Gemini Historical Artifact

1. INTRODUCTION

1.1. Project Overview

Gemini Historical Artifact is an AI-enabled web application developed to generate comprehensive and structured descriptions of historical artifacts using Google's Generative AI technology. The application allows users to enter the name of a historical artifact, monument, manuscript, or historical period and specify the required word count for the generated content. Based on this input, the system produces an informative and well-organized description within seconds.

The primary goal of this project is to simplify the process of creating historical content for students, researchers, museum professionals, and history enthusiasts. Traditionally, preparing detailed descriptions of artifacts requires extensive research, careful structuring, and time-consuming writing. This application automates that process through artificial intelligence.

The system is implemented using Python and Streamlit for the front-end interface, while the Gemini Generative AI model is integrated through an API for dynamic content generation. During processing, the application also presents an interesting historical fact to enhance user engagement.

Overall, the Gemini Historical Artifact project demonstrates the practical implementation of generative AI in the field of digital humanities by providing a fast, structured, and user-friendly solution for automated historical content generation.

1.2. Objectives

The major objectives of the Gemini Historical Artifact project are:

- To design and develop a web-based application that generates historical artifact descriptions using artificial intelligence

- To enable users to input artifact names or historical topics and receive structured content

- To allow customization of description length through user-defined word count

- To minimize the time required for researching and writing historical content manually

- To integrate Google Gemini Generative AI for efficient and reliable text generation

- To provide an intuitive and easy-to-use user interface using Streamlit

- To improve user engagement by displaying interesting historical facts during content generation

- To create a scalable system that can be deployed online for broader accessibility

2. Ideation Phase

2.1. Problem Statement

In academic and cultural institutions, creating detailed descriptions of historical artifacts requires significant research effort, writing skill, and formatting precision. Students and researchers often spend hours gathering information from multiple sources before compiling structured content. Additionally, many online resources provide generic or incomplete descriptions that may not meet specific academic or professional requirements.

This leads to reduced productivity, repetitive work, and difficulty in maintaining consistency across multiple descriptions. There is a need for an intelligent system that can automatically gener ate accurate and structured historical descriptions based on user input.

The Gemini Historical Artifact application addresses this issue by using a generative AI model to instantly produce detailed and customized artifact descriptions tailored to the required word count.

History Research & Content Creation: User Persona Analysis

Persona	Core Goal	Primary Obstacle	Impact
PS-1: History Researcher documentation.	Prepare detailed artifact	Manual research and rewiting take too much time.	Feels OVEROLADED and productive.
PS-2: Museum Curator	Create engaging, structured exhibit descriptions.	Existing sources are repeitive; lack of AI-structured tools.	Feels FRUSTATED and creatively constrained.
Publish highish-qualigty, consistent articles.	Publish high-quality, consistent articles.	Research and formating consume significant time.	Frequent DELAYS in content publishing.

2.2. Empathy Map Canvas

Empathy Map Canvas:

An empathy map is a visualization tool used to understand user behavior, expectations, and challenges. It helps in identifying what users think, feel, say, and do while interacting with a system.

For the Gemini Historical Artifact application, the empathy map focuses on understanding students, historians, and cultural professionals who require structured historical information.

User Perspective:

Thinks:

- "I need reliable and structured historical information."
- "Researching from multiple websites is time-consuming."

Feels:

- Pressured by deadlines
- Overwhelmed by lengthy research processes

Says:

