## Project Planning: Gemini Landmark Explorer

Project Phase: Project Design & Planning (Main Folder 02)

Sub Folder: Project Planning

Date Prepared: June 20, 2025 (Reflecting a plan for June 12 - June 19, 2025)

Project Title: Gemini Landmark Explorer – An AI-Powered Multimodal Landmark Description App1. Executive Summary

This document outlines the project plan for the Gemini Landmark Explorer application, focusing on the period of June 12th to June 19th, 2025. It details the project scope, key deliverables, a phased work breakdown structure, a conceptual timeline (Gantt chart), required resources, and risk mitigation strategies. The goal is to deliver a functional AI-powered web application that enhances tourist experiences by providing detailed, scenario-based landmark descriptions from images, leveraging Google's Gemini model.

### 2. Project Scope Confirmation

**In Scope:**

* Development of a Streamlit-based web application for user interaction.
* Integration with Google Gemini Pro Vision API (gemini-2.0-flash-001 or gemini-pro-vision) for image and text processing.
* Scenario-based prompt engineering (Traveler, Tour Guide, Educator, Personal Explorer) [Source: Project Report Landmark Explorer.docx].
* Multilingual output translation (English, Hindi, Spanish, French, Japanese, etc.) [Source: app.py, Project Report Landmark Explorer.docx].
* Download functionality for AI-generated descriptions.
* Session-based history of generated descriptions.
* Secure API key handling via .env file.
* In-memory processing of user-uploaded images (no persistent storage) [Source: Gemini\_Landmark\_Description\_App\_Final\_Documentation.docx].
* Clear and intuitive user interface [Source: Project Report Landmark Explorer.docx].

**Out of Scope (for this phase):**

* Real-time camera integration for image capture.
* Persistent cloud storage for user profiles or landmark data.
* External knowledge base grounding for AI outputs.
* Advanced features like voice output or AR/VR integration.
* Sophisticated user authentication or profile management.

### 3. Key Deliverables

Upon completion of the project timeline (June 19th), the following deliverables are expected:

1. **Functional Gemini Landmark Explorer Web Application:** Deployable Streamlit application (app.py).
2. **Project Documentation:**
   * Ideation Document (already created).
   * Final Project Report (Project Report Landmark Explorer.docx) [Source: Project Report Landmark Explorer.docx].
   * Technical Documentation (e.g., Gemini\_Landmark\_Description\_App\_Final\_Documentation.docx) [Source: Gemini\_Landmark\_Description\_App\_Final\_Documentation.docx].
3. **Requirements File:** (requirements.txt) detailing all necessary Python libraries.
4. **Demo/Presentation Readiness:** Application ready for demonstration.

### 4. Work Breakdown Structure (WBS) & Task List

Below is a breakdown of major project phases and their associated tasks, aligning with the project's development.

**Phase 1: Project Setup & Core Environment (Approx. June 12)**

* **Task 1.1:** Finalize project requirements and scope.
* **Task 1.2:** Set up Python environment and virtual environment.
* **Task 1.3:** Install core libraries (streamlit, python-dotenv, Pillow).
* **Task 1.4:** Generate and configure Google API Key (.env file).
* **Task 1.5:** Initialize Streamlit app structure (app.py).

**Phase 2: Gemini Integration & Core Logic (Approx. June 13 - June 14)**

* **Task 2.1:** Integrate google.generativeai library.
* **Task 2.2:** Develop get\_gemini\_response function for image and prompt handling.
* **Task 2.3:** Implement input\_image\_setup for image preparation.
* **Task 2.4:** Initial prompt engineering for a basic "tour guide" description.
* **Task 2.5:** Test basic image upload and AI response functionality.

**Phase 3: User Interface & Feature Development (Approx. June 15 - June 17)**

* **Task 3.1:** Implement image upload component (st.file\_uploader, st.image).
* **Task 3.2:** Develop UI for displaying AI-generated descriptions.
* **Task 3.3:** Implement scenario selection UI (st.sidebar.selectbox).
* **Task 3.4:** Define and implement scenario\_prompts dictionary [Source: app.py].
* **Task 3.5:** Integrate googletrans for multilingual translation (translate\_text function) [Source: app.py].
* **Task 3.6:** Implement language selection UI (st.sidebar.selectbox).
* **Task 3.7:** Develop download button for descriptions [Source: app.py].
* **Task 3.8:** Implement session history management (st.session\_state.history) [Source: app.py].

