## **Project Title:**

**EchoVerse App: AI-Powered Audiobook Creation**

## **Team Name:**

Team Prompt Smiths

## **Team Members:**

* Penumudi Anand Mahadev
* Kolli Ravi
* Devara Adithya Kumar

## **Phase-1: Brainstorming & Ideation**

### **Objective:**

The primary goal is to develop an AI-powered audiobook creation tool. The tool will use a Large Language Model (LLM) to rewrite text with different tones and a Text-to-Speech (TTS) service to generate audio.

### **Key Points:**

1. **Problem Statement:**
   * Many content creators and users want an easy way to transform text into expressive audio but lack the technical tools or time to do so. Existing solutions often lack customization for tone and voice.
2. **Proposed Solution:**
   * EchoVerse is an AI-powered application that uses a Hugging Face LLM (IBM Granite) to rewrite text and a TTS service to create audiobooks.
   * It allows users to input text via files, pasted text, or voice, and provides options for tones and voices.
3. **Target Users:**
   * The app targets content creators, students, and general users who want to convert articles, documents, or personal notes into audio for listening on the go.
4. **Expected Outcome:**
   * A functional **AI-powered Audio Book Generator tool** that provides Audio Books from user driven input.

## **Phase-2: Requirement Analysis**

### **Objective:**

Define the technical and functional requirements for the EchoVerse Web App.

### **Key Points:**

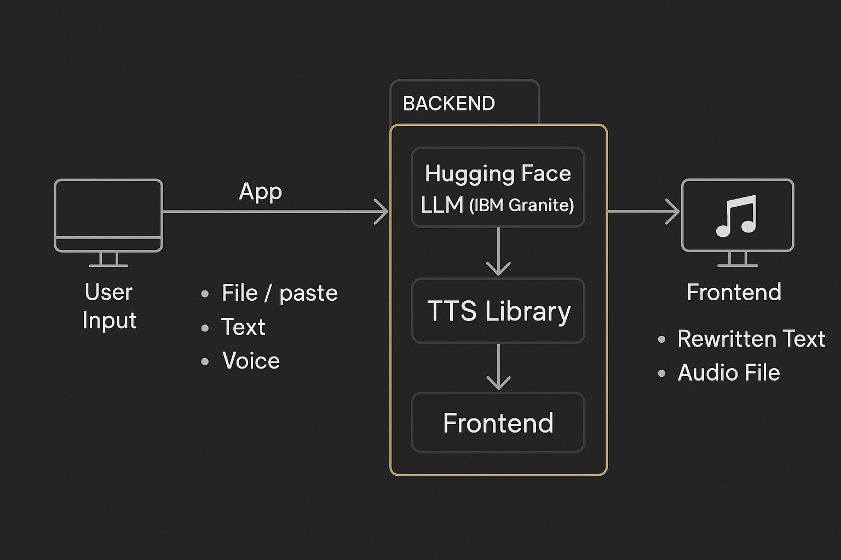
1. **Technical Requirements:**
   * Programming Language: **Python**
   * Backend: **Hugging Face LLM API (IBM Granite) & gTTS**
   * Frontend: **Streamlit Web Framework**
   * Database: **Not required initially (API-based queries)**
2. **Functional Requirements:**
   * Ability to fetch text from a file, text area, or microphone: This remains a core requirement, supported by the libraries streamlit for the UI, SpeechRecognition for transcribing audio from the microphone, and PyAudio to handle the microphone input.
   * Rewrite text based on a selected tone and LLM: The application must be able to take user-provided text and rewrite it to match a specific tone (e.g., Neutral, Suspenseful, Inspiring). This is powered by the transformers and torch libraries, which will be used with an LLM from huggingface-hub.
   * Convert rewritten text to audio: The application needs to use a Text-to-Speech (TTS) service to convert the final rewritten text into a downloadable audio file. This is handled by the gtts library.
   * Provide a customizable and intuitive UI: The user interface, built with streamlit, must allow users to input text, select their tone and voice, and view the rewritten text and audio player. The streamlit-mic-recorder library can be used to add a direct, easy-to-use microphone recording widget to the UI.
   * Optimize model performance: The application must efficiently handle the LLM, particularly with regards to memory usage and speed. The accelerate library helps in optimizing the model for the available hardware.
3. **Constraints & Challenges:**
   * Ensuring real-time updates from **Gemini API**.
   * Handling **API rate limits** and optimizing API calls.
   * Providing a **smooth UI experience** with Streamlit.

