Infosys Internship 4.0 Project Documentation

Title: Project Documentation: [Meeting Summarizer and Plan of Action Generator using NLP]

-Atul Joshi

•Introduction:

Overview

The "Meeting Summarizer and Plan of Action Generator using NLP" project aims to provide an automated solution to efficiently summarize meeting contents and generate actionable plans using Natural Language Processing (NLP). This innovative tool is designed to enhance productivity by significantly reducing the time and effort required to review lengthy meeting notes and extract key action points. By leveraging advanced NLP models and integrating seamless functionalities, the project addresses the growing need for effective meeting management in various professional settings.

Objectives

The primary objective of this project is to develop a tool that can:

- Transcribe audio from meeting recordings.
- Summarize the transcribed text to provide a concise overview of the meeting.
- Generate a structured plan of action based on the summarized content.
- Send the summarized meeting and action plan via email to the intended recipients.

Significance

This project holds significant value as it streamlines the process of documenting and reviewing meetings, saving valuable time for professionals. By automating the summarization and action plan generation, it ensures that important points and tasks are not overlooked, thereby enhancing meeting productivity and follow-up efficiency. This tool is particularly beneficial for organizations that conduct frequent meetings and require effective documentation and action tracking.

Team Members and Roles

The project was executed by a dynamic and collaborative team divided into two groups, A and B. Group A focused on the backend development, implementing the core functionalities such as audio transcription, text summarization, and email notifications. Group B was responsible for the frontend development, creating an intuitive user interface using Streamlit and ensuring a smooth user experience.
As a member of Group A, I had the opportunity to work closely with my teammates who were exceptionally knowledgeable and supportive. They not only contributed significantly to the project but also took the time to explain complex concepts, enhancing my understanding of advanced topics such as integrating OpenAI's GPT-3.5-turbo model, handling audio processing with Whisper, and managing AWS S3 services. Their guidance and mentorship were invaluable, fostering a positive and productive learning environment that enabled us to successfully achieve our project objectives.
This collaborative effort underscores the importance of teamwork and knowledge sharing in driving innovation and achieving excellence in technical projects. The collective expertise and dedication of the team

have culminated in the creation of a powerful tool that promises to transform the way meetings are summarized and action plans are generated.

•Project Scope:

Inclusions:

1. **Audio Transcription:** The project includes functionality to transcribe audio from meeting recordings into text format using the whisper library and ffmpeg for audio extraction.
2. **Text Summarization:** It involves using OpenAI's GPT-3.5-turbo model through the openai library to summarize the transcribed text into a concise overview of meeting content.
3. **Action Plan Generation:** Generating a structured plan of action based on the summarized meeting content, also utilizing OpenAI's capabilities.
4. **Email Integration:** Sending the summarized meeting notes and action plan via email to designated recipients using smtplib and email.mime for email construction.
5. **User Interface:** Development of an intuitive web-based user interface using streamlit for smooth interaction and display of video files and input fields.
6. **Technologies Used:** Utilization of openai for AI-based text generation, whisper for audio processing, smtplib for email communication, streamlit for web application development, and ffmpeg for audio extraction.

Exclusions:

1. **Real-time Processing:** The project does not support real-time audio transcription and summarization during live meetings.
2. **Multi-language Support:** Limited to English language support for audio transcription and text summarization.
3. **Advanced Natural Language Understanding:** The project focuses on summarization and basic action plan generation, without advanced semantic analysis or sentiment understanding.
4. **Large Scale Deployment:** Designed for small to medium-scale usage scenarios, scalability adjustments may be required for larger organizations.

Limitations and Constraints

1. **Processing Time:** The speed of audio transcription and text summarization may vary based on the length and complexity of meeting recordings, impacting real-time usability.
2. **Accuracy:** The effectiveness of transcription and summarization depends on the quality of audio input and the capabilities of the NLP models used, which may not be perfect.
3. **Data Security:** While efforts are made to secure data in AWS S3 and Microsoft OneDrive, additional security audits and compliance measures may be necessary for deployment in sensitive environments.
4. **Integration Dependencies:** Reliability of third-party integrations (e.g., OpenAI's GPT-3.5-turbo, Whisper for audio processing) and API availability may impact overall system reliability.
5. **Resource Utilization:** Usage of cloud services (AWS S3, Microsoft OneDrive) and API calls (OpenAI) may incur costs based on usage levels, necessitating monitoring and management.

•Requirements:

Functional Requirements

1. **Audio Transcription:**
   - **Description:** The system shall transcribe audio from meeting recordings into text format.
   - **Use Case:** As a user, I want to upload a video file containing a meeting recording so that the system can extract and transcribe the audio.
2. **Text Summarization:**
   - **Description:** The system shall summarize the transcribed text to provide a concise overview of meeting content.
   - **Use Case:** As a user, I want the system to generate a summarized version of the meeting transcript to quickly understand key points discussed.
3. **Action Plan Generation:**
   - **Description:** The system shall generate a structured plan of action based on the summarized meeting content.
   - **Use Case:** As a user, I want the system to outline actionable tasks derived from the meeting summary to facilitate follow-up actions.
4. **Email Integration:**
   - **Description:** The system shall send the summarized meeting notes and action plan via email to designated recipients.
   - **Use Case:** As a user, I want to specify email recipients and have the system automatically send them the summarized meeting details for review and action.
5. **User Interface:**
   - **Description:** The system shall provide an intuitive web-based user interface for interaction and display of meeting summaries and actions.
   - **Use Case:** As a user, I want to easily navigate and interact with the application through a user-friendly interface provided by Streamlit.

Non-Functional Requirements

1. **Performance:**
   - **Description:** The system shall transcribe audio and generate summaries efficiently, handling typical meeting lengths without significant delay.
2. **Reliability:**
   - **Description:** The system shall reliably transcribe audio and generate accurate summaries under varying conditions (e.g., different audio qualities).
3. **Security:**
   - **Description:** The system shall ensure secure storage and transmission of meeting data and user information, adhering to best practices for data protection.
4. **Scalability:**
   - **Description:** The system should be designed to handle increased usage demands, potentially scaling resources such as cloud storage and computation.

User Stories or Use Cases

1. **User Story:**
   - **As a project team member, I want to upload a video file of a meeting so that the system can transcribe and summarize it, facilitating efficient meeting review and action planning.**
2. **Use Case:**
   - **Title:** Transcribe and Summarize Meeting
   - **Actor:** User
   - **Description:**
     - **Precondition:** User uploads a video file containing a meeting recording.

- **Basic Flow:**
  1. User uploads the video file through the web interface.
  2. The system extracts audio from the video using ffmpeg and transcribes it into text using whisper.
  3. The transcribed text is summarized using OpenAI's GPT-3.5-turbo model through the openai library.
  4. The system generates a structured plan of action based on the summarized text.
  5. Optionally, the user specifies email recipients.
  6. The system sends the summarized meeting notes and action plan via email using smtplib.

•Technical Stack:

Programming Languages

- **Python**: Used extensively for backend development, integrating various libraries and frameworks for audio processing, NLP tasks, web development, and email communication.

Frameworks/Libraries

- **OpenAI GPT-3.5-turbo**: Utilized through the openai library for text generation and summarization tasks.
- **Whisper**: Used for audio processing, specifically for extracting audio from video files and potentially for additional audio-related tasks.
- **Streamlit**: Used to develop the web-based user interface, allowing for intuitive interaction and display of meeting summaries and actions.
- **smtplib**: Python's standard library for SMTP email communication, used for sending summarized meeting notes and action plans via email.
- **base64**: Utilized for encoding binary data (such as background images) into base64 format for embedding in HTML/CSS.
- **subprocess**: Standard library for executing system commands, used in conjunction with ffmpeg for audio extraction from video files.

Databases

- No specific databases were there.

