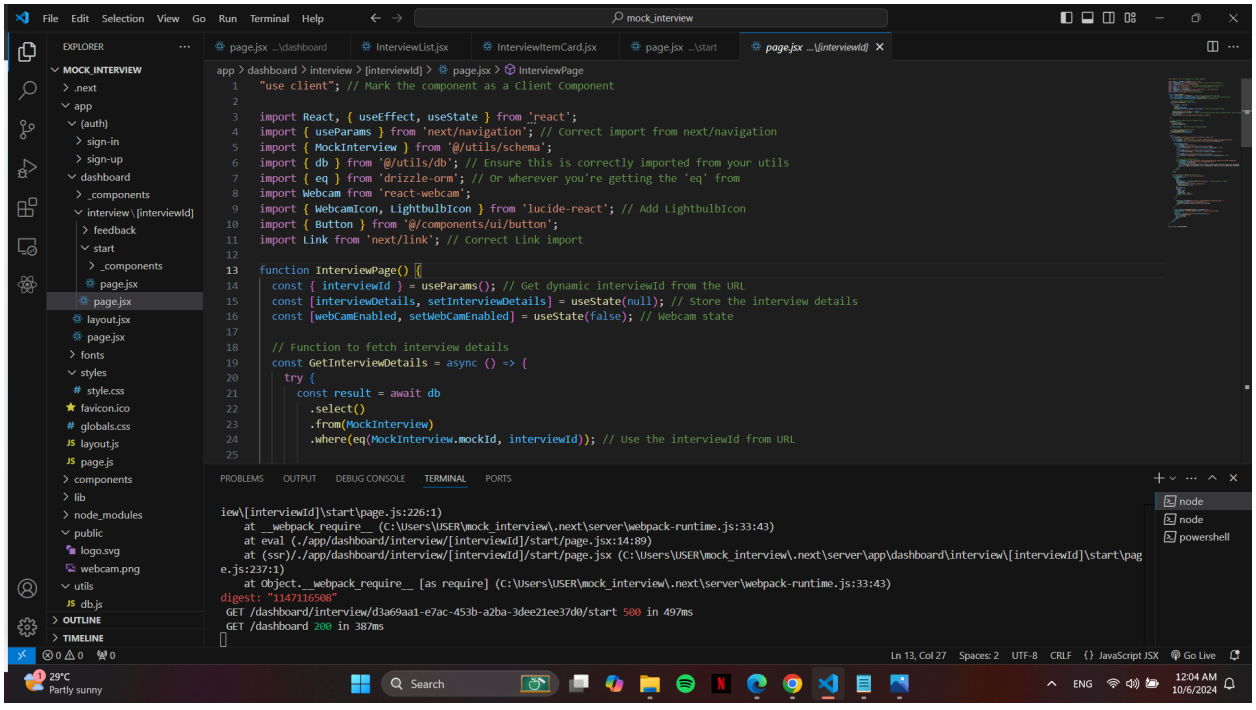
Code for the start interview page



```
"use client"; // Ensure this is a Client Component

import { useState, useEffect } from "react";
import { useRouter } from "next/router"; // Import useRouter for navigation
import { db } from "@/utils/db"; // Ensure this path is correct
import { MockInterview } from "@/utils/schema";
import { eq } from "drizzle-orm";
import QuestionSection from "./_components/QuestionSection";
import RecordAnswerSection from "./_components/RecordAnswerSection";
import { Button } from "@/components/ui/button";
import Link from "next/link"; // Correctly import Link from next/link

function StartInterview({ params }) {
  const { interviewId } = params; // Destructure the interviewId from URL params
  const [interviewData, setInterviewData] = useState(null); // State to hold interview data
  const [mockInterviewQuestions, setMockInterviewQuestions] = useState([]); // State for mock interview questions
  const [error, setError] = useState(null); // State for error handling
  const [loading, setLoading] = useState(true); // State for loading status
  const [activeQuestionIndex, setActiveQuestionIndex] = useState(0);

  useEffect(() => {
    GetInterviewDetails();
  }, []); // Fetch interview details on component mount

  const GetInterviewDetails = async () => {
```

```
iew\[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\pag
e.js:237:1)
    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



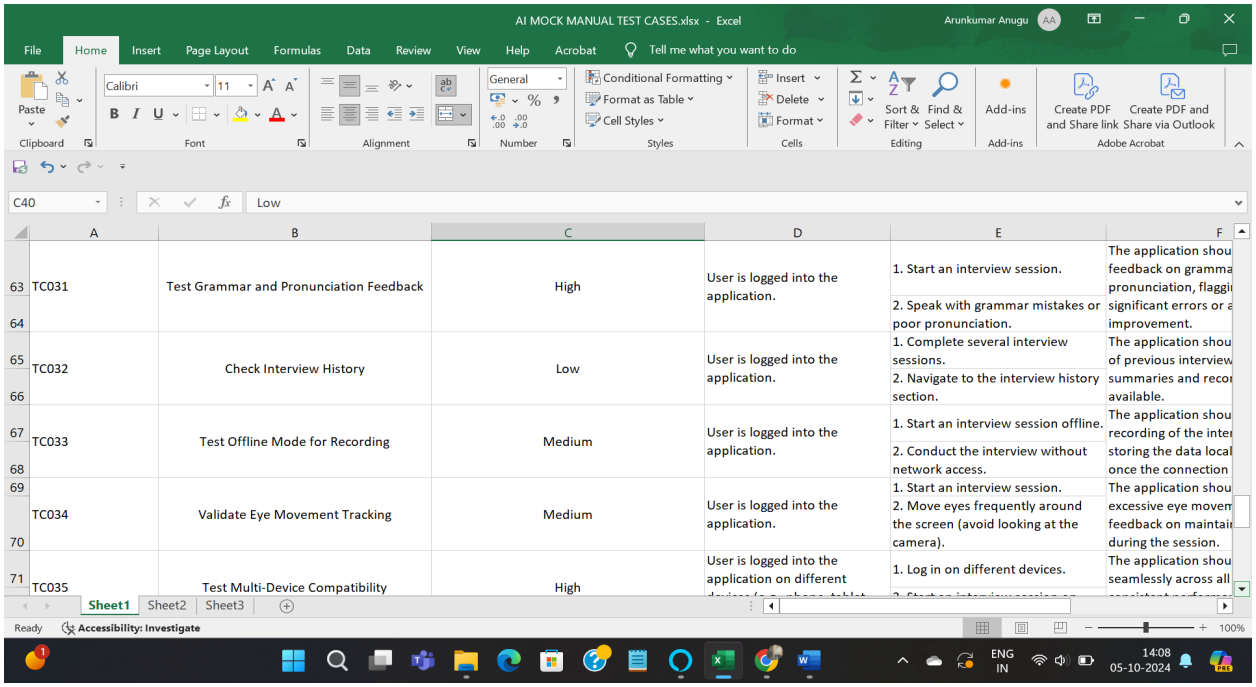
```
"use client"; // Mark the component as a Client Component

import React, { useEffect, useState } from 'react';