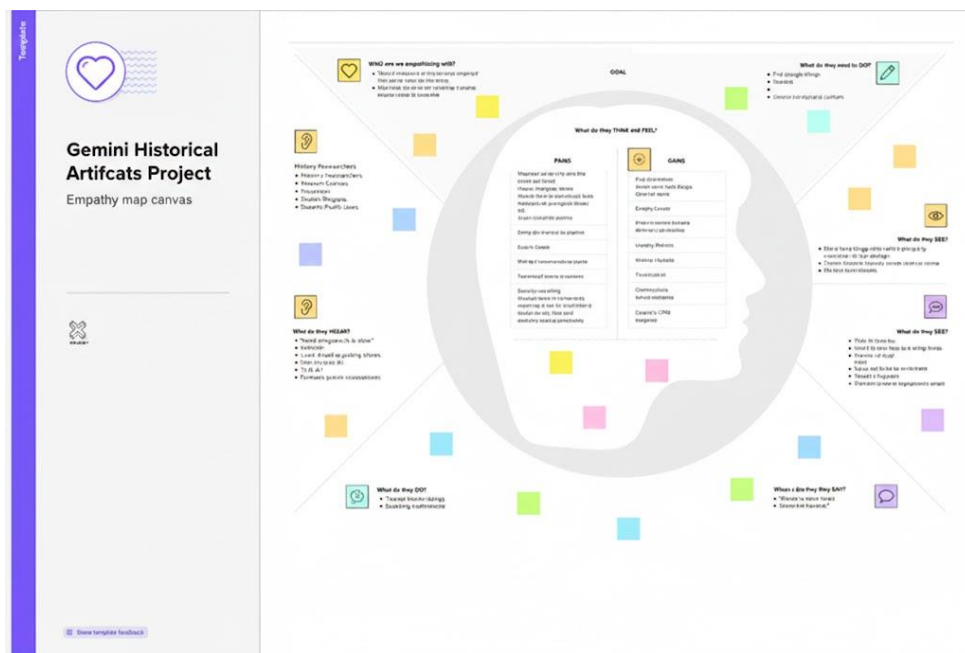- "I wish I had a tool that generates historical content instantly."

- "I need well-organized descriptions for academic or exhibition purposes."

Does:

- Searches online for artifact details

- Copies and edits information manually

- Spends time formatting content

By understanding these behaviors, the solution was designed to provide automated, structured, and customizable historical descriptions.

Example:

2.3 Brainstorming

Brainstorm & Idea Prioritization:

The brainstorming phase involved identifying multiple ways to apply generative AI in the historical domain. The team discussed several potential ideas including:

- AI-based historical chatbot

- Virtual museum guide system

- Image-based artifact recognition

- Automated historical content generator

- AI-powered timeline generator

After evaluating feasibility, development complexity, and implementation time, the team selected the concept of an AI-based artifact description generator as it provided:

- Clear problem definition

- Achievable scope

- Strong academic relevance

- Practical real-world application

Brainstorming Steps

Step 1: Problem Identification

Identified challenges in manual historical documentation and content writing.

Step 2: Idea Listing and Grouping

Generated multiple AI-related ideas and grouped them based on feasibility and impact.

Step 3: Idea Prioritization

Selected the historical artifact description generator based on practicality, innovation level, and technical viability.

3. Requirement Analysis

3.1. Solution Requirement

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Artifact Input Module	Enter artifact name / historical period & select word count
FR-2	Input Validation	Validate empty input & check word count limits