## 

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

### **Objective:**

Develop the architecture and user flow of the application.



### **Key Points:**

1. **System Architecture:**

* User enters text via UI (file, paste, or voice).
* The app processes the text using a **Hugging Face LLM (IBM Granite).**
* The **LLM rewrites the text** in a chosen tone.
* The **gTTS** library fetches and processes the data to **generate audio**.
* The frontend displays the rewritten text and a playable audio file.

1. **User Flow:**
   * **Step 1:** User enters text via file upload, text area, or voice input.
   * **Step 2:** The backend calls the Hugging Face API to rewrite the text.
   * **Step 3:** The app processes the text and calls the gTTS library to generate audio.
   * **Step 4:** The frontend displays the rewritten text and the playable audio file.
2. **UI/UX Considerations:**
   * **Minimalist, user-friendly interface** for seamless navigation.
   * **Tones and voices** are selected via filters in the sidebar.
   * A **clear and concise layout** for easy readability.

## **Phase-4: Project Planning (Agile Methodologies)**

### **Objective:**

Break down development tasks for efficient completion.

### 

### **Sprint Planning with Priorities****:**

* **Sprint 1 – Setup & Integration (Day 1):**
* **🔴** High Priority: Set up the environment and install dependencies.
* 🔴 High Priority: Integrate Hugging Face and TTS APIs.
* 🟡 Medium Priority: Build a basic UI with input fields.
* **Sprint 2 – Core Features & Debugging (Day 2):**
* 🔴 High Priority: Implement text rewriting and audio generation functionalities.
* 🔴 High Priority: Debug API issues & handle errors in queries.
* **Sprint 3 – Testing, Enhancements & Submission (Day 2):**
* 🟡 Medium Priority: Test API responses, refine UI, & fix UI bugs.
* 🟢 Low Priority: Final demo preparation & deployment.

## **Phase-5: Project Development**

### **Objective:**

### Implement core features of the EchoVerse App.

**Technology Stack Used:**

* **Frontend:** Streamlit
* **Backend:** Hugging Face LLM (IBM Granite) & gTTS
* **Programming Language:** Python

**Development Process:**

* Implement API key authentication and Hugging Face and TTS API integration.
* Develop text rewriting and audio generation logic.
* Optimize search queries for performance and relevance.

**Challenges & Fixes:**

* **Challenge:** Delayed API response times. **Fix:** Implement caching to store frequently queried results.
* **Challenge:** Limited API calls per minute. **Fix:** Optimize queries to fetch only necessary data.

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

### **Objective:** Ensure that the EchoVerse-App works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Category | Test Scenario | Expected Outcome | Status | Tester |
| TC-001 | Functional Testing | Testing the tone selection feature | If a neutral tone is selected, the text should be rewritten in a neutral tone. | ✅ Passed | K Ravi |
| TC-002 | Functional Testing | Querying the voice input feature | Transcribing a user's voice should provide accurate text. | ❌ Failed - UI broken | K Ravi |
| TC-003 | Performance Testing | LLM response time for text rewriting | LLM should return results quickly. | ⚠ Needs Optimization | Aditya Kumar |
| TC-004 | Bug Fixes & Improvements | Improved LLM and gTTS responses Intensity | Data accuracy should be improved. | ✅ Fixed | Anand Mahadev |
| TC-005 | Final Validation | Ensure UI is responsive across devices | UI should work on mobile & desktop. | ❌ Failed - UI broken on mobile | Aditya Kumar |