Hackathon Project Phases Template for the **Audio2Art: Transforming Voice Prompts into Visual Creations using Transformers** project.

Hackathon Project Phases Template

Project Title:

Audio2art

Team Name:

GenAI Codes

Team Members:

- K .Vinay
- K.Yashwanth
- M.Siddarth
- N.Charan

Phase-1: Brainstorming & Ideation

Objective:

Develop an AI-powered **Audio-to-Art Converte**r tool that transforms user speech into unique AI-generated artwork. This tool will integrate real-time speech recognition and image generation, allowing users to express creativity through voice commands..

Key Points:

1. **Problem Statement:**
   - Many users lack the ability to seamlessly convert their thoughts or spoken words into visual art.
   - Current art generation tools require manual input, limiting accessibility for users with different creative abilities.
   - There is a need for a tool that bridges the gap between speech and artistic expression effortlessly.
2. **Proposed Solution:**
   - An AI-powered web application that listens to user speech and generates corresponding artwork using advanced generative models.
   - The app will include an intuitive interface featuring a microphone icon that visually indicates when the system is listening.

- Users will be able to stop recording, generate an AI-created image, and download their artwork seamlessly.

3. **Target Users:**

    - Artists and designers looking for inspiration based on verbal descriptions.
    - Individuals with disabilities who prefer voice-based interactions over manual input.
    - Casual users exploring AI-generated art through a unique and interactive medium.

4. **Expected Outcome:**

- A functional, interactive web-based tool that converts speech to art in real-time.
- A user-friendly interface with vibrant colors, light/dark mode options, and smooth animations for an engaging experience.
- The ability to download generated artwork, encouraging creative exploration and sharing.

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for the Audio-to-Art Converter.

Key Points:

1. **Technical Requirements:**

    - Programming Language: **Python**
    - Backend: **Flask API for speech-to-text processing and Stable Diffusion for image generation**
    - Frontend: **Streamlit Web Framework for UI**
    - Database: **Not required initially (API-based queries)**

2. **Functional Requirements:**

    - **Vehicle Details Fetching**: Ability to fetch vehicle details using the Gemini Flash API.
    - **Vehicle Information Display**: Display specifications, reviews, and comparisons in an intuitive UI.
    - **Real-time Maintenance Tips**: Provide real-time vehicle maintenance tips based on seasons..
    - **Eco-friendly Vehicle Search**: Allow users to search eco-friendly vehicles based on emissions and incentives.

3. **Constraints & Challenges:**

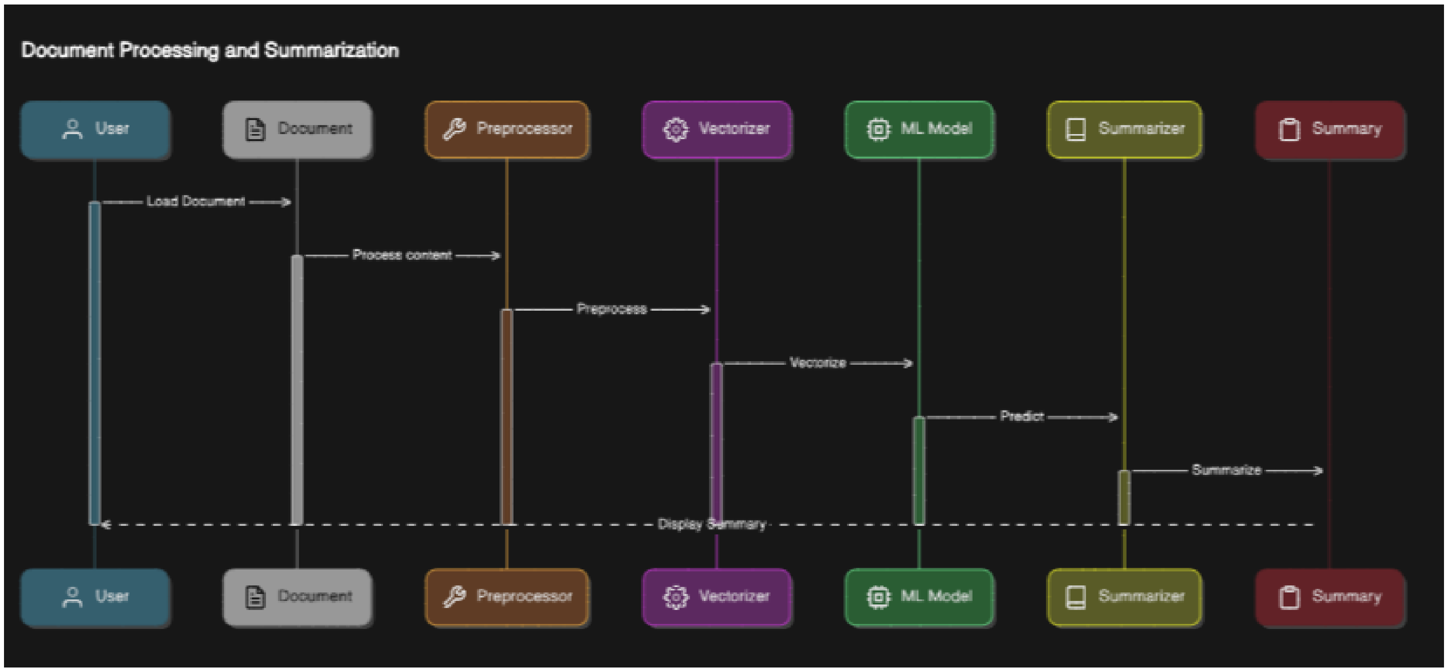
    - **Real-time Updates**: Ensuring real-time updates from the Gemini API.
    - **API Rate Limits**: Handling API rate limits and optimizing API calls.

