

FINAL REPORT

GEMINI HISTORICAL ARTIFACT DESCRIPTION

1. INTRODUCTION

1.1 Project Overview

Gemini Historical Artifact Description is a web-based Generative AI application designed to automate the creation of structured and detailed historical artifact descriptions. The system leverages Google's Gemini (gemini-flash-latest) model to generate comprehensive, academically styled artifact descriptions based on user input.

The application allows users to:

- Enter an Artifact Name
- Enter a Historical Period / Civilization
- Select desired word count (100–2000 words)
- Generate a structured historical artifact description
- Download the output as a Markdown (.md) file

The solution is implemented using Streamlit for the frontend interface and Python for application logic, with Google Gemini API integration for AI-driven content generation.

1.2 Purpose

The purpose of this project is to:

- Reduce the time required to research and write structured historical artifact descriptions
- Provide AI-assisted content generation for students, researchers, museum curators, and history bloggers
- Demonstrate practical integration of Generative AI into an educational and documentation-based web application
- Deliver a lightweight, scalable, and deployable AI-powered system
- Enhance accessibility to structured historical knowledge through automation

2. IDEATION PHASE

2.1 Problem Statement

Students, researchers, museum curators, and history enthusiasts spend significant time researching and writing structured historical artifact descriptions. Preparing detailed content that includes origin, material, historical context, artistic style, and cultural significance requires strong research and academic writing skills.

Traditional history websites provide static and scattered information, but they do not assist users in generating customized, structured, and publication-ready artifact descriptions instantly.

Gemini Historical Artifact Description addresses this gap by providing AI-powered automated generation of structured historical artifact descriptions, reducing manual effort while maintaining professional and academic presentation.

Customer Problem Statement

Project Context: Flavor Fusion – AI Recipe Blog Generator

| | |
|--|--|
| I am Describe the customer and their attributes | PS-1: I am a food blogger Describe the customer and their attributes |
| I'm trying to List the thing they are trying to achieve here | PS-1: I am a beginner home cook PS-2: I'm trying to consistently publish engaging and structured recipe blog posts PS-2: I'm trying to share my recipes online in a professional blog format |
| but Describe the problems or barriers that get in | PS-1: But I spend too much time drafting introductions, formatting ingredients, and writing detailed instructions PS-2: But I don't know how to structure content or write engaging introductions Describe the problems or barriers that get in the way here |
| which makes me feel Describe the emotions the result from experience | PS-1: Because writing high-quality content requires creativity, structure, and time investment PS-2: Because I lack professional writing skills and blogging experience |

2.2 Empathy Map Canvas

Target User

Museum curator, history student, researcher, or history blogger with limited time for detailed writing.

Says

- “Writing detailed artifact descriptions takes too much time.”
- “I need professional and structured content.”
- “I want my exhibition or blog to look academically strong.”

Thinks

- “My description may not sound scholarly enough.”
- “I might miss important historical context.”
- “I need to maintain consistency across multiple artifacts.”

Does

- Searches academic websites and museum archives
- Refers to multiple sources before writing
- Spends hours structuring and formatting descriptions

Feels

- Overwhelmed by research workload
- Pressured by deadlines
- Insecure about academic writing quality

Gains Expected

- Faster artifact description generation
- Well-structured and professional output
- Consistent formatting across descriptions
- Increased confidence in publishing academic or blog content

2.3 Brainstorming

During the ideation phase, the team evaluated multiple AI-based content generation ideas including:

- AI museum catalog description generator
- Historical timeline generator
- Academic research summary tool

- Cultural heritage storytelling assistant
- AI historical artifact description generator

The AI historical artifact description generator was selected due to:

- Clear and identifiable user problem
- Feasible implementation using Gemini API
- Strong demonstration of Generative AI capability
- Educational and academic real-world relevance
- Structured output suitable for documentation and publication

Features were prioritized based on user impact and development effort using a prioritization matrix.

Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

⌚ 10 minutes to prepare
⌚ 1 hour to collaborate
👤 2-8 people recommended

Before you collaborate
A little bit of preparation goes a long way with this session. Here's what you need to do to get going.
⌚ 10 minutes

Define your problem statement
What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.
⌚ 5 minutes

Key rules of brainstorming
To run a smooth and productive session

| | |
|-------------------|----------------------------|
| 💡 Stay in topic. | 💡 Encourage wild ideas. |
| ⌚ Defer judgment. | 👤 Listen to others. |
| 📍 Go for volume. | 👁️ If possible, be visual. |

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

1. User opens the web application

2. Enters artifact name or description
3. Selects desired word count
4. Clicks “Generate Description”
5. Views AI-generated artifact description
6. Downloads the Markdown file

Pain Point Addressed:

Manual research and academic drafting effort is replaced with instant AI-generated structured artifact descriptions.

3.2 Solution Requirement

Functional Requirements

- Accept artifact name or historical context input
- Accept word count selection
- Generate structured artifact description using Gemini API
- Include sections such as Artistic Style, Historical Context, Scene Depicted, and Cultural Significance
- Display generated content clearly in UI
- Provide Markdown (.md) download option
- Show loading indicator during AI processing
- Handle API errors and invalid inputs gracefully

Non-Functional Requirements

- Usability: Simple, clean, and intuitive interface
- Performance: Description generated within acceptable API response time
- Reliability: System should not crash during API or network failure
- Security: API key stored securely using environment variables
- Scalability: Cloud-deployable and horizontally scalable architecture

3.3 Data Flow Diagram (Description)

User → Streamlit UI → Application Logic → Gemini API

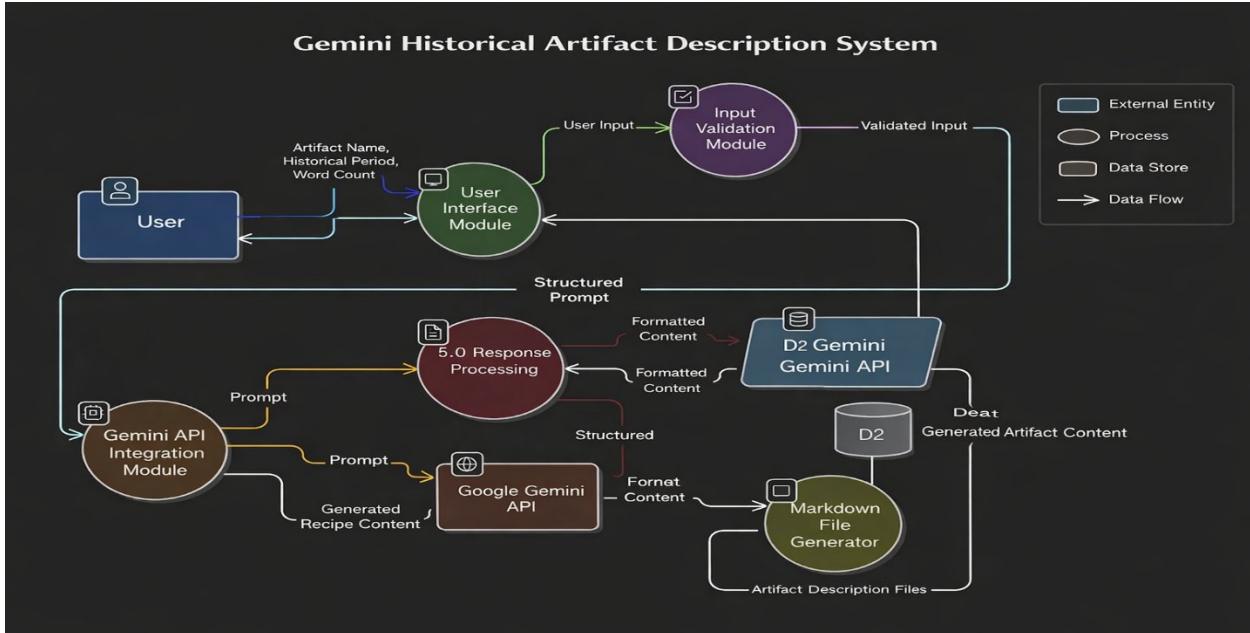
Gemini API → Application Logic → Content Formatter → Markdown Generator → User

The system processes user input, sends a structured prompt to the Gemini model, formats the AI response into a professional catalog-style description, and returns it for viewing and download.