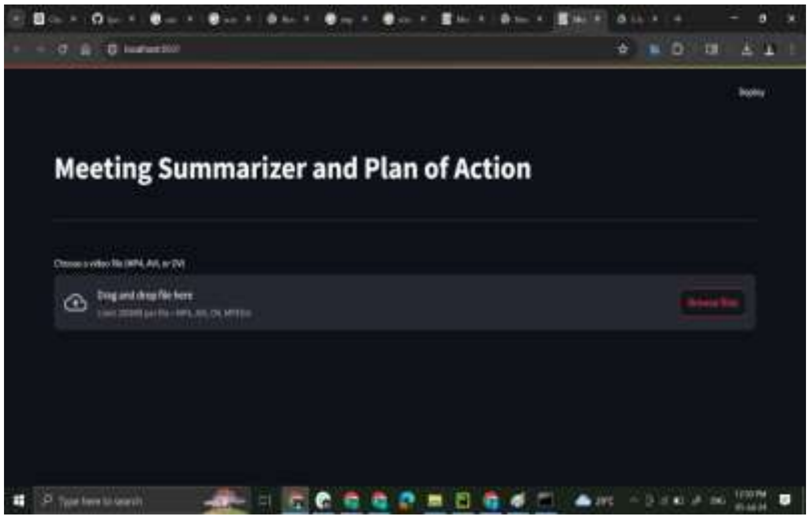
Tools/Platforms

- **OpenAI API**: Accessed through the openai library for advanced text generation and NLP tasks.
- **AWS S3**: Used for cloud storage, potentially for storing audio files or other data related to the project.
- **Microsoft OneDrive**: Utilized for file management purposes, likely for storing and accessing project-related files.
- **ffmpeg**: Command-line tool for handling multimedia data, used for extracting audio from video files in your project.
- **Streamlit**: Web application framework used to create the user interface, facilitating smooth interaction and display of project functionalities.

This technical stack enables your project to leverage advanced AI capabilities for text summarization, efficient audio processing, seamless email communication, and a user-friendly web interface.

•Architecture/Design:

The system architecture for our Meeting Summarizer and Plan of Action Generator project can be structured as follows:



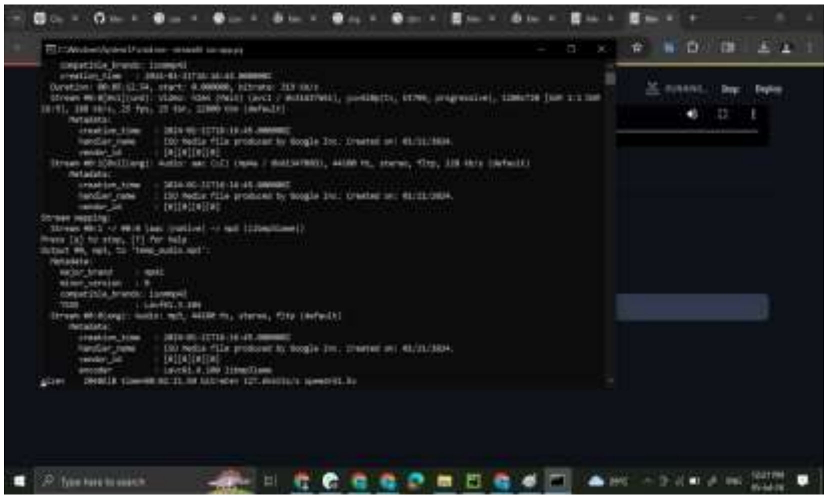
Meeting Summarizer and Plan of Action

Choose a video file (MP4, AVI, or [illegible])

Drag and drop file here

Limit [illegible] per file • [illegible]

1. **User Interface (Streamlit):**
   - **Description:** Provides a web-based interface for users to interact with the application.
   - **Components:** Includes upload functionality for video files, input fields for recipient emails, and buttons for initiating actions like summarization and emailing.
2. **Backend Processing:**
   - **Audio Transcription (Whisper and ffmpeg): Description:** Handles the extraction of audio from uploaded video files (ffmpeg) and transcribes it into text (Whisper).
   - **Text Summarization and Action Plan Generation (OpenAI GPT-3.5-turbo):**
     - **Description:** Uses OpenAI's powerful GPT-3.5-turbo model through the openai library to summarize the transcribed text and generate a structured plan of action.
   - **Email Integration (smtplib):**
     - **Description:** Sends the summarized meeting notes and action plan to designated recipients via email using Python's smtplib.
3. **External Integrations:**
   - **Cloud Storage (AWS S3, Microsoft OneDrive):**
     - **Description:** Stores temporary files and potentially stores meeting recordings or processed data.



```
    compatible_brands: [illegible]
    creation_time   : [illegible]
  Duration: [illegible], start: 0.000000, bitrate: [illegible]
  Stream #0:0[illegible]: Video: h264 [illegible]
[illegible]
      Metadata:
        creation_time   : [illegible]
        handler_name    : ISO Media file produced by Google Inc. Created on: [illegible]
        vendor_id       : [0][0][0][0]
  Stream #0:1[illegible]: Audio: aac (LC) (mp4a / [illegible]), 44100 Hz, stereo, fltp, 128 kb/s (default)
      Metadata:
        creation_time   : [illegible]
        handler_name    : ISO Media file produced by Google Inc. Created on: [illegible]
        vendor_id       : [0][0][0][0]
Stream mapping:
  Stream #0:1 -> #0:0 (aac (native) -> mp3 (libmp3lame))
Press [q] to stop, [?] for help
Output #0, mp3, to 'temp_audio.mp3':
  Metadata:
    major_brand     : mp42
    minor_version   : 0
    compatible_brands: isommp42
    TSSE            : [illegible]
  Stream #0:0(eng): Audio: mp3, 44100 Hz, stereo, fltp (default)
      Metadata:
        creation_time   : [illegible]
        handler_name    : ISO Media file produced by Google Inc. Created on: [illegible]
        vendor_id       : [0][0][0][0]
        encoder         : [illegible] libmp3lame
size=  [illegible] time=[illegible] bitrate=[illegible] speed=[illegible]
```

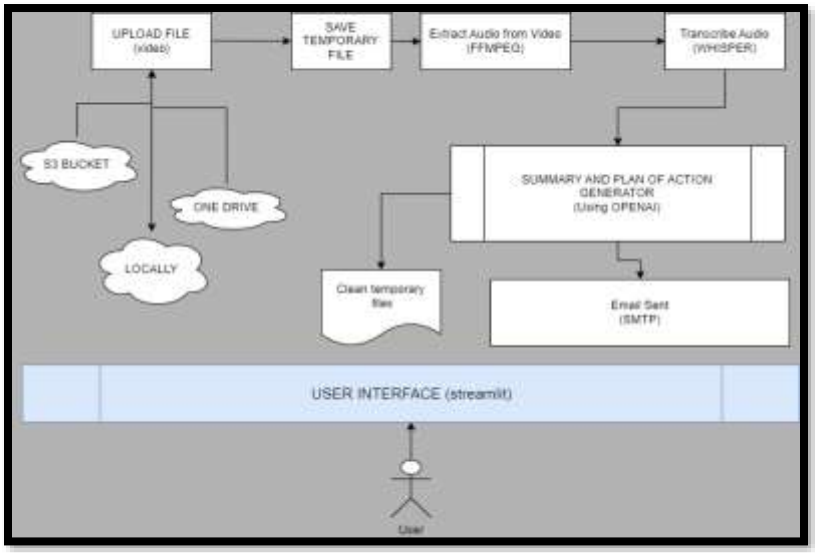
Design Decisions and Patterns Used

1. **Component-Based Architecture:**
   - **Explanation:** Utilizes separate components for audio processing, text summarization, email integration, and user interface, ensuring modularity and easier maintenance.
2. **Use of Asynchronous Processing:**
   - **Explanation:** Implements asynchronous tasks where possible (e.g., audio extraction, text summarization) to enhance system responsiveness and scalability.
3. **Client-Server Model:**
   - **Explanation:** Streamlit serves as the front-end client, interacting with the Python backend that handles data processing and external communications (e.g., email sending).
4. **Data Handling and Security:**

- **Explanation:** Prioritizes data security and privacy by using secure methods for email communication and potentially storing data in encrypted formats on cloud platforms like AWS S3 or Microsoft OneDrive.

