

PROJECT DOCUMENTATION

Project Title:

IntelliSQL – An Intelligent SQL Querying System

Team Details

Team ID: LTVIP2026TMIDS88549

Team Size : 4

Team Leader : Hima Varsha Palle

Team Member : Basetty Niharika

Team Member : Thupakula Kavya

Team Member : Nazeem Hussain Patan

GitHub Repository:

<https://github.com/HimaVarshaPalle/IntelliSQL>

Phase 1: Brainstorming & Ideation

IntelliSQL

Brainstorm & idea prioritization

Use this template to brainstorm features and improvements for the IntelliSQL system — an intelligent NL-to-SQL querying tool using Google Gemini Pro and Streamlit.

10 minutes to prepare
1 hour to collaborate
1 team member

IntelliSQL — Brainstorm & idea prioritization

Use this template to brainstorm features and improvements for IntelliSQL to enhance intelligent SQL querying and natural language database interaction.

TIP
If you have a Slack to notify your team, send out copies of facilitation content and tools during session.

1 Before you collaborate

A bit of preparation goes a long way with this session. Here's what you need to do.

⌚ 10 minutes

A Team gathering
Define the relevant participant in the session and set the meeting to prevent scheduling conflicts ahead of time.

B Send out the agenda and problem to be solved
Read and understand the session goals; set devices to "do not disturb" to dedicate focus.

C Learn how to use the facilitation tools
Review the facilitation templates and tools to ensure you're familiar with them beforehand.

D Problem Statement
Define a clear problem — users selecting scope, impact and identifying core layers undetermined.

TIP
If you have a Slack to notify your team, send out copies of facilitation content and tools during session.

1 Define your problem statement

Framing your problem as a How Might We statement. This will be the focus of your brainstorm.

PROBLEM

How might we make it possible for anyone — regardless of SQL knowledge — to query a database and explore data using plain, everyday English?

Key rules of brainstorming

To run a smooth and productive session

- ✓ NL-to-SQL Core
- ✓ Personalized query suggestions
- ✓ AI-driven SQL optimization
- ✓ User Experience
- ✓ Improve user interface
- ✓ Customizable query templates

Database & API Integration

- Integrate real-time data sources
- Partnerships with database services

Query Assistance

- From complex queries made easy
- Smart budget-aware services

Future Scope Expansion

- Expand database types covered
- Integrate with cloud platforms

TIP
Encourage open and judgment-free idea sharing. Use the HMW question as your anchor point to guide discussion back to keeping ideas aligned to the main problem.

1 Group Ideas

TIP
Add customizable tags to sticky notes to make it easier to find, browse, and categorize important ideas as themes.

Convert English to SQL using Gemini Pro	Query DB without writing SQL	Auto-suggest SQL from input
---	------------------------------	-----------------------------

User Experience

Show results as table in Streamlit	Track query history with timestamps	One-click example questions
------------------------------------	-------------------------------------	-----------------------------

Future Scope Expansion

Support MySQL, BigQuery, PostgreSQL	Explain SQL in plain English	Visualize results as charts
-------------------------------------	------------------------------	-----------------------------

Prioritize — Importance vs Feasibility

TIP
Participants use their cursor to point to where ideas should go on the grid. The facilitator confirms the spot using the laser pointer or H key on keyboard.

1.1 Objective

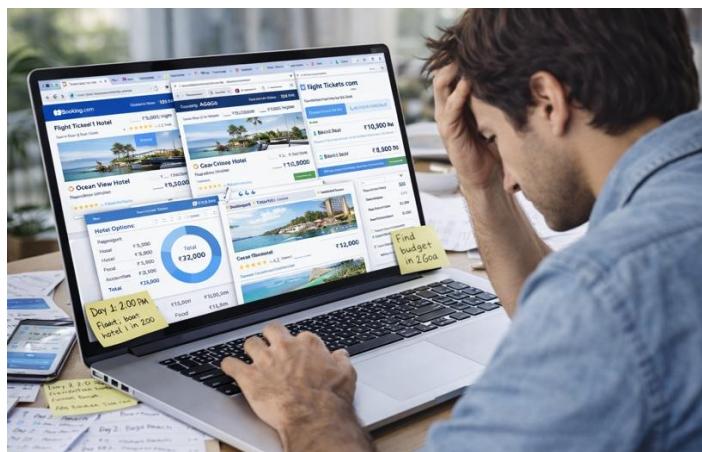
To design and develop an intelligent SQL querying system that converts natural language questions into SQL queries using Google's Gemini Pro model, enabling non-technical users to interact with databases using plain English without writing any SQL.

