Hackathon Project Phases

Project Title:

Audio Transcription app with open AI Whisper

Team Name:

Risers

Team Members:

- K. Abhinay Reddy
- P. Sai Pavan
- S. Varun
- K. Vardhan

Phase-1: Brainstorming & Ideation

Objective:

- Existing transcription methods are slow, inaccurate, and costly. An AI-powered app using OpenAI Whisper enables real-time, multilingual, and highly accurate speech-to-text conversion, enhancing accessibility, productivity, and global communication.
- An audio transcription app using OpenAI Whisper enhances accessibility, automates speech-to-text, supports multiple languages, boosts productivity, enables real-time translations, improves content analysis, aids media creators, and facilitates seamless global communication.

Key Points:

- **Problem Statement:**
  Manual transcription is slow & error-prone, Existing solutions are expensive, Need for accurate, real-time transcription.

- **Proposed Solution:**
  Develop an AI-powered transcription app using OpenAI Whisper for real-time, multilingual, and highly accurate speech-to-text conversion, integrating language detection, translation, speaker diarization, and content analysis to enhance accessibility and productivity.

- **Target Users:**
  Journalists & Content Creators – Automate interview transcriptions, podcast captions, and video subtitles.
  Students & Educators – Convert lectures into text for easy note-taking and study materials.
  Businesses & Professionals – Transcribe meetings, calls, and conferences for documentation.
  Hearing-Impaired Individuals – Improve accessibility with real-time captions and subtitles.
  Multilingual Users & Translators – Enable speech-to-text conversion with language detection and translation.
  Legal & Medical Professionals – Automate case notes, medical records, and documentation.
  Customer Support & Call Centers – Analyze customer interactions for service improvements.

- **Expected Outcome:**
  Accurate, real-time, multilingual speech-to-text conversion enhances accessibility, productivity, and content analysis. The app automates transcription, translation, and speaker identification, improving efficiency, reducing costs, and enabling seamless global communication.

Phase-2: Requirement Analysis

Objective:

- Develop an AI-powered app using OpenAI Whisper for accurate, real-time, multilingual speech-to-text conversion, enhancing accessibility, automating transcription, enabling translation, improving productivity, and supporting diverse users across industries.

Key Points:

1. **Technical Requirements:**
   Languages: Python
   Dependencies: Streamlit, Numpy, Scipy, pyannote.audio, Whisper,Langdetect, sounddevice, keybert, transformers, deep_translator

2. **Functional Requirements:**
   Real-time & File-based Transcription
   Speaker Diarization

3. **Constraints & Challenges:**
   Performance & Resource Constraints
   Accuracy & Language Limitations
   Integration & Scalability Challenges

Phase-3: Project Design

Objective:

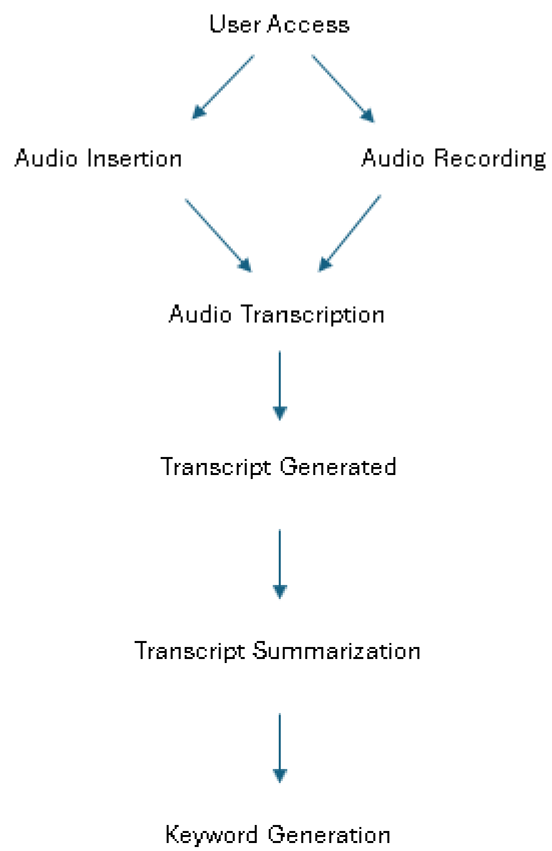
- Create the architecture and user flow for the audio transcription app using OpenAI Whisper