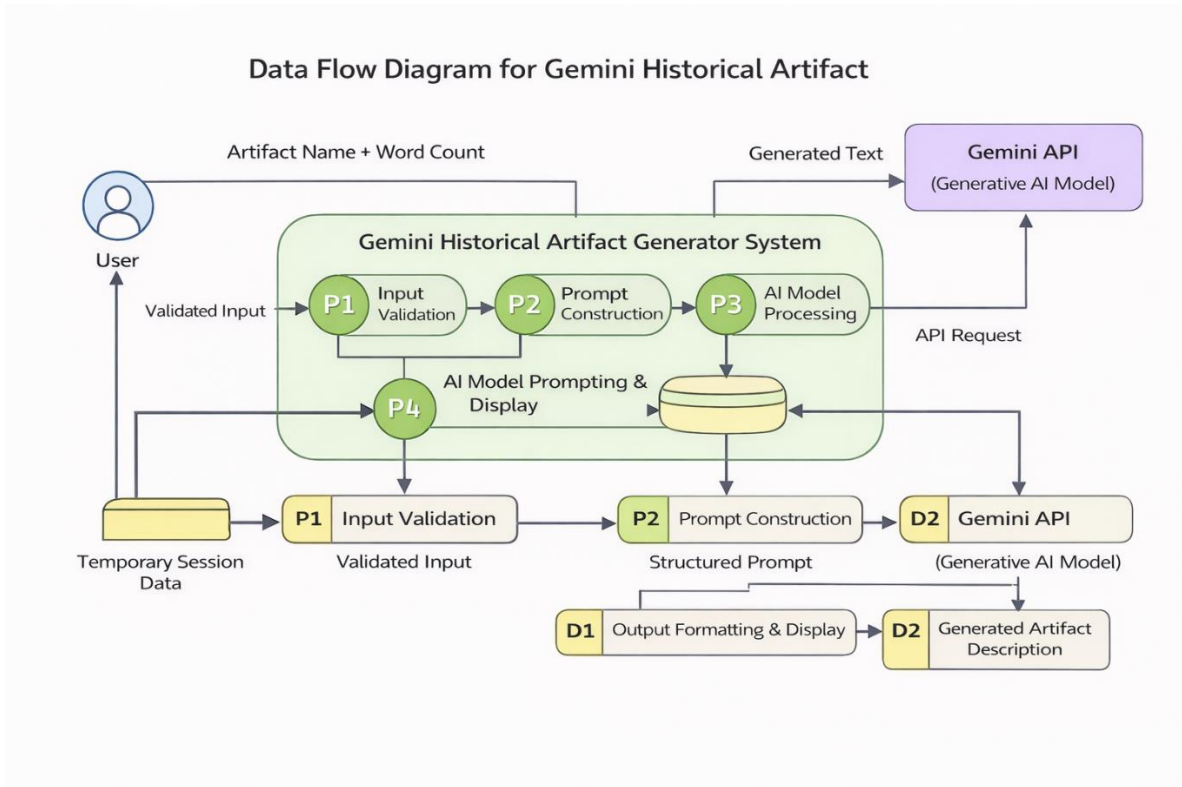| FR-3	AI Integration	Connect to Gemini Generative AI API and send structured prompt
FR-4	Description Generation	Generate structured historical artifact description
FR-5	Output Display	Display generated content in web interface
FR-6	Engagement Feature	Display an interesting historical fact during generation

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	The interface must be simple and intuitive for all types of users
NFR-2	Security	API keys must be securely stored using environment variables
NFR-3	Reliability	The system must generate structured and relevant descriptions for valid inputs
NFR-4	Performance	Description generation should complete within a few seconds
NFR-5	Availability	The application must be accessible when deployed online
NFR-6	Scalability	The system should handle multiple user requests efficiently

3.2. Data Flow Diagram

Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Data Flow Diagram for Gemini Historical Artifact

User Stories

User Type	Functional Requirement	User Story No	User Story	Acceptance Criteria	Priority	Release
Student / Researcher	User Interface Setup	USN-1	As a user, I want to enter an artifact name and word count easily.	I can access and input data successfully	High	Sprint-1
Administrator	Input Validation	USN-2	As a user, I want input validation before generation.	System prevents invalid inputs	High	Sprint-1
Research Professional	AI Integration	USN-3	As a user, I want the system to generate artifact descriptions using Gemini AI.	AI generates structured content	High	Sprint-2
Student	Historical Fact Feature	USN-4	As a user, I want to see an interesting historical fact while content loads.	Fact appears during processing	Medium	Sprint-2

User Type	Functional Requirement	User Story No	User Story	Acceptance Criteria	Priority	Release
All Users	Output Display	USN-5	As a user, I want to view the generated description clearly.	Output displayed properly in UI	High	Sprint-3
Administrator	Deployment	USN-6	As a user, I want the system deployed online.	Application accessible via browser	Medium	Sprint-3

3.3. Technology Stack

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1	User Interface	Web interface for entering artifact details	Streamlit (Python Framework)
2	Application Logic-1	Input validation and prompt construction	Python
3	Application Logic-2	AI API request handling and response processing	Google Generative AI API
4	Application Logic-3	Output formatting and structured display	Python
5	File Storage	Local storage for source code and logs	Local File System
6	External API	Generative AI service for description generation	Gemini Generative AI Model
7	Machine Learning Model	Pre-trained large language model	Gemini Model
8	Infrastructure	Application hosting and deployment	Streamlit Cloud / Local Hosting

Table-2: Application Characteristics:

