**Hackathon Project Phases Template** for the **Landmark Description App** project.

# **Hackathon Project Phases Template**

## **Project Title:**

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

## **Team Name:**

**EchoQuest**

## **Team Members:**

* Varshini Mane
* MVL.Sowjanya
* Keerthana Malleshwaram
* Maligireddy Anitha
* Lakuma Rajini

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

### **Objective:**

### To develop an AI-powered landmark description app that enhances tourist experiences by providing intelligent, real-time, and multilingual insights about landmarks using Gemini 1.5 Pro.

### **Key Points:**

1. **Problem Statement:**

Tourist often struggle with:

* + Lack of accurate and detailed historical or cultural context about landmarks
  + Language barriers preventing them from understanding local guides or signage.
  + Inconsistent and unreliable information from online sources.

1. **Proposed Solution:**

A Flask-based web application deployed on Vercel that integrates Gemini 1.5 Pro to provide:

* + Real-time landmark descriptions image recognition.
  + Multilingual support, enabling tourists to access information in their preferred language.

1. **Target Users:**
   * Domestic and international tourists.
   * Travel bloggers and content creators.
   * Local tourism boards and guides**.**
2. **Expected Outcome:**
   * + Enhanced tourist experiences with AI-driven, reliable landmark insights.
     + Support for accessibility by offering alternative content formats..

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

### **Objective:**

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

### **Key Points:**

1. **Technical Requirements:**
   * Programming Language: **Python(Flask)**
   * Backend: **Gemini-1.5-pro**
   * Frontend: **Vercel Web Framework**
   * Database: **Not required initially (API-based queries)**
2. **Functional Requirements:**

* + - AI-Generated Landmark Descriptions: Provide detailed, AI-generated historical and cultural insights.
    - Image-Based Recognition: Users can upload a photo to identify and learn about a landmark.
    - Multilingual Support: AI descriptions available in multiple languages.
    - User Preferences : preferred languages.

1. **Constraints & Challenges:** Ensuring quick response time for AI-generated descriptions.

Ensuring high-quality translations across multiple languages.

Making the app user-friendly .

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

### **Objective:**

Develop the architecture and user flow of the application.



### **Key Points:**

1. **System Architecture:**
   * User uploads a image .
   * Image is processed using **Google Gemini API**.
   * Data Processing Module to refine and structure the response.
   * The AI-generated description is then displayed on the Front-end.
2. **User Flow:**
   * Step 1: User Uploads a image.
   * Step 2: The backend **calls the Gemini Flash API** to process the image
   * Step 3: The app processes the image and **displays results** in an easy-to-read format.
3. **UI/UX Considerations:**
   * **Minimalist, user-friendly interface** for seamless navigation.
   * **Multilingual Support**: Easy language switch for travelers.
   * **Fast & Responsive**: Quick processing to avoid user frustration.

## 

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

### **Key Points:**

1. **Technology Stack Used:**
   * **Frontend:** Vercel
   * **Backend:** Gemini-1.5 pro
   * **Programming Language:** Python(Flask)
2. **Development Process:**
   * Implement **API key authentication** and **Gemini 1.5 pro** .
   * Develop **simple UI for user input** .
   * Optimize **UI for multilingual support for better performance.**
3. **Challenges & Fixes:**
   * **Challenge:** Delayed API response times.  
      **Fix:** Implement request catching and optimize API calls.
   * **Challenge:** Landmark recognition accuracy.  
      **Fix:** Refine Vision API parameters.

## **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 | Uploaded image “Charminar.jpeg” | Relavent information about the landmark. | ✅ Passed | Rajini |
| TC-002 | Functional Testing | Uploaded image “Images.jpeg” | Relavent information about the landmark. | ✅ Passed | Keerthana |
| 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 API responses. | 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 Vercel 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**