

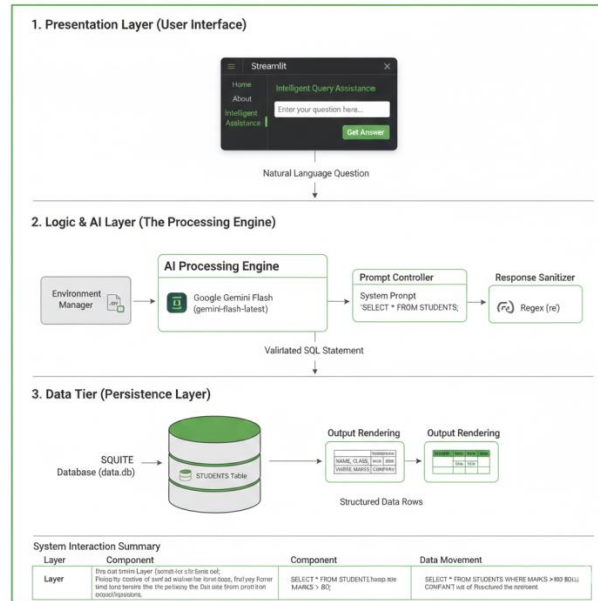
## Project Design Phase-II

### Technology Stack (Architecture & Stack)

Date	15 February 2026
Team ID	LTVIP2026TMIDS65940
Project Name	IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro
Maximum Marks	4 Marks

#### Technical Architecture:

- Presentation: Managed by Streamlit with custom CSS accents (#4CAF50) for a modern appearance.
- Logic Execution: Orchestrated by Python, ensuring that the database connection is safely opened and closed during each query.
- Intelligence: Powered by the Gemini Flash model to ensure the "Intelligent Query Assistance" remains fast and reliable.



**Table-1 : Components & Technologies:**

S.No	Component	Description	Technology
1.	User Interface	A web-based dashboard with a professional dark theme that allows users to navigate between Home, About, and the Query Tool.	Streamlit, Custom CSS
2.	Application Logic-1	Core backend processing for environment setup, model configuration, and coordinating data flow between the UI and LLM.	Python
3.	Application Logic-2	Natural language processing logic to translate English questions into structured SQL queries.	<b>Google Gemini Flash API</b> (gemini-flash-latest)
4.	Application Logic-3	Data cleaning and sanitization logic to extract valid SQL from conversational AI responses.	Python Regex (re)
5.	Database	Relational database containing the STUDENTS table with columns for Name, Class, Marks, and Company.	<b>SQLite</b> (data.db)
6.	Cloud Database	N/A - The current project utilizes a local database setup for rapid development.	Local SQLite
7.	File Storage	Local storage for database files and environment configuration.	Local Filesystem
8.	External API-1	Primary generative AI interface used to perform the conversion of text to SQL.	Google Generative AI SDK
9.	External API-2	Secure management of environment variables to protect the system's API keys.	python-dotenv
10.	Machine Learning Model	High-speed Large Language Model (LLM) optimized for fast inference and accurate code generation.	Gemini 1.5 Flash
11.	Infrastructure (Server / Cloud)	Deployment configuration for local testing and interactive web hosting.	Local System / Streamlit Cloud

**Table-2: Application Characteristics:**

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	The project utilizes a high-performance open-source framework for building the web interface. It also relies on environmental management libraries to handle configuration safely.	Streamlit, Python-dotenv
2.	Security Implementations	Access control is managed by storing the sensitive <code>GOOGLE_API_KEY</code> in a local environment file rather than the source code. A <code>.gitignore</code> file prevents these credentials from being exposed in public repositories	Environment Variable Masking (.env)
3.	Scalable Architecture	The application follows a modular 3-tier architecture—separating the <b>UI Layer</b> , <b>AI Logic Layer</b> , and <b>Database Layer</b> . This allows for the AI model to be upgraded or the database to be migrated with minimal impact on other components.	3-Tier Architecture (Python/Gemini/SQLite)
4.	Availability	As a local executable or cloud-hosted Streamlit app, the tool is available as long as the host server is active and the Google AI services are reachable.	3-Tier Architecture (Python/Gemini/SQLite)
5.	Performance	Performance is optimized by using the <b>Gemini 1.5 Flash</b> model, specifically chosen for its low-latency response times and efficient token processing for code generation. Data retrieval is kept fast by using a lightweight SQLite local store.	Gemini 1.5 Flash, SQLite Indexed Storage