Google Cloud Generative AI

Project Documentation

1. Introduction

- **Project Title: Gemini Pro Financial Decoder**
- **Team Members:**

Member 1	Frontend & UI Developer	Designing Streamlit user interface, file upload sections, buttons, layout, and displaying summaries and charts
Member 2	Backend & Data Processing Developer	Implementing file reading, data validation, data processing using Pandas, and handling CSV/XLSX files
Member 3	AI Integration & Logic Developer	Integrating Google Gemini API, implementing AI summary generation, handling API quota issues, and implementing fallback local summary logic
Member 4	Testing & Documentation Lead	Performing functional testing, user acceptance testing, performance testing, preparing documentation, diagrams, and final report

---

2. Project Overview

Purpose

The purpose of this project is to simplify the analysis of financial statements using Generative AI. The system helps non-finance users understand Balance Sheets, Profit & Loss Statements, and Cash Flow Statements through easy-to-read summaries and visualizations.

Features

- **Upload financial statements in CSV or Excel format**
- **Automated financial data processing**
- **AI-generated financial summaries using Google Gemini**
- **Local rule-based summary fallback when API quota is exceeded**
- **Interactive charts and tabular visualization**
- **Simple and user-friendly web interface**

---

3. Architecture

Frontend

The frontend is developed using Streamlit, a Python-based web framework. It provides components for file upload, buttons, text display, and charts, enabling quick interaction with users.

Backend

The backend logic is implemented using Python. It handles file validation, data processing using Pandas, prompt preparation, AI invocation, and fallback summary logic.

Database

No persistent database is used. Uploaded files are processed in-memory using Pandas DataFrames to ensure simplicity and data privacy.

---

4. Setup Instructions

Prerequisites

- **Python 3.9 or above**
- **pip (Python package manager)**
- **Internet connection (for Gemini API)**

Installation

1. **Clone the project repository**
2. **Install required libraries using:**
3. **pip install streamlit pandas google-generativeai**
4. **Add Google Gemini API key in the application code**
5. **Run the application using Streamlit**

---

5. Folder Structure

- **app.py – Main Streamlit application file**
- **requirements.txt – Python dependencies**
- **sample_data/ – Sample CSV/Excel financial**

Category	Item	Amount
Assets	Cash	120000
Assets	Bank Balar	350000
Assets	Accounts F	210000
Assets	Inventory	180000
Assets	Prepaid Ex	40000
Assets	Short Term	90000
Assets	Plant and N	750000
Assets	Office Equi	130000
Assets	Furniture	85000
Assets	Intangible /	95000
Liabilities	Accounts F	160000
Liabilities	Short Term	120000
Liabilities	Accrued Ex	70000
Liabilities	Long Term	450000
Liabilities	Deferred T	60000
Equity	Share Capi	500000
Equity	Retained E	315000
Equity	Reserves	135000

---

6. Running the Application

To start the application locally:

streamlit run app.py

The application will be accessible in the browser at:

http://localhost:8501

---

7. API Documentation

External API Used

- **Google Gemini Generative AI API**

Purpose

- **Generate financial summaries and explanations from processed financial data.**

Request Type

- **Text-based prompt sent using Python SDK**

Response

- **Natural language financial summary**

---

8. Authentication

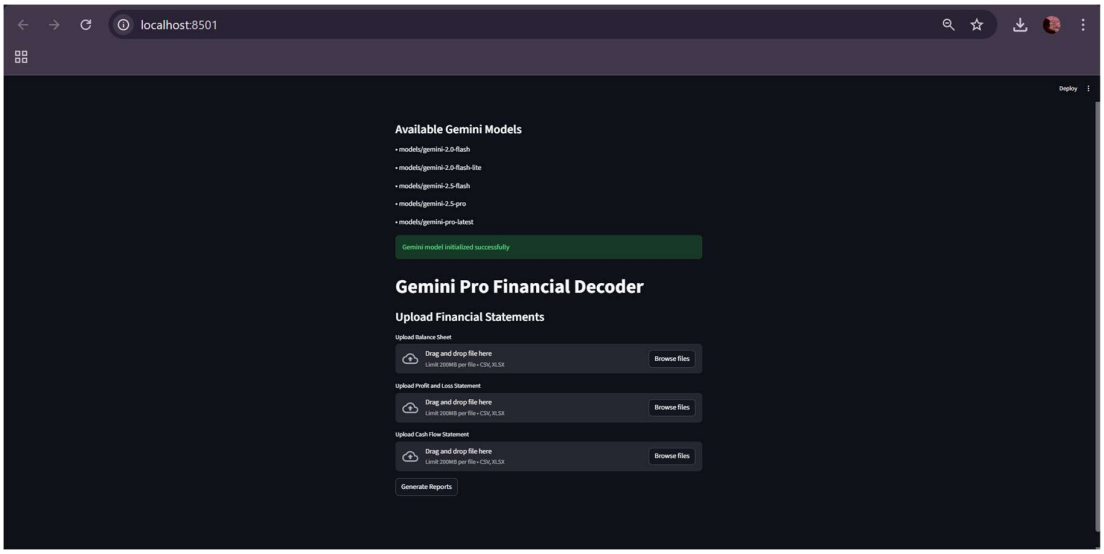
The application uses API key-based authentication for accessing Google Gemini services.
No user login or session management is implemented, as this is an academic prototype focused on functionality rather than user accounts.

