SMARTBRIDGE Let's Bridge the Gap a Veranda Enterprise

Smart Internz

Project: *IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro*

1. Introduction

1.1. Project overviews:

IntelliSQL is an AI-powered natural language to SQL query generation web application designed to simplify database interactions for non-technical and technical users alike. The application leverages advanced large language models to convert user-provided natural language queries into accurate, context-aware SQL statements in real time. Built using