**Project Documentation: Gemini Landmark Description App**

**Version:** 1.0.0  
**Date:** June 19, 2025  
**Author(s):**

Kamlesh Chowdhary ( [kamlesh.23bce10260@vitbhopal.ac.in](mailto:kamlesh.23bce10260@vitbhopal.ac.in) )

Prashasti Joshi ( [prashasti.23bce10893@vitbhopal.ac.in](mailto:prashasti.23bce10893@vitbhopal.ac.in) )

Somil Asati ( [somil.23bce10364@vitbhopal.ac.in](mailto:somil.23bce10364@vitbhopal.ac.in) )

Devansh Tyagi ( [devansh.23bce10247@vitbhopal.ac.in](mailto:devansh.23bce10247@vitbhopal.ac.in) )

**1. Introduction**

**1.1 Project Overview**

The **Gemini Landmark Description App** is a Streamlit-based AI application designed to provide users with rich, descriptive information about landmarks based on images. By harnessing the power of Google's **Gemini Pro Vision model**, the app identifies visual features of a landmark and responds with historical, architectural, and cultural narratives. It bridges the gap between real-world exploration and instant access to knowledge, transforming every photo into an educational experience.

**1.2 Goals and Objectives**

**Goal:** To enhance travel, educational, and exploratory experiences using AI-powered landmark interpretation.

**Objectives:**

* Develop a responsive and user-friendly web interface for image upload and multilingual display.
* Leverage the Gemini model to generate contextual, engaging landmark descriptions.
* Enable customization of output based on user roles, including tourists, tour guides, educators, and enthusiasts.

**1.3 Scope**

This project focuses on multimodal AI inference using user-uploaded images and scenario-specific prompts. It includes:

* Image handling and preprocessing
* Dynamic prompt selection based on user scenario
* Text generation and translation
* History tracking and download functionality

It does **not** include persistent storage, real-time image capture via camera, or geolocation-based suggestions in this version.

**1.4 Target Audience**

The app serves a diverse user base:

* **Travelers** seeking insights during exploration
* **Professional tour guides** enhancing visitor engagement
* **Educators** designing virtual learning tools
* **Architecture and history enthusiasts** pursuing personal curiosity

**2. Problem Definition and Use Cases**

**2.1 Problem Statement**

In real-world tourism or education, individuals often come across significant landmarks without accessible, concise, or engaging historical context. Existing solutions like guidebooks or web searches are time-consuming, language-dependent, and often require prior knowledge of the landmark's name. There is a clear need for a **vision-based AI tool** that can process visual input and return meaningful content instantly.

**2.2 Use Case Scenarios**

**Scenario 1: Discovering Iconic Landmarks**

A traveler exploring a new city stumbles upon a majestic monument. Without needing to know its name, they upload a photo into the app and instantly receive a well-structured description containing its name, significance, and architectural features. This elevates their sightseeing experience from passive observation to informed appreciation.

**Scenario 2: Tour Guide Assistance**

A professional tour guide uses the app to enhance live narration. By uploading a photo and selecting the "Tour Guide" scenario, the app generates interesting facts, relevant anecdotes, and conversation starters. This supports the guide’s efforts to keep their audience engaged with contextually rich storytelling.

**Scenario 3: Virtual Tours and Educational Resources**

Educators designing virtual museum or site tours can upload images into the app and use AI-generated descriptions as part of their content. This use case highlights the app’s value in remote learning environments, where visual and textual integration fosters better understanding of global heritage.

**Scenario 4: Personal Exploration and Curiosity**

Independent learners with an interest in history or architecture can use the app to explore local or international landmarks. It provides a self-guided, educational journey where users gain insights from AI without needing formal resources.

**3. Generative AI Model and System Architecture**

**3.1 Model Selection**

The application utilizes **Gemini Pro Vision** (gemini-2.0-flash-001) from Google’s Generative AI suite. It supports multimodal input, specifically combining **image and text** to produce relevant, descriptive output.

**3.2 Architecture Diagram**

[User Uploads Image]

|

▼

[Streamlit Frontend UI]

|

[Selected Scenario Prompt + Image]

|

▼

[Gemini Pro Vision API (Google)]

|

▼

[Generated Landmark Description]

|

▼

[Multilingual Translation + Display + Download]

**3.3 Key Components**

| **Component** | **Description** |
| --- | --- |
| **Streamlit** | Front-end interface for uploading images and displaying results |
| **Gemini Model** | AI engine for generating text from multimodal input |
| **PIL** | Image processing for API compatibility |
| **Googletrans** | Translation engine to support Indian and foreign languages |
| **Session State** | Tracks past interactions in-memory |

**3.4 Input/Output Specifications**

**Input:**

* Image file (.jpg, .jpeg, .png)
* Scenario selection
* Optional text prompt

**Output:**

* Textual description covering name, location, history, architecture, and significance
* Translated content (optional)
* Downloadable .txt file

**4. Data Management**

**4.1 Data Sources**

The app does not rely on pre-collected datasets. Instead, all inputs are **real-time user uploads** processed during the session.

