**Hackathon Project Phases Template** for the **Gemini Landmark Explorer App** project.

# **Hackathon Project Phases Template**

## **Project Title:**

**Gemini Landmark Description App Enhancing Tourist Experiences with AI**

## **Team Name:**

Explorers United

## **Team Members:**

* Varshini Santhosham
* Rakshitha Sai Sri Tamada
* Sigilipalli Haritha
* Arigela Haripriya

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

### **Objective:**

Develop an AI-powered Gemini Landmark Description App Enhancing Tourist Experiences.

### **Key Points:**

1. **Problem Statement:**
   * Tourists often struggle to obtain detailed, real-time, and contextually relevant information about landmarks, leading to a less engaging and informative travel experience.
   * Users also need guidance on nearby attractions, historical insights, and travel tips for a richer experience.
2. **Proposed Solution:**
   * An AI-powered application using Gemini Flash to provide real-time landmark descriptions, historical insights, and travel tips.
   * The app offers multilingual support, audio guides, and personalized recommendations for an enhanced tourist experience.
3. **Target Users:**

○ Tourists seeking real-time landmark descriptions and historical insights.  
 ○ Travelers looking for personalized recommendations and nearby attractions.  
 ○ Accessibility-focused users needing multilingual support and audio guides.

1. **Expected Outcome:**
   * A functional AI-powered landmark guide app that provides real-time descriptions, historical insights, and personalized travel recommendations.

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

### **Objective:**

Define the technical and functional requirements for the Gemini Landmark Explorer App.

### **Key Points:**

1. **Technical Requirements:**
   * Programming Language: **Python**
   * Backend: **Google Gemini Flash API**
   * Frontend: **Streamlit Web Framework**
   * Database: **Not required initially (API-based queries)**
2. **Functional Requirements:**

○ Ability to identify landmarks using Gemini Flash API and Google Vision API.  
○ Display historical facts, cultural significance, and trivia in an intuitive UI.  
○ Provide real-time travel tips based on location, weather, and season.  
○ Allow users to explore nearby landmarks and attractions using geolocation.

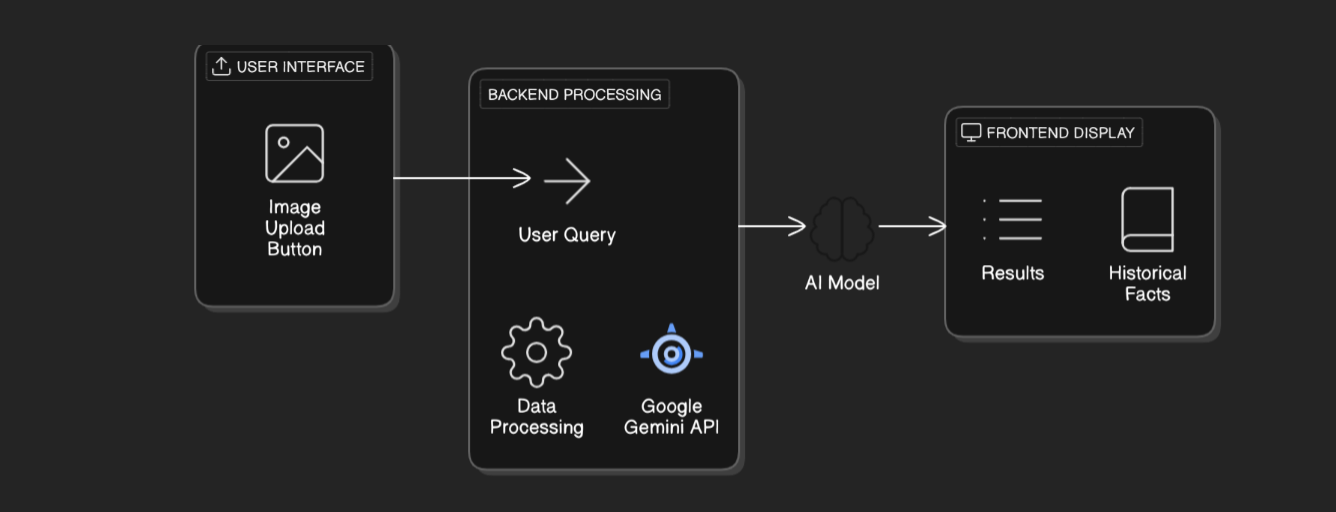
1. **Constraints & Challenges:**

○ Ensuring real-time landmark recognition and descriptions from Gemini Flash API.  
○ Handling API rate limits and optimizing API calls for high user traffic.  
○ Providing a smooth and responsive UI experience with Streamlit for seamless user interaction.