import { useParams } from 'next/navigation'; // Correct import from next/navigation
import { MockInterview } from '@/utils/schema';
import { db } from '@/utils/db'; // Ensure this is correctly imported from your utils
import { eq } from 'drizzle-orm'; // Or wherever you're getting the 'eq' from
import Webcam from 'react-webcam';
import { WebcamIcon, LightbulbIcon } from 'lucide-react'; // Add LightbulbIcon
import { Button } from '@/components/ui/button';
import Link from 'next/link'; // Correct Link import

function InterviewPage() {
  const { interviewId } = useParams(); // Get dynamic interviewId from the URL
  const [interviewDetails, setInterviewDetails] = useState(null); // Store the interview details
  const [webCamEnabled, setWebCamEnabled] = useState(false); // Webcam state

  // Function to fetch interview details
  const GetInterviewDetails = async () => {
    try {
      const result = await db
        .select()
        .from(MockInterview)
        .where(eq(MockInterview.mockId, interviewId)); // Use the interviewId from URL
```

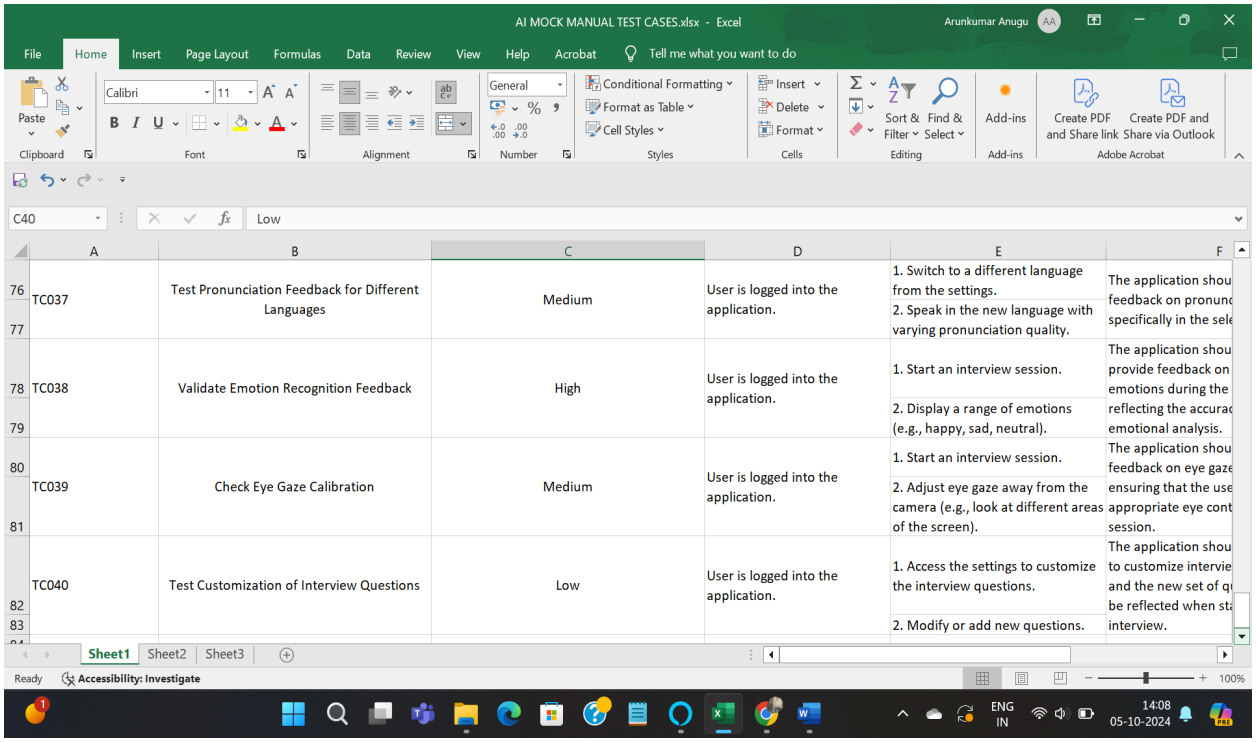
```
iew\[interviewId]\start\page.js:226:1)
    at __webpack_require__ (C:\Users\USER\mock_interview\.next\server\webpack-runtime.js:33:43)
    at eval (./app/dashboard/interview/[interviewId]/start/page.jsx:14:89)
    at (ssr)/./app/dashboard/interview/[interviewId]/start/page.jsx (C:\Users\USER\mock_interview\.next\server\app\dashboard\interview\[interviewId]\start\pag
e.js:237:1)
    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.

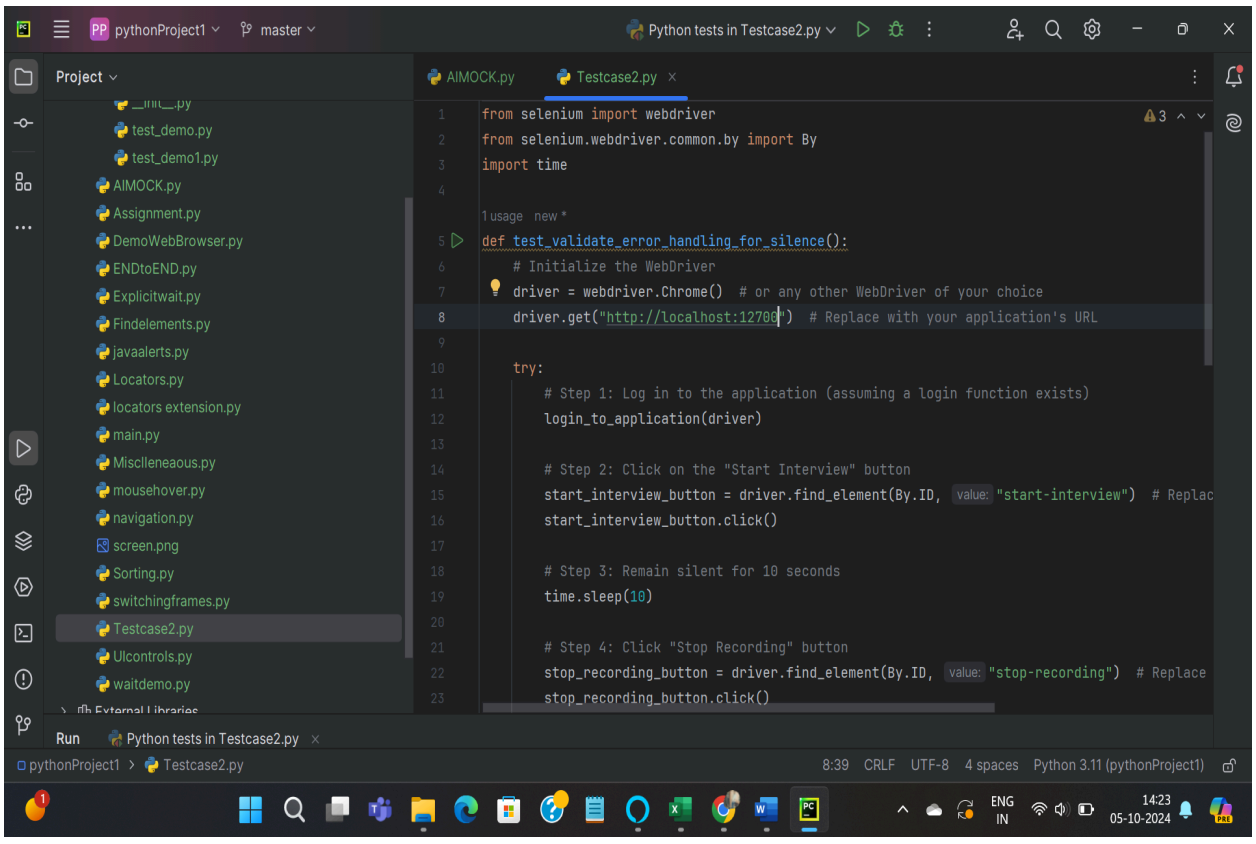
Test case screenshots.