S.No	Component	Description	Technology
1	User Interface	Web interface for entering artifact details	Streamlit (Python Framework)
2	Application Logic-1	Input validation and prompt construction	Python
3	Application Logic-2	AI API request handling and response processing	Google Generative AI API
4	Application Logic-3	Output formatting and structured display	Python
5	File Storage	Local storage for source code and logs	Local File System

4. PROJECT DESIGN

4.1. Problem Solution Fit

Problem – Solution Fit Template:

Problem–solution fit ensures that the developed system directly addresses the real challenges faced by users. Many students, historians, and researchers require detailed historical information for academic writing, museum documentation, and content creation. However, collecting and organizing this information manually requires significant time and effort.

The Gemini Historical Artifact system addresses this issue by providing an automated platform that generates structured descriptions instantly using generative AI. The solution aligns with user needs by offering customizable content length, accurate information flow, and quick generation time.

Purpose of Problem–Solution Fit

- To address difficulties in manual historical content creation

- To provide fast and structured artifact descriptions

- To improve productivity for students and researchers

- To enhance user experience through automation

- To provide easily accessible historical information

The proposed system effectively matches user requirements by delivering automated, reliable, and structured historical descriptions based on user input.

Template:

4.2. Proposed Solution

Proposed Solution Template:

Project team shall fill the following information in the proposed solution template.

S.No	Parameter	Description
1	Problem Statement	Students, researchers, and historians often spend excessive time researching and writing detailed descriptions of historical artifacts. Manual documentation requires effort, consistency, and proper formatting. There is a need for an automated system that can generate structured historical descriptions quickly.
2	Idea / Solution Description	Gemini Historical Artifact is an AI-powered web application that generates detailed descriptions of historical artifacts based on user input. Users enter an artifact name or historical topic and specify word count. The system uses Gemini Generative AI to produce structured and informative content instantly through a Streamlit-based interface.
3	Novelty / Uniqueness	Unlike traditional history websites that provide static content, this system dynamically generates customized descriptions in real time. Users can control the word count and receive unique, structured content instantly.
4	Social Impact / Customer Satisfaction	The application supports students, educators, historians, and bloggers by reducing research time and improving productivity. It provides quick access to structured historical information, enhancing learning and content creation.
5	Business Model	The system can be expanded using a freemium model where basic description generation is free and advanced features such as multi-language support, image generation, and premium templates are available through subscription.
6	Scalability	The application can be deployed on cloud platforms and scaled to handle multiple users. Additional features such as user accounts, database storage, and artifact image recognition can be integrated in future versions.

4.3 Solution Architecture

Solution Architecture:

      The solution architecture of Flavour Fusion consists of a Streamlit-based web application that collects user inputs and validates them. The application constructs a structured prompt and sends it to the Gemini Flash Lite generative AI model through an API call. The AI model generates a structured recipe blog, which is then formatted and displayed to the user. The system can be deployed locally or on a cloud platform such as Streamlit Cloud.

5. PROJECT PLANNING & SCHEDULING

5.1. Project Planning

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement	User Story No	User Story / Task	Story Points	Priority	Team Members
Sprint-1	UI Development	USN-1	Develop Streamlit interface for input	2	High	All members
Sprint-1	Input Validation	USN-2	Validate artifact name and word count	1	High	All members
Sprint-2	AI Integration	USN-3	Integrate Gemini API for description generation	3	High	All members

Sprint	Functional Requirement	User Story No	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Historical Fact Feature	USN-4	Display random historical fact during generation	1	Medium	All members
Sprint-3	Output Display	USN-5	Display generated description clearly	2	High	All members
Sprint-3	Deployment	USN-6	Deploy application on web platform	2	Medium	All members

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint 1	20	4 Days	28 January 2026	31 January 2026	20	31 January 2026
Sprint 1	20	4 Days	28 January 2026	31 January 2026	20	31 January 2026
Sprint 2	20	8 Days	02 February 2026	09 February 2026	20	09 February 2026
Sprint 2	20	8 Days	02 February 2026	09 February 2026	20	09 February 2026
Sprint 3	20	7 Days	12 February 2026	18 February 2026	20	18 February 2026
Sprint 3	20	7 Days	12 February 2026	18 February 2026	20	18 February 2026

