Hackathon Project Phases Template

# Project Title:

Audio Transcription using OpenAI Whisper

# Team Name:

# Ctrl Alt Elite

# Team Members:

* Geethika Chatakonda
* Chakrika Bandi
* Ranabothu Varshitha Reddy
* Jarupla Gangothri



# Phase-1: Brainstorming & Ideation

## Objective:

* The website uses OpenAI Whisper to provide an accurate, efficient, and multilingual solution for converting spoken audio into written text. Leveraging Whisper’s advanced speech-to-text capabilities, it supports multiple languages and improves accessibility for users with hearing impairments. The website offers fast, high-quality transcriptions with easy editing options, ensuring data security and seamless integration with other platforms, making it ideal for use cases like podcast transcriptions, meeting notes, and content creation.

## 

## Key Points:

1. **Problem Statement:**

The Audio Transcription App using OpenAI Whisper is an AI-driven tool designed to transcribe speech from audio files into accurate text. It leverages OpenAI's Whisper model, known for its multilingual and robust speech recognition capabilities. This tool aims to enhance accessibility, productivity, and automation across various industries, including education, journalism, healthcare, and customer service.

1. **Proposed Solution:**

The proposed solution is a web application that utilizes OpenAI Whisper for accurate voice-to-text transcription. The backend will be built using Flask to handle audio processing, while Streamlit will provide a simple and user-friendly interface for users to upload or record audio. The application will process the audio using Whisper, generate transcriptions, and offer additional features like language detection, translation, and the ability to download or copy transcriptions.

This solution addresses the business challenge by providing a fast, efficient, and highly accurate transcription tool, reducing manual effort and improving productivity for professionals such as journalists, researchers, and content creators. Additionally, it enhances accessibility for individuals with hearing impairments, making speech content more inclusive and easy to use.

1. **Target Users:**
   * Content Creators
   * Businesses
   * Researchers and Academics
   * Journalists and Reporters
   * Students
   * Accessibility Advocates
   * Transcription Professionals
   * Language Learners
2. **Expected Outcome:**

The expected outcome is to provide fast transcriptions with easy editing, improved accessibility, and seamless integration, enhancing productivity for users across various industries.



# Phase-2: Requirement Analysis

## Objective: to clearly define the technical and functional needs of the audio transcription web application using OpenAI Whisper. It helps in identifying potential challenges and aligning the project with business goals and technical feasibility.

## Key Points:

* **Technical Requirements:**
  + - Basic programming language: Python
    - Backend: Flask
    - Frontend: Streamlit
    - Transcription Engine: OpenAI Whisper (Local or API-based)
* **Functional Requirements:**

**User Features**

* Upload audio files (MP3, WAV, FLAC, etc.).
* Download transcribed text (TXT).

**System Features**

* Background processing for long audio files.
* Error handling for unsupported/corrupt files.
* Display transcription status.

**Admin Features**

* Set file size and duration limits.
* **Constraints and Challenges:**
  + Accuracy in Challenging Scenarios
  + File Size and Processing Limitations