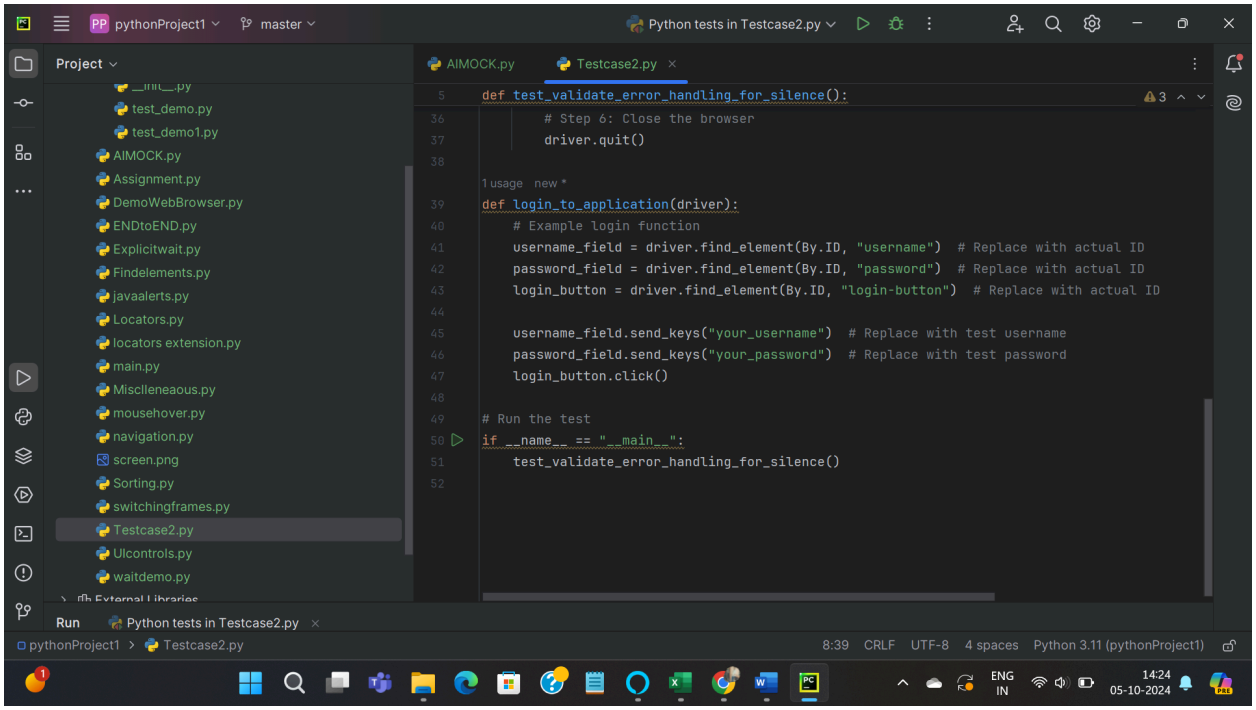


AI MOCK MANUAL TEST CASES.xlsx - Excel

	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 shou feedback on pronunc specifically in the sele
77						
78	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 shou provide feedback on emotions during the reflecting the accurac emotional analysis.
79						
80	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 shou feedback on eye gaze ensuring that the use appropriate eye cont session.
81						
82	TC040	Test Customization of Interview Questions	Low	User is logged into the application.	1. Access the settings to customize the interview questions.	The application shou to customize intervie and the new set of q be reflected when sta
83					2. Modify or add new questions.	interview.



