# Full Stack Development Project Documentation

# 1. Introduction

### • Project Title:

AI Historical Artifact Description App

### • Team Members:

P Poojitha Sai

Poli Nagajagadeshwar Reddy – Full Stack Developer & AI Integration

P Venu gopal

# 2. Project Overview

### • Purpose

The purpose of this project is to develop an AI-powered web application that analyzes historical artifact images and generates detailed historical descriptions using transformer-based machine learning models.

### • Features

Image upload functionality (JPG, PNG)

Image preview display

AI-based image caption generation

Structured historical description generation

Minimum 10+ sentence output

Fully offline operation

Clean and responsive web interface

# 3. Architecture

Although not MERN, the project follows a layered full-stack architecture.

### • Frontend (UI Layer)

Built using Streamlit

Handles image upload and display

Provides interactive “Generate Description” button

Displays formatted AI-generated output

### • Backend (Application Logic Layer)

Written in Pytho

Handles image preprocessing

Integrates AI models

Controls caption generation and text expansion

### • Database

No external database used

Pre-trained transformer models are used

Model caching handled locally

# 4. Setup Instructions

### • Prerequisites

Python 3.10+

pip

Internet connection (first-time model download)

### • Installation

Clone the repository

git clone https://github.com/YOUR\_USERNAME/Gemini-Historical-Artifact-Description.git  
cd Gemini-Historical-Artifact-Description

Install dependencies

pip install -r requirements.txt

Run application

streamlit run app.py

# 5. Folder Structure

Gemini-Historical-Artifact-Description/  
│  
├── app.py  
├── requirements.txt  
├── README.md  
├── screenshots/  
└── models (cached automatically)

# 6. Running the Application

Start the application locally:

streamlit run app.py

The application will run at:

http://localhost:8501

# 7. API Documentation

This project does not expose REST APIs.  
Instead, it uses internal model inference pipelines:

### AI Pipeline:

Image Upload

BLIP Image Caption Model

FLAN-T5 Language Model

Structured Historical Description Output

# 8. Authentication

Authentication is not implemented in this version.  
Future versions can integrate JWT-based authentication for multi-user access.

# 9. User Interface

UI includes:

Image Upload Section

Image Preview

Generate Button

AI Output Section

(Screenshots included in GitHub repository)

# 10. Testing

Testing was conducted through:

Functional Testing (Upload, Output generation)

Performance Testing (Response time measurement)

Edge Case Testing (Invalid file formats)

Manual Validation of AI output quality

# 11. Screenshots or Demo

Screenshots included in GitHub:

Application Interface

Uploaded Artifact

Generated Historical Description

Demo: Local Streamlit deployment.

# 12. Known Issues

Initial model download takes time (~2–5 minutes)

High RAM usage during first run

Output depends on generalization of pre-trained models

# 13. Future Enhancements

Cloud deployment (AWS/Render

User authentication

Multilingual support

Database integration

PDF export feature

Fine-tuned domain-specific historical model