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

Hackathon Project Phases Template

# Project Title:

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

# Team Name:

Infinity Squads

# Team Members:

* Avusula Shirisha
* Amrutha Potta
* Sai Sankeerthana Choudekari
* K Hari Priya

# Phase-1: Brainstorming & Ideation

## Objective:

To enhance tourist experiences by providing AI-generated descriptions of landmarks based on uploaded images.

## Key Points:

1. **Problem Statement:**

Travelers often lack instant access to detailed historical, architectural, and cultural insights about landmarks.

1. **Proposed Solution:**

The Gemini Landmark Description App allows users to upload landmark images and receive AI-generated descriptions, ensuring accessibility and enriched learning. It supports multilingual features for inclusivity.

3. **Target Users:**

* + Tourists exploring new places
  + Travel guides providing historical insights
  + History and architecture enthusiasts
  + Students and researchers studying landmarks.

1. **Expected Outcome:**

Instant AI-generated landmark descriptions enhance tourist experiences by providing historical, architectural, and cultural insights. The app ensures accessibility with multilingual support, benefiting travelers, guides, and researchers.

# Phase-2: Requirement Analysis

## Objective:

To identify and define the functional and non-functional requirements for developing the Gemini Landmark Description App, ensuring seamless user experience, accuracy, and accessibility.

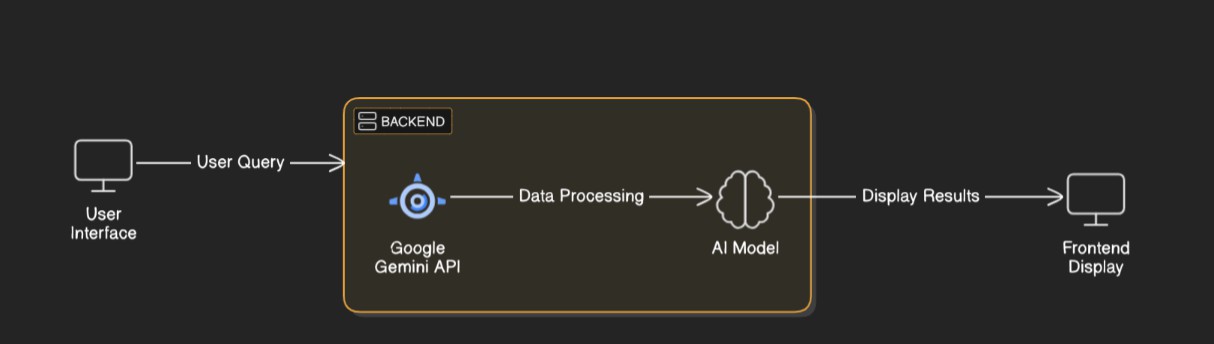
## Key Points:

1. **Technical Requirements:**
   * Programming Language: **Python**
   * Frontend: **React.js (for UI)**
   * Backend**: Node.js (Express.js)**
   * AI Models**: Google Vision API (Landmark Detection) & Gemini AI (Text Generation)**
   * Databas**e: Firebase (for storing user preferences)**
   * Authentication**: Firebase Auth (Google Sign-In)**
   * Cloud Storage**: Google Cloud Storage (for images)**
2. **Functional Requirements:**
   * Users can upload images of landmarks for AI-generated descriptions.
   * AI provides historical, architectural, and cultural insights based on the image.
   * Multilingual support for accessibility and inclusivity.
   * User-friendly interface for easy navigation and interaction.
   * Option to save and share landmark descriptions.
   * Integration with maps for location-based information..
3. **Constraints & Challenges:**
   * Ensuring high accuracy in AI-generated landmark descriptions.
   * Handling diverse image qualities and varying lighting conditions.
   * Supporting multiple languages while maintaining translation accuracy.

# Phase-3: Project Design

## Objective:

Develop the architecture and user flow of the application.



## Key Points:

1. **System Architecture:**

✅ **User uploads a landmark image & enters a brief prompt via UI.**  
✅ **Query is processed using Hugging Face API for image recognition.**  
✅ **AI model generates a detailed description covering history, architecture, and cultural significance.**  
✅ **Frontend displays the landmark details with multilingual support and public reviews.**

1. **User Flow:**

1️⃣ **Step 1:** User uploads an image & enters a query (e.g., “Describe the Taj Mahal”).  
2️⃣ **Step 2:** The backend calls Hugging Face’s Vision Transformer (ViT) model for landmark identification.  
3️⃣ **Step 3:** The landmark name is passed to GPT-Neo (text-generation model) to generate a **500-word** description.  
4️⃣ **Step 4:** The app processes the data and displays results with historical significance, architecture, and cultural details.  
5️⃣ **Step 5:** User can **read reviews, switch languages, and share descriptions** publicly.

1. **UI/UX Considerations:**

**Minimalist, user-friendly interface** for smooth navigation.  
 **Filters for historical importance, architectural style, and region**

# 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 | Shirisha | Google API Key, Python, Streamlit setup | API connection established & working |
| Sprint 1 | Frontend UI Development | 🟡  Medium | 2 hours  (Day 1) | End of Day 1 | Amrutha | API response format finalized | Basic UI with input fields |
| Sprint 2 | Vehicle Search & Comparison | 🔴 High | 3 hours  (Day 2) | Mid-Day 2 | Haripriya | API response, UI elements ready | Search functionality with filters |
| Sprint 2 | Error Handling & Debugging | 🔴 High | 1.5 hours  (Day 2) | Mid-Day 2 | Sai Sankeerthana | API logs, UI inputs | Improved API stability |
| Sprint 3 | Testing & UI Enhancements | 🟡  Medium | 1.5 hours  (Day 2) | Mid-Day 2 | Amrutha &Shrisha | 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 Description App.

## Key Points:

1. **Technology Stack Used:**
   * **Frontend:** React.js(for ui)
   * **Backend:** Google Gemini Flash API
   * **Programming Language:** Python
2. **Development Process:**
   * Implement **API key authentication** and **Gemini API integration**.
   * Develop **vehicle comparison and maintenance tips logic**.
   * Optimize **search queries for performance and relevance**.
3. **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 AutoSage App works as expected.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Category** | **Test Scenario** | **Expected Outcome** | **Status** | **Tester** |
| TC-001 | Functional Testing | Query "Best budget cars under ₹10 lakh" | Relevant budget cars should be displayed. | ✅ Passed | Tester 1 |
| TC-002 | Functional Testing | Query "Motorcycle maintenance tips for winter" | Seasonal tips should be provided. | ✅ 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 API responses. | Data accuracy should be improved. | ✅ Fixed | Develop er |
| 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**