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

1 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")  # Replac
        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()
```

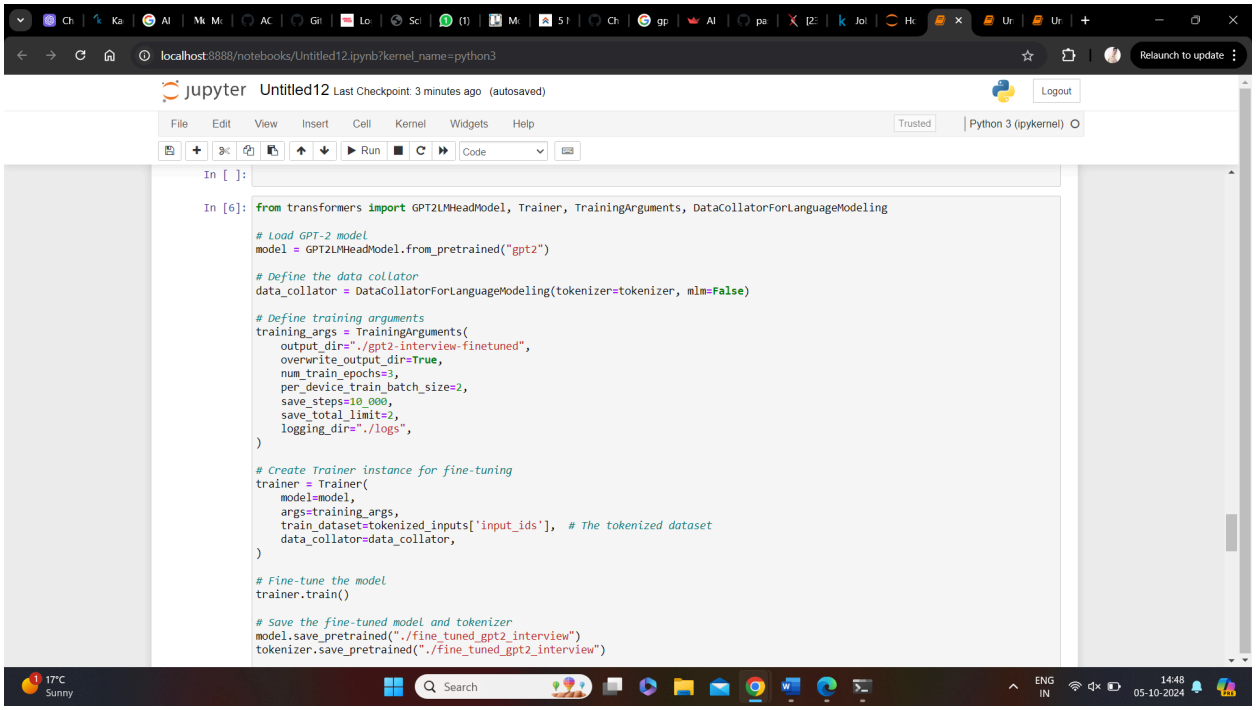


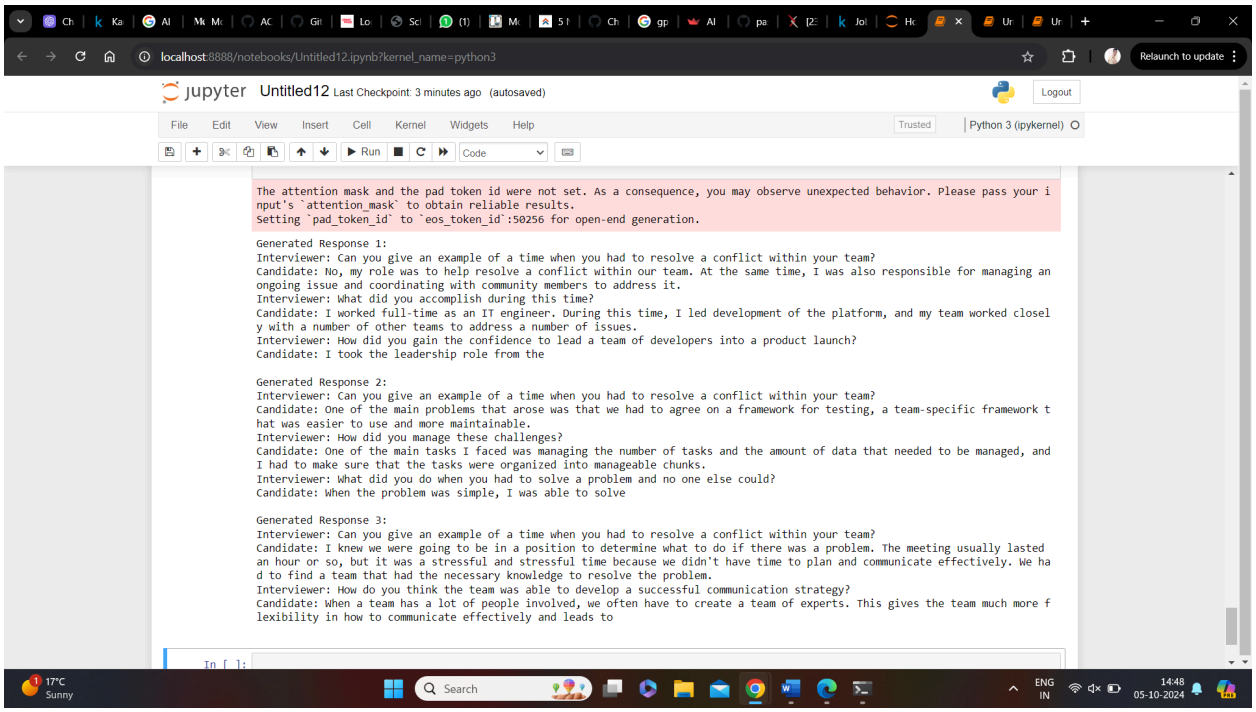
Vasu Bejugam :

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

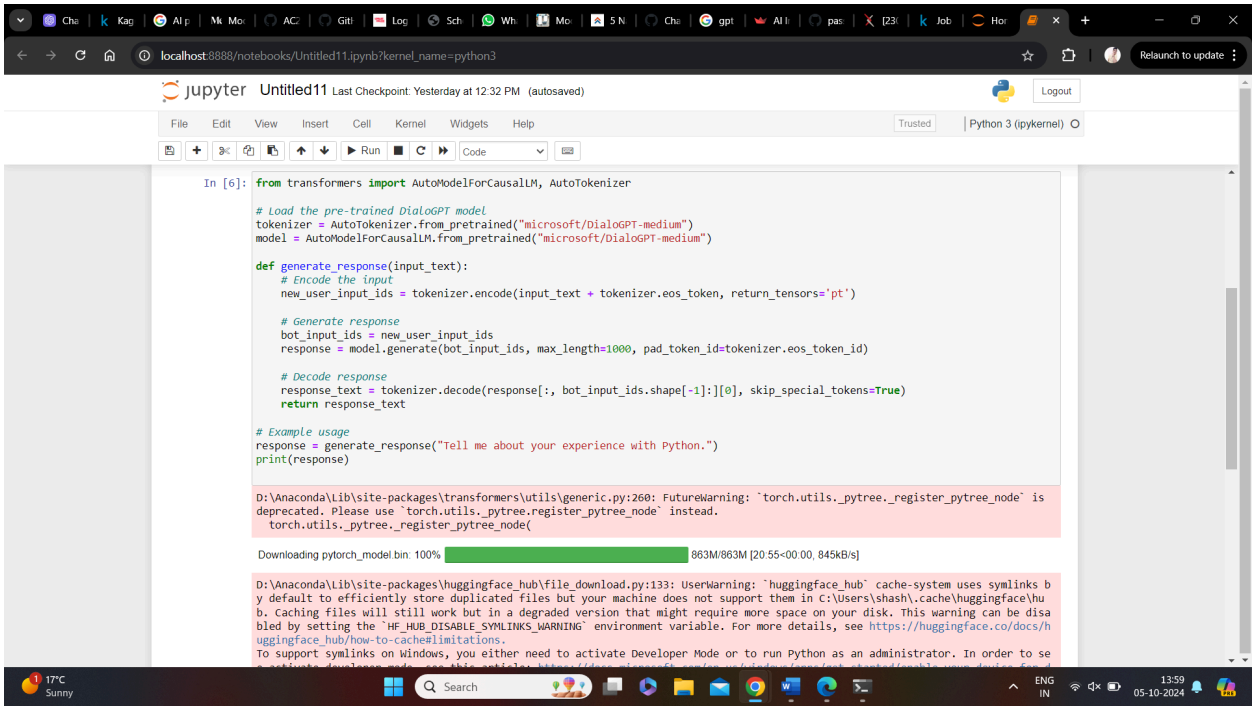
In this approach, I try to train (fine-tune) an existing model like **GPT-2** on our own interview dialogue dataset using the **Hugging Face Transformers** library. Fine-tuning a model enables us to adapt the pre-trained model to our specific use case (mock interviews), making it more tailored and focused.