**Phase 4: Refinement, Testing & Documentation (Approx. June 18 - June 19)**

* **Task 4.1:** Conduct comprehensive testing (functional, UI/UX, responsiveness, error handling).
* **Task 4.2:** Refine AI prompts based on test outputs for accuracy and coherence.
* **Task 4.3:** Implement robust error handling (e.g., try-except blocks).
* **Task 4.4:** Finalize UI/UX improvements (styling, messages, loading indicators).
* **Task 4.5:** Prepare requirements.txt file.
* **Task 4.6:** Complete project documentation (update existing docs, write new sections).
* **Task 4.7:** Final review and preparation for submission.

### 5. Conceptual Timeline (Gantt Chart / Milestone Plan)

This is a retrospective timeline showing how tasks could have been distributed across the project duration.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Task ID** | **Task Description** | **Start Date** | **End Date** | **Duration (Days)** | **Status (Conceptual)** |
| 1.0 | **Project Setup & Environment** | Jun 12 | Jun 12 | 1 | Completed |
| 1.1 | Requirements & Scope Confirmation | Jun 12 | Jun 12 | 0.5 | Completed |
| 1.2 | Environment & Library Setup | Jun 12 | Jun 12 | 0.5 | Completed |
| 1.3 | API Key Configuration | Jun 12 | Jun 12 | 0.5 | Completed |
| 2.0 | **Gemini Integration & Core Logic** | Jun 13 | Jun 14 | 2 | Completed |
| 2.1 | Gemini API Integration & Functions | Jun 13 | Jun 13 | 1 | Completed |
| 2.2 | Initial Prompt Engineering | Jun 14 | Jun 14 | 0.5 | Completed |
| 2.3 | Core Functionality Testing | Jun 14 | Jun 14 | 0.5 | Completed |
| 3.0 | **UI & Feature Development** | Jun 15 | Jun 17 | 3 | Completed |
| 3.1 | Image Upload & Display UI | Jun 15 | Jun 15 | 1 | Completed |
| 3.2 | Scenario Selection & Prompts | Jun 16 | Jun 16 | 1 | Completed |
| 3.3 | Multilingual Translation | Jun 17 | Jun 17 | 1 | Completed |
| 3.4 | Download & History Functionality | Jun 17 | Jun 17 | 0.5 | Completed |
| 4.0 | **Refinement, Testing & Documentation** | Jun 18 | Jun 19 | 2 | Completed |
| 4.1 | Comprehensive Testing & Debugging | Jun 18 | Jun 18 | 1 | Completed |
| 4.2 | UI/UX Polishing & Error Handling | Jun 18 | Jun 18 | 0.5 | Completed |
| 4.3 | Documentation & Report Finalization | Jun 19 | Jun 19 | 1 | Completed |
| 4.4 | Final Review & Submission Prep | Jun 19 | Jun 19 | 0.5 | Completed |

**Recommended Diagram:**

* **Gantt Chart:** A visual representation of the project schedule, showing tasks, their dependencies, durations, and start/end dates. This would be highly beneficial for tracking progress and managing dependencies in a real-time project. Tools like Microsoft Project, Smartsheet, or online Gantt chart generators can create these.

### 6. Resource Allocation

* **Human Resources:**
  + **Project Lead:** (1) Overall project management, coordination, strategic decisions.
  + **Developer(s):** (1-3) Python programming, Streamlit development, AI model integration, testing.
  + **(Optional) QA/Tester:** (1) Dedicated testing and bug reporting.
* **Technical Resources:**
  + **Development Environment:** Python 3.9+, pip, virtual environments.
  + **Libraries:** streamlit, google-generativeai, python-dotenv, Pillow, googletrans.
  + **API Access:** Google Gemini API Key.
  + **Hardware:** Standard personal computer/laptop (min. 4GB RAM) [Source: Project Report Landmark Explorer.docx].
  + **Network:** Stable internet connection for API calls.