6. FUNCTIONAL AND PERFORMANCE TESTING

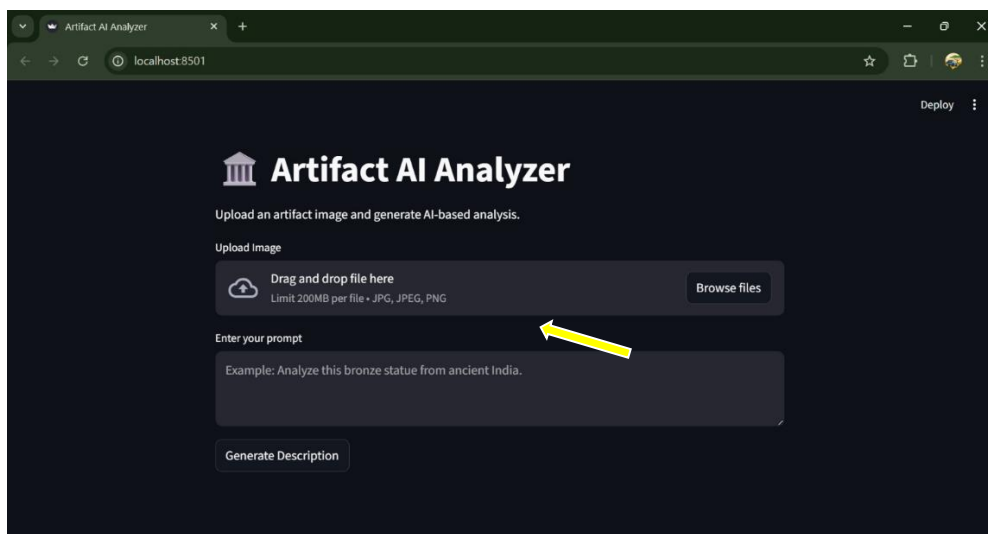## 6.1 Performance Testing

Test Scenarios & Results

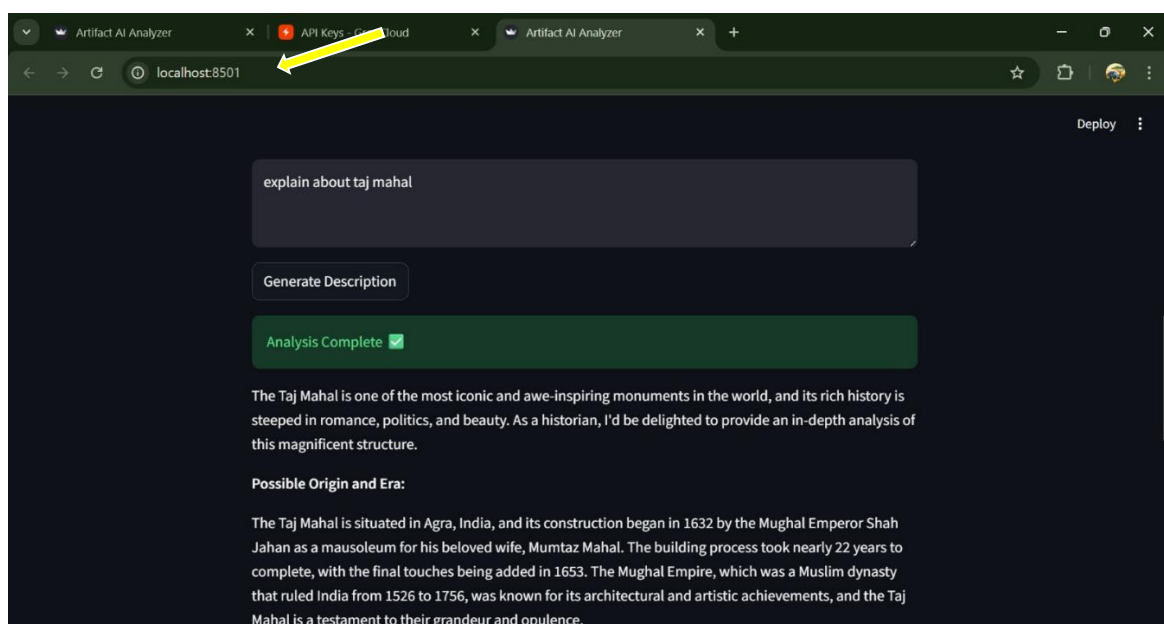| Test Case ID	Scenario	Test Steps	Expected Result	Actual Result	Status
FT-01	Input Validation	Enter valid and empty inputs	Valid accepted, empty rejected	As expected	Pass
FT-02	Word Count Validation	Enter different word limits	Accept valid range	As expected	Pass
FT-03	AI Description Generation	Click generate after input	Structured description generated	As expected	Pass
FT-04	API Connection	Use valid API key	Successful response	As expected	Pass
FT-05	Historical Fact Feature	Generate description	Fact displayed during generation	As expected	Pass
PT-01	Response Time	Measure generation time	Output within few seconds	Within limit	Pass
PT-02	Multiple Requests	Generate repeatedly	Stable performance	Stable	Pass
PT-03	Deployment Test	Access via browser	App runs correctly	Working	Pass