**4.2 Preprocessing**

* Image files are read as byte streams and converted into API-acceptable formats.
* Input is accompanied by scenario-based prompt templates.

**4.3 Data Storage**

* All images and descriptions are processed in memory.
* No user data is stored persistently.
* Descriptions are stored temporarily in session history for the current session.

**5. Training and Experimentation**

**5.1 Development Environment**

* **Python Version:** 3.9+
* **Libraries:** streamlit, google-generativeai, Pillow, googletrans, python-dotenv
* **Platform:** Any OS with Python support

**5.2 Training Configuration**

No model training is required. The app directly uses Google’s pre-trained model for inference.

**5.3 Experimentation Log**

Various prompt structures and image types were tested to determine optimal response quality across the four primary scenarios.

**6. Evaluation and Metrics**

**6.1 Qualitative Evaluation**

| **Metric** | **Criteria** |
| --- | --- |
| Relevance | Matches landmark with correct name and context |
| Coherence | Structured, fluent, grammatically correct descriptions |
| Detail Level | Provides depth of history, architecture, and cultural importance |
| Accuracy | Based on known facts of landmarks (validated manually) |
| User Satisfaction | Observed through test cases and exploratory usage |

**6.2 Findings**

* High-quality outputs for known global landmarks
* Decreased precision for obscure, lesser-known images
* Translations mostly accurate with minor exceptions in regional idioms

**7. Deployment and Inference**

**7.1 Deployment Strategy**

The app is deployed via Streamlit and is compatible with cloud platforms like:

* Streamlit Community Cloud
* Hugging Face Spaces
* Custom virtual servers (Linux/Windows)

**7.2 Inference Requirements**

* Active internet connection for API access
* .env file containing a valid GOOGLE\_API\_KEY

**7.3 Scalability**

* Front-end scalability depends on host
* Gemini API scales automatically via Google infrastructure

**8. Ethical Considerations and Responsible AI**

| **Risk** | **Mitigation Strategy** |
| --- | --- |
| Misinformation | Use disclaimers; encourage user fact-checking |
| Cultural Biases | Scenario-based prompt engineering to reduce insensitive content |
| Privacy Concerns | No data retention; user content not stored |
| Misuse of Platform | Limit inputs to educational/public content use only |

**9. Future Work and Enhancements**

**9.1 Planned Improvements**

* Add **location services** to suggest nearby landmarks
* Build a **cache layer** for high-frequency landmarks
* Expand **language support** to 25+ languages
* Integrate **voice narration** for visually impaired users
* Enable **offline batch processing** for educators

**9.2 Research Directions**

* Ground generative outputs to verified knowledge bases
* Analyze hallucination rates across languages and image types

**10. Version Control and Repository**

| **Item** | **Description** |
| --- | --- |
| Code Repository | [Add your GitHub/Bitbucket/GitLab link here] |
| Branching Model | main for stable, dev for active development |
| Current Version | v1.0.0 |

**11. How to Use / Getting Started**

**11.1 Folder Structure**

gemini-landmark-app/

├── app.py

├── .env

├── requirements.txt

└── README.md

**11.2 Setup Instructions**

1. Clone or download the project.
2. Create a .env file with:
3. GOOGLE\_API\_KEY="your\_google\_api\_key"
4. Install dependencies:
5. pip install -r requirements.txt
6. Run the application:
7. streamlit run app.py

**12. References and Acknowledgements**

* [Google Gemini API Documentation](https://ai.google.dev/gemini-api)
* [Streamlit Documentation](https://docs.streamlit.io/)
* [Googletrans Python Library](https://py-googletrans.readthedocs.io/)
* OpenAI (for best practices on generative model usage)

**13. Glossary**

| **Term** | **Definition** |
| --- | --- |
| Generative AI | A class of AI models capable of creating content (text, image, etc.) |
| Gemini Pro Vision | Google's multimodal model for text + image understanding |
| Streamlit | Python-based framework for building interactive web apps |
| Prompt Engineering | The design of model input instructions to guide AI outputs effectively |
| .env File | Configuration file storing sensitive credentials outside source code |