---

9. User Interface

The user interface consists of:

- **File upload sections for financial statements**
- **A “Generate Reports” button**
- **Summary sections for each statement**
- **Graphical visualizations using charts**



---

10. Testing

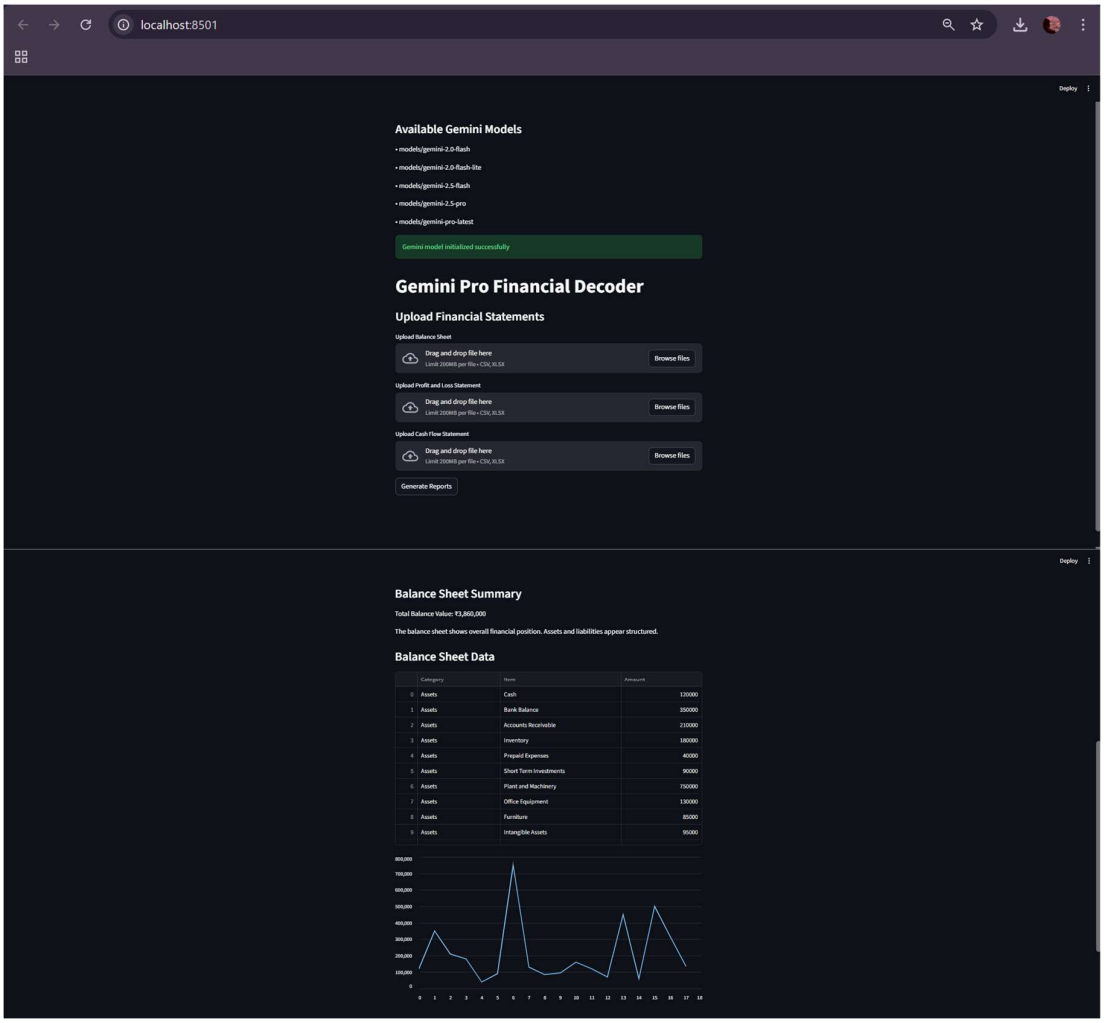
The project was tested using:

- **Functional testing for file uploads and report generation**
- **User Acceptance Testing (UAT) to validate end-user workflows**
- **Performance testing for API quota handling and fallback logic**

All test cases passed successfully.

---

11. Screenshots or Demo



---

12. Known Issues

- **AI summary generation depends on Gemini API quota**
- **When quota is exceeded, AI responses are replaced by local summaries**

This limitation is handled gracefully without application failure.

---

13. Future Enhancements

- **Export reports as PDF files**
- **Multi-language summary support**

- **Advanced financial ratio analysis**
- **Cloud deployment for multi-user access**
- **Role-based user authentication**