7. RESULTS

7.1. Output Screenshots

The complete execution of Flavour Fusion: AI-Driven Recipe Blogging application is represented step by step in the following screenshots.
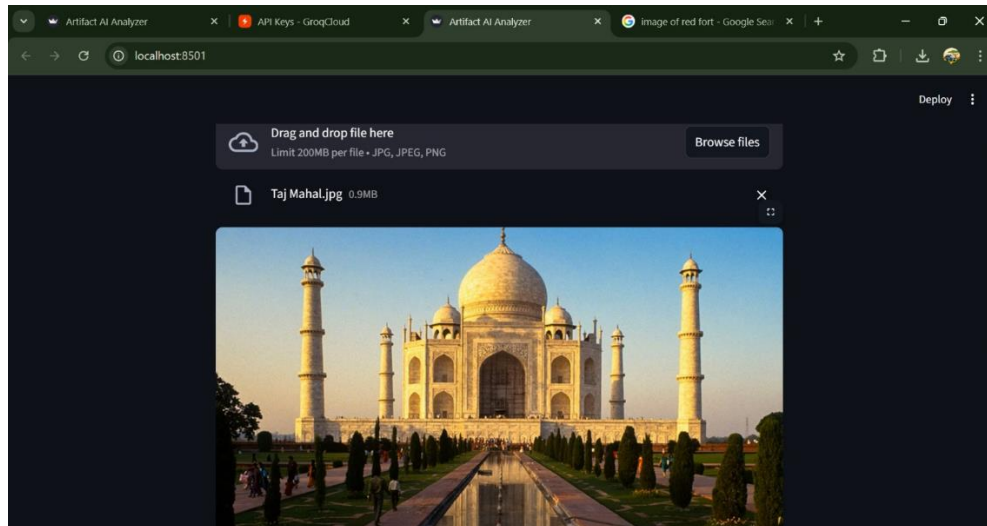
Step 1: To run the Streamlit Application we have to use the command streamlit run app.py in the terminal in path where the app.py file is located.
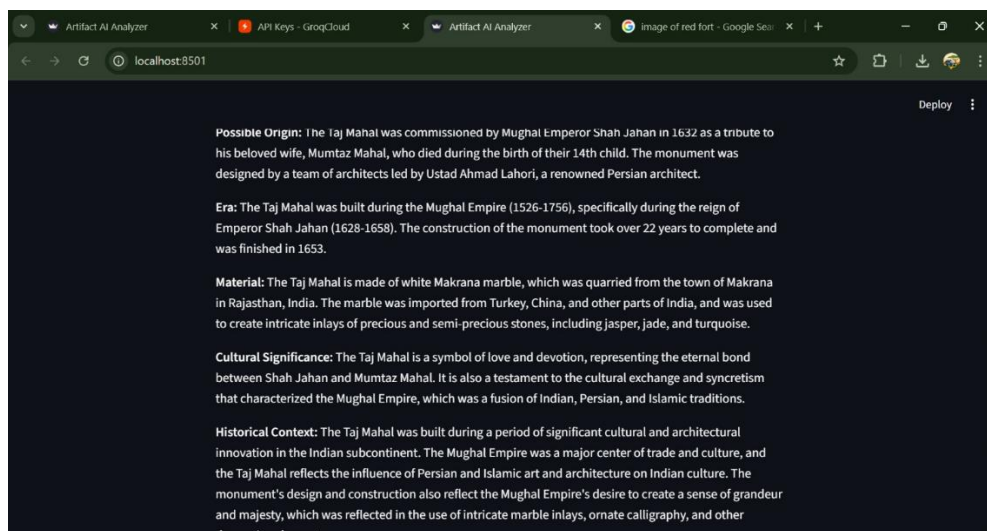