DFD Level 0 (Industry Standard)



DFD Level 1 (Industry Standard)



The system follows a stateless architecture with no persistent database.

3.4 Technology Stack

Frontend: Streamlit

Backend: Python

AI Model: Google Gemini (gemini-flash-latest)

File Format: Markdown (.md)

Deployment: Local / Streamlit Cloud

4. PROJECT DESIGN

4.1 Problem-Solution Fit

Problem:

Writing detailed, structured historical artifact descriptions requires extensive research, domain knowledge, and academic writing skills. This process is time-consuming and may lead to inconsistent formatting across multiple artifacts.

Solution:

An AI-powered web application that automatically generates professionally structured historical artifact descriptions using Google Gemini. The output includes artistic style, historical context, scene depiction, and cultural significance in a catalog-ready format.

Fit Justification:

The solution directly reduces research and writing effort while maintaining professional, academic-quality output. It ensures consistency, saves time, and supports museums, researchers, students, and content creators in producing structured documentation efficiently.

| | | | | | | |
|--|----------------|--|-----------|---|-----------|--|
| 1. CUSTOMER SEGMENT(S) Who is your customer? i.e. working parents of 0-5 y.o. kids | CS | 6. CUSTOMER CONSTRAINTS What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices. | CC | 5. AVAILABLE SOLUTIONS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking | AS | Explore AS, differentiate |
| 2. JOBS-TO-BE-DONE / PROBLEMS Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides. | J&P | 9. PROBLEM ROOT CAUSE What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations. | RC | 7. BEHAVIOUR What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; Indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace) | BE | Focus on J&P, map into BE; understand RC |
| 3. TRIGGERS What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news. | TR | 10. YOUR SOLUTION If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business idea, leave the first section blank, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour. | SL | 8. CHANNELS OF BEHAVIOUR 8.1 ONLINE What kind of actions do customers take online? Extract online channels from #7 | CH | Extract online & offline Ch of BE |
| 4. EMOTIONS: BEFORE / AFTER How do customers feel when they face a problem or a job and afterwards? i.e. lost, insecure > confident, in control - use it in your communication strategy & design. | EM | | | 8.2 OFFLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development. | | |

4.2 Proposed Solution

The Artifact AI application is a lightweight AI-powered web system that:

- Dynamically generates structured historical artifact descriptions
- Provides customizable word length
- Produces academically styled content (Artistic Style, Historical Context, Cultural Significance, etc.)
- Offers export-ready Markdown (.md) files
- Enhances user experience through an interactive and simple UI

The system bridges the business need (automated academic content generation) with Generative AI technology.

4.3 Solution Architecture

The architecture consists of four layers:

Business Layer

User need for faster, structured, and professional artifact documentation.

Application Layer

Streamlit UI + Python-based application logic modules.

