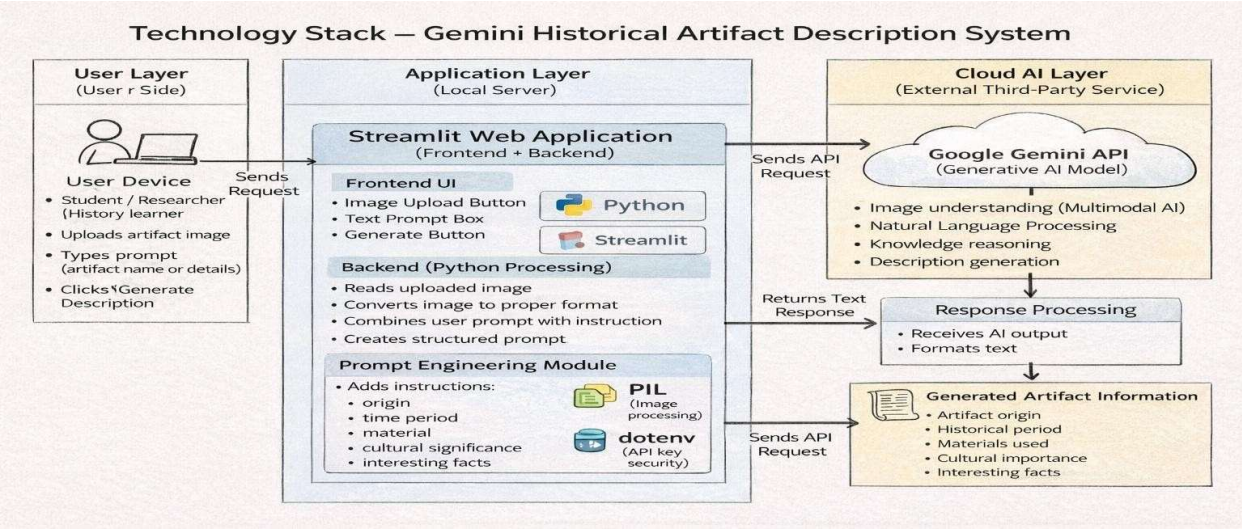
Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	2 February 2026
Team ID	LTVIP2026TMIDS74087
Project Name	Gemini Historical Artifact Description
Maximum Marks	4 Marks

Technical Architecture:

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

Example:



Guidelines:

1. **Include all main components:**

   Show every part of the system: User (student), Web Application (Streamlit), Backend (Python), and AI model (Gemini API).

2. **Separate Local and Cloud clearly:**
   - Local → Streamlit app + Python processing (your laptop/server)
   - Cloud → Google Gemini AI (external service)

3. **Show external API connection:**

   Draw an arrow from your application to the Gemini API to represent sending image + text prompt and receiving generated description.

4. **Show data flow direction:**

   Use arrows to explain the process:

   Upload Image → Processing → AI Model → Response → Display Result.

5. **Include data storage (if used):**

   Mention where temporary images, prompts, or results are stored (local memory / session state).

6. **Mention AI model usage:**

   Clearly indicate the system communicates with a Machine Learning model (Gemini) for:
   - image understanding
   - natural language generation

7. **Explain input and output:**
    - Input → artifact image + user text prompt
    - Output → historical description (origin, period, materials, facts)
8. **Keep diagram simple:**

    Do NOT draw too many boxes. Only important blocks:

    User → Web App → Python Processing → Gemini API → Result

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	User uploads artifact image and enters prompt through web page.	Streamlit Web UI (Python), HTML, CSS.
2.	Application Logic-1	Handles image upload, validation, and preprocessing.	Python (Streamlit + PIL library).
3.	Application Logic-2	Converts image into suitable format and prepares input prompt.	Python Image Processing & Prompt Engineering.
4.	Application Logic-3	Sends request to AI model and receives generated description.	Google Generative AI SDK (Gemini API).
5.	Database	Temporary storage of session data and prompts.	Streamlit Session State / Local Memory (No permanent DB used).
6.	Cloud Database	Not required — system generates response dynamically.	Not Applicable (Real-time AI response).
7.	File Storage	Stores uploaded artifact images temporarily.	Local File System / RAM (Temporary Storage).
8.	External API-1	AI model used to generate artifact description.	Google Gemini Pro Vision API.

9.	External API-2	API key authentication and request handling.	Google AI Studio API Key Service.
10.	Machine Learning Model	Understands artifact image and generates historical explanation.	Gemini Multimodal Generative AI Model.
11.	Infrastructure (Server / Cloud)	Application runs locally and connects to cloud AI service.	Local System (Python environment) + Google Cloud AI Service.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	The application is developed using open-source Python libraries for UI, image handling, and API communication. It allows easy development and modification without license cost.	Python, Streamlit, Pillow (PIL), Requests, dotenv.
2.	Security Implementations	The system protects API access using a private API key stored in environment variables. Only authorized requests are allowed to communicate with the AI model. User data is not permanently stored.	Google API Key Authentication, .env file, HTTPS secure connection.
3.	Scalable Architecture	The application follows a lightweight client-server architecture. The frontend (Streamlit UI) runs locally while heavy processing is handled by the cloud-based Gemini AI model, allowing multiple users without heavy local hardware.	Google API Key Authentication, .env file, HTTPS secure connection.
4.	Availability	The AI service runs on Google Cloud infrastructure, so the system remains available whenever internet connection exists. No local server dependency for AI computation.	Google Cloud AI Services (High Availability Cloud Servers).