Ideation Phase
Brainstorm & Idea Prioritization Template

Date	14 February 2026
Team ID	LTVIP2026TMIDS66015
Project Name	IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro
Maximum Marks	4 Marks

Brainstorm & Idea Prioritization Template:

Project Overview: IntelliSQL is an innovative platform designed to revolutionize how users interact with databases by leveraging the **Google Gemini Flash** architecture. By translating natural language into structured SQL queries, the project removes the technical barriers associated with database management, specifically for tracking student data such as marks and company placements.

The Creative Goal: The brainstorming process for IntelliSQL focused on creating a "zero-friction" environment for data retrieval. We prioritized developing a system that not only understands intent but also ensures high security through **environment variable protection** and **regex-based query sanitization** to prevent execution errors.

Step-1: Team Gathering, Collaboration and Select the Problem Statement

Team Gathering & Collaboration To ensure a successful development cycle, the team focused on a cross-functional approach involving UI/UX design and AI integration. We utilized a collaborative stack consisting of **Streamlit** for the frontend, **Python** for the backend logic, and **Google Gemini Flash** as the core intelligence engine. Preparation involved setting up a secure environment using .env files and .gitignore to protect sensitive API credentials before starting the build.

Set the Goal The primary objective of the IntelliSQL session was to create a platform that revolutionizes database querying by offering an intelligent, intuitive interface for interacting with SQL databases. We aimed to develop a system capable of interpreting natural language to retrieve records from a STUDENTS table, which includes attributes like Name, Class, Marks, and Company.

Define Your Problem Statement

- **Problem:** Non-technical users often struggle to extract data from relational databases because they do not know SQL syntax.
- **How Might We Statement:** "How might we create an intelligent assistant that allows users to ask questions in plain English and automatically generates and executes the corresponding SQL queries to provide instant data insights?"

Step-2: Brainstorm, Idea Listing and Grouping

Brainstorm & Idea Listing During the ideation phase, we listed several features and technical requirements needed to make the project viable:

- **Dynamic SQL Generation:** Leveraging the gemini-flash-latest model to handle real-time English-to-SQL translation.
- **Database Foundation:** Creating a lightweight SQLite data.db to store student metrics like names, classes, and marks.
- **Safety Measures:** Implementing a strict "SQL-only" response prompt to ensure the LLM doesn't return unnecessary conversational text.
- **Visual Interface:** Building a dark-themed, professional dashboard using Streamlit to display query results in an organized table format.
- **Security Protocols:** Storing API credentials in a .env file and excluding it from version control via .gitignore.

Group Ideas We clustered these ideas into three distinct groups to streamline the development process:

- **Group 1: The Intelligent Engine (AI Logic)**
    - System prompting for "Expert SQL Converter" persona.
    - Regex extraction to isolate the generated SELECT statements from the LLM output.
    - Generative model configuration and API integration.
- **Group 2: The Data Layer (Storage & Retrieval)**
    - SQL table creation for the STUDENTS schema.
    - Development of the read_query function to execute SQL safely via sqlite3.
    - Initial data seeding with sample records for testing.
- **Group 3: User Interaction (UI/UX)**
    - Multi-page navigation for "Home," "About," and "Intelligent Query Assistance".
    - Custom CSS styling with green accents (#4CAF50) for a modern look.
    - Interactive tooltips and buttons to guide users during the querying process.

Step-3: Idea Prioritization

Prioritization Strategy The team utilized a grid to determine which features of **IntelliSQL** should be prioritized for the final executable phase. We focused on ideas that maximized user accessibility while maintaining the high performance of the **Gemini Flash** model.

The Prioritization Grid

- **High Importance / High Feasibility (Immediate Implementation)**
    - **Natural Language to SQL Translation:** The core function of converting user prompts into SELECT statements using Gemini.

    - **SQLite Integration:** Using a lightweight local database (data.db) to ensure fast response times and easy setup.
    - **Regex SQL Cleaning:** Ensuring the application remains stable by stripping away any conversational text from the LLM response.
- **High Importance / Medium Feasibility (Future Scope)**
    - **Complex Trend Analysis:** Expanding the prompt logic to handle advanced statistical queries across multiple database tables.
    - **Performance Optimization:** Suggesting more efficient SQL syntax to the user for large-scale datasets.
- **Low Importance / High Feasibility (Polishing)**
    - **Custom Dark Theme UI:** Applying specific CSS styles like #2E2E2E and green accents to enhance the professional feel.
    - **Sidebar Navigation:** Organizing the "Home," "About," and "Tool" sections for a cleaner user flow.

Group Ideas Conclusion By clustering similar notes, we identified that the **Intelligent Query Assistance** module was the most critical cluster for the project's success. Larger ideas were broken down into smaller sub-groups, such as separating the **Database Initializer (sql.py)** from the **Main Application (app.py)** to maintain clean code architecture.