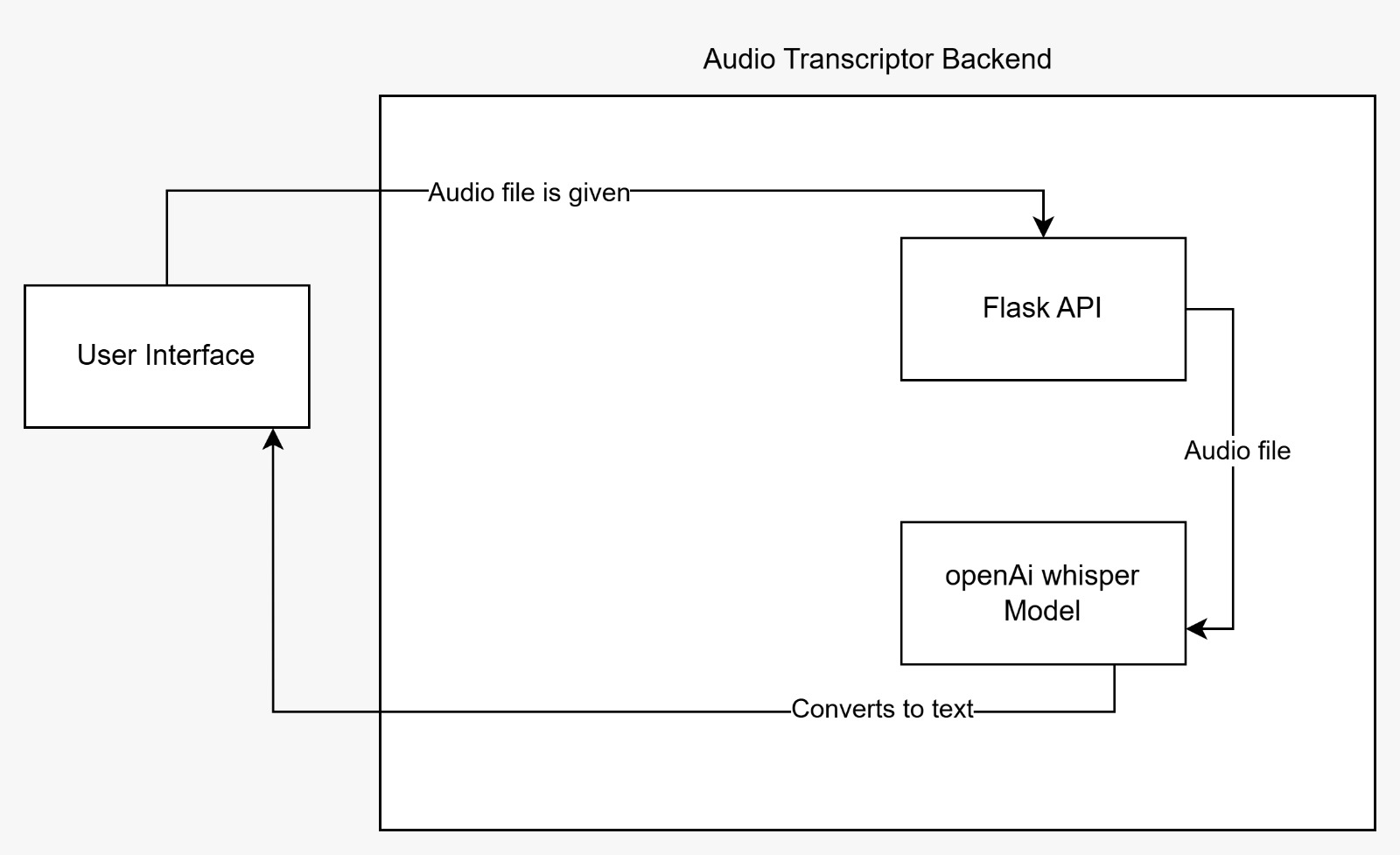
******Phase-3: Project Design**

## Objective: Develop the architecture and user flow of Audio Transcription through OpenAI Whisper

## Key Points:

1. **System Architecture Diagram:**

User uploads the audio file via UI The Audio file is connected to the whisper model through Flask API and is transcribed to text. The transcribed text is displayed to the user in a legible format



1. **User Flow:**

**** User uploads an audio file via the web UI.

 File is sent to Flask API, which processes it with OpenAI Whisper.

 Whisper transcribes the audio to text and sends it back.

 The transcribed text is displayed in a readable format.

 User can copy, edit, or download the text

1. **UI/UX Considerations:**

 Simple Layout: File upload, transcription display, transcribe button.

 Loading Indicator: Shows status during transcription.

 Readability & Accessibility: Clear fonts.