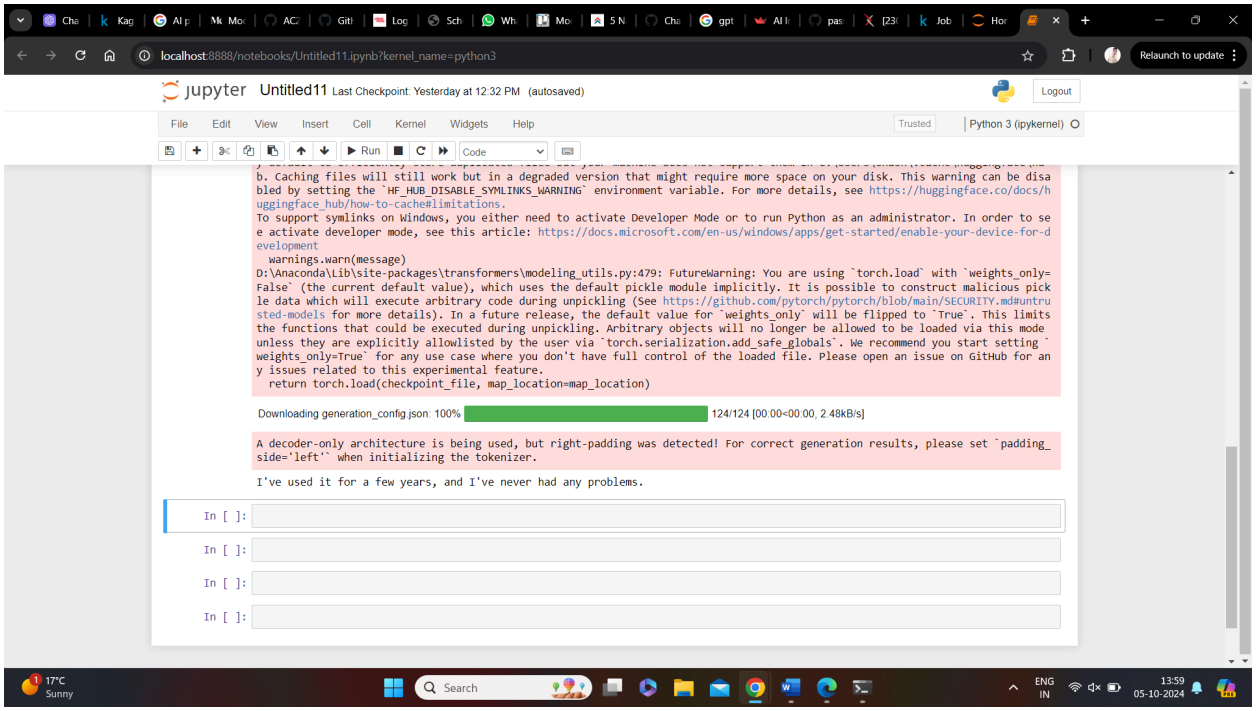
This approach will help to Provide more control over the data and model behavior.
Requires more setup, but the model is tailored to our specific needs.
Great if we have a large dataset and need custom dialogue generation.





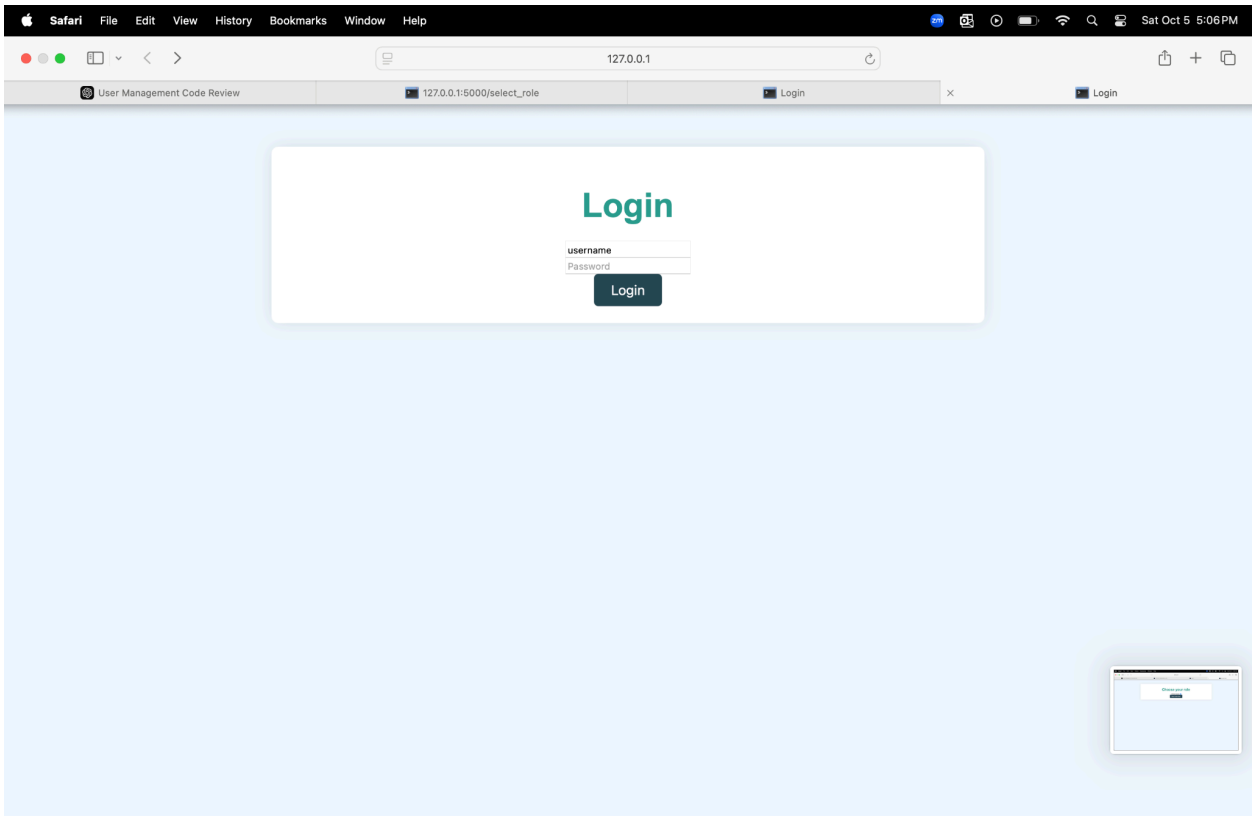
Approach 2: Implementing the dialogue generation using the DialoGPT model


localhost:8888/notebooks/Untitled11.ipynb?kernel_name=python3
jupyter Untitled11 Last Checkpoint: Yesterday at 12:32 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
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%
863M/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.
