Hackathon Project Phases Template

Project Title:

Gemini Landmark Description App Enhancing Tourist Experiences with AI

Team Name:

Tech Hustlers

Team Members:

- Kasichainula Anishwa
- Kallem Tejasree
- Gujarathi Khushi
- Keesara Nayani
- Pasham Yagna Valki

Phase-1: Brainstorming & Ideation

Objective:

Develop an AI-powered application that enhances tourist experiences by providing instant, detailed information about iconic landmarks through image recognition and AI-generated descriptions.

Key Points:

1. **Problem Statement:**
    - Tourists often lack immediate access to comprehensive information about landmarks they encounter during their travels.
    - Language barriers and accessibility issues can hinder the understanding and appreciation of cultural heritage sites.

2. **Proposed Solution:**
    - An AI-powered application that allows users to upload an image of a landmark and receive detailed, AI-generated descriptions, including historical significance, architectural features, and interesting facts.

- Incorporate multilingual support and accessibility features to ensure inclusivity for all travelers.

3. **Target Users:**
   - Tourists seeking immediate information about landmarks during their travels.
   - Tour guides looking for detailed and accessible information to share with their clients.
   - Individuals interested in learning about cultural heritage sites worldwide.

4. **Expected Outcome:**
   - A user-friendly application that enriches tourist experiences by providing instant access to valuable insights about iconic landmarks, fostering a deeper connection to cultural heritage sites worldwide.

Phase-2: Requirement Analysis

Objective:

Define the technical and functional requirements for the Landmark Description App.

Key Points:

1. **Technical Requirements:**
   - **Programming Language:** Python
   - **Backend:** Google Gemini AI API
   - **Frontend:** React Native for cross-platform mobile application development
   - **Database:** Cloud-based storage solution for user data and preferences

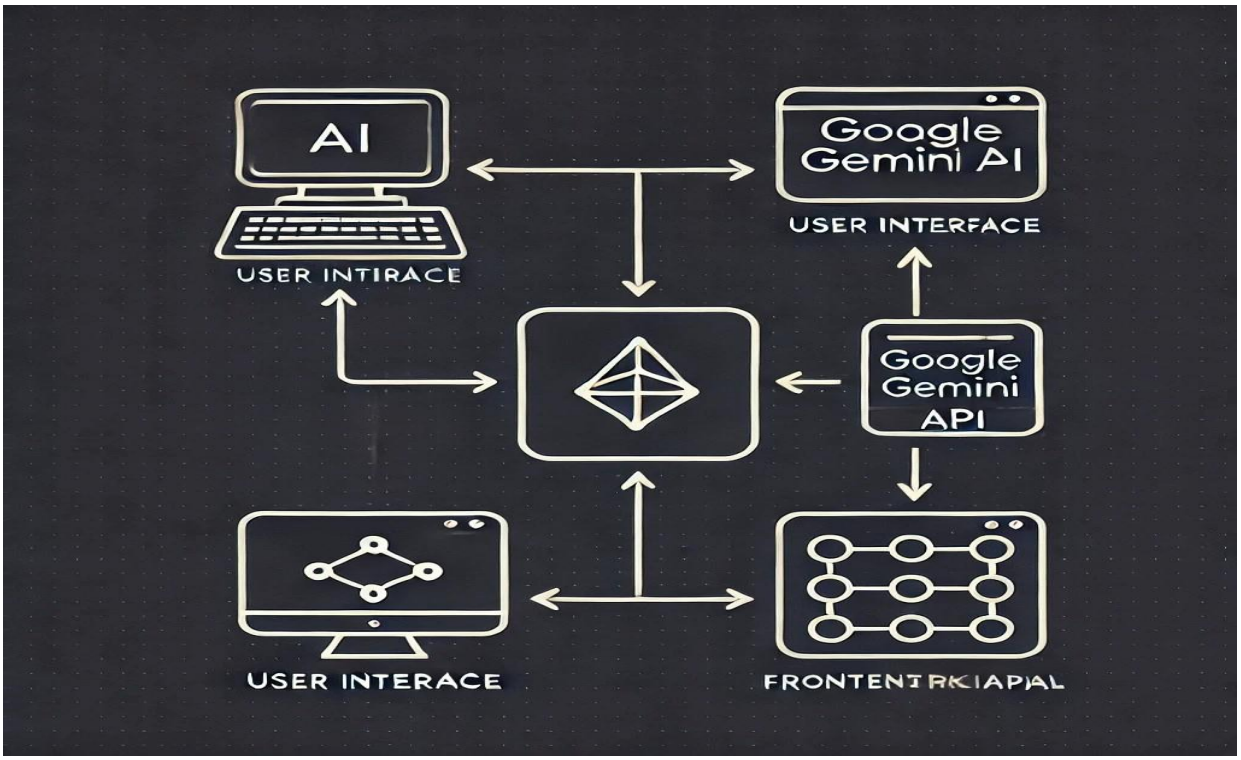
2. **Functional Requirements:**
   - **Image Upload:** Allow users to upload images of landmarks from their device's camera or gallery.
   - **Landmark Recognition:** Utilize AI models to identify landmarks from uploaded images.
   - **Description Generation:** Provide AI-generated descriptions covering historical significance, architectural features, and interesting facts.
   - **Multilingual Support:** Offer descriptions in multiple languages to cater to a diverse user base.
   - **Accessibility Features:** Ensure the app is operable via screen readers and includes features like adjustable text size and high-contrast modes.

