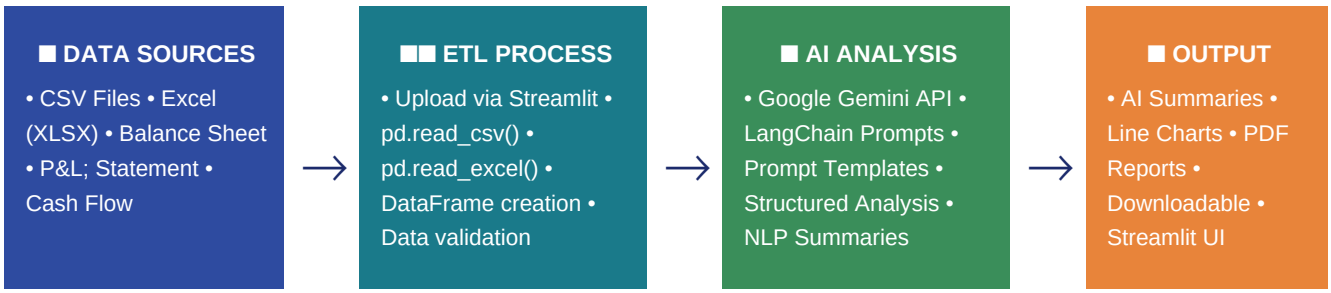
Project Design Phase-II

Technology Stack (Architecture & Stack)

Date	19 February 2026
Team ID	LTVIP2026TMIDS49230
Project Name	Gemini Pro Financial Decoder
Maximum Marks	4 Marks

Technical Architecture:

The Gemini Pro Financial Decoder follows a layered architecture — from raw data input through AI processing to final output delivery. Each layer uses purpose-built technologies to ensure efficiency and accuracy.



Technology Stack Components:

S.No	Component	Description	Technology
1	**User Interface**	How users interact with the financial analysis application.	**Streamlit (Web App Framework)**
2	**Data Sources**	Where raw financial data originates and is uploaded from.	**CSV Files, Excel (XLSX), Local File System**
3	**Data Processing & ETL**	Processes for loading, cleaning, and transforming uploaded financial data.	**Python, Pandas (pd.read_csv, pd.read_excel)**
4	**AI / NLP Engine**	Core AI engine for analyzing financial data and generating insights.	**Google Gemini API (google.generativeai)**

5	**Prompt Management**	Framework for managing and formatting AI prompt templates.	**LangChain (langchain.prompts, langchain_google_genai)**
6	**Visualization Engine**	Software used to create and display financial charts and graphs.	**Streamlit Charts (st.line_chart, st.bar_chart)**
7	**PDF Report Generation**	Library used to auto-generate downloadable structured PDF reports.	**ReportLab (Python PDF Library)**
8	**Security / Config**	Mechanisms to securely store API keys and sensitive configuration.	**Python dotenv, Environment Variables (.env file)**
9	**Version Control**	Source code management and project deployment pipeline.	**Git & GitHub**

Application Characteristics:

S.No	Characteristic	Description	Technology
1	**Open-Source Frameworks**	Open-source technologies used within the solution.	**Python, Streamlit, Pandas, LangChain, ReportLab**
2	**Security Implementations**	Measures to protect API keys and user-uploaded data.	**Environment Variables (.env), No persistent data storage**
3	**Scalable Architecture**	How the system handles growing document sizes and user load.	**Modular Streamlit app, Stateless API calls, Cloud deployment-ready**
4	**Availability**	Ensuring the application is consistently accessible.	**Streamlit Cloud / Local server, Graceful API error handling**
5	**Performance**	Speed and responsiveness of AI analysis and report generation.	**Gemini API streaming, Pandas in-memory processing, ReportLab**