Gemini Historical Artifact Description

Project Documentation format

1. Introduction

- **Project Title:** Gemini Historical Artifact Description

2. Project Overview

- **Purpose:** To provide a Generative AI-powered curatorial tool that generates scholarly, long-form historical descriptions of artifacts based on text or image inputs.
- **Features:**
    - **Multimodal Input:** Support for both text-based artifact names and image uploads.
    - **Customizable Length:** Generates content from 100 up to 1500 words.
    - **Engagement Features:** Real-time historical facts displayed during AI processing.
    - **Data Export:** One-click "Copy to Clipboard" and "Export as Text" functionality.

3. Architecture

- **Frontend:** Built using **Streamlit**, providing a responsive, museum-themed user interface.
- **Backend:** Logic layer handled via **Python**, managing prompt engineering and state management.
- **AI Engine (Database/Intelligence): Google Gemini 1.5 Flash API** acts as the intelligent core for identifying artifacts and generating text.

4. Setup Instructions

- **Prerequisites:** Python 3.10+, pip, and a Google AI Studio API Key.
- **Installation:**
    1. Clone the repository: git clone [Your Repo Link]
    2. Install dependencies: pip install -r requirements.txt
    3. Set Environment Variables: Create a .env file and add GOOGLE_API_KEY=your_key_here.

5. Folder Structure

- **Main Application:**
  - app.py: Contains the Streamlit UI and Gemini API logic.
  - requirements.txt: Lists libraries like google-generativeai, streamlit, and Pillow.
  - .gitignore: Configured to exclude .env for security.

6. Running the Application

- Navigate to the project directory and run:
  - **Command:** streamlit run app.py

7. API Documentation

- **Model:** gemini-1.5-flash-latest
- **Input Parameters:** prompt (String), image (PIL Image object), word_count (Integer).
- **Response:** A Markdown-formatted historical blog post.

8. Authentication

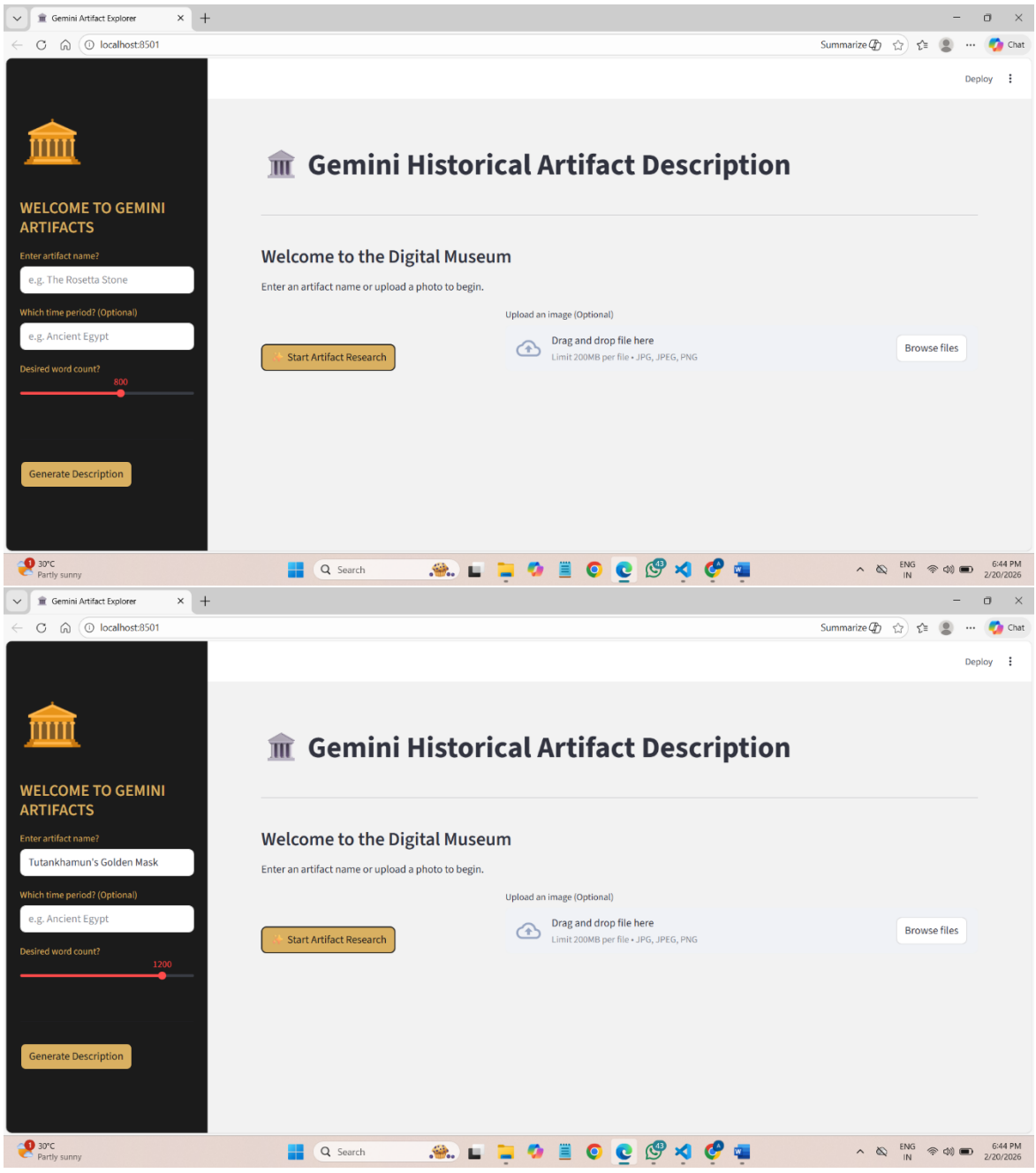
- **Method:** API Key-based authentication.
- **Handling:** The key is securely loaded from a .env file using the python-dotenv library to ensure it is never exposed in the source code.