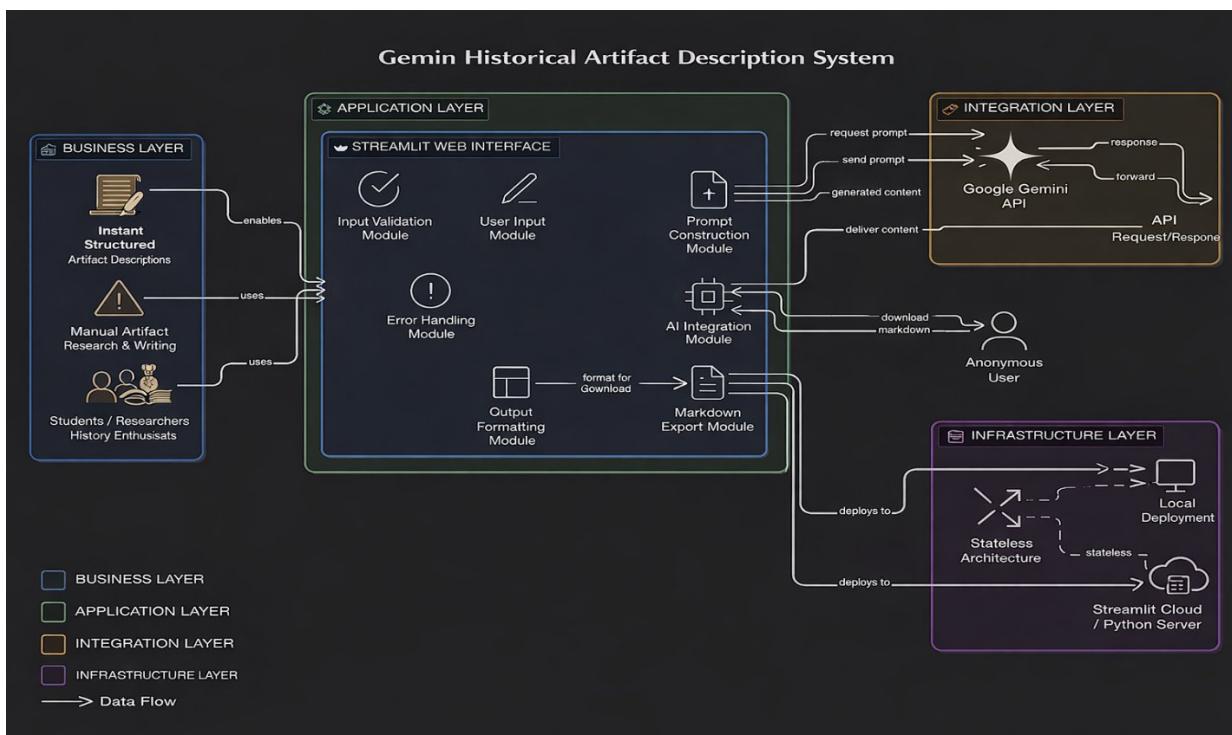
Integration Layer

Google Gemini API for AI-powered content generation.

Infrastructure Layer

Local system or cloud-based deployment (e.g., Streamlit Cloud).

The application is stateless and horizontally scalable. Since AI processing is handled through the Gemini API, scaling depends primarily on API limits and cloud infrastructure capacity.



5.1 Project Planning

Development was completed in two sprints:

Sprint 1:

- User Interface development using Streamlit
- Google Gemini API integration
- Structured prompt engineering for artifact description generation

Sprint 2:

- Markdown export functionality
- Loading indicator during AI processing
- Input validation and error handling
- Functional testing and local/cloud deployment

Average team velocity: 19 Story Points per sprint.

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

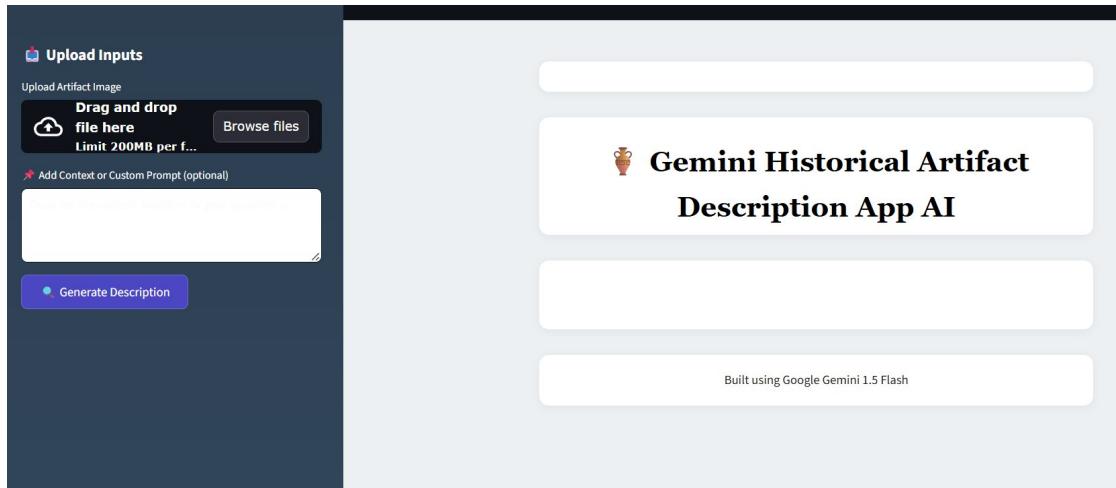
- AI generation time was tested under multiple word count conditions.
- Short description (~200 words): 3–5 seconds.
- Long description (~1500 words): 5–8 seconds depending on API latency.
- System tested under simulated API delay conditions.
- No application crash observed during API timeout simulation.

