Audio transcription app using OpenAi Whisper

Project Title:

Audio transcription app using OpenAi Whisper

Team Name:

EchoChasers

Team Members:

- M.Akshaya(22B81A05CB)
- K.Archithа(22B81A05CE)
- A.Ashritha(22B81A05CF)
- B.Divya(22B81A05CL)
- B.Greeshma(22B81A05CR)

Phase-1: Brainstorming & Ideation

Objective:

To develop an audio transcription tool that converts MP4 audio files into WAV format and transcribes the audio to text using OpenAI's Whisper model. This tool will allow users to efficiently convert audio to text with minimal effort.

Key Points:

Problem Statement:

Users may have audio or video files (such as MP4 format) containing important speech data that needs to be transcribed into text. However, manual transcription is time-consuming, and automatic transcription tools require support for different audio formats. The need is for a tool that can convert MP4 files to a suitable audio format (WAV) and then transcribe that audio into text.

Proposed Solution:

○ Create a script that performs two key tasks:

Convert MP4 to WAV: Using the ffmpeg library to convert MP4 files (audio or video) to WAV format.

Transcribe Audio using Whisper: Utilize OpenAI's Whisper ASR model to transcribe the audio file into text.

○ This program will streamline the transcription process by handling both file format conversion and transcription, allowing users to simply provide an MP4 file and receive a text transcription.

Target Users:

- Content creators (e.g., podcasters, YouTubers) who need to transcribe interviews, podcasts, or speeches stored in MP4 format.
- Professionals (e.g., journalists, researchers) who need to transcribe meetings, lectures, or interviews recorded as MP4.
- Students who need transcripts of recorded lectures or class discussions.
- Businesses that require transcription for meetings, customer calls, and video content.
- Deaf and Hard of Hearing Individuals who require accurate transcriptions of video or audio content for better accessibility and understanding.

Expected Outcome:

- A reliable script that automates the transcription of MP4 files by first converting them into WAV and then using Whisper to generate accurate text.
- Users will save time compared to manual transcription, with accurate results provided by Whisper's robust language model.
- Enhanced accessibility to transcription services for people working with different audio or video formats.

Phase-2: Requirement Analysis

Objective:

The goal of this phase is to define the technical and functional requirements necessary to build an efficient and scalable audio transcription application. The app leverages OpenAI Whisper for speech recognition and FFmpeg for audio processing. It supports MP4-to-WAV conversion and transcribes speech into text. Additionally, we aim to identify key challenges, constraints, and potential enhancements to optimize performance, accuracy, and usability. This phase ensures a structured development approach, reducing risks and improving the overall application efficiency.

Key Points

1. **Technical Requirements**

- **Languages & Tools:**
  - **Primary Language:** Python (used for backend processing)
  - **Audio Processing:** FFmpeg (converting MP4 to WAV)
  - **Speech Recognition:** OpenAI Whisper
  - **System Utilities:** subprocess, os, warnings (for file handling and execution)
- **Frameworks & Libraries:**
  - **Whisper API:** Handles transcription tasks
  - **FFmpeg:** Converts audio formats
  - **Error Handling Modules:** Python's exception handling for managing errors
  - **Flask**: A lightweight web framework to build and serve the transcription application, allowing users to upload files and retrieve transcripts easily.
- **Storage & File Handling:**
  - Local storage is used for storing input MP4 and output WAV files
  - Efficient file management needed to avoid unnecessary disk usage
- **Hardware Considerations:**
  - The app currently runs on CPU with FP32 precision (due to lack of GPU support)
  - For improved performance, future versions may leverage GPU acceleration

2. **Functional Requirements**

- **Core Features:**
  - **Audio File Processing:** Convert MP4 files to WAV using FFmpeg
  - **Speech Transcription:** Process WAV files through OpenAI Whisper for text extraction
  - **Error Handling:**
    - Ensure the input file exists before processing
    - Handle errors in file conversion and transcription
  - **Console Output:** Display transcription results in the terminal

- **User Interaction:**
  - Users provide an MP4 file as input
  - The app converts it into a WAV file and then transcribes the speech
  - The transcription is displayed in the console
- **Additional Features (Future Enhancements):**
  - Improve file handling with automatic cleanup of temporary files
  - Web or GUI-based interface for improved user experience
  - Cloud storage integration for scalability

3. Constraints & Challenges

- **Performance Considerations:**
  - Running Whisper on CPU may be slow, especially for long files
  - Optimizations needed for handling large audio files efficiently
- **Audio Quality & Accuracy:**
  - Background noise, accents, and poor-quality recordings may reduce accuracy
  - Potential need for pre-processing audio (e.g., noise reduction)
- **Error Handling & System Reliability:**
  - Ensure FFmpeg and Whisper errors are handled gracefully
  - Improve logging and debugging mechanisms for troubleshooting

Phase-3: Project Design

Objective:

To design the system architecture, user flow, and UI/UX considerations for the transcription tool.

