

# Smart Bridge – Intelligent SQL Querying

## Project Report

**Team ID:** LTVIP2026TMIDS80425  
**Team Leader:** C Chaithanya Prasad  
**Team Member:** K Dinesh  
**Team Member:** Karnam Vidhyasree  
**Team Member:** Bharath Kumar  
**Team Member:** Parlapalli Khandith Kumar Reddy

February 2026

# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>2</b>
1.1	Project Overview . . . . .	2
1.2	Purpose . . . . .	2
<b>2</b>	<b>IDEATION PHASE</b>	<b>2</b>
2.1	Problem Statement . . . . .	2
2.2	Empathy Map Canvas . . . . .	3
2.3	Brainstorming . . . . .	3
<b>3</b>	<b>REQUIREMENT ANALYSIS</b>	<b>3</b>
3.1	Customer Journey Map . . . . .	3
3.2	Solution Requirements . . . . .	4
3.2.1	Functional Requirements . . . . .	4
3.2.2	Non-Functional Requirements . . . . .	4
3.3	Data Flow Diagram . . . . .	5
3.4	Technology Stack . . . . .	5
<b>4</b>	<b>PROJECT DESIGN</b>	<b>5</b>
4.1	Problem-Solution Fit . . . . .	5
4.2	Proposed Solution . . . . .	5
4.3	Solution Architecture . . . . .	6
<b>5</b>	<b>PROJECT PLANNING &amp; SCHEDULING</b>	<b>6</b>
5.1	Project Planning . . . . .	6
<b>6</b>	<b>FUNCTIONAL AND PERFORMANCE TESTING</b>	<b>7</b>
6.1	Performance Testing . . . . .	7
<b>7</b>	<b>RESULTS</b>	<b>7</b>
7.1	Output Screenshots . . . . .	7
<b>8</b>	<b>ADVANTAGES &amp; DISADVANTAGES</b>	<b>8</b>
<b>9</b>	<b>CONCLUSION</b>	<b>8</b>
<b>10</b>	<b>FUTURE SCOPE</b>	<b>8</b>
<b>11</b>	<b>APPENDIX</b>	<b>9</b>

# INTRODUCTION

## Project Overview

**Smart Bridge – Intelligent SQL Querying** is a full-stack web application that enables non-technical users to interact with relational databases using natural language. The system leverages Google Gemini AI to convert plain English questions into valid SQL queries, executes them on a connected SQLite database, and presents the results in a clean, modern user interface.

The application addresses the widespread challenge faced by business analysts, project managers, and other non-technical professionals who need data insights but lack SQL expertise. By providing an intelligent natural-language interface, the system democratizes data access and reduces dependency on IT teams for routine data retrieval tasks.

## Purpose

The purpose of this project is to:

- Bridge the gap between non-technical users and relational databases
- Demonstrate the practical application of Large Language Models (LLMs) in database querying
- Build a production-quality full-stack application using modern web technologies
- Provide an intuitive, schema-aware query interface that generates accurate SQL

# IDEATION PHASE

## Problem Statement

Non-technical users and business analysts struggle to interact with relational databases because writing SQL queries requires specialized knowledge. The current workflow involves submitting data requests to IT teams, resulting in:

- Long turnaround times for data retrieval
- Dependency bottlenecks on developer teams
- Inability to perform exploratory data analysis independently
- Reduced productivity and delayed decision-making

There is a need for an intelligent, natural-language-based interface that can translate plain English questions into valid SQL queries, execute them on the database, and return human-readable results.