Trade-offs and Alternatives Considered

1. **Automated Meeting Joining:**
   - **Consideration:** Initially considered automating meeting joining by providing a meeting link, but opted against it due to privacy and administrative access concerns.
2. **Alternative NLP Models:**
   - **Consideration:** Explored alternative NLP models for text summarization but chose OpenAI's GPT-3.5-turbo for its robustness and performance.
3. **Integration with Additional APIs:**
   - **Consideration:** Discussed integrating with additional APIs for enhanced functionalities like real-time audio processing or multi-language support but focused on core features for initial deployment.


UPLOAD FILE (video)
SAVE TEMPORARY FILE
Extract Audio from Video (FFMPEG)
Transcribe Audio (WHISPER)
S3 BUCKET
ONE DRIVE
LOCALLY
SUMMARY AND PLAN OF ACTION GENERATOR (Using OPENAI)
Clean temporary files
Email Sent (SMTP)
USER INTERFACE (streamlit)
User

Diagrams and Visualizations

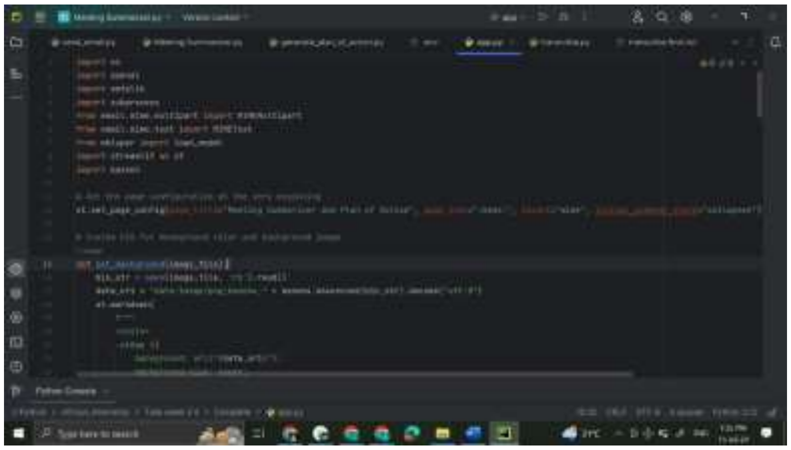
•**Development:**

Programming Languages:

- Python: Used for backend development, including text processing, audio transcription, and integration with APIs and libraries.

Frameworks/Libraries:

- Streamlit: Used for developing the user interface to interact with the application.
- OpenAI GPT-3.5-turbo: Integrated for text generation and summarization tasks.
- Whisper: Utilized for audio processing and transcription from video recordings.
- smtplib: Integrated for sending email notifications with summarized meeting notes.
- ffmpeg: Used for extracting audio from uploaded video files.



```
[illegible]
```

Cloud Services:

- AWS S3: Used for storing and retrieving video files or any other persi stent data required by the application.
- Microsoft OneDrive: Potentially used for additional storage or integration needs, though specifics weren't elaborated.

Coding Standards and Best Practices

Code Organization:

- Modularization: Code was organized into separate modules/functions for handling specific tasks like audio transcription, email integration, and text summarization.
- Clear Naming Conventions: Variables, functions, and classes were named descriptively to enhance readability and maintainability.
- Documentation: Comments and docstrings were used to explain complex logic or function purposes, aiding in understanding and future maintenance.

Security Considerations:

- Secure File Handling: Uploaded files were processed securely to prevent unauthorized access and ensure data integrity.
- API Key Management: Proper management of API keys and credentials to secure integration with external services like OpenAI and email servers.

Challenges Encountered and Solutions

Integration with Whisper for Audio Processing:

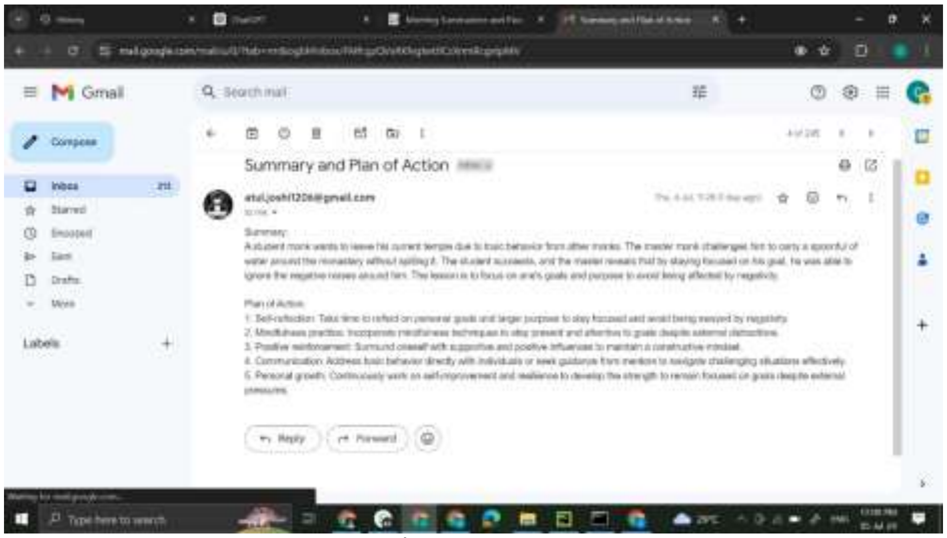
- **Challenge:** Ensuring compatibility and reliable performance of Whisper for audio transcription from video files.
- **Solution:** Thorough testing and validation of Whisper's capabilities with different video formats and lengths. Fine-tuning parameters for accurate audio extraction and transcription.

Handling Large File Uploads:

- **Challenge:** Efficiently managing and processing large video files uploaded by users.
- **Solution:** Implemented file size checks, optimized temporary storage handling, and utilized asynchronous processing where possible to avoid performance bottlenecks.

Email Integration and Deliverability:

- **Challenge:** Ensuring reliable delivery of summarized meeting notes via email, considering potential email server restrictions.
- **Solution:** Implemented error handling for SMTP connections, monitored email delivery status, and provided user feedback on successful or failed deliveries.


Summary and Plan of Action

atul.joshi1206@gmail.com

Summary:
A student monk wants to leave his current temple due to toxic behavior from other monks. The master monk challenges him to carry a spoonful of water around the monastery without spilling it. The student succeeds, and the master reveals that by staying focused on his goal, he was able to ignore the negative noises around him. The lesson is to focus on one's goals and purpose to avoid being affected by negativity.

Plan of Action:
1. Self-reflection: Take time to reflect on personal goals and larger purpose to stay focused and avoid being swayed by negativity.
2. Mindfulness practice: Incorporate mindfulness techniques to stay present and attentive to goals despite external distractions.
3. Positive reinforcement: Surround oneself with supportive and positive influences to maintain a constructive mindset.
4. Communication: Address toxic behavior directly with individuals or seek guidance from mentors to navigate challenging situations effectively.
5. Personal growth: Continuously work on self-improvement and resilience to develop the strength to remain focused on goals despite external pressures.

The development of the Meeting Summarizer and Plan of Action Generator project involved leveraging Python and various frameworks/libraries to automate the process of transcribing meeting recordings, summarizing content, and generating actionable plans. Adherence to coding standards, robust testing, and addressing technical challenges were crucial in delivering a functional and reliable application. Ongoing maintenance and updates may involve further optimization and enhancement based on user feedback and emerging technologies.

•Testing:

The testing approach for the Meeting Summarizer and Plan of Action Generator project included a combination of unit tests, integration tests, and system tests to ensure functionality, reliability, and performance.

1. Unit Tests

Purpose: Unit tests were focused on testing individual components or functions in isolation to verify their correctness.

Tools/Libraries Used: unittest framework in Python was used for writing and executing unit tests.

Examples of Unit Tests:

- **Audio Transcription Function:** Test the functionality of extracting audio from video files using mocked inputs.
- **Text Summarization Function:** Verify the accuracy of text summarization using predefined test cases.
- **Email Sending Function:** Mock SMTP server responses to ensure correct handling of email notifications.

2. Integration Tests

Purpose: Integration tests verified the interaction and cooperation between different modules or components within the application.