The system was confirmed stable and ready for User Acceptance Testing (UAT).

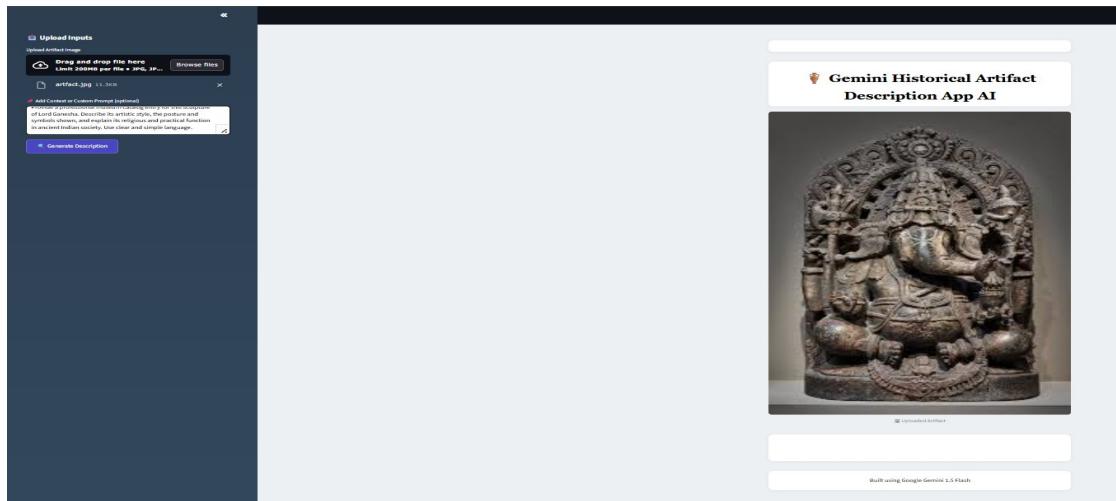
7. RESULTS

7.1 Output Screenshots

- User Launches Application



- Inputs Artifact Details



- AI Description Generation Loading

Upload Inputs

Upload Artifact Image

Drag and drop file here
Limit 200MB per file • JPG, JPEG, PNG

[Browse files](#)

aircraftimages.jpg X
11.3K

Add Context or Custom Prompt (optional)

Generate Description



Uploaded Artifact

Analyzing artifact and generating insights...

- Display Output

Upload Inputs

Upload Artifact Image

Drag and drop file here
Limit 200M...

[Browse files](#)

air... X
11.3K

Add Context or Custom Prompt (optional)

Generate Description

Description generated successfully!

AI-Generated Artifact Description

As a historian specializing in South Asian art and culture, I can provide you with a detailed analysis of this remarkable artifact.

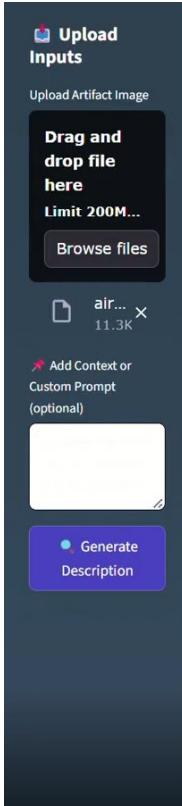
Identity and Description

This sculpture depicts **Ganesha** (also known as Ganapati or Vinayaka), one of the most popular and ubiquitous deities in the Hindu pantheon. He is easily recognizable by his elephant head and human body.