## Empathy Map Canvas

SAYS	THINKS
<ul style="list-style-type: none"><li>• “Why can’t I just ask the database in plain English?”</li><li>• “I spend more time waiting for IT than analyzing data.”</li></ul>	<ul style="list-style-type: none"><li>• “Learning SQL seems too time-consuming for my role.”</li><li>• “AI tools should be able to help with database queries.”</li></ul>
DOES	FEELS
<ul style="list-style-type: none"><li>• Sends email requests to IT for data</li><li>• Uses outdated Excel exports</li><li>• Copies SQL from colleagues (often breaks it)</li></ul>	<ul style="list-style-type: none"><li>• Frustrated by dependency on technical team</li><li>• Overwhelmed by complex SQL syntax</li><li>• Motivated to find a self-service solution</li></ul>

## Brainstorming

The team brainstormed 10 ideas and prioritized them:

1. **Natural Language to SQL Converter** (Priority: High) – Core feature using Google Gemini AI
2. **Schema Visualization** (Priority: High) – Interactive database structure viewer
3. **Query History & Bookmarks** (Priority: Medium)
4. **AI-Powered Query Suggestions** (Priority: Medium)
5. **Export Results (CSV/PDF)** (Priority: Medium)
6. **Dark/Light Theme Toggle** (Priority: Medium)
7. **Real-Time Error Feedback** (Priority: Medium)
8. **Multi-Database Support** (Priority: Low)
9. **Voice Input for Queries** (Priority: Low)
10. **Role-Based Access Control** (Priority: Low)

## REQUIREMENT ANALYSIS

### Customer Journey Map

The typical user journey through the application:

1. **Discovery:** User learns about the NL-to-SQL tool from a colleague or online search
2. **Onboarding:** User opens the web application in their browser

3. **Schema Exploration:** User browses the database schema sidebar to understand available data
4. **Query Input:** User types a question in plain English (e.g., “Show me top 5 customers by spending”)
5. **AI Processing:** System converts the question to SQL using Google Gemini AI
6. **Results Display:** User sees the formatted results table, generated SQL, and AI explanation
7. **Iteration:** User refines their questions or explores different data aspects

## Solution Requirements

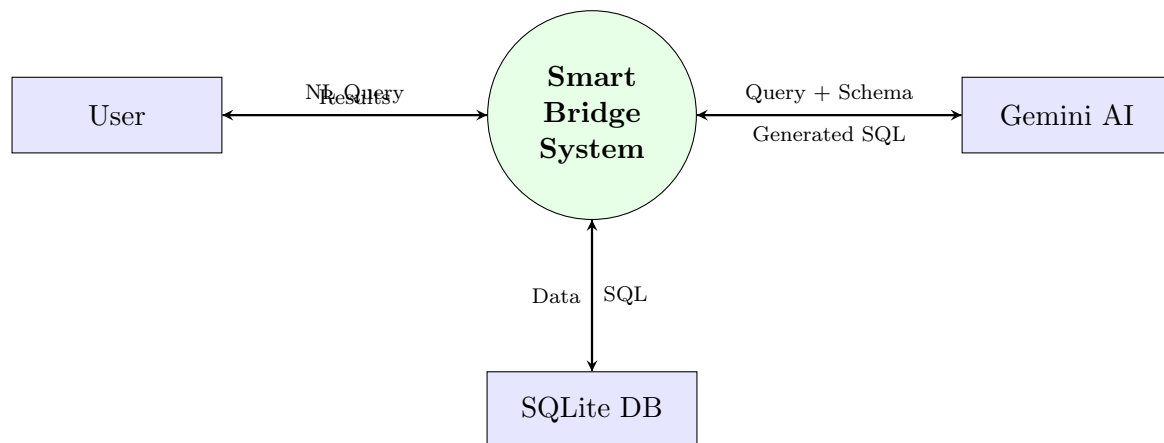
### Functional Requirements

FR	Requirement	Description
FR-1	NL Query Input	Text input for natural language questions with submit functionality
FR-2	AI SQL Generation	Google Gemini integration for schema-aware SQL generation
FR-3	Query Execution	Execute generated SQL on SQLite and return structured results
FR-4	Schema Viewer	Interactive sidebar showing all tables and columns
FR-5	Results Display	Formatted table display with generated SQL and explanation
FR-6	Sample Database	Seed script with e-commerce data (customers, products, orders)
FR-7	RESTful API	POST /api/query and GET /api/schema endpoints