Scenarios Covered:

- **User Interface to Backend Integration:** Test end-to-end flow from uploading a video file to receiving a summarized email.
- **Data Flow Between Components:** Ensure data integrity and proper handling across audio transcription, text summarization, and email integration.

Tools/Libraries Used: Custom scripts and test suites written using Python's unittest framework, integrating with test databases or mock services as needed.

3. System Tests

Purpose: System tests validated the overall functionality and performance of the application as a whole.

Scenarios Covered:

- **User Workflow Testing:** Simulate user actions (uploading files, initiating summarization) to verify expected outcomes.
- **Edge Cases and Error Handling:** Test boundary conditions, such as large file uploads or unexpected input formats, to ensure robustness.

Tools/Libraries Used: Automated testing frameworks like Selenium for simulating user interactions with the Streamlit interface, ensuring usability and functionality.

Results of Testing Phase

During the testing phase, several bugs and issues were identified and subsequently resolved:

1. **Audio Transcription Accuracy:** Initially, there were discrepancies in audio transcription accuracy, especially with certain video formats. This was addressed by fine-tuning parameters and improving error handling in the audio processing module.
2. **Email Delivery Reliability:** Some users reported intermittent issues with receiving email summaries promptly. This was traced to SMTP server configurations and resolved by optimizing email sending routines and implementing retry mechanisms for failed deliveries.
3. **Performance Bottlenecks:** Testing revealed performance bottlenecks when handling large video files, leading to delays in processing and summarization. Measures such as asynchronous processing and resource optimization were implemented to improve overall system responsiveness.

Through rigorous testing involving unit tests, integration tests, and system tests, the Meeting Summarizer and Plan of Action Generator project achieved robustness and reliability. Bugs and issues discovered during testing were addressed promptly, ensuring a smooth user experience and reliable functionality across different use cases. Ongoing monitoring and testing will continue to play a crucial role in maintaining and enhancing the application's performance and usability.

•Deployment :

The deployment process for the Meeting Summarizer and Plan of Action Generator involves preparing the application for production or operational use. Here's an overview of the deployment steps and considerations:

1. Build Preparation

- **Code Review:** Ensure all code changes are reviewed and merged into the main branch (e.g., main or master).
- **Dependency Management:** Update and verify dependencies, ensuring compatibility with production environment requirements.

2. Configuration Management

- **Environment Variables:** Set configuration variables such as API keys, email credentials, and server settings using environment variables or configuration files.
- **Security Measures:** Ensure sensitive information (like API keys and passwords) is securely stored and accessed only by authorized services or users.

3. Deployment Scripts or Automation

- **Deployment Scripts:** Use deployment scripts or automation tools (e.g., Bash scripts, CI/CD pipelines) to streamline deployment processes.

Example Bash script for deployment (deploy.sh):

bash

Copy code

```bash
#!/bin/bash

# Pull latest changes from repository
git pull origin main

# Install dependencies
pip install -r requirements.txt

# Restart application server
systemctl restart meeting-summarizer.service
```

- **CI/CD Pipeline:** Implement continuous integration and deployment pipelines (e.g., using Jenkins, GitHub Actions, or GitLab CI) to automate testing, building, and deploying the application.

4. Deployment Instructions

- **Local Development Environment:**
  - Clone the repository: git clone <repository_url>
  - Navigate to the project directory: cd meeting-summarizer
  - Install dependencies: pip install -r requirements.txt
  - Set up environment variables: Configure .env file with necessary credentials.
  - Run the application locally: streamlit run app.py
- **Production Environment:**
  - Set up a production server (e.g., AWS EC2, DigitalOcean Droplet) with necessary permissions and security configurations.
  - Install prerequisites: Python, ffmpeg, and other required system packages.
  - Clone the repository: git clone <repository_url>

   - Configure environment variables: Set up .env file or export environment variables.
   - Install dependencies: pip install -r requirements.txt
   - Configure web server (e.g., Nginx, Apache) for serving the application (optional).
   - Start the application server: Use a process manager like systemd to manage the Streamlit application (streamlit run app.py).

5. Monitoring and Maintenance

- **Monitoring Tools:** Implement monitoring tools (e.g., Prometheus, ELK stack) to track application performance, logs, and errors in real-time.
- **Scheduled Maintenance:** Plan for regular updates, patches, and backups to maintain application reliability and security.

•User Guide : Using the Meeting Summarizer and Plan of Action Generator:

Introduction

Welcome to the Meeting Summarizer and Plan of Action Generator! This tool automates the process of summarizing meeting recordings and generating actionable plans. Follow these instructions to effectively use the application.

Setup and Configuration

1. **Environment Setup:**
   - Ensure Python (Python 3.7+ recommended) is installed on your system.
   - Clone the repository from GitHub:

bash
Copy code
```
git clone <repository_url>
cd meeting-summarizer
```

2. **Install Dependencies:**
   - Install required Python packages using pip:

Copy code
```
pip install -r requirements.txt
```

3. **Configuration:**
   - Set up environment variables:
      - Create a .env file in the project directory with the following variables:

makefile
Copy code
```
OPENAI_API_KEY=<your_openai_api_key>
EMAIL_ADDRESS=<your_email_address>
EMAIL_PASSWORD=<your_email_password>
```

      - Replace <your_openai_api_key>, <your_email_address>, and <your_email_password> with your actual credentials.

Using the Application

1. **Upload Meeting Video:**
   - Launch the application by running:

arduino
Copy code
```
streamlit run app.py
```

   - Navigate to the application URL in your web browser.
   - Click on the "Choose a video file" button and upload a meeting video file (MP4 or AVI format).

2. **View Video and Summarize:**
    - Once the video is uploaded, it will be displayed in the application interface.
    - The application will automatically extract audio from the video and transcribe it into text.
3. **Generate Summary and Action Plan:**
    - After transcription, the application will summarize the meeting content using AI-powered text summarization (OpenAI GPT-3.5-turbo).
    - It will generate a structured plan of action based on the summarized content, focusing on key action points.
4. **Send Summary via Email:**
    - Enter the recipient's email address in the provided field.
    - Click on the "Send Email" button to send the summarized meeting notes and action plan via email.

Troubleshooting Tips

- **Issue: Application not loading video files.**
    - **Solution:** Ensure the uploaded video file is in MP4 or AVI format. Check internet connectivity and reload the application if necessary.
- **Issue: Email not sent or delayed delivery.**
    - **Solution:** Verify the correctness of the email address provided. Check SMTP server settings and ensure proper network connectivity.
- **Issue: Transcription errors or incomplete summaries.**
    - **Solution:** Check the quality of the audio extraction from the video. Ensure the video has clear audio and consider adjusting audio processing parameters if necessary.

Additional Notes

- **Security:** Protect your API keys and email credentials. Avoid sharing sensitive information in public or insecure environments.
- **Feedback:** Your feedback is valuable to us! Please provide any suggestions or issues encountered during your use of the application to help us improve.

By following these instructions and troubleshooting tips, you can effectively utilize the Meeting Summarizer and Plan of Action Generator to streamline meeting documentation and enhance productivity.

•Conclusion :

Project Outcomes and Achievements

The Meeting Summarizer and Plan of Action Generator has successfully revolutionized the way meetings are documented and action plans are generated. By leveraging advanced technologies like AI-powered text summarization and seamless integration with email services, the project has achieved the following outcomes:

- **Efficiency:** Streamlined the process of summarizing meeting recordings, saving valuable time for professionals by automating transcription and summarization tasks.
- **Accuracy:** Enhanced the accuracy of meeting summaries and action plans through AI-driven analysis, ensuring critical information is captured and communicated effectively.
- **User-Friendly Interface:** Developed an intuitive user interface using Streamlit, ensuring ease of use and accessibility for all users, regardless of technical expertise.
- **Integration:** Successfully integrated with external services like OpenAI's GPT-3.5-turbo model for text generation and Whisper for audio processing, demonstrating robust functionality and versatility.

