Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	6 February 2026
Team ID	LTVIP2026TMIDS91648
Project Name	Gemini Historical Artifact Description
Maximum Marks	4 Marks

Technical Architecture: The Deliverable includes the architectural diagram as below and the information as per the table1 & table 2

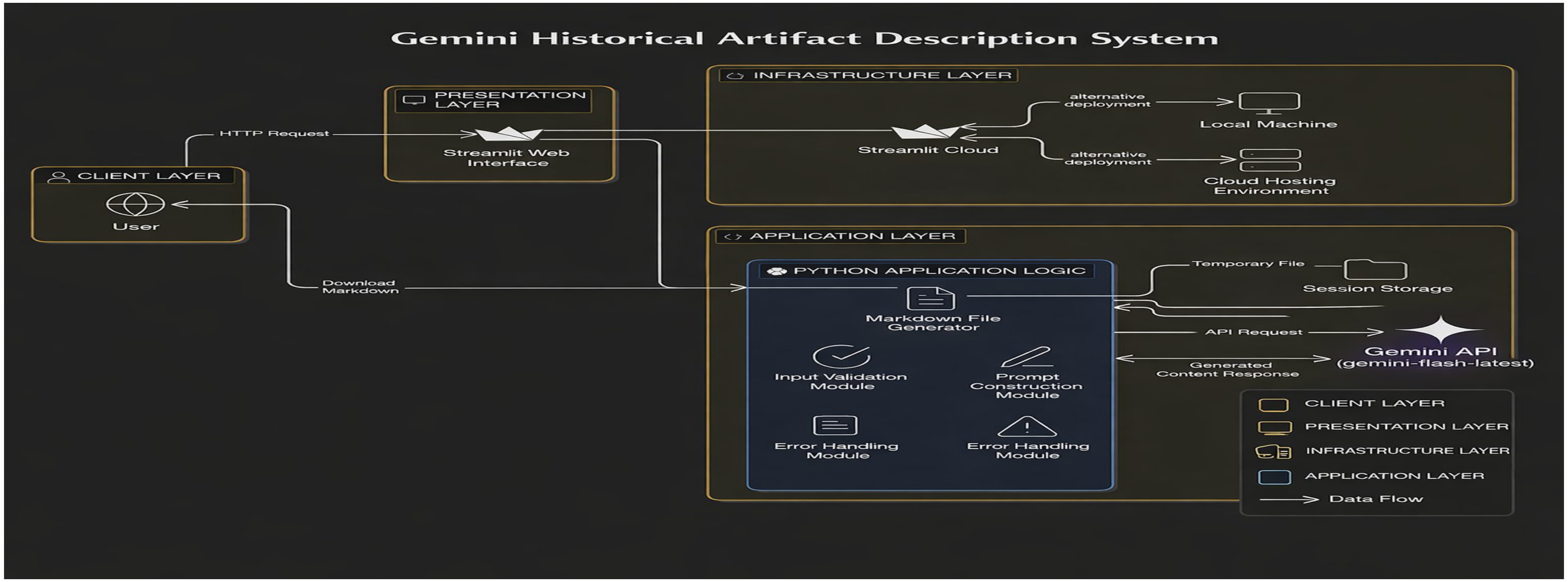

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1. 1	User Interface	Web-based interface where users enter Artifact Name, Historical Period, select word count, view generated description, and download Markdown file	Streamlit (Python-based Web UI)
2. 2	Application Logic-1	Handles user input, validation, and workflow control	Python
3. 3	Application Logic-2	Prompt construction logic for structured historical artifact description	Python (Prompt Engineering Logic)
4.	Application Logic-3	AI content generation logic via external model integration	Google Gemini API (gemini-flash-latest)
5.	Database	No persistent database used (stateless application)	Not Applicable
6.	Cloud Database	Not used in current implementation	Not Applicable
7.	File Storage	Temporary generation of Markdown file for download	Local Runtime (Streamlit Session State)
8.	External API-1	AI text generation for structured historical artifact descriptions	Google Gemini API
9.	External API-2	Not used	Not Applicable
10.	Machine Learning Model	Large Language Model used for generative artifact descriptions	Gemini Flash Model
11.	Infrastructure (Server / Cloud)	Application can be deployed locally or on cloud hosting platforms	Local System / Streamlit Cloud / Any Python-supported Cloud Environment

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Framework used to build and deploy the web interface	Streamlit (Open-source Python framework)
2.	Security Implementations	API key secured via environment variables or Streamlit secrets; no sensitive user data stored	Streamlit Secrets / Environment Variables
3.	Scalable Architecture	Stateless architecture; scalability depends on hosting environment and Gemini API limits	Streamlit + Cloud Hosting (Horizontal scaling possible via cloud deployment)
4.	Availability	Application availability depends on hosting platform uptime	Streamlit Cloud / Cloud Hosting Provider
5.	Performance	AI response time depends on Gemini API latency; lightweight UI ensures minimal frontend delay	Python + Gemini API