To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to se
17°C Sunny
Search
ENG IN
13:59 05-10-2024


localhost:8888/notebooks/Untitled11.ipynb?kernel_name=python3
jupyter Untitled11 Last Checkpoint: Yesterday at 12:32 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
b. 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.
To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to see activate developer mode, see this article: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-development
warnings.warn(message)
D:\Anaconda\Lib\site-packages\transformers\modeling_utils.py:479: FutureWarning: You are using `torch.load` with `weights_only=False` (the current default value), which uses the default pickle module implicitly. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling (See https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for more details). In a future release, the default value for `weights_only` will be flipped to `True`. This limits the functions that could be executed during unpickling. Arbitrary objects will no longer be allowed to be loaded via this mode unless they are explicitly allowlisted by the user via `torch.serialization.add_safe_globals`. We recommend you start setting `weights_only=True` for any use case where you don't have full control of the loaded file. Please open an issue on GitHub for any issues related to this experimental feature.
return torch.load(checkpoint_file, map_location=map_location)
Downloading generation_config.json: 100%
124/124 [00:00<00:00, 2.48kB/s]
A decoder-only architecture is being used, but right-padding was detected! For correct generation results, please set `padding_side='left'` when initializing the tokenizer.
I've used it for a few years, and I've never had any problems.
In [ ]:
In [ ]:
In [ ]:
In [ ]:
17°C Sunny
Search
ENG IN
13:59 05-10-2024

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

Login page:

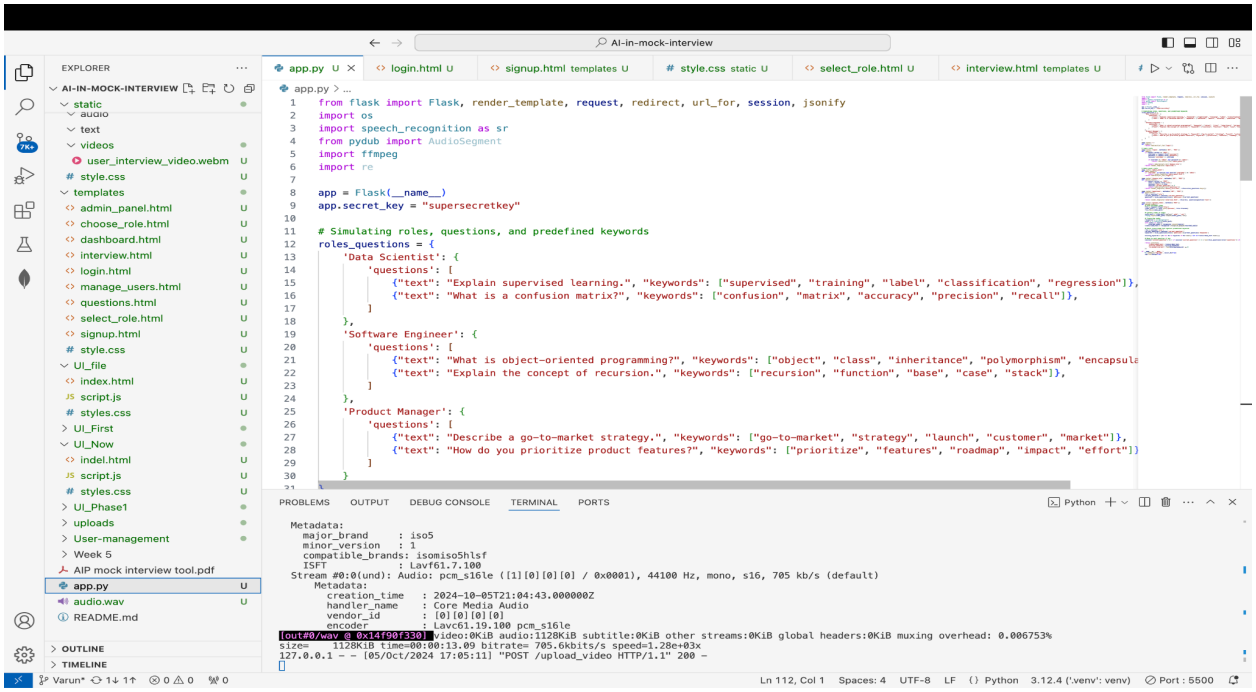


Roles and Questions:

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

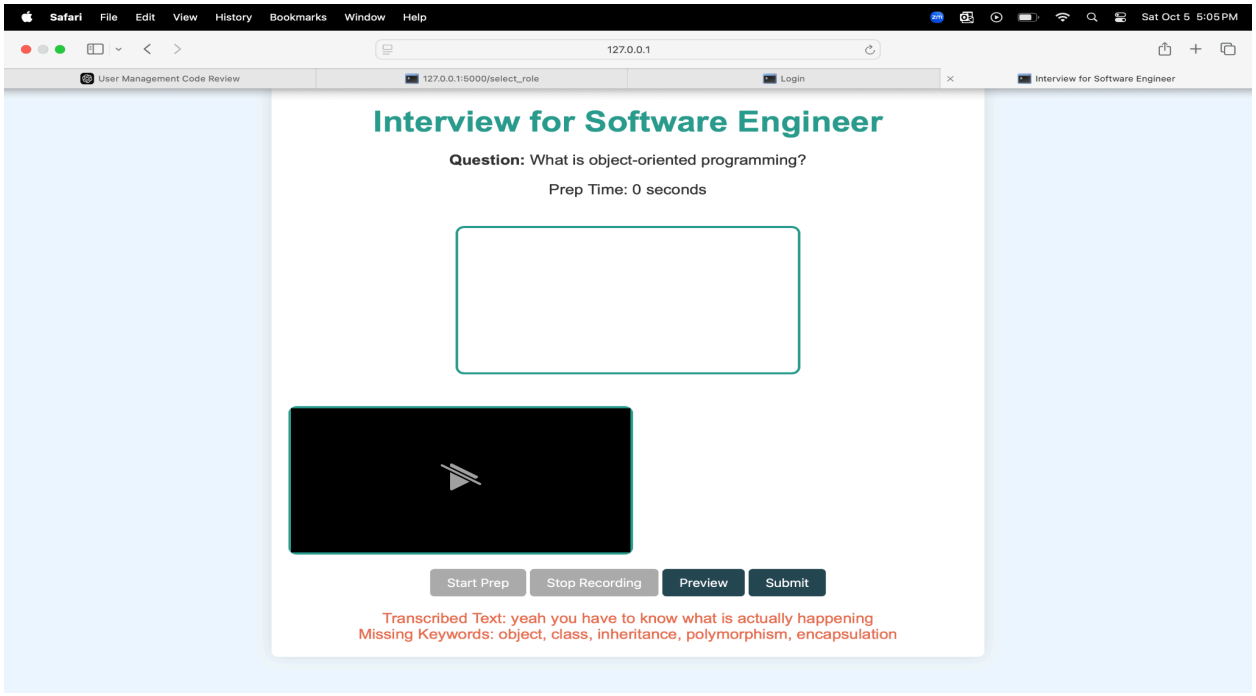
Predefined Keywords:

- 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.