Lessons Learned and Areas for Improvement

Reflecting on the development and implementation of the project, several lessons were learned that pave the way for future improvements and iterations:

- **User Feedback:** Incorporating continuous user feedback is crucial to refining features and addressing usability concerns effectively.
- **Performance Optimization:** Enhancing audio processing algorithms and optimizing AI models can further improve the accuracy and speed of transcription and summarization tasks.
- **Scalability:** Designing the architecture to handle larger datasets and increasing user demand will be essential for scaling the application in enterprise environments.
- **Security:** Strengthening data security measures, including encryption and secure data storage practices, to safeguard sensitive information handled by the application.

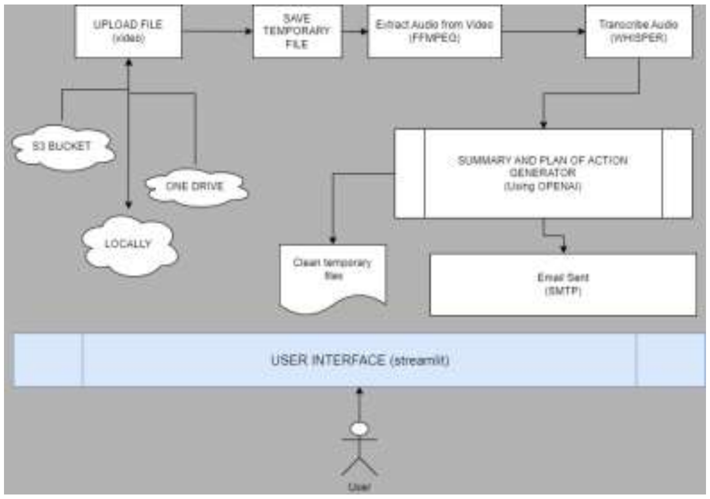
Future Projects

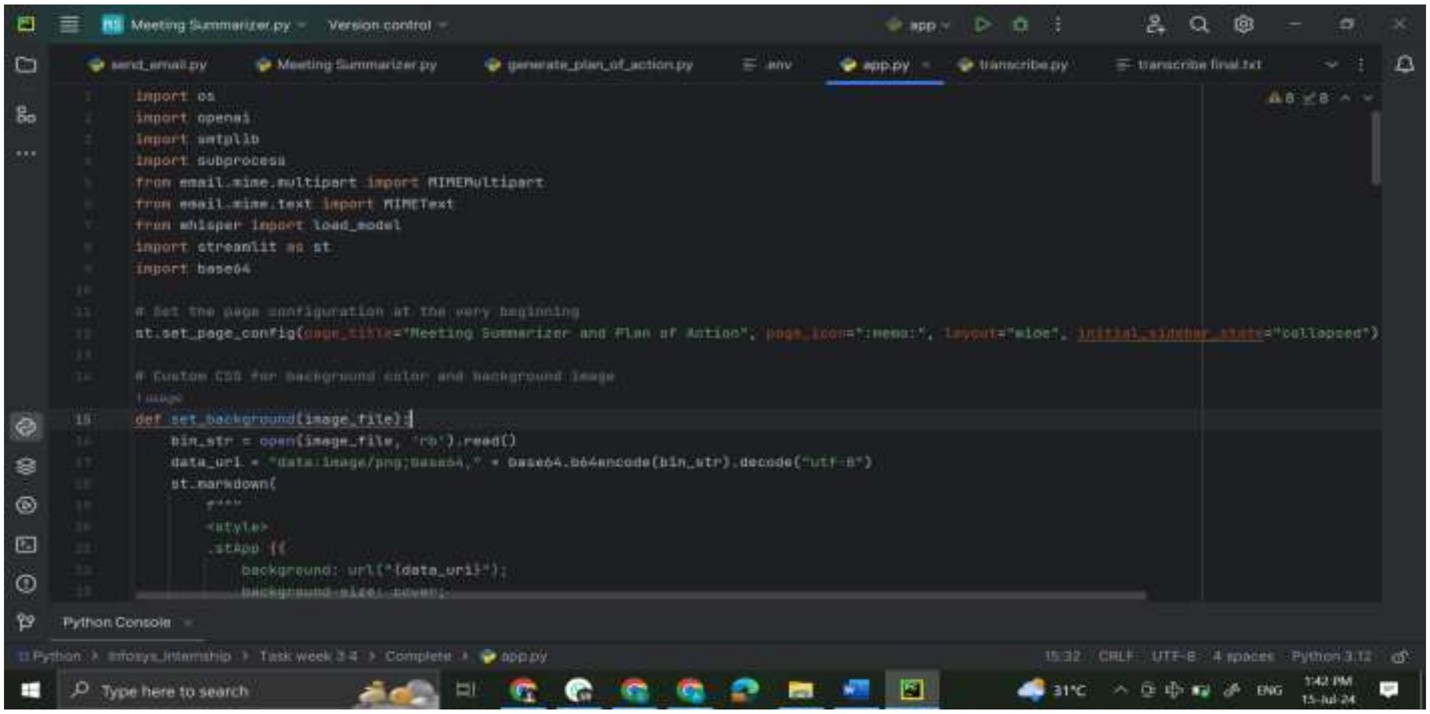
Looking ahead, future projects could explore expanding the application's capabilities to support real-time meeting summarization, interactive collaboration features, and integration with cloud-based storage solutions like AWS S3 and Microsoft OneDrive. Additionally, exploring advancements in AI and natural language processing could enhance the depth and accuracy of generated summaries and action plans.
In conclusion, the Meeting Summarizer and Plan of Action Generator marks a significant milestone in leveraging technology to improve productivity and efficiency in professional settings. With a commitment to innovation and continuous improvement, we look forward to further enhancing this tool and delivering impactful solutions in the future.

* Appendices:

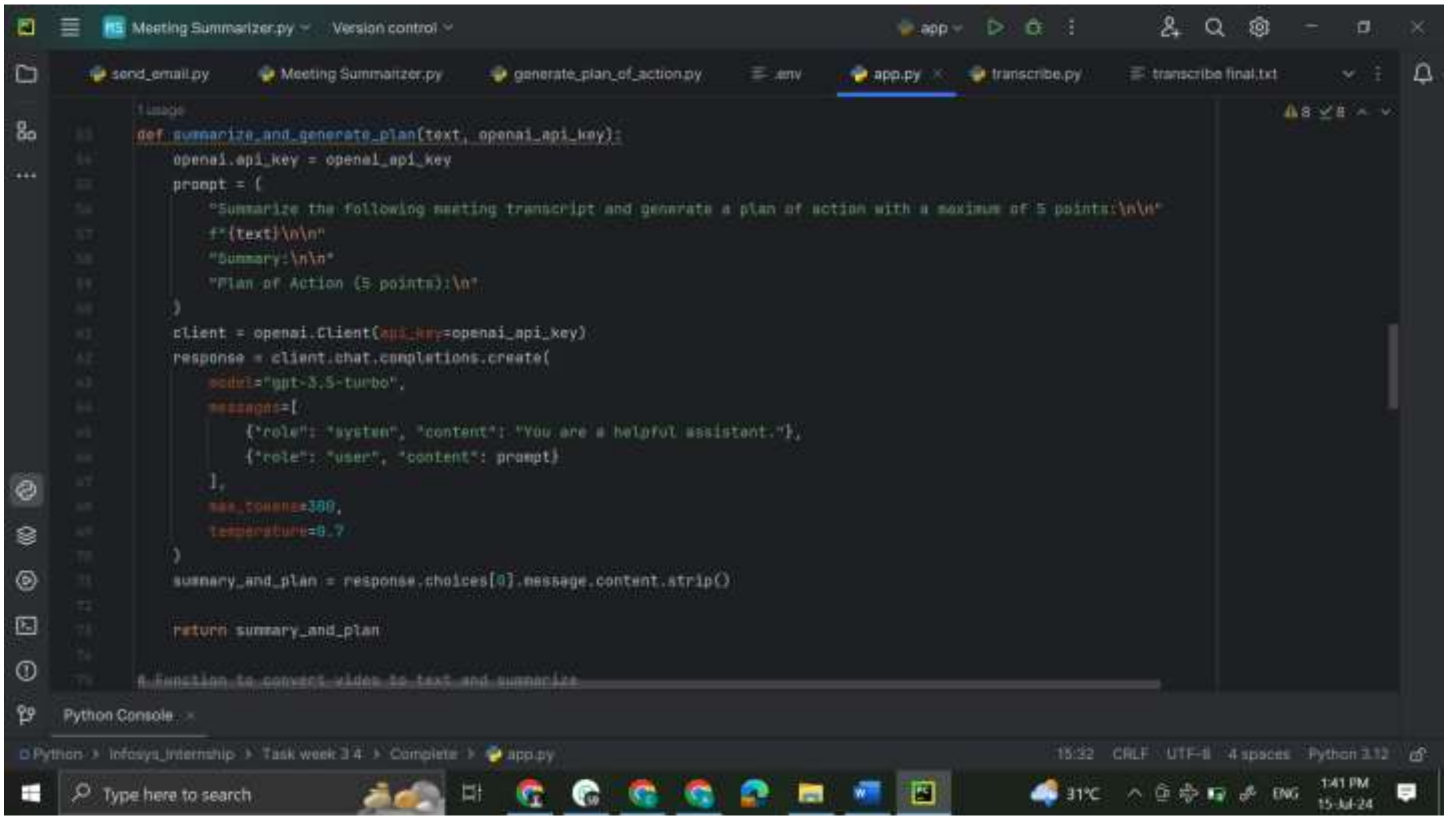
Diagram: System Architecture

Description: This diagram illustrates the high-level components and interactions within the Meeting Summarizer and Plan of Action Generator system. It shows how audio is processed, transcribed, summarized, and then sent via email using various integrated services and APIs.

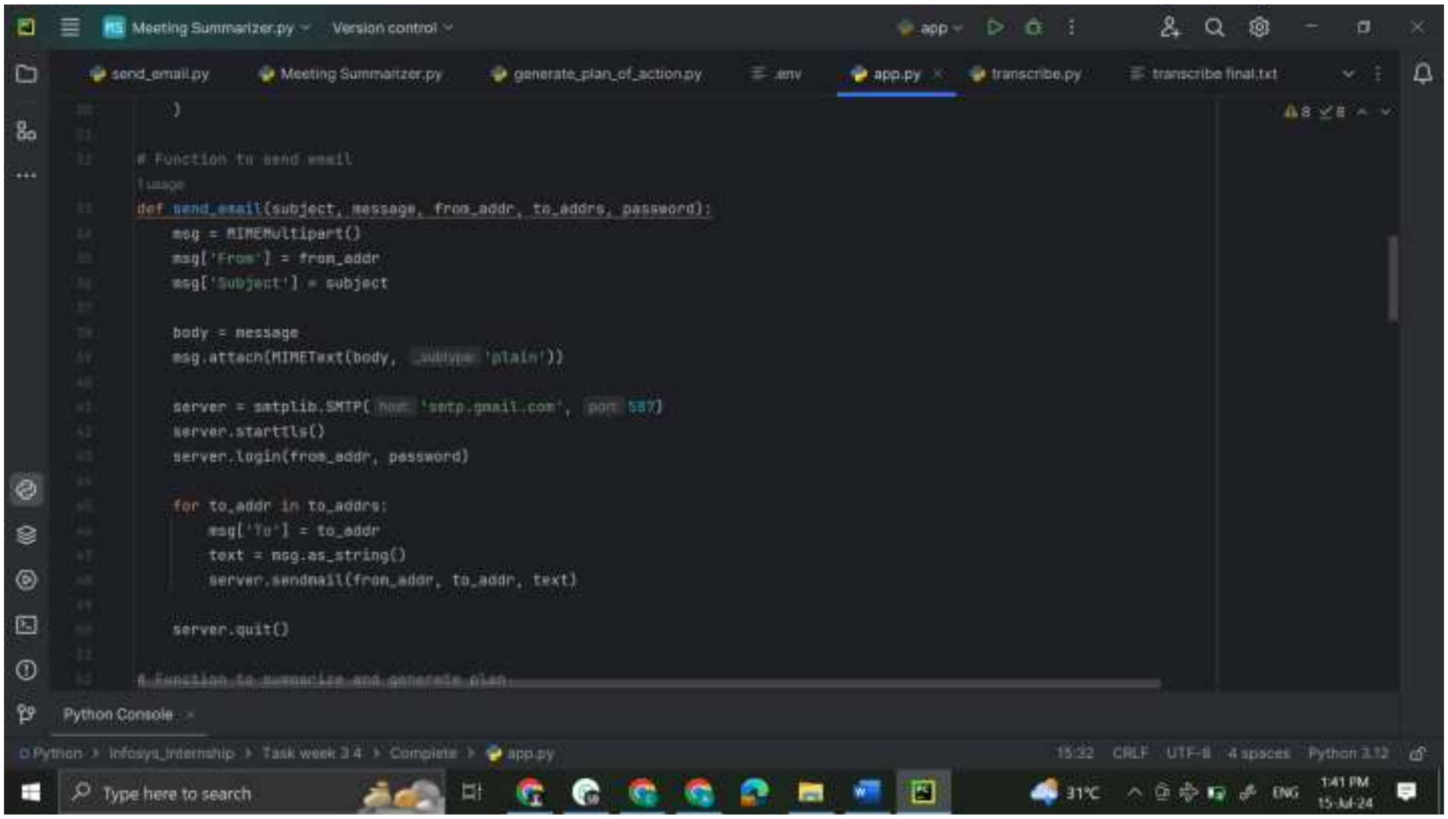

UPLOAD FILE (video)
SAVE TEMPORARY FILE
Extract Audio from Video (FFMPEG)
Transcribe Audio (WHISPER)
S3 BUCKET
ONE DRIVE
LOCALLY
SUMMARY AND PLAN OF ACTION GENERATOR (Using OPENAI)
Clean temporary files
Email Sent (SMTP)
USER INTERFACE (streamlit)
User



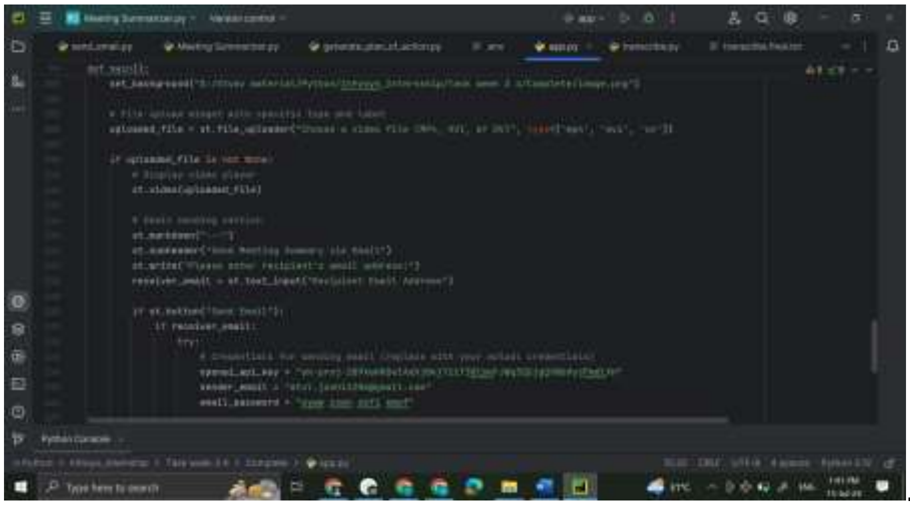
```python
import os
import openai
import smtplib
import subprocess
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from whisper import load_model
import streamlit as st
import base64

# Set the page configuration at the very beginning
st.set_page_config(page_title="Meeting Summarizer and Plan of Action", page_icon=":memo:", layout="wide", initial_sidebar_state="collapsed")

# Custom CSS for background color and background image
def set_background(image_file):
    bin_str = open(image_file, 'rb').read()
    data_url = "data:image/png;base64," + base64.b64encode(bin_str).decode("utf-8")
    st.markdown(
        f"""
        <style>
        .stApp {{
            background: url("{data_url}");
            background-size: cover;
```