In this specific depiction, Ganesha is shown in a seated posture. He is "pot-bellied" (*lambodara*), a characteristic symbolizing his ability to swallow and digest the sorrows of the world. He is four-armed, a common iconographic trait signifying divine power.

- **Iconography (Attributes):** In his upper right hand, he holds a battle-ax (*parashu*); in his upper left, a lotus flower or possibly a noose; in his lower right, his own broken tusk; and in his lower left, a bowl of *modakas* (sweet dumplings), which his trunk is reaching toward.
- **Ornamentation:** He is adorned with incredibly intricate jewelry, including a beaded crown (*karanda mukuta*), heavy necklaces, armlets, and a sacred thread (*yagnopavita*) in the form of a serpent wrapped around his belly.
- **The Halo:** Behind him is an elaborate, circular decorative arch (*prabharamandala*) featuring deeply carved floral and geometric patterns, characteristic of the height of regional craftsmanship.

- Markdown Export



carved floral and geometric patterns, characteristic of the height of regional craftsmanship.

Historical Context

- Name: Seated Ganesha.
- Origin: Southern India, specifically the region of modern-day Karnataka.
- Time Period: Hoysala Empire, circa 12th–13th century CE.
- Material: Chloritic Schist (commonly referred to as soapstone). This material was favored by Hoysala sculptors because it is relatively soft when first quarried, allowing for the minute, filigree-like detail seen here, before hardening upon exposure to air.

Cultural and Artistic Significance

The Hoysala period is considered a "golden age" of South Indian temple architecture and sculpture. This artifact is a prime example of the **Hoysala style**, which is defined by its extreme attention to ornamental detail, high-relief carving that almost detaches the figure from the backstone, and a sense of "jewel-like" precision.

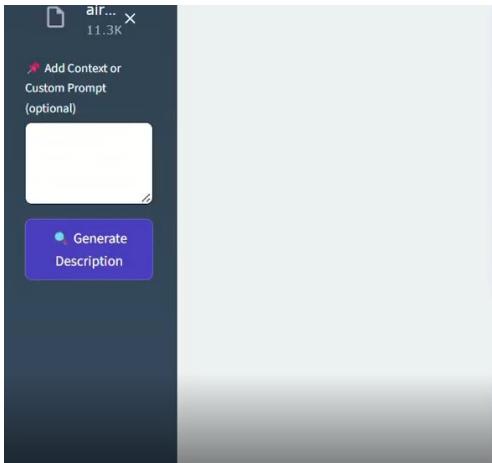
Religious Significance: Ganesha is revered as the **Remover of Obstacles** (*Vighneshvara*) and the **Lord of Beginnings**. In Hindu tradition, he is the first deity invoked at the start of any new venture, prayer, or wedding. He is also the patron of arts, sciences, and wisdom.

The broken tusk he holds refers to a famous myth where Ganesha broke off his own tusk to use as a pen to transcribe the epic poem, the *Mahabharata*, as it was dictated to him by the sage Vyasa, symbolizing the sacrifice required for the pursuit of knowledge.

Summary of Facts

- **The Vahana:** Although difficult to see in this specific view, Ganesha's "mount" or *vahana*, a small shrew or mouse (*Musika*), is typically carved near his feet at the base, representing the taming of desire and ego.
- **Location:** Sculptures of this quality were originally housed in the niches of grand temples, such as the Hoysaleswara Temple at Halebidu or the Chennakeshava Temple at Belur.

- Additional Features Demonstrated



Symbolizing the sacrifice required for the pursuit of knowledge.

Summary of Facts

- **The Vahana:** Although difficult to see in this specific view, Ganesha's "mount" or *vahana*, a small shrew or mouse (*Musika*), is typically carved near his feet at the base, representing the taming of desire and ego.
- **Location:** Sculptures of this quality were originally housed in the niches of grand temples, such as the Hoysaleswara Temple at Halebidu or the Chennakeshava Temple at Belur.
- **Preservation:** The dark, lustrous patina on the stone is often the result of centuries of ritual anointing with oils and pastes during worship (*puja*), which has helped preserve the stone's integrity while giving it a polished appearance.

