**Project Title:**

The Gemini Landmark Description App

**Team Name:**

Pixella

**Team Members:**

● 22wh1a1293- Harshitha Yarra

● 22wh1a12c7-Vishruthi Kongarapu

● 22wh1a1267-Srihita Karempudi

**Phase-1: Brainstorming & Ideation**

**Objective:**

Develop an AI-powered tool using Gemini Flash to provide detailed descriptions of iconic landmarks for tourists and enthusiasts.

**Key Points:**

1. **Problem Statement:**

○Tourists often lack access to comprehensive, reliable information about landmarks during visits.

○ Limited multilingual or accessibility features exclude many users.

2. **Proposed Solution:**

○ An AI-powered app that uses image uploads and prompts to generate detailed descriptions of landmarks, including historical, architectural, and cultural insights.

○ Incorporate multilingual support and accessibility options for inclusivity.

3. **Target Users:**

○ Tourists exploring new cities.

○ Tour guides seeking enriched narratives.

○ History enthusiasts and curious travellers.

4. **Expected Outcome:**

○ A functional AI-driven app delivering real-time, personalized landmark descriptions.

**Phase-2: Requirement Analysis**

**Objective:**

Define the technical and functional requirements for the Gemini Landmark Description 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 upload landmark images and process prompts.

○ Fetch landmark details using the Gemini Flash API.

○ Display AI-generated descriptions in a user-friendly UI.

Multilingual and accessibility features for inclusivity.

3. **Constraints & Challenges:**

○ Ensuring accurate image recognition via the Gemini Flash API.

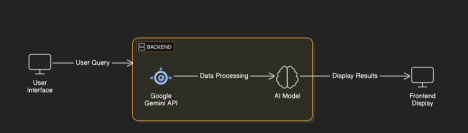
○ Handling **API rate limits** and optimizing API calls.

○ Providing a seamless, multilingual user experience.

**Phase-3: Project Design**

**Objective:**

Develop the architecture and user flow of the application.

**Key Points:**

1. **System Architecture:**

○ User uploads a landmark image and enters a prompt via UI.

○ Backend calls Gemini Flash API to process the image and prompt.

○ AI model generates detailed descriptions.

○ The frontend displays descriptions with multilingual options

2. **User Flow:**

○ Step 1: User uploads a landmark image and enters a prompt (e.g., "Historical background of this landmark”).

○ Step 2: The backend process the image and prompt through gemini flash API.

○ Step 3: App displays descriptions in an intuitive, multilingual format.

3. **UI/UX Considerations:**

○ **Minimalist, intuitive interface for each of use.**

○ **Options for language selection and font size adjustments for accessibility**

○ **Adaptive design for mobile and desktop platforms**

**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 | Member 1 | 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 2 | API response  format finalized | Basic UI with input fields |
| Sprint 2 | Vehicle Search & Comparison | �� High | 3 hours  (Day 2) | Mid-Day 2 | Member 1& 2 | 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&4 | API logs, UI  inputs | Improved API  stability |
| Sprint 3 | Testing & UI  Enhancements | ��  Medium | 1.5 hours (Day 2) |  | Mid-Day 2 Member 2& 3 | 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 AutoSage App.

**Key Points:**

1. **Technology Stack Used:**

○ **Frontend:** Streamlit

○ **Backend:** Google Gemini Flash API

○ **Programming Language:** Python

2. **Development Process:**

○ Implement **API key authentication** and **Gemini API integration**.

○ Develop **image recognition and prompt processing logic**.

○ Optimize **multilingual support for description generation**

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 "History of Eiffel Tower" | Detailed description displayed. | ✅ Passed | Tester 1 |
| TC-002 | Functional  Testing | Upload Taj Mahal image | Accurate details retrieved. | ✅ 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 image recognition | 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 |