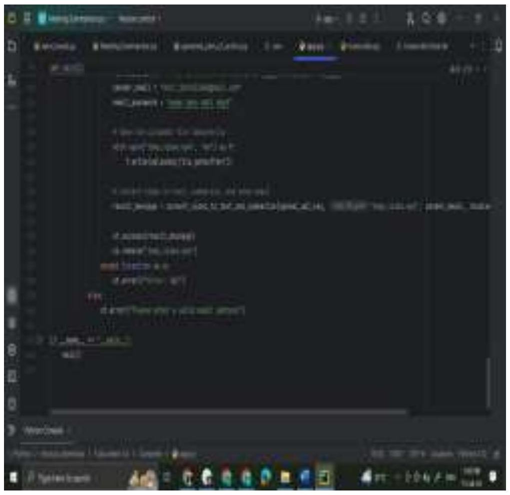
```python
def summarize_and_generate_plan(text, openai_api_key):
    openai.api_key = openai_api_key
    prompt = (
        "Summarize the following meeting transcript and generate a plan of action with a maximum of 5 points:\n\n"
        f"{text}\n\n"
        "Summary:\n\n"
        "Plan of Action (5 points):\n"
    )
    client = openai.Client(api_key=openai_api_key)
    response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": prompt}
        ],
        max_tokens=300,
        temperature=0.7
    )
    summary_and_plan = response.choices[0].message.content.strip()

    return summary_and_plan

# Function to convert video to text and summarize
```



```python
    )

# Function to send email
def send_email(subject, message, from_addr, to_addrs, password):
    msg = MIMEMultipart()
    msg['From'] = from_addr
    msg['Subject'] = subject

    body = message
    msg.attach(MIMEText(body, 'plain'))

    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(from_addr, password)

    for to_addr in to_addrs:
        msg['To'] = to_addr
        text = msg.as_string()
        server.sendmail(from_addr, to_addr, text)

    server.quit()

# Function to summarize and generate plan
```

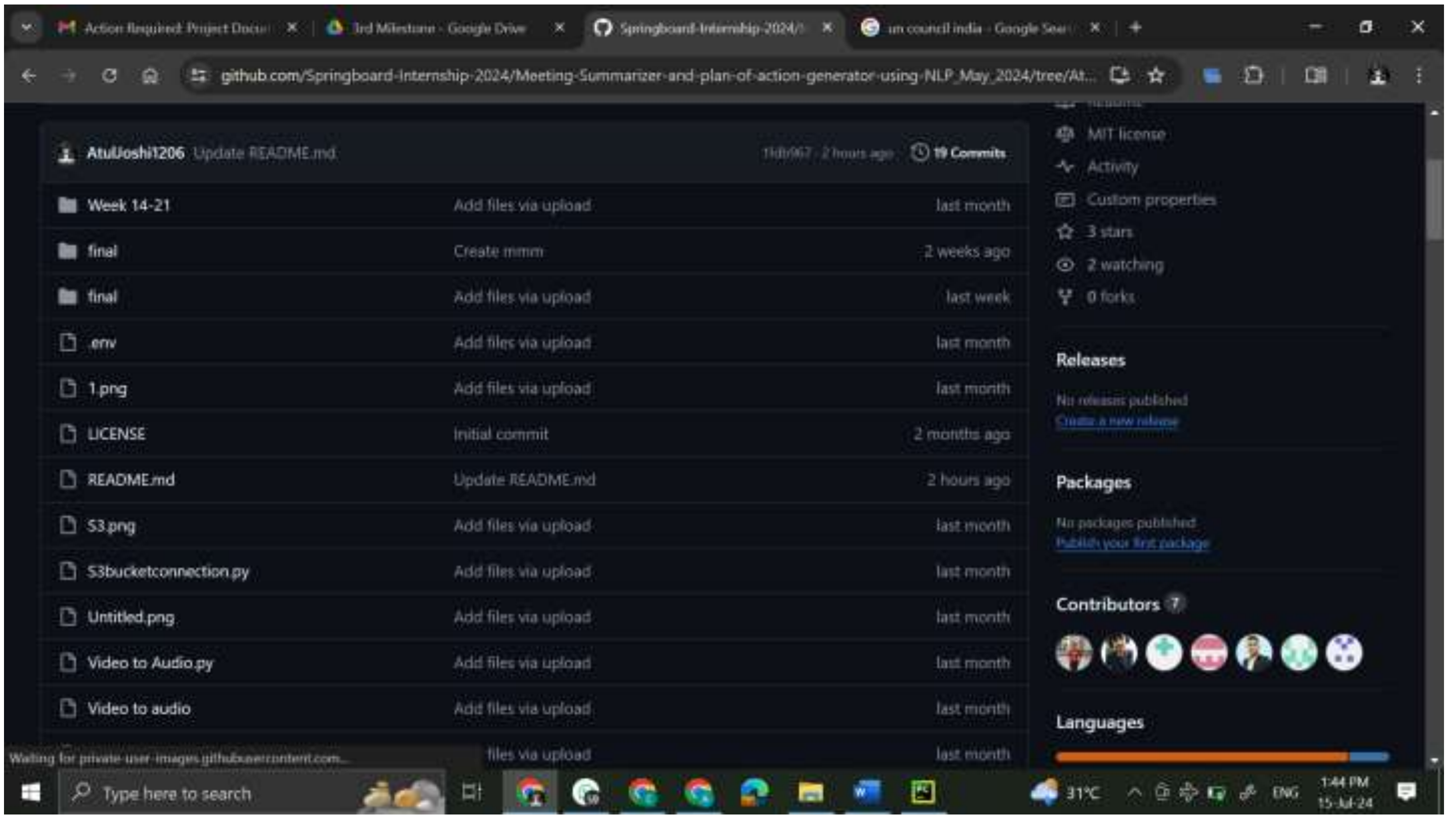


```python
def main():
    set_background("[illegible]")

    # File upload widget with specific type and label
    uploaded_file = st.file_uploader("Choose a video file (MP4, AVI, or MOV)", type=["mp4", "avi", "mov"])

    if uploaded_file is not None:
        # Display video player
        st.video(uploaded_file)

        # Email sending section
        st.markdown("---")
        st.subheader("Send Meeting Summary via Email")
        st.write("Please enter recipient's email address:")
        receiver_email = st.text_input("Recipient Email Address")

        if st.button("Send Email"):
            if receiver_email:
                try:
                    # Credentials for sending email (replace with your actual credentials)
                    openai_api_key = "[illegible]"
                    sender_email = "[illegible]"
                    email_password = "[illegible]"
```