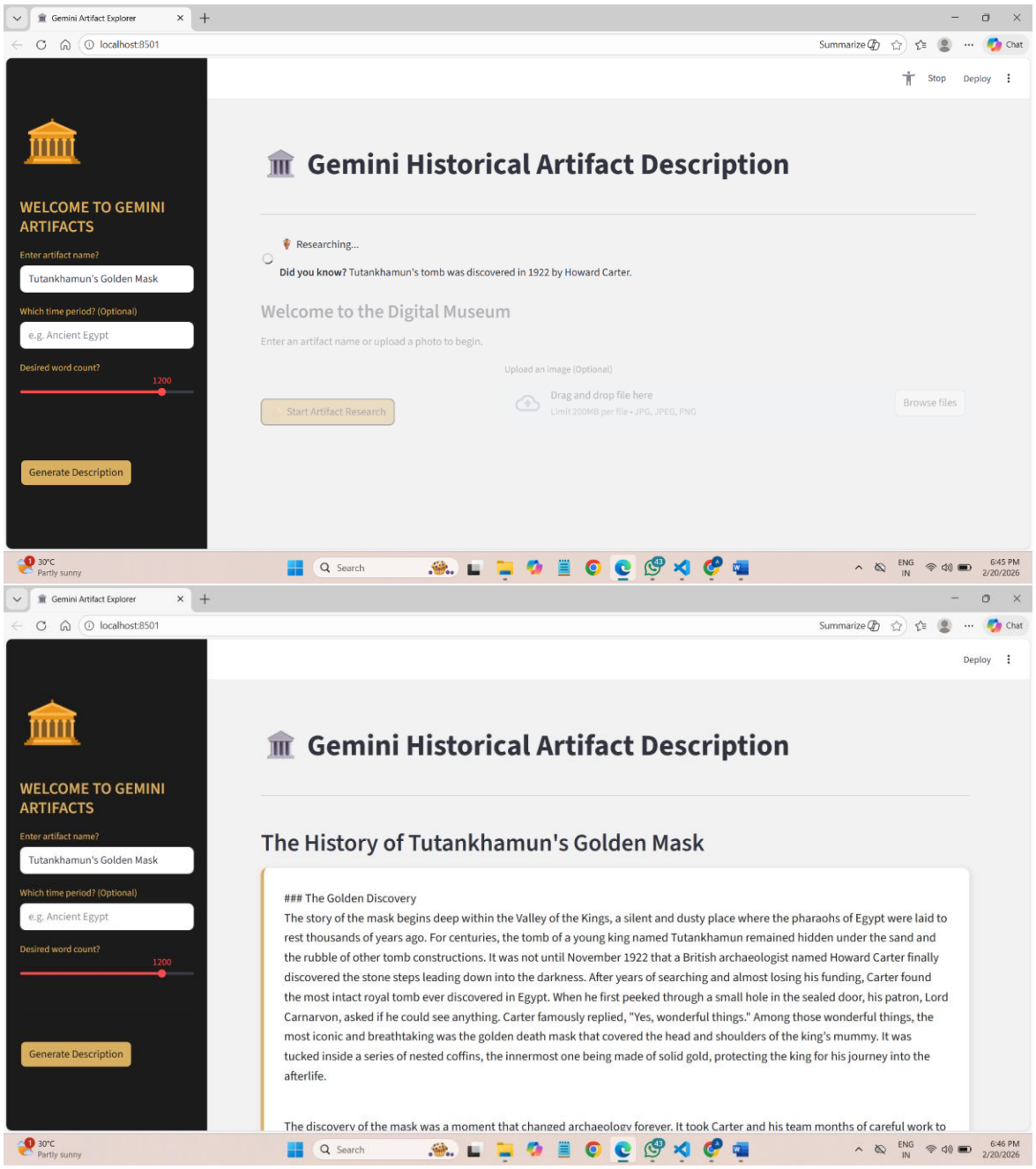
9. Known Issues & Resolutions

Issue Found	Resolution Applied
Credential Leak: API key was initially visible in the commit history.	Resolved: Rotated the API key in Google Cloud, added .env to .gitignore, and force-pushed a clean history.
Repetitive Content: AI generated similar paragraphs for very high word counts (1500+).	Resolved: Refined the system prompt to require specific scholarly headings (e.g., 'Craftsmanship', 'Legacy') to force diverse content.
Image Recognition Failure: Uploading an image without a name sometimes confused the model.	Resolved: Implemented a "Smart Prompt" that explicitly instructs Gemini to "Identify the artifact in the image first if no name is provided."

State Reset: UI refreshed and cleared results when users clicked the "Copy" button.	Resolved: Used st.session_state to preserve generated text across button-click reruns.

10. User Interface & Screenshots





11. Future Enhancements

- **Voice Narrator:** Integration of Text-to-Speech (TTS) to provide audio museum tours.
- **Language Support:** Translating historical descriptions into multiple international languages.
- **Database Integration:** Saving user history and favorite artifacts to a persistent database