Download Description

Generated Outputs Demonstrated:

- Properly structured academic artifact description
- Inclusion of sections such as Artistic Style, Historical Context, Cultural Significance, and Interpretation

- Correct word count range based on user selection
- Professional and catalog-ready presentation format

8. ADVANTAGES & DISADVANTAGES

Advantages

- Significant time savings in artifact documentation
- Structured and academically styled professional output
- Lightweight and modular architecture
- Easy cloud deployment
- Beginner-friendly and intuitive UI
- Consistent formatting across multiple artifact descriptions

Disadvantages

- Dependent on external AI API (Gemini)
- Requires active internet connection
- API response time may vary depending on latency
- No persistent database storage for previously generated descriptions

9. CONCLUSION

The Gemini Historical Artifact Description project successfully demonstrates the practical integration of Generative AI into a real-world academic and documentation-focused web application.

The system significantly reduces manual research and drafting effort while maintaining structured, professional-quality output. It validates the feasibility of AI-driven automation for cultural heritage documentation, museum catalog preparation, and academic content creation.

The project achieves strong problem-solution alignment and demonstrates a scalable, cloud-ready AI application architecture.

10. FUTURE SCOPE

- Multi-language artifact description generation to support global museums, researchers, and students.

- Academic citation support (auto-suggested references or formatted citation styles like APA/MLA).
- AI-based image generation or enhancement for artifact visualization and catalog presentation.
- Integration with museum databases or digital archives for automated metadata enrichment.
- User account system with description history storage for managing multiple artifacts.
- Cloud auto-scaling deployment for handling large-scale institutional usage.
- Advanced customization options (tone selection: academic, storytelling, museum label, research format).