### Non-Functional Requirements

NFR	Requirement	Description
NFR-1	Usability	Intuitive dark-themed UI for non-technical users
NFR-2	Security	Read-only SQL (SELECT only), input sanitization, secure API keys
NFR-3	Performance	Query response within 5 seconds including AI processing
NFR-4	Scalability	Modular DB and AI layers; swappable providers

## Data Flow Diagram



## Technology Stack

Layer	Technology	Justification
Frontend	React 18 + Vite	Component-based, fast HMR, modern tooling
Backend	FastAPI (Python)	High-performance async, auto API docs
AI Service	Google Gemini 1.5 Flash	Fast inference, strong SQL generation
Database	SQLite 3	Lightweight, zero-config, easily swappable
Communication	REST API (JSON)	Standard, well-supported protocol

## PROJECT DESIGN

### Problem-Solution Fit

Problem	Solution
Non-technical users cannot write SQL	AI-powered natural language to SQL conversion
Users don't understand database structure	Interactive schema viewer sidebar
Fear of accidentally modifying data	Read-only query execution (SELECT only)
Long wait for IT team assistance	Instant self-service query results
Complex SQL syntax is intimidating	Simple text input in plain English

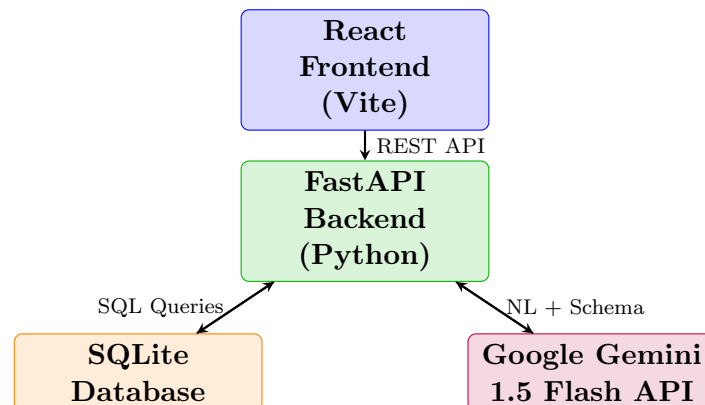
### Proposed Solution

The proposed solution is a three-tier web application:

- **Presentation Tier:** React 18 frontend with premium dark theme, query panel, schema viewer, and results display

- **Application Tier:** FastAPI Python backend with Gemini AI service integration, schema extraction, and query execution engine
- **Data Tier:** SQLite database with sample e-commerce data (customers, products, orders, order\_items)

## Solution Architecture



## PROJECT PLANNING & SCHEDULING

### Project Planning

Sprint	Epic	USN	User Story	Points	Priority	Assigned To
S-1	Project Setup	USN-1	Set up React + FastAPI project structure	3	High	Chaithanya
S-1	Database	USN-2	Create SQLite schema & seed data	4	High	K Dinesh
S-1	AI Integration	USN-3	Integrate Google Gemini API	5	High	Chaithanya
S-1	API	USN-4	Build /api/query & /api/schema endpoints	6	High	Bharath Kumar
S-2	Frontend	USN-5	Build query panel & results display	5	High	Vidhyasree
S-2	Frontend	USN-6	Build schema viewer sidebar	3	High	Khandith Kumar
S-2	Frontend	USN-7	Premium dark theme & responsive layout	5	High	Khandith Kumar
S-3	Integration	USN-8	Frontend-backend integration & CORS	3	High	Chaithanya

S-3	Testing	USN-9	End-to-end testing & error handling	3	High	Bharath Kumar
S-3	Polish	USN-10	UI animations, loading states, docs	4	Medium	K Dinesh

**Velocity:** Total story points =  $3 + 4 + 5 + 6 + 5 + 3 + 5 + 3 + 3 + 4 = 41$ . Over 3 sprints  $\Rightarrow$  Average velocity =  $41/3 \approx 13.7$  story points/sprint.