Key Points:

1. **System Architecture Diagram:**

```
|    User (Browser)   |
+---------------------+
          |
          |  Audio Upload
          |
+---------------------+
|   Streamlit Frontend  |
|  (streamlit_app)    |
+---------------------+
          |
          |  HTTP POST (audio)
          |
+---------------------+     +---------------------+
|   FastAPI Backend   |     |   OpenAI Whisper    |
|  (fastapi_backend)  |     |  (ASR Model)        |
+---------------------+     +---------------------+
          |                      |
          |  Receive Audio      |
          |-------------------->
          |                      |
          |  Process Audio       |
          |  (Whisper ASR)      |
          |-------------------->| Perform Transcription|
          |                     |<---------------------|
          |  Transcribed Text   |
          |                      |
+---------------------+     +---------------------+
          |
          |  HTTP Response (text)
          |
+---------------------+
|   Streamlit Frontend  |
+---------------------+
          |
          |  Display Text
          |
+---------------------+
|    User (Browser)   |
+---------------------+
```

2. **User Flow:**



3. **UI/UX Considerations:**
   Streamlit library is used to enhance the user interface which is user friendly and intuitive to even non-technical users.

Phase-4: Project Planning (Agile Methodologies)

Objective:

- Break down the tasks using Agile methodologies.

Key Points:

1. **Sprint Planning:**
   Each sprint will last 2 days. During sprint planning, we'll break the project into smaller, manageable tasks, assign story points, and prioritize based on urgency and importance.

2. **Task Allocation:**
   Pre-requisites: K. Vardhan
   Coding and Testing: K. Abhinay Reddy, P. Sai Pavan, S. Varun
   Deployment: K. Abhinay Reddy
   Solution Submission: K. Abhinay Reddy

3. **Timeline & Milestones:**
   Pre-requisites: Done by K.Vardhan from 10:00 AM to 1:10 PM
   Solution building: Done by K. Abhinay Reddy, P. Sai Pavan, S. Varun from 2:00 PM to 11:00AM
   Deployment: Done by K. Abhinay Reddy from 11:00 AM to 12:00 PM
   Documentation: Done by P. Sai Pavan from 12:00 PM to 12:30 PM
   Presentation: Done By S. Varun from 12:00 PM to 12:30 PM

Phase-5: Project Development

Objective:

- Code the project and integrate components.

Key Points:

1. **Technology Stack Used:**
   Programming Language: Python
   API: Fast API
   Libraries: Streamlit, sounddevice, deep_translator, pyannote.audio, pyannote.audio.speakerdiarization, Whisper

2. **Development Process:**
   Development Process for Audio Transcription Web Application

To develop and maintain the Audio Transcription Web Application, follow this step-by-step guide for building, testing, and enhancing the project.

1. Setup and Environment Configuration

Step 1.1: Clone the Repository
Clone the repository to your local system:
git clone https://github.com/Abhinay6227/Audio-Transcription-App.git
cd Audio-Transcription-App
Step 1.2: Set up Backend Environment (FastAPI)
Navigate to the backend folder and set up the virtual environment:
cd fastapi_backend
python -m venv venv
venv\Scripts\activate
Upgrade pip and install the required dependencies:
python -m pip install --upgrade pip
pip install -r requirements.txt
Step 1.3: Resolve NumPy/Numba Compatibility Issue
The Whisper library requires specific versions of NumPy and Numba. To resolve compatibility issues:
pip uninstall numpy -y
pip install numpy==2.1.6
pip uninstall whisper -y
pip install whisper
Step 1.4: Set up Frontend Environment (Streamlit)
Open a new terminal window and navigate to the frontend directory:
cd streamlit_app
python -m venv venv
venv\Scripts\activate
Install the required frontend dependencies:
python -m pip install --upgrade pip
pip install -r requirements.txt
2. Configure API Key for Whisper
Step 2.1: Obtain OpenAI API Key
Create an account on [OpenAI] and obtain an API key.
Step 2.2: Securely Store API Key
In the fastapi_backend directory, create a .env file:
OPENAI_API_KEY=YOUR_OPENAI_API_KEY
Add .env to .gitignore to avoid exposing your API key:
.env
Step 2.3: Access API Key in Code
In your backend code, access the API key as follows:

```
import os
openai_api_key = os.environ.get("OPENAI_API_KEY")
if not openai_api_key:
    raise ValueError("OPENAI_API_KEY environment variable not set.")
```