 Error Handling: Messages for unsupported files or failed uploads



# Phase-4: Project Planning (Agile Methodologies)

## Objective:

* + Break down the tasks using Agile methodologies.

## Key Points:

1. **Sprint Planning:**

Prerequisite

Backend Development

Frontend Development

Testing

Submit the solution

1. **Task Allocation:**

Prerequisite - J Gangothri, R Varshitha Reddy

Backend Development - C Geethika, Chakrika B

Frontend Development - C Geethika, Chakrika B

Testing - C Geethika, J Gangothri

Submit the solution - R Varshitha Reddy, Chakrika B

1. **Timeline & Milestones:**

Prerequisite – 1 hour

Backend Development – 6 hours

Frontend Development – 2 hours

Testing – 2 hours

Submit the solution – 1 hour

# Phase-5: Project Development.

## Objective:

* + Code the project and integrate components.

## Key Points:

1. **Technology Stack Used:** 
   * + Basic programming language: Python
     + Backend: Flask
     + Frontend: Streamlit
     + Transcription Engine: OpenAI Whisper (Local or API-based)
2. **Development Process:**

Phase 1: Initial Setup

* Set up the development environment (Python, Flask, Streamlit).
* Install Whisper and required dependencies.
* Create a GitHub repository for version control.

Phase 2: Backend Development

* Initialize Flask App
* Load Whisper Model
* Handle File Uploads
* Transcribe audio
* Return JSON response

Phase 3: Frontend Development (Streamlit UI)

* Initialize Streamlit app
* Upload audio file
* Send file to backend
* Receive transcription
* Display transcription

1. **Challenges & Fixes:**

**Challenge: Managing Different Audio Formats**

* Issue: Some audio formats (e.g., AAC, FLAC) weren’t supported directly.
* Fix:
  + Integrated FFmpeg to convert unsupported formats into Whisper-compatible formats (WAV/MP3).
  + Added format validation and conversion before transcription starts.

**Challenge: Importing OpenAI Whisper API (Paid Version Restriction)**

* Issue: The official OpenAI Whisper API requires a paid subscription, which was a limitation.
* Fix:
  + Instead of using the paid OpenAI Whisper API, we opted for the open-source Whisper model (openai/whisper).
  + Installed and used the local Whisper model via pip pip install openai-whisper.
  + Allowed users to choose between different Whisper model sizes (tiny, base, small, medium, large) based on available computing resources.

******Phase-6: Functional & Performance Testing**

## Objective:

## Functional Testing Objective: Ensure the audio transcription app correctly processes uploads, transcribes speech accurately using OpenAI Whisper, and handles errors effectively. Verify UI responsiveness and multi-language support. Test authentication, file validation, and transcription retrieval for a seamless user experience.

## Performance Testing Objective: Assess the app’s speed, scalability, and resource usage under varying loads to ensure smooth transcription processing. Test large file handling, concurrent user requests. Optimize response times, background processing, and system stability for high efficiency.

## Key Points:

1. **Test Cases Executed:**

 Audio File Upload: Verified that users can upload various audio formats (MP3,

WAV, M4A).

 Whisper Model Accuracy: Tested transcription quality on different audio samples (clear, noisy, multiple speakers).

 Performance & Speed: Measured transcription speed for small (1 min), medium (5 min), and large (30 min) files.

 User Interface: Checked UI responsiveness and ease of navigation in Streamlit/Flask.

 Error Handling: Ensured proper error messages for unsupported file formats or long audio durations.

1. **Bug Fixes & Improvements:**

 Improved UI Feedback: Displays status while transcribing.

 Enhanced Error Messages: More descriptive error handling for file format and size issues.

1. **Final Validation:**

Meets Initial Requirements: The system successfully transcribes audio into text, supports multiple formats, and provides a user-friendly interface. Whisper’s accuracy meets expectations, especially for clear speech.

# 

# Final Submission

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