Key Points:

1. **System Architecture Diagram:**

User Uploads MP4 → Convert to WAV (ffmpeg) → Whisper Transcription Model → Return Text Output

2. **User Flow:**

User uploads MP4 file.

Program checks file existence and converts to WAV (if necessary).

Whisper transcribes the audio to text.

User receives the transcribed text (view, copy).

3. **UI/UX Considerations:**

   Simple Interface: "Upload MP4" button for easy user interaction.
   Error Handling: Display clear error messages (e.g., file not found, conversion failed).
   Result Display: Allow users to copy or download the transcribed text

Phase-4: Project Planning (Agile Methodologies)

Objective:

- Break down the tasks using Agile methodologies.
- To efficiently develop the audio transcription tool using Agile methodologies within a 2-day sprint.

Key Points:

Sprint Planning:

Sprint 1 (Day 1 & 2):

Objective: Complete the entire project (MP4 to WAV conversion, Whisper transcription, front-end CLI, and documentation).
Tasks:
Team Member 1 (Project Manager): Set up repository, manage tasks, create documentation.
Team Member 2 (Backend Developer): Implement MP4-to-WAV conversion and optimize backend.
Team Member 3 (Backend Developer): Integrate Whisper transcription model and test.
Team Member 4 (Frontend Developer): Develop the CLI interface for file upload and feedback.
Team Member 5 (QA/Testing): Test core functionality, handle error scenarios, and ensure end-to-end flow works.

Task Allocation:

Project Manager: Manage tasks, document processes.
Backend Developers: Focus on MP4-to-WAV conversion and Whisper integration.
Frontend Developer: Build the CLI interface for user interaction.
QA/Testing: Conduct final testing for all components.

Timeline & Milestones:

Sprint 1 (2 days):
Milestone 1: Complete MP4-to-WAV conversion, Whisper integration, and CLI interface.
Milestone 2: Testing and bug fixes completed.

Final Delivery: All tasks completed by the end of Day.
Phase-5: Project Development.

Objective:

To implement the audio transcription tool, integrating components and ensuring smooth functionality.

Key Points:

Technology Stack Used:

Programming Language: Python
Libraries/APIs: Whisper, ffmpeg, subprocess, os
Version Control: Git & GitHub

Development Process:

Step 1: Set up Git repository and install dependencies.
Step 2: Implement MP4-to-WAV conversion using ffmpeg.
Step 3: Integrate Whisper model for transcription.
Step 4: Perform unit tests, integration testing, and debug.

Challenges & Fixes:

Challenge: File conversion failure due to codec issues.
Fix: Add detailed error handling for specific codec errors.
Challenge: Slow transcription with large files.
Fix: Use the smaller Whisper model to improve speed.
Challenge: Noisy audio impacting accuracy.
Fix: Suggest noise reduction or provide clearer audio guidance.

Phase-6: Functional & Performance Testing

Objective:

To ensure the audio transcription app using OpenAI Whisper functions correctly, meets performance expectations, and provides accurate transcriptions. The testing phase will validate the app's reliability, efficiency, and usability by executing test cases, identifying bugs, and implementing necessary improvements.

Key Points:

1.Test Cases Executed:

- Audio File Handling:
- Verify MP4 to WAV conversion using FFmpeg
- Ensure handling of different audio formats (MP3, WAV)
- Functionality Testing:
- Check correct transcription output
- Validate output format (text structure, punctuation, accuracy)
- Ensure proper error handling for missing or corrupted audio files
- Usability Testing:
- Ensure easy-to-follow console outputs
- Verify user error messages are clear and actionable

2. Bug Fixes & Improvements:

- Fixed incorrect file handling when converting MP4 to WAV
- Improved error handling for missing FFmpeg dependencies
- Enhanced logging for better debugging and troubleshooting

3. Final Validation:

- Ensured that all technical and functional requirements are met
- Confirmed that performance is within acceptable limits for different file sizes and formats

4. Deployment:

- Final demo or GitHub repository for project access

Final Submission

1. **Project Report Based on the templates**
2. **Demo Video (3-5 Minutes)**
3. **GitHub/Code Repository Link**
4. **Presentation**