3. Configure Backend and Frontend URL
Step 3.1: Configure Backend URL in Frontend
Open streamlit_app/app.py or other relevant frontend files and set the correct backend URL:
BACKEND_URL = "http://127.0.0.1:8000/transcribe"
4. Run the Application Locally
Step 4.1: Start FastAPI Backend
Open a terminal, activate the backend virtual environment, and run the FastAPI app:
cd D:\...your path to Audio-Transcription-App\fastapi_backend
venv\Scripts\activate
uvicorn main:app --reload
Step 4.2: Start Streamlit Frontend
Open a new terminal, activate the frontend virtual environment, and run the Streamlit app:
cd D:\...your path to Audio-Transcription-App\streamlit_app
venv\Scripts\activate

streamlit run app.py

5. Transcribing Audio Files

Open the Streamlit app in your browser.
Upload an audio file (MP3 or WAV format).
Click the "Transcribe" button to send the file to the FastAPI backend.
View the transcribed text displayed in the Streamlit interface.

6. Troubleshooting

ImportError: If you encounter missing package errors, make sure the correct virtual environment is activated and install the missing packages:
pip install <missing_package>

Connection Refused: If the frontend cannot connect to the backend:
Ensure the backend server is running (uvicorn main:app --reload).
Ensure the BACKEND_URL is correctly configured in the frontend.

Numba/NumPy Error: If errors related to Numba and NumPy arise, downgrade NumPy as specified earlier.

7. Testing and Validation

Step 7.1: Backend Testing

Test the backend API endpoints with a tool like Postman or cURL:
Send a POST request to http://127.0.0.1:8000/transcribe with an audio file.
Validate that the response contains transcribed text.

Step 7.2: Frontend Testing

Interact with the Streamlit frontend:
Test the file upload functionality.
Ensure the transcribed text is displayed correctly.

Step 7.3: End-to-End Testing

Test the entire flow, from uploading an audio file in the frontend to receiving the transcription in the frontend.

8. Enhancements and Feature Development

Multilingual Support: Extend the application to support multiple languages using Whisper’s multilingual capabilities.
Real-Time Transcription: Implement real-time transcription for live audio streaming or microphone input.
UI Improvements: Enhance the frontend to display timestamps or speaker identification.
File Format Support: Add support for additional audio file formats like FLAC, OGG, etc.

9. Deployment (Optional)

Step 9.1: Deploy Backend to a Server

Use a cloud service like AWS, Azure, or Heroku to deploy the FastAPI backend.

Step 9.2: Deploy Frontend to a Hosting Service

Host the Streamlit app on a platform like Streamlit Sharing, Heroku, or other web hosting services.

10. Contributing to the Project

Bug Reports: Submit issues if you encounter bugs or unexpected behavior.
Feature Requests: Suggest new features for further improvements.
Pull Requests: Fork the repository, implement changes, and submit a pull request for review.

3. **Challenges & Fixes:**

   We faced an issue while adding the summarization feature to the project and we fixed it by dividing the generated transcript into small modules for the summarization feature to work and summarize the generated transcript.

Phase-6: Functional & Performance Testing

Objective:

- Ensure the project works as expected.

Key Points:

1. **Test Cases Executed:**
   An Audio file is taken and uploaded to the app and started the transcription process and a transcript is generated with summarization feature and keyword definition features working in good condition.

2. **Bug Fixes & Improvements:**
   No bugs were found while deploying the project and tested the application thoroughly to check the working status of the project.

3. **Final Validation:**
   The project meets the initial requirements of the taken problem statement and is running without any issues.
4. **Deployment (if applicable):**

   https://github.com/Abhinay6227/Audio-Transcription-App

Final Submission

1. **Project Report**
2. **Demo Video (3-5 Minutes)**
3. **https://github.com/Abhinay6227/Audio-Transcription-App**
4. **https://github.com/Abhinay6227/Audio-Transcription-App/tree/main/PPT%20file**