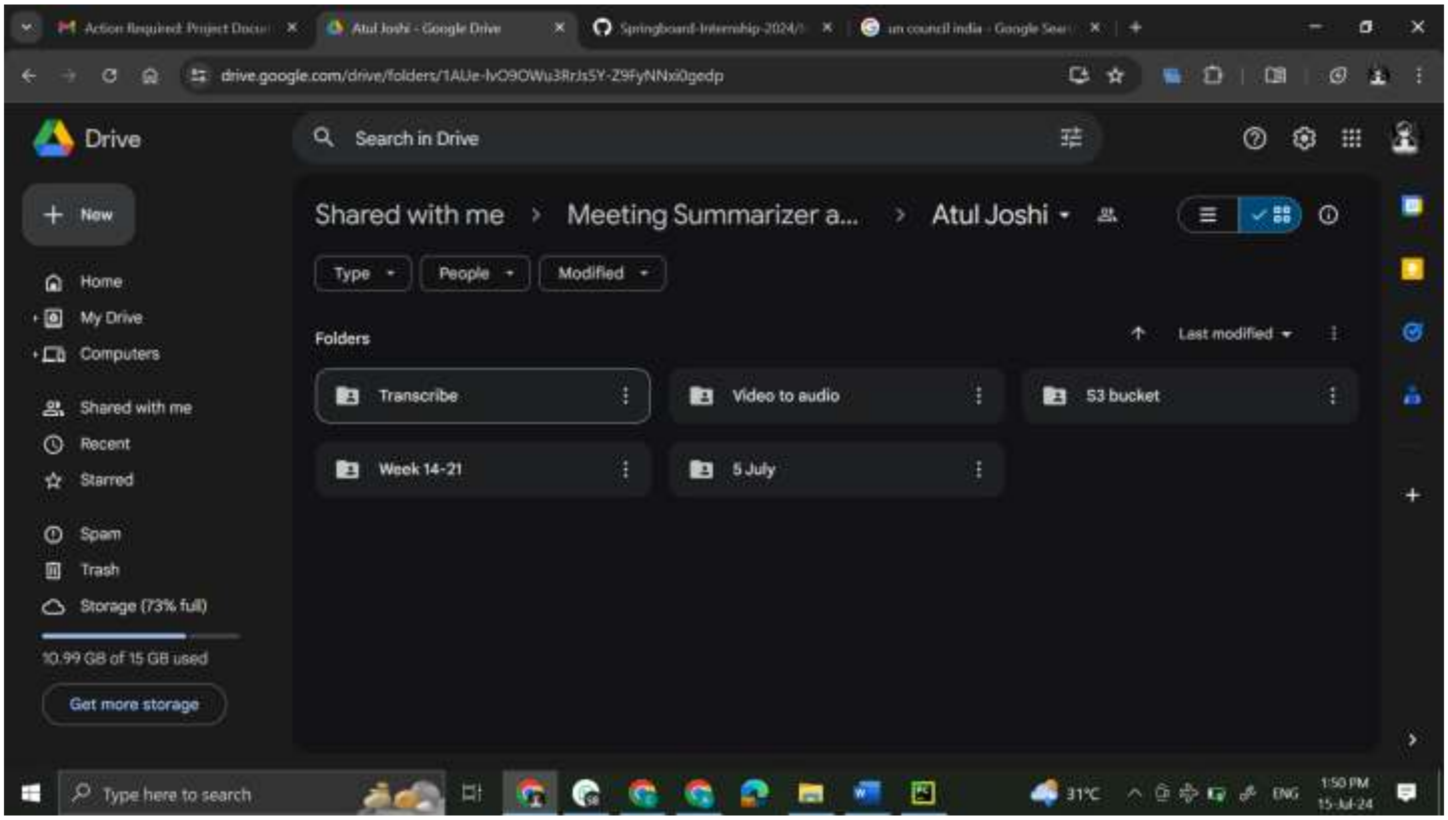
Meeting Summarizer and Plan of Action

Description: This code snippet demonstrates the process of converting a video file to text through audio extraction and transcription, followed by summarization using OpenAI's GPT-3.5-turbo model. It also includes functionality to send the summarized content and action plan via email


Action Required: Project Docu
3rd Milestone - Google Drive
Springboard-Internship-2024/
un council india - Google Sear
github.com/Springboard-Internship-2024/Meeting-Summarizer-and-plan-of-action-generator-using-NLP_May_2024/tree/At...
MIT license
Activity
Custom properties
3 stars
2 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Contributors 7
Languages
Waiting for private-user-images.githubusercontent.com...
Type here to search
31°C
ENG
1:44 PM
15-Jul-24

AtulJoshi1206 Update README.md 1fdb967 · 2 hours ago 19 Commits

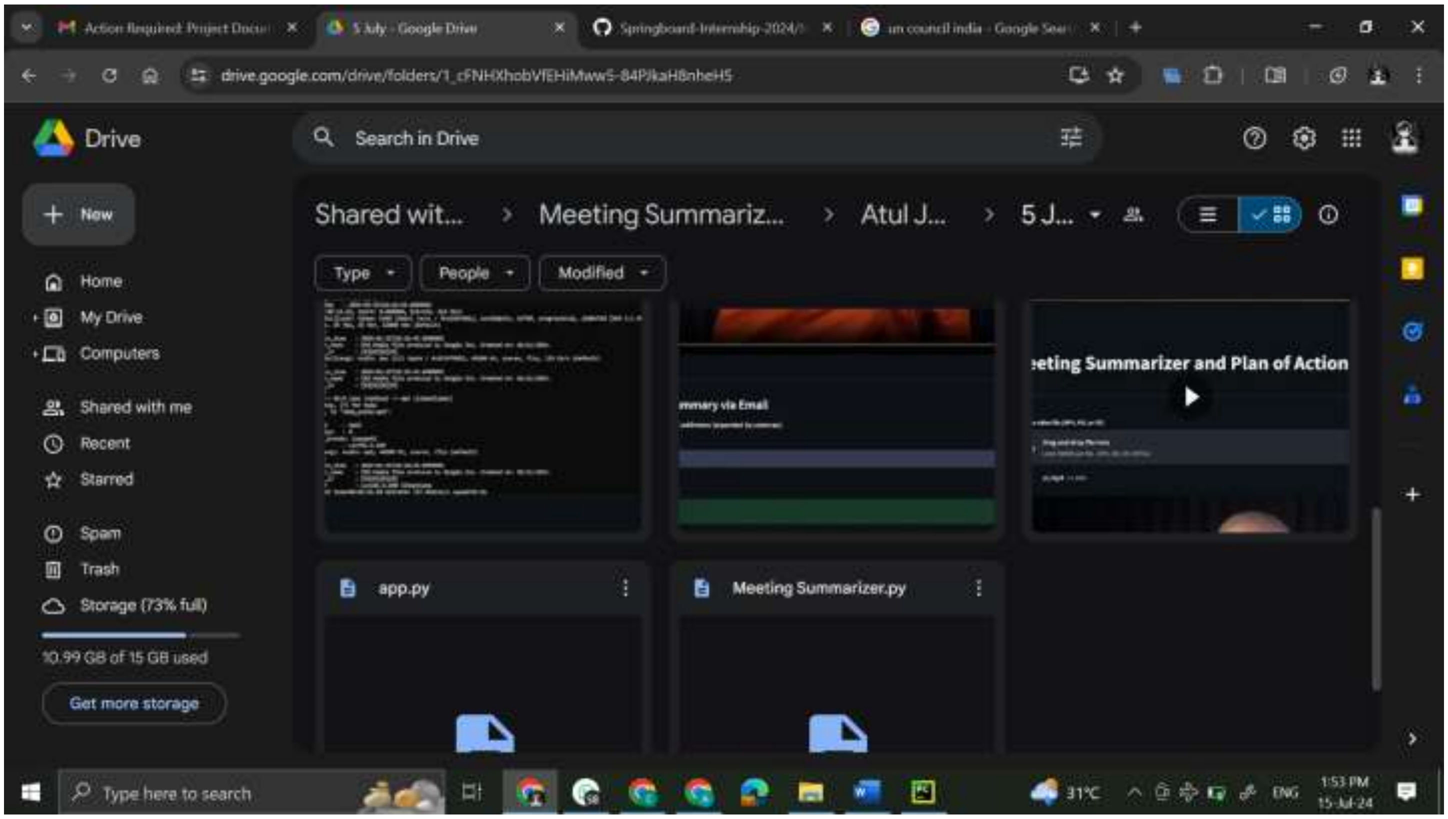
Week 14-21	Add files via upload	last month
final	Create mmm	2 weeks ago
final	Add files via upload	last week
.env	Add files via upload	last month
1.png	Add files via upload	last month
LICENSE	Initial commit	2 months ago
README.md	Update README.md	2 hours ago
S3.png	Add files via upload	last month
S3bucketconnection.py	Add files via upload	last month
Untitled.png	Add files via upload	last month
Video to Audio.py	Add files via upload	last month
Video to audio	Add files via upload	last month
	Add files via upload	last month



Shared with me > Meeting Summarizer a... > Atul Joshi

Folders

Transcribe	Video to audio	S3 bucket
Week 14-21	5 July	



Sharing Work with Mentors: Google Drive and GitHub

Google Drive:

- **Description:** Project updates and documentation, including architecture diagrams, design decisions, and progress reports, were regularly shared with mentors via Google Drive. This platform facilitated seamless collaboration and ensured mentors had access to the latest project artifacts.

GitHub:

- **Description:** The project's source code was hosted on GitHub, providing a central repository for collaborative development and version control. Regular updates, including code commits and feature implementations, were shared with mentors through this platform.