### 7. Risk Management

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risk ID** | **Risk Description** | **Likelihood** | **Impact** | **Mitigation Strategy** | **Contingency Plan** |
| R1 | Gemini API Rate Limits/Cost | Medium | Medium | Optimize API calls, implement basic caching (future work), monitor usage. | Explore alternative, cheaper models for initial responses, or implement user warning if limits are near. |
| R2 | API Key Compromise | Low | High | Store API key in .env file, ensure .env is not committed to version control, educate team on security best practices. | Revoke compromised API key immediately, generate new one, and update configurations. |
| R3 | AI Model Hallucinations/Inaccuracies | Medium | Medium | Design prompts to encourage factual responses, add clear disclaimers to users that output is AI-generated and may require cross-referencing [Source: Gemini\_Landmark\_Description\_App\_Final\_Documentation.docx]. | Implement a "report inaccuracy" feature (future work), prioritize clear, well-known landmarks for demos. |
| R4 | googletrans API Issues | Medium | Medium | Be aware of googletrans limitations (unofficial API, potential throttling). | Suggest using official translation APIs if googletrans proves unreliable for a production environment (future work). |
| R5 | Image Recognition Failures (Obscure/Poor Quality) | Medium | Medium | Display user-friendly error messages if image processing fails or if no relevant information can be extracted [Source: Gemini\_Landmark\_Description\_App\_Final\_Documentation.docx]. | Recommend users upload clear, well-lit images of prominent landmarks; provide examples. |
| R6 | Short Timeline (9 days) | High | High | Strict adherence to WBS, minimal scope creep, efficient task delegation, pre-emptive problem-solving. | Prioritize core features, defer less critical enhancements to future iterations, extend timeline if absolutely necessary (but avoided for this project). |

### 8. Communication Plan

* **Daily Stand-ups:** (15 mins) Quick team meetings to discuss progress, blockers, and next steps.
* **Code Reviews:** Peer reviews for critical code sections.
* **Documentation Updates:** Regular updates to project documentation (Ideation, Project Report).
* **Version Control:** Utilize a Git repository (e.g., GitHub) for collaborative code management.
* **Issue Tracking:** Simple method (e.g., shared document or Trello board) to track bugs and tasks.

### 9. Evaluation Metrics (Recap from Project Documentation)

The project's success will be primarily evaluated qualitatively:

* **Relevance:** How well descriptions match the landmark and selected scenario.
* **Accuracy:** Correctness of factual information (name, history, architecture).
* **Coherence & Readability:** Quality of generated text (structure, grammar, clarity).
* **Detail Level:** Sufficiency and interestingness of provided details.
* **User Satisfaction:** Overall helpfulness and engagement.
* **Performance:** Response time (under 10 seconds for typical inputs) [Source: Project Report Landmark Explorer.docx].

### 10. Diagrams / Flowcharts Required

To support the project planning and implementation, the following diagrams are highly recommended:

1. **Gantt Chart (as conceptualized in Section 5):**
   * **Purpose:** To visually represent the project schedule, task durations, dependencies, and overall timeline. It's crucial for project managers to track progress and allocate resources effectively.
   * **Details:** It would show the start and end dates for each task listed in the WBS (Section 4) against the June 12th - June 19th timeline. Dependencies (e.g., UI development depends on core AI integration) would also be clearly marked.
2. **System Workflow Diagram / Activity Diagram (Already in Project Report):**
   * **Purpose:** To illustrate the end-to-end flow of how the Gemini Landmark Explorer app processes user input and delivers contextual landmark descriptions. This helps in understanding the sequence of operations.
   * **Details:** The existing diagram [Source: Project Report Landmark Explorer.docx, Section 5.2.1] adequately covers this:
     + User Uploads Image
     + Streamlit Frontend Interface
     + Scenario-specific Prompt Selection
     + Gemini Multimodal Model via API
     + AI-Generated Description
     + Displayed on UI + Download Option
   * **Recommendation:** While this is already provided, for more granular planning, specific activity diagrams could detail the sub-processes within "Gemini Multimodal Model via API" (e.g., image preprocessing, API call, response parsing) or within "Displayed on UI" (e.g., translation, history update).
3. **Component Diagram (Optional but Recommended for larger projects):**
   * **Purpose:** To show the high-level architecture of the system, depicting the relationships and interfaces between different software components (e.g., Streamlit Frontend, Gemini API, googletrans library, .env config).
   * **Details:** Would visually represent how app.py interacts with google.generativeai and googletrans, and how these communicate with external services (Gemini API, Google Translate).