Step 2: After running the command in terminal, the code will get executed and the webpage will open directly. Another way to open webpage is that a localhost link will get generated in the terminal, we can access the webpage using that link.

Step 3: The Streamlit webpage opens as shown in the figure given below. This is an automated webpage. No secondary HTML codes required to build this webpage. Python code itself consists the webpage building code.

Step 3: The user has to give inputs in the website such as a Recipe Name and Number of Words. The Number of words means with how many words the recipe should be generated. After entering the required details, the user should click on Generate Recipe button to generate the recipe. Here I chose the Chocolate Cake Recipe with 400 words count.

Step 4: After clicking the Generate Recipe button, in fraction of seconds a simple joke will be generated as shown in the below figure to engage the users if recipe generation get delayed. The Food Recipe will be generated based on the user inputs as shown in the following four images.

8. ADVANTAGES AND DISADVANTAGES

Advantages

- Reduces time required for researching and writing historical descriptions

- Automatically generates structured and informative artifact content

- Provides customized output based on user input and word count

- Simple and user-friendly interface built using Streamlit

- Fast and efficient content generation using Gemini Generative AI

- No need for dataset collection or manual training of models

- Useful for students, historians, researchers, and bloggers

- Can be accessed online when deployed on cloud

Disadvantages

- Requires internet connection for AI model access

- Depends on third-party API services for content generation

- Output accuracy depends on quality of user input

- Limited to text-based descriptions in current version

- Advanced features require further development

9. CONCLUSION

The Gemini Historical Artifact project demonstrates the effective use of generative artificial intelligence in automating historical content creation. The system allows users to generate well-structured and detailed descriptions of historical artifacts by simply entering the artifact name and required word count. This reduces manual research effort and saves significant time for students, researchers, and historians.

By integrating the Gemini Generative AI model with a Streamlit-based interface, the application provides fast and reliable results while maintaining simplicity and usability. The project successfully highlights how modern AI tools can be utilized in educational and research domains to enhance productivity and accessibility of information.

Overall, the developed system provides an efficient and scalable solution for generating historical artifact descriptions and can be further enhanced with additional intelligent features in the future.

10. FUTURE SCOPE

The Gemini Historical Artifact project can be expanded with several advanced features to improve functionality and user experience. Future enhancements may include support for multiple languages, allowing users from different regions to generate descriptions in their preferred language. Image-based artifact recognition can also be integrated, enabling users to upload artifact images and receive detailed descriptions automatically.

Additional improvements such as voice input support, user login systems, cloud database storage, and mobile application deployment can make the system more interactive and accessible. Integration with virtual museum platforms and augmented reality applications could further extend the usability of the system in educational and cultural domains.

With continuous development and integration of advanced AI capabilities, the system has strong potential to evolve into a comprehensive digital history assistant for academic and professional use.

11. APPENDIX

11.1. Source Code

The source code of the Gemini Historical Artifact project includes the implementation of the Streamlit user interface, integration of the Gemini Generative AI model using the Google Generative AI API, input validation module, description generation logic, and historical fact feature.

The application is developed using Python programming language and follows a modular structure for easy maintenance and scalability.

11.2 GitHub & Demo

11.2. GitHub & Project Demo Link

GitHub Repository Link: https://github.com/Sravs78/Gemini-Historical-Artifact-Description

Demo Link:
https://drive.google.com/file/d/1D6UwxBjJPlbXR1wBCFyA_2xynSX5259N/view?usp=driv esdk