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

### **Objective:**

Develop the architecture and user flow of the application.



### **Key Points:**

1. **System Architecture:**
   * **User Input:** The user enters a landmark-related query via the UI (text, image, or voice).
   * **Backend Processing:** The query is processed using the **Google Gemini API** for landmark recognition.
   * **AI Model Processing:** The AI model analyzes the landmark and fetches historical, cultural, and tourist-related details.
   * **Frontend Display:** The system presents the results, including **landmark descriptions, historical facts, and travel recommendations** in a user-friendly interface.
2. **User Flow:**
   * **Step 1:** The user enters a query (e.g., uploads an image or asks, "Tell me about the Eiffel Tower").
   * **Step 2:** The backend calls the **Google Gemini API** to recognize the landmark and retrieve relevant historical and cultural details.
   * **Step 3:** The app processes the data and presents the results in an easy-to-read format, including **landmark descriptions, historical facts, and travel recommendations**.
3. **UI/UX Considerations:**
   * **Minimalist, user-friendly interface** for seamless navigation and easy access to landmark details.
   * **Search and filter options** for landmarks based on location, historical significance, or user preferences.
   * **Dark & light mode** support for an enhanced user experience in different lighting conditions.

## **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 | Member4 | Google API Key, Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡 Medium | 2 hours (Day 1) | End of Day 1 | Member 1 | API response format finalized | Basic UI with input fields |
| Sprint 2 | Landmark Search & Comparison | 🔴 High | 3 hours (Day 2) | Mid-Day 2 | Member 3 | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours (Day 2) | Mid-Day 2 | Member 1&2 | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | 🟡 Medium | 1.5 hours (Day 2) | Mid-Day 2 | Member 4&2 | 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**

### **Objective:**

Implement core features of the Gemini Landmark Explorer App.

### **Key Points:**

1. **Technology Stack Used:**
   * **Frontend:** Streamlit / React
   * **Backend:** Google Gemini Flash API
   * **Programming Language:** Python
   * **AI Model:** CNN/ Transformer-based model
   * **Database:** Firebase / postgreSQL
2. **Development Process:**
   * **Implement API key authentication** and integrate the **Google Gemini API** for landmark recognition and description generation.
   * **Develop landmark comparison and travel recommendation logic** to enhance the user experience.
   * **Optimize search queries** for faster performance and more relevant landmark information.
3. **Challenges & Fixes:**
   * **Challenge:** Delayed API response times.  
      **Fix:** Implement caching to store frequently queried landmark details and reduce API calls.
   * **Challenge:** Limited API calls per minute.  
      **Fix:** Optimize queries to fetch only the necessary data and use request batching where possible.

## **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 | Query "Tell me about the Eiffel Tower" | A detailed description of the landmark should be displayed. | ✅ Passed | Tester 1 |
| TC-002 | Functional Testing | Upload an image of a historical site | The app should correctly identify the landmark and provide relevant details. | ✅ Passed | Tester 2 |
| TC-003 | Performance Testing | API response time under 500ms | API should return results quickly. | ⚠ Needs Optimization | Tester 3 |
| TC-004 | Bug Fixes & Improvements | Fixed incorrect landmark identification | Data accuracy should be improved. | ✅ Fixed | Developer |
| TC-005 | Final Validation | Ensure UI is responsive across devices. | UI should work on mobile & desktop. | ❌ Failed - UI broken on mobile | Tester 2 |
| TC-006 | Deployment Testing | Host the app using 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**