Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	8 February 2026
Team ID	LTVIP2026TMIDS66048
Project Name	IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro
Maximum Marks	4 Marks

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	System Configuration	Secure API Key loading via .env environment variables. Initialization of Google Generative AI with the gemini-flash-latest model.
FR-2	Database Management	Local SQLite database (data.db) creation and connection. Definition of the STUDENTS table schema (NAME, CLASS, MARKS, COMPANY). Initial data seeding for testing purposes.
FR-3	Natural Language Processing	Implementation of a "System Prompt" to guide the LLM's SQL generation. Translation of natural language English questions into executable SQL queries.
FR-4	Query Sanitization	Regex-based extraction of raw SQL strings from AI conversational output. Filtering out markdown tags or non-SQL text to prevent execution errors.
FR-5	Data Retrieval & Display	Execution of generated SQL queries against the SQLite engine. Rendering of retrieved datasets into interactive Streamlit tables.
FR-6	User Interface & Navigation	Multi-page sidebar navigation (Home, About, Query Tool). Custom CSS theme application (Dark background with green accents). Interactive "Get Answer" trigger buttons and text input fields.
FR-7	Error Handling	Implementation of try-except blocks for database connection failures. Validation to detect if a valid SQL query was successfully generated.

Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	The application shall feature a professional dark-themed UI with custom CSS (#2E2E2E and #4CAF50) to provide an "effortless and intuitive" user experience. It must include sidebar navigation to allow users to switch between tools and documentation without confusion.
NFR-2	**Security**	Sensitive credentials, specifically the GOOGLE_API_KEY, must be stored in a .env file and never hardcoded in the source files. The .gitignore file must be configured to prevent accidental leakage of this API key to public repositories.
NFR-3	**Reliability**	The system must utilize Regex-based sanitization (re.search) to ensure that only valid SQL queries are passed to the database, preventing application crashes caused by non-SQL text in LLM responses.
NFR-4	**Performance**	The application shall leverage the **Gemini Flash** model to ensure low-latency natural language processing and rapid SQL generation for a "real-time" querying experience.
NFR-5	**Availability**	As a local Streamlit application, the tool must remain accessible as long as the Python environment is active and the Google API service is reachable.
NFR-6	**Integrity**	The read_query function must maintain database integrity by closing the SQLite connection immediately after fetching results to prevent file locking or data corruption.
NFR-7	**Scalability**	The current architecture, using a modular GenerativeModel setup, should allow for easy upgrades to newer or more powerful LLM versions (e.g., transitioning from Flash to Pro) with minimal code changes.