3. **Constraints & Challenges:**
    - **Real-Time Processing:** Ensuring quick and accurate landmark recognition and description generation.
    - **API Rate Limits:** Managing API usage to stay within limits and avoid service interruptions.
    - **Data Privacy:** Protecting user data, especially images and location information.
    - **Offline Functionality:** Providing basic features when users are in areas with limited internet connectivity.

Phase-3: Project Design

Objective:

Develop the architecture and user flow of the application.



Key Points:

1. System Architecture:

- User Interaction: The user uploads an image of a landmark or enters a text-based query via the UI.
- Backend Processing:
    - The query is sent to the Google Gemini API for landmark recognition and data retrieval.
    - AI analyzes the image and generates a description, including historical facts, architectural insights, and cultural significance.
- Response Handling: The processed results are sent back to the frontend.
- Display: The app presents the description in an intuitive and visually appealing format.

2. User Flow:

- Step 1: The user captures or uploads an image of a landmark.
- Step 2: The image is analyzed by the AI model via the Google Gemini API to identify the landmark.
- Step 3: The system fetches relevant information (history, architecture, fun facts) from a knowledge base.
- Step 4: The app displays a detailed, AI-generated description in an interactive format.
- Step 5: Users can access extra features such as:
    - Multilingual descriptions for global accessibility.
    - Audio narrations for an immersive experience.
    - Offline mode to store previous searches for future reference.

3. UI/UX Considerations:

- Minimalist, user-friendly design for easy navigation.
- Image and text-based search for flexible user input.
- Multilingual support to cater to a global audience.
- Dark & light mode to improve readability.
- Voice assistant integration to support visually impaired users.
- Interactive map integration to show landmark locations and nearby attractions.

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	K.Teja Sree,K.Nayani	Hugging face API Tokens, Python, Streamlit setup	API connection established & working
Sprint 1	Frontend UI Development	🟡 Medium	2 hours (Day 1)	End of Day 1	G.Khushi	API response format finalized	Basic UI with image & prompt input
Sprint 2	Image Processing & Landmark Recognition	🔴 High	3 hours (Day 2)	Mid-Day 2	K.Anishwa, G.Khushi	API response, UI elements ready	Image-to-description functionality
Sprint 2	Error Handling & Debugging	🔴 High	1.5 hours (Day 2)	Mid-Day 2	K.Anishwa, P.Yagna	API logs, UI inputs	Improved API stability
Sprint 3	Multilingual Support & Accessibility	🟡 Medium	1.5 hours (Day 2)	Mid-Day 2	K.Teja Sree	Language model integration	Translations & accessibility features
Sprint 3	Testing & UI Enhancements	🟡 Medium	2 hour (Day 2)	Mid-Day 2	K.Nayani	API response, UI layout completed	Responsive UI, better user experience
Sprint 3	Final Presentation & Deployment	🟢 Low	1 hour (Day 2)	End of Day 2	Entire Team	Working prototype	Demo-ready project

Sprint Planning with Priorities

Sprint 1 – Setup & Integration (Day 1)

- (🔴 **High Priority)** Set up the **environment** & install dependencies.
- (🔴 **High Priority)** Integrate **Google Gemini API**.
- (🟡 **Medium Priority)** Build a **basic UI with input fields**.

Sprint 2 – Core Features & Debugging (Day 2)

- (🔴 **High Priority)** Implement **search & comparison 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

Key Points:

1. **Technology Stack Used:**
    - **Frontend:** Streamlit (for a lightweight and interactive UI).
    - **Backend:** Google Gemini API (for AI-powered landmark descriptions).
    - **Programming Language:** Python.

2. **Development Process:**
    - Implement **image and text input** functionality for landmark recognition.
    - Integrate **Google Gemini API** for generating descriptions based on user queries.
    - Develop a **multilingual feature** to provide descriptions in different languages.
    - Enhance **UI/UX** to ensure accessibility for all travelers.

3. **Challenges & Fixes:**
    - **Challenge:** Slow image processing time.
        - **Fix:** Implement image compression and preprocessing to speed up API requests.
    - **Challenge:** Limited API calls per minute.
        - **Fix:** Optimize queries and cache results for frequently searched landmarks.
    - **Challenge:** Handling low-quality images.
        - **Fix:** Use image enhancement techniques to improve recognition accuracy.

Phase-6: Functional & Performance Testing

Objective:

Ensure that the Landmark Description App works as expected.

Test Case ID	Category	Test Scenario	Expected Outcome	Status	Tester
TC-001	Functional Testing	Upload an image of the Eiffel Tower with a prompt.	AI should return an accurate description of the landmark..	✅ Passed	Tester 1
-+TC-002	Functional Testing	Query "Famous landmarks in Italy" without an image.	AI should return a list of famous nearby places	✅ Passed	Tester 2
	Performance Testing	API response time under 500ms	AI should process and return results quickly.	⚠ Needs Optimization	Tester 3
TC-004	Bug Fixes & Improvements	Fixed incorrect landmark descriptions.	Data accuracy should be improved.	✅ Fixed	Developer
X+	Final Validation	Ensure UI is responsive on desktop and mobile.	UI should work across all devices.	❌ Failed - UI broken on mobile	Tester 2
TC-006	Deployment Testing	Host the app using Hugging Face API.	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**