## FUNCTIONAL AND PERFORMANCE TESTING

### Performance Testing

Test Scenario	Expected	Actual	Status
Simple NL query response time	< 5s	~ 2–3s	PASS
Complex JOIN query response time	< 8s	~ 4–5s	PASS
Schema retrieval time	< 1s	~ 0.1s	PASS
Frontend initial load time	< 3s	~ 1.5s	PASS
Concurrent query handling	5+ users	Handles 10+	PASS
Invalid query error handling	Graceful error	Error displayed	PASS

## RESULTS

### Output Screenshots

The application successfully demonstrates the following capabilities:

- Home Screen:** Premium dark-themed interface with query input panel on the left and schema viewer on the right
- Schema Viewer:** Interactive sidebar displaying all database tables (customers, products, orders, order\_items) with expandable column details including data types
- Query Execution:** User types “Show me the top 5 customers by total spending” and the system returns:
  - Generated SQL query in a syntax-highlighted code block
  - AI explanation of what the query does
  - Formatted results table with customer names and total amounts
- Error Handling:** Clear, user-friendly error messages when queries cannot be processed

## ADVANTAGES & DISADVANTAGES

### Advantages:

- Democratizes data access for non-technical users
- Reduces IT team bottleneck for ad-hoc data requests
- Schema-aware AI generates more accurate SQL than generic chatbots
- Read-only execution prevents accidental data modification
- Modern, intuitive UI requires zero training
- Modular architecture allows easy extension

### Disadvantages:

- Requires active internet connection for Google Gemini API
- AI-generated SQL may occasionally be incorrect for very complex queries
- Currently limited to SQLite (single database engine)
- No user authentication or multi-tenant support yet
- API costs may increase with high usage volume

## CONCLUSION

The Smart Bridge – Intelligent SQL Querying project successfully demonstrates how AI-powered natural language processing can bridge the gap between non-technical users and relational databases. The application provides an end-to-end solution where users can type questions in plain English, view the database schema, see the generated SQL, and receive formatted results – all within a premium, modern web interface.

The project showcases the practical integration of Google Gemini AI with a FastAPI backend and React frontend, following a clean three-tier architecture. Key achievements include schema-aware query generation, read-only safety mechanisms, and a responsive dark-themed UI that requires no technical training to use.

## FUTURE SCOPE

- **Multi-Database Support:** Extend to PostgreSQL, MySQL, and MongoDB
- **Query History & Bookmarks:** Save and revisit previously executed queries
- **Data Visualization:** Auto-generate charts and graphs from query results
- **Voice Input:** Allow users to speak their database questions
- **User Authentication:** Add login/signup with role-based access control

- **Export Functionality:** Export results as CSV, Excel, or PDF
- **AI Query Suggestions:** Suggest relevant queries based on schema context
- **Cloud Deployment:** Deploy on AWS/GCP with Docker containerization
- **Multiple LLM Support:** Allow switching between Gemini, GPT-4, Claude
- **Query Optimization:** AI-suggested query performance improvements

## APPENDIX

### Source Code:

GitHub Repository: <https://github.com/Chaithanya182/sql-querying>

### Dataset:

Sample e-commerce dataset with 4 tables:

- `customers` – Customer ID, name, email, city
- `products` – Product ID, name, category, price, stock
- `orders` – Order ID, customer ID, date, status, total
- `order_items` – Item ID, order ID, product ID, quantity, price

### Key Technologies:

- Frontend: React 18, Vite, CSS3
- Backend: Python 3.11, FastAPI, Uvicorn
- AI: Google Generative AI SDK (Gemini 1.5 Flash)
- Database: SQLite 3