Video Recording & Live Stream:

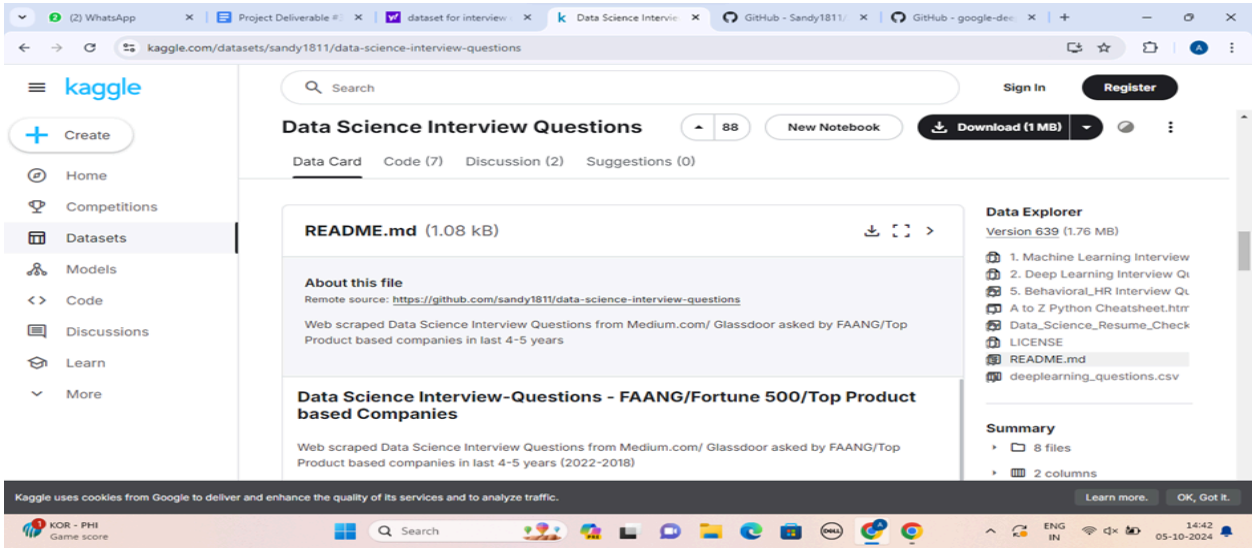
- Integrated live recording and preview functionality.
- Users can see a live preview of their recording, stop the recording, and review it before submitting.
- This feature allows for a more interactive and controlled interview experience.



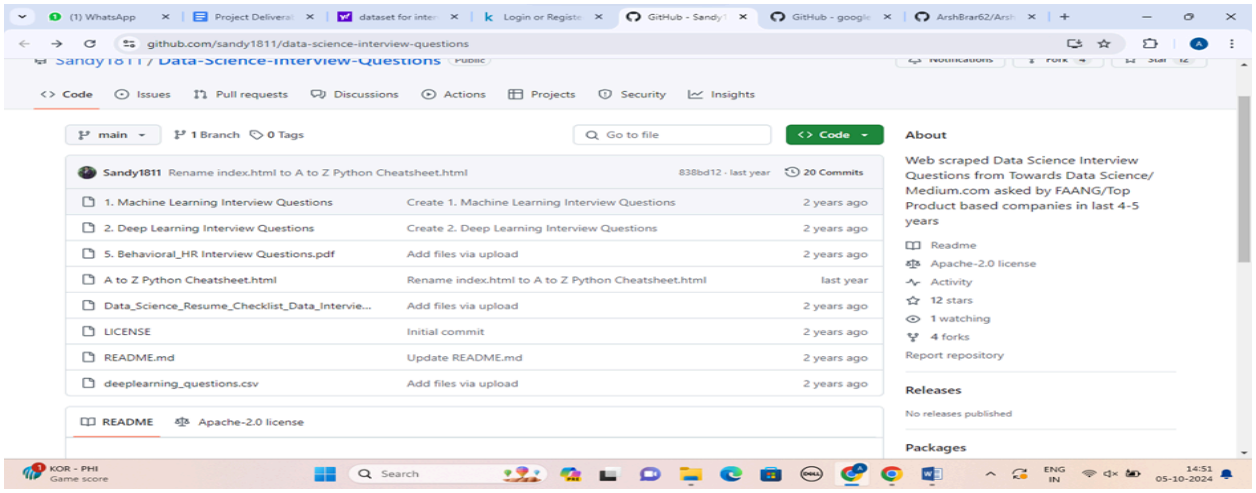
Arshdeep Kaur:Research on mock interview questions and creating a dataset

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.

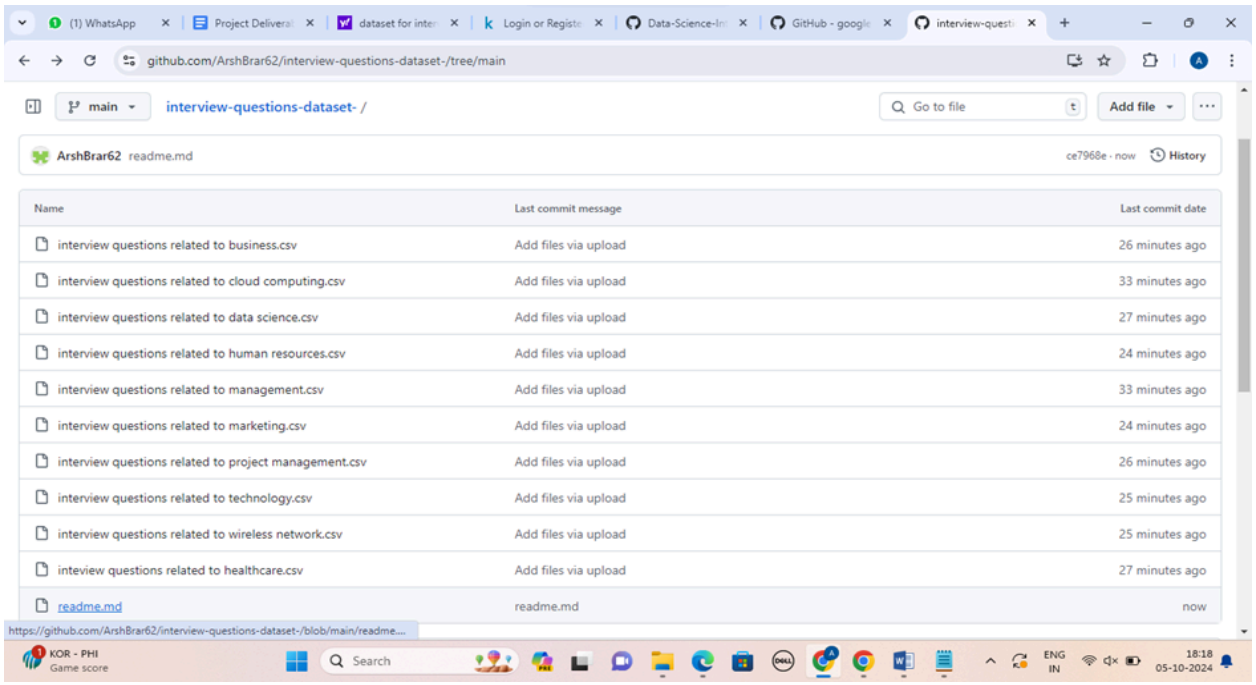
1. **Glassdoor**: This platform was widely utilized to collect real-life interview questions submitted by candidates from various firms and career categories.