1.2 Problem Statement

Most people who work with databases — including students, analysts, and business users — do not have strong SQL skills. Writing accurate SQL queries requires technical knowledge that creates a bottleneck, forcing people to depend on developers just to extract basic information from their own data.

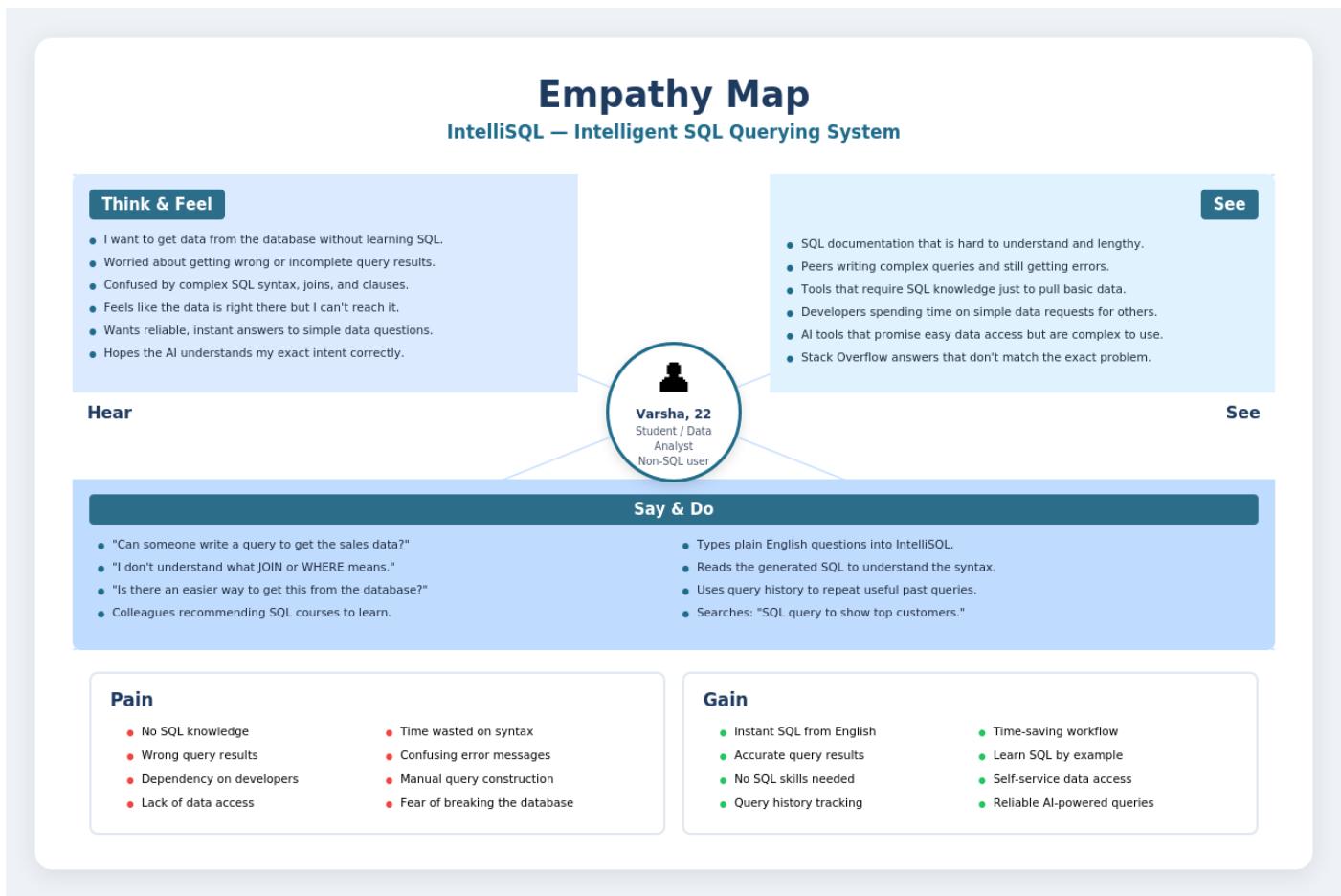
Customer Problem Statement:

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	I am a student or data analyst with no SQL background.	I'm trying to get specific data from a database quickly and independently.	But I do not know how to write SQL queries and always get syntax errors.	Because SQL requires technical knowledge that I have not been trained in.	Which makes me feel frustrated, stuck, and dependent on developers for basic data access.
PS-2	I am a business user or manager who needs database reports.	I'm trying to pull data from the system without asking a developer every time.	But I cannot write SQL and there is no self-service tool available.	Because there is no tool that allows natural language interaction with the database.	Which makes me feel slowed down and unable to make data-driven decisions independently.



