## **Audio to Text Converter with Summarization**

### **1. Overview**

This application converts audio files into transcribed text, summarizes the content, and allows users to download both the full transcription and summarized text as .docx or .txt files. It uses the **AssemblyAI** API for audio transcription and the **Hugging Face Transformers** library for summarization.

### **2. Objectives**

* **Transcribe audio to text**: Convert user-uploaded audio files into text using AssemblyAI’s Speech-to-Text API.
* **Summarize transcriptions**: Generate concise summaries of transcribed content.
* **Provide export options**: Allow users to download the transcription and summary as text or Word documents.
* **User Interface**: Provide a clean and intuitive UI using Streamlit for ease of interaction.

### **3. Key Features**

* **Audio Upload**: Users can upload audio files in wav or mp3 format.
* **Audio Playback**: Once uploaded, the app allows users to listen to the audio.
* **Transcription**: Converts uploaded audio into text via the AssemblyAI API.
* **Real-time Summary Generation**: Summarizes the transcription using Hugging Face's BART model.
* **Download Options**:
  + Download transcription as a .txt file.
  + Download summary as a .txt or .docx file.

### **4. System Architecture**

* **Frontend (UI)**:
  + **Platform**: Streamlit (Python web framework for creating interactive apps).
  + **UI Components**:
    - **File Uploader**: Allows users to upload audio files (wav, mp3).
    - **Audio Player**: Plays the uploaded audio.
    - **Transcription Area**: Displays the full transcribed text.
    - **Summary Area**: Displays the summarized text in bullet points.
    - **Download Buttons**: Provides buttons for downloading transcriptions and summaries in different formats (.txt, .docx).
* **Backend**:
  + **Audio Transcription**:
    - Uses **AssemblyAI API** to transcribe audio into text.
    - Audio file is uploaded via the API and the transcription request is made.
  + **Summarization**:
    - **Hugging Face’s BART model** is used to generate a summary of the transcribed text.
  + **File Creation**:
    - **python-docx** is used to create .docx files from the summary.
* **Integration**:
  + **API**: Use **AssemblyAI API** for transcription.
  + **Transformer Model**: Use Hugging Face’s pre-trained **BART** model for summarization.

### **5. Technology Stack**

* **Frontend**:
  + **Streamlit**: For building the user interface.
  + **Python**: The primary language for both backend and frontend logic.
* **Backend**:
  + **Requests**: For making HTTP requests to the AssemblyAI API.
  + **python-docx**: For creating .docx files for download.
  + **Hugging Face Transformers**: For text summarization using the pre-trained BART model.
* **External Services**:
  + **AssemblyAI API**: Provides Speech-to-Text functionality for transcribing audio.
  + **Hugging Face**: Summarizes the transcribed text using pre-trained models.

### **6. Workflow**

1. **Audio Upload**:
   * The user uploads an audio file (wav or mp3) via the file uploader in the Streamlit interface.
   * The audio is displayed as an audio player for listening.
2. **Audio Transcription**:
   * The user presses the **"Transcribe"** button.
   * The uploaded audio file is sent to the **AssemblyAI API** for transcription.
   * The API processes the audio, returning the transcribed text, which is displayed in a text area.
3. **Text Summarization**:
   * Once the transcription is complete, the text is sent to the **Hugging Face BART model** for summarization.
   * A concise, bullet-point style summary is generated from the transcription.
4. **Download Options**:
   * Users can download the summary in either .txt format or as a .docx file.
   * The .txt file contains the summary in a simple format.
   * The .docx file contains the summary in bullet-point format, using the **python-docx** library to generate the document.

### **7. Data Flow**

1. **User Input**:
   * Audio file uploaded by the user.
2. **Audio Processing**:
   * File sent to the **AssemblyAI API** for transcription.
   * The transcription is retrieved and displayed on the UI.
3. **Text Processing**:
   * The transcription is fed into the **Hugging Face BART model** for summarization.
   * The summary is returned and formatted in bullet points.
4. **Download Options**:
   * The summarized content is saved as .txt or .docx files.
   * The user is provided with buttons to download these files.

### **8. Error Handling and Edge Cases**

* **File Upload Errors**:
  + Handle unsupported file types (ensure only audio files are accepted).
  + Display a message if the file is too large or not valid.
* **API Errors**:
  + Handle cases where the **AssemblyAI API** fails (e.g., due to an invalid API key or network issues).
  + Handle errors when the **Hugging Face summarization model** fails to process the text.
* **Empty Transcription**:
  + If the transcription returns empty text, display a meaningful error message to the user.

### **9. Non-Functional Requirements**

* **Performance**:
  + Ensure that transcription is completed in a reasonable time (depending on file length, should not exceed 2 minutes for small files).
  + Summarization should take less than 10 seconds for an average transcription.
* **Scalability**:
  + The app should handle multiple simultaneous users.
  + Ensure efficient API calls to the AssemblyAI API and Hugging Face API.
* **Security**:
  + The AssemblyAI API key is stored in a .env file and never exposed in the frontend.
  + Ensure that all user data is stored securely (if applicable).
* **Usability**:
  + Simple, intuitive user interface that minimizes the number of steps to complete the transcription and summarization.
  + Ensure compatibility across different devices (desktop/mobile).

### **10. Conclusion**

This design document outlines the architecture and components of the **Audio to Text Converter with Summarization** application. By utilizing **AssemblyAI** for transcription and **Hugging Face’s BART model** for summarization, the app provides a seamless experience for users to convert audio into text and generate concise summaries. The app also includes export features for easy sharing of transcription and summaries.