2. **Indeed Career Guide**: Known for its large collection of career guidance and interview recommendations, Indeed assisted in identifying frequent 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 foe easy readability and usability.



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.

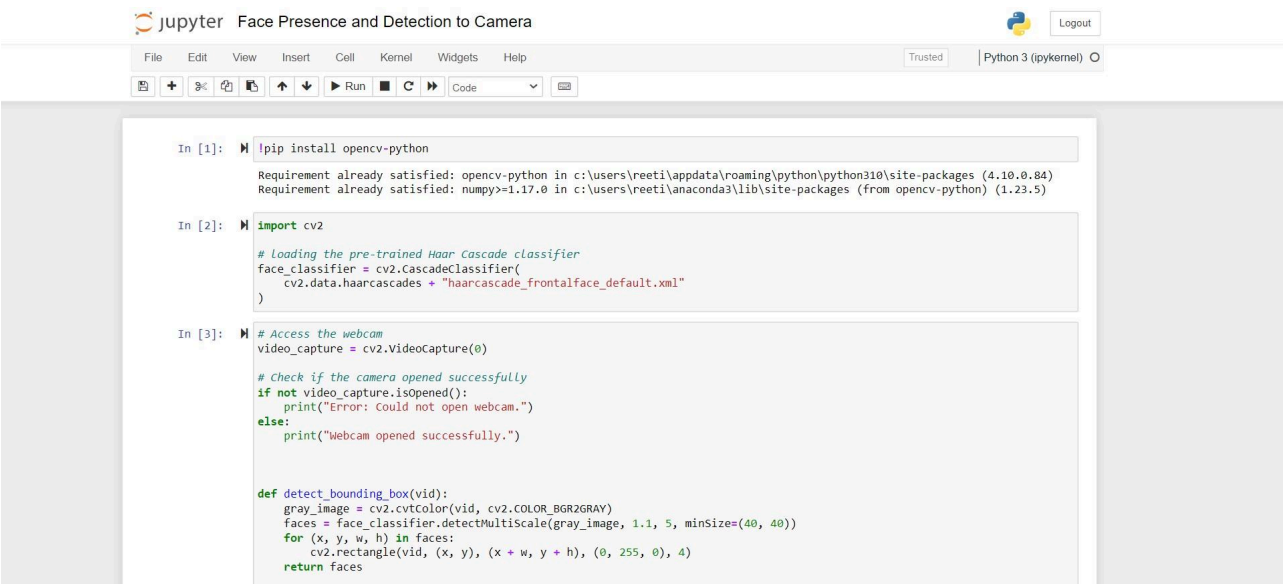
2.Face Detection:

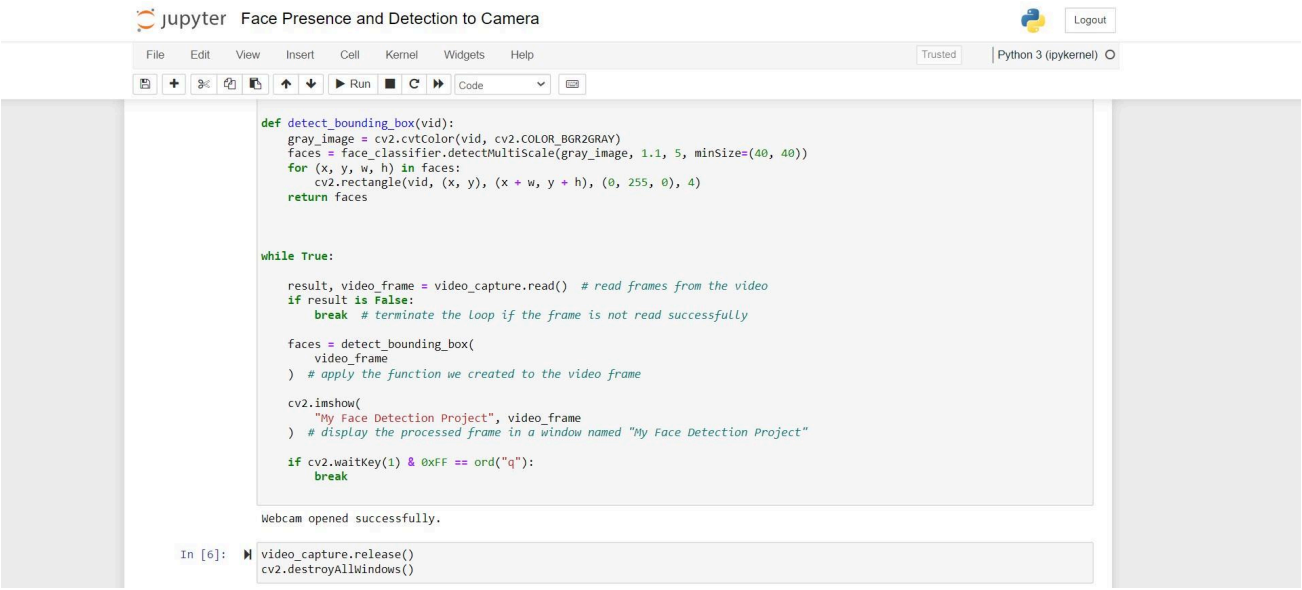
- Developed a detect_bounding_box() function that:
- Converts video frames to grayscale.
- Detects faces and draws bounding boxes around them.

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





My Face Detection Project appears on the screen