11. APPENDIX

Source Code (task.py):

```
from dotenv import load_dotenv
import streamlit as st
import os
import google.generativeai as genai
from PIL import Image

# --- Configuration ---
load_dotenv()
api_key = os.getenv("GEMINI_API_KEY")
genai.configure(api_key=api_key)

# --- Utility Functions ---
def input_image_setup(uploaded_file):
    if uploaded_file is not None:
        bytes_data = uploaded_file.getvalue()
        return [{"mime_type": uploaded_file.type, "data": bytes_data}]
    else:
```

```
raise FileNotFoundError("No file uploaded")

def get_gemini_response(input_text, image_data, prompt):
    full_prompt = f"{prompt}\n\nUser Instruction: {input_text}"
    model = genai.GenerativeModel("gemini-flash-latest")
    response = model.generate_content([full_prompt, image_data[0]])
    return response.text

# --- Streamlit Page Settings ---
st.set_page_config(
    page_title="💡 Gemini Historical Artifact Description App",
    page_icon="💡",
    layout="centered",
)

# --- Custom CSS Styling ---
st.markdown("""
<style>
.stApp {
    background-color: #ecf0f3;
    font-family: 'Verdana', sans-serif;
    color: #222222;
}

.title {
    font-family: 'Georgia', serif;
    font-size: 36px;
    font-weight: 700;
    color: black;
}
""")
```

```
text-align: center;  
margin-top: 0.5rem;  
margin-bottom: 0.2rem;  
}  
  
section[data-testid="stSidebar"] {  
background: linear-gradient(to bottom, #2c3e50, #34495e);  
color: #ffffff;  
}  
  
section[data-testid="stSidebar"] h1,  
section[data-testid="stSidebar"] h2,  
section[data-testid="stSidebar"] h3,  
section[data-testid="stSidebar"] h4,  
section[data-testid="stSidebar"] h5,  
section[data-testid="stSidebar"] h6,  
section[data-testid="stSidebar"] label,  
section[data-testid="stSidebar"] span {  
color: #ffffff !important;  
font-weight: bold;  
}  
  
main[data-testid="stAppViewContainer"] {  
background-color: #ecf0f3;  
}  
  
main[data-testid="stAppViewContainer"] > div:first-child {  
padding: 0 !important;  
margin: 0 !important;
```

```
}
```

```
.stButton button {  
background-color: #4B47C3;  
color: white;  
border-radius: 8px;  
font-weight: bold;  
padding: 0.5em 1.4em;  
transition: background-color 0.3s ease;  
}
```

```
.stButton button:hover {  
background-color: #3934a1;  
}
```

```
.stTextInput input {  
background-color: #ffffff;  
border: 1px solid #ffffff;  
border-radius: 6px;  
padding: 0.5em;  
color: #000000;  
}
```

```
.stTextArea textarea {  
background-color: #ffffff;  
border: 1px solid #ffffff;  
border-radius: 6px;  
padding: 0.5em;  
color: #000000;
```

```
}
```

```
.stMarkdown {  
background-color: white;  
padding: 1.2rem;  
border-radius: 12px;  
box-shadow: 0 2px 10px rgba(0, 0, 0, 0.05);  
margin-top: 1rem;  
}
```

```
.stDownloadButton button {  
background-color: #6E6E6E;  
color: white;  
border-radius: 6px;  
}
```

```
.stDownloadButton button:hover {  
background-color: #4f4f4f;  
}.css-1d391kg, .css-1dp5vir {  
background-color: #ffffff !important;  
border-radius: 12px;  
padding: 1rem !important;  
box-shadow: 0 2px 10px rgba(0, 0, 0, 0.06);  
}
```

```
.css-qrbaxs, .css-1v3fvcr {  
color: #4B47C3 !important;  
font-weight: 600 !important;  
}
```

```

</style>

"""", unsafe_allow_html=True)

# --- Header ---

st.markdown('<div class="title">⭐ Gemini Historical Artifact Description App AI</div>', unsafe_allow_html=True)

# --- Sidebar Input Section ---

with st.sidebar:

    st.header("📸 Upload Inputs")

    uploaded_file = st.file_uploader("Upload Artifact Image", type=["jpg", "jpeg", "png"])

    input_text = st.text_area("📌 Add Context or Custom Prompt (optional)", placeholder="Describe the context, location, or your question...")

    submit = st.button("🔍 Generate Description")

# --- Display Uploaded Image ---

if uploaded_file:

    st.image(uploaded_file, caption="🖼️ Uploaded Artifact", use_container_width=True)

# --- Predefined System Prompt ---

system_prompt = """"

You are a historian. Please describe the historical artifact in the image and provide detailed information, including its name, origin, time period, material, cultural significance, and any other relevant facts.

"""

# --- Output Area ---

if submit:

    try:

        with st.spinner("🔍 Analyzing artifact and generating insights..."):

            image_data = input_image_setup(uploaded_file)

            response = get_gemini_response(input_text, image_data, system_prompt)

```

```
# ✅ Soft Pastel Blue Full-Width Container Box

st.markdown(""""

<div style='

width: 100%;

background-color: #d8f0ff;

padding: 1rem 1.5rem;

border-radius: 12px;

border: 1px solid #b9e6fa;

color: #114a63;

font-family: Verdana, sans-serif;

font-weight: 500;

text-align: center;

box-sizing: border-box;

margin: 1.5rem auto;

box-shadow: 0 2px 8px rgba(0,0,0,0.05);

'>

Description generated successfully!

</div>

"""", unsafe_allow_html=True)

st.markdown("### 📝 AI-Generated Artifact Description")

st.markdown(response)

st.download_button(

label="📝 Download Description",

data=response,

file_name="artifact_description.txt",

mime="text/plain"
```

```
)  
except Exception as e:  
    st.error(f"⚠ Error: {str(e)}")  
  
# --- Footer ---  
st.markdown("---")  
st.markdown(  
    "<p style='text-align: center;'>Built using Google Gemini 1.5 Flash </p>",  
    unsafe_allow_html=True  
)
```

GitHub Repository:

<https://github.com/AnushaYerramsetti16/-GEMINI-HISTORICAL-ARTIFACT-DESCRIPTION->

Project Demo Link:

https://drive.google.com/file/d/1Yc34QaskqthJmM9nHTu-GxAHDv-wJnUf/view?usp=drive_link