1.3 Empathize & Discover

Empathy Map:



The Empathy Map was created to understand the user's database querying challenges, emotions, and expectations. It helped identify key problems faced by non-technical users and ensured that IntelliSQL is designed as a user-centered solution.

Key Pain Points:

- No SQL knowledge or technical skills
- Dependency on developers for data access
- Wrong query results from manual SQL attempts
- Time wasted searching SQL syntax online
- Confusing error messages from the database
- Lack of direct, self-service data access

Key Gains:

- Instant SQL generation from plain English questions
- Accurate query results every time
- No SQL skills required to use the tool
- Query history tracking for easy re-use
- Time-saving automated data workflow
- Learn SQL by reading the generated queries

Key Insights from Ideation:

- Users struggle to write correct SQL queries without training
- Data access is blocked by technical barriers
- Non-technical users need automated SQL generation
- A simple natural language interface is required

1.4 User Journey Map



Journey Overview:

The user journey analysis shows that non-technical users move from frustration when trying to write SQL manually, to relief and delight when they discover IntelliSQL. IntelliSQL simplifies this journey by providing automated SQL generation, transparent query display, and a centralized natural language querying interface.

Key Insights from Ideation:

- Users struggle to write correct SQL queries without technical training
- Data access is difficult without a SQL background
- Non-technical users need automated SQL generation from plain English
- A centralized intelligent querying platform is required

1.5 Proposed Solution

Develop IntelliSQL — a web application that allows users to type a natural language question (e.g. 'show all customers from Hyderabad') and instantly receive the generated SQL query along with the query results, all powered by Google's Gemini Pro large language model and displayed through an interactive Streamlit interface.

Phase 2: Requirement Analysis

2.1 Functional Requirements

FR No	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Natural Language Input	User types a plain English question into the Streamlit text box System accepts the input and sends it to Gemini Pro API
FR-2	Input Validation	System validates the input is not empty before processing System shows an error message if input is empty
FR-3	SQL Generation via Gemini	Gemini Pro generates a SQL query from the natural language question System strips markdown formatting from the generated SQL
FR-4	SQL Execution	System executes the generated SQL against the SQLite database System handles and displays SQL execution errors gracefully
FR-5	Result Display	Query results displayed as an interactive pandas DataFrame table Row and column count shown as metrics above the table
FR-6	Query History	All queries tracked with timestamp, generated SQL, and row count Full history displayed in the session sidebar
FR-7	Example Questions	Sidebar shows clickable example questions for quick use Clicking an example auto-fills the input field
FR-8	API Key Management	User enters Google Gemini API key in the Streamlit sidebar Key input is masked (password field) for security

2.2 Non-Functional Requirements

NFR No	Non-Functional Requirement	Description
NFR-1	Usability	The system must provide a simple and intuitive Streamlit web interface.
NFR-2	Performance	SQL query must be generated and results returned within a few seconds.
NFR-3	Reliability	The system must generate consistent and accurate SQL for valid English questions.
NFR-4	Scalability	The system should support adding more database tables and schemas with minimal changes.
NFR-5	Maintainability	Code should be modular, well-commented, and easy to extend or update.

NFR-6	Availability	The system should be accessible whenever the user runs the Streamlit application locally.
NFR-7	Compatibility	The system should run on any OS with Python 3.x installed and a modern web browser.

2.3 Technology Stack

Component	Technology Used	Purpose / Description
Programming Language	Python 3.12	Used to implement backend logic, SQL execution, and query history management.
Frontend Framework	Streamlit	Used to build the interactive web-based user interface for IntelliSQL.
AI / LLM	Google Gemini Pro (google-genai)	Converts natural language English questions into SQL queries via API.
Database	SQLite3	Lightweight relational database used to store and query customer records.
Version Control	GitHub	Used for code management, version tracking, and project submission.

2.4 User stories

User Type	Epic	User Story No	User Story	Acceptance Criteria	Priority	Release
Customer	Natural Language Input	US-1	As a user, I can type a plain English question and the system generates the SQL and returns results.	System accepts input and displays generated SQL + results correctly.	High	Sprint 1
Customer	SQL Generation	US-2	As a user, I receive an accurate SQL query generated by Gemini Pro from my natural language question.	System generates correct SQL for all valid English questions.	High	Sprint 1
Customer	Result Display	US-3	As a user, I can view the query results in a clear interactive table with row and column count metrics.	System displays a formatted DataFrame with correct metrics.	High	Sprint 1
Customer	Query History	US-4	As a user, I can view my full query history with timestamps and generated SQL during the session.	System shows complete query history after each successful query run.	High	Sprint 2

Phase 3: Project Design

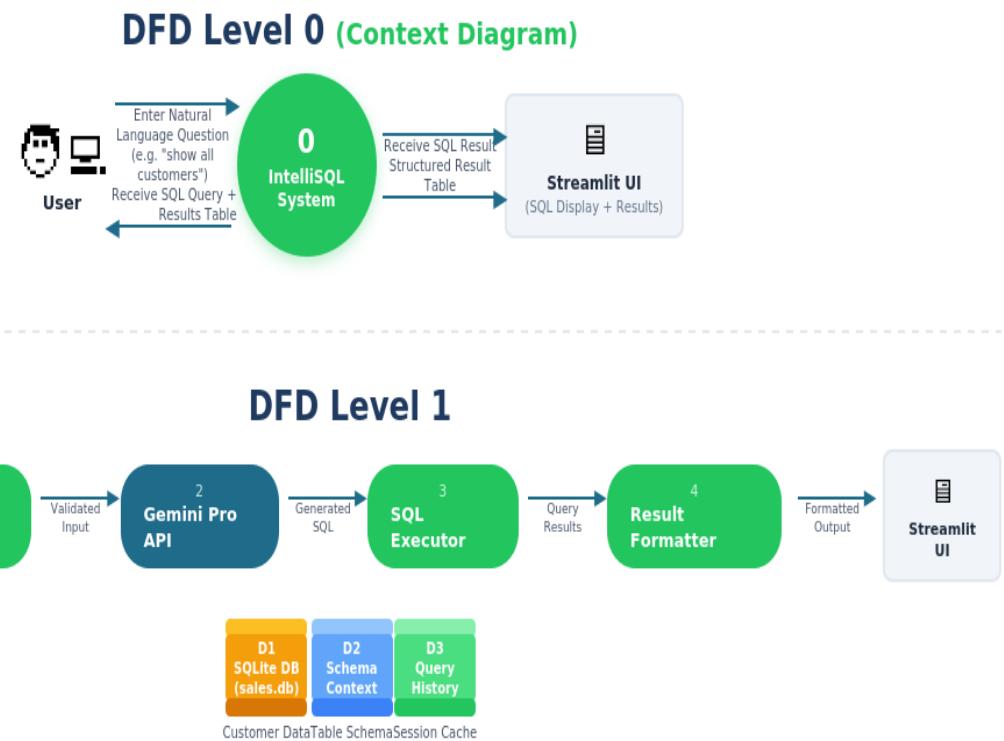
3.1 System Architecture

IntelliSQL uses a modular architecture to process user questions and generate accurate SQL results. The system includes an Input Handler, Gemini Pro API Integration, SQL Execution Engine, and Result Formatter. It uses a SQLite database and table schema context to generate correct queries and display structured results.

User Input → Input Handler → Gemini Pro API → SQL Executor → Result Formatter → Streamlit Display

3.2 Data Flow Diagram (DFD)

The following Data Flow Diagram illustrates the overall data movement and internal processing of the IntelliSQL system.



Level 0 (Context Diagram) presents the overall interaction: the user enters a natural language question, and the IntelliSQL system returns the generated SQL query along with the query results displayed in the Streamlit interface.

Level 1 explains the internal processes, including Input Handling, Gemini Pro API Call, SQL Execution, and Result Formatting. The system uses the SQLite database (sales.db), schema context, and session query history to produce accurate SQL results and display them to the user.

This diagram clearly illustrates the flow of data from the user's natural language question to the final SQL result output.

Phase 4: Project Planning (Agile)

4.1 Product Backlog, Sprint Schedule and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Natural Language Input	US-1	As a user, I can type a plain English question and the system accepts and processes it.	2	High	Developer
Sprint-1	Input Validation	US-2	As a system, I validate that the input is not empty before sending to Gemini Pro API.	1	High	Developer
Sprint-2	SQL Generation	US-3	As a user, I receive a correct SQL query generated by Gemini Pro from my natural language input.	4	High	Developer
Sprint-2	SQL Execution	US-4	As a user, the generated SQL is executed against the SQLite database and returns accurate results.	3	High	Developer
Sprint-3	UI Development	US-5	As a user, I can interact with the text input, sidebar API key, and example questions in Streamlit.	3	High	Developer
Sprint-3	Result Display	US-6	As a user, I see a structured results table with row count, column count, and generated SQL shown.	2	High	Developer
Sprint-4	Query History	US-7	As a user, I can see the full session query history with timestamps and SQL at any time.	2	High	Developer
Sprint-4	Integration & Testing	US-8	Integrate all modules and perform end-to-end system testing before final submission.	3	High	All