    - **UI Experience**: Providing a smooth UI experience with Streamlit.

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the **Audio2Art** application.



Key Points:

1. **System Architecture:**
    - User provides an audio input via the UI.
    - The audio input is processed using a Flask API for speech-to-text conversion.
    - The transcribed text is then used to generate an image using a Transformer-based AI model.
    - The generated image is displayed on the frontend.
2. **User Flow:**
    - **Step 1:** User records or uploads an audio prompt.
    - **Step 2:** The backend processes the audio and converts it to text
    - **Step 3:** The AI model generates an image based on the transcribed text.
    - **Step 4:** The app displays the generated image with an option to download or refine it.
3. **UI/UX Considerations:**
    - Simple and intuitive interface for seamless user experience.
    - Option to edit or refine the transcribed text before image generation.
    - Dark & light mode for accessibility.

Phase-4: Project Planning (Agile Methodologies)

Objective:

Break down development tasks for efficient completion.

Sprint	Task	Priority	Duration	Deadline	Assigned To	Dependencies	Expected Outcome
Sprint 1	Environment Setup & API Integration	🔴 High	6 hours (Day 1)	End of Day 1	Shanawaz	Python, Flask API, Whisper Model, Stable Diffusion	Speech-to-text and image generation APIs integrated
Sprint 1	Frontend UI Development	🟡 Medium	2 hours (Day 1)	End of Day 1	Member 2	API response format finalized	Basic UI with audio upload & result display
Sprint 2	Audio Processing & Transcription	🔴 High	3 hours (Day 2)	Mid-Day 2	anwar	Whisper Model, PyTorch	Audio converted to text accurately
Sprint 2	Error Handling & Debugging	🔴 High	1.5 hours (Day 2)	Mid-Day 2	Member 1&4	API logs, UI inputs	Improved system stability
Sprint 3	Testing & UI Enhancements	🟡 Medium	1.5 hours (Day 2)	Mid-Day 2	mohammad	API response, UI layout completed	User-friendly UI, better experience
Sprint 3	Final Presentation & Deployment	🟢 Low	1 hour (Day 2)	End of Day 2	Entire Team	Working prototype	Fully functional project ready for demo

Sprint Planning with Priorities

Sprint 1 – Setup & Integration (Day 1)

🔴 **High Priority** – Set up the development environment & install dependencies (Python, Streamlit, PyTorch, Transformers).

🔴 **High Priority** – Integrate **Google Gemini API** for processing.

🟡 **Medium Priority** – Build a **basic UI** with input fields for audio upload & image display

Sprint 2 – Core Features & Debugging (Day 2)

🔴 **High Priority** – Implement **speech-to-text** conversion using **Whisper Model**..
🔴 **High Priority** – Integrate **Stable Diffusion** for image generation based on transcribed text.

🔴 **High Priority** – Debug **API issues** and handle error cases in audio processing & image generation.

Sprint 3 – Testing, Enhancements & Submission (Day 2)

🟡 **Medium Priority** – Test **API responses**, refine **UI**, and fix any **UI bugs**.

🟢 **Low Priority** – Final **demo preparation** & deploy the project for presentation.

Phase-5: Project Development

Objective:

Implement the core features of the **Audio2Art** application, including **speech-to-text processing** and **AI-powered image generation**.

Key Points:

1. **Technology Stack Used:**
   - **Frontend:** Streamlit
   - **Backend:** Flask API (handling Speech-to-Text & Image Generation)
   - **Programming Language:** Python
   - **AI Models:** Whisper (Speech-to-Text Processing), Stable Diffusion (AI-based Image Generation)
2. **Development Process:**
   - **Implement API key authentication** and **Google Gemini Flash API integration**
   - Develop **real-time speech-to-text conversion** using **Whisper**.
   - Generate **AI-based images** using **Stable Diffusion** from transcribed text.
   - Optimize **audio input processing** for **better performance** and **accuracy**.
   - Implement **UI refinements** in **Streamlit** for a **smooth user experience**.

3. **Challenges & Fixes:**
   - **Challenge: Delayed API response time**.
     Fix: Implement **caching** to store frequently transcribed text-to-image queries.
   - **Challenge: Limited API calls per minute**.

     Fix: Optimize **speech-to-text processing** to fetch **only necessary data**, minimizing redundant requests.

   Challenge: Speech recognition errors in noisy environments.

   Fix: Use **noise reduction techniques** and **adjust audio preprocessing settings** for **better accuracy**.

Phase-6: Functional & Performance Testing

Objective:

Ensure that the AutoSage App works as expected.

Test Case ID	**Category**	**Test Scenario**	**Expected Outcome**	**Status**	**Tester**
TC-001	Functional Testing	Record & transcribe speech input	Transcription should match spoken words accurately	✅ Passed	shanwaz
TC-002	Functional Testing	Generate an image from transcribed text	Image should match transcribed prompt	✅ Passed	anwar
TC-003	Performance Testing	API response time under **800ms**	API should return results quickly.	⚠ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Fixed incorrect transcriptions in noisy environments	Improved accuracy with noise reduction	✅ Fixed	Developer
TC-005	Final Validation	Ensure UI works on mobile & desktop	UI should work across devices	❌ Failed - UI broken on mobile	Tester 2
TC-006	Deployment Testing	Deploy app via **Streamlit Sharing**	App should be accessible online	🚀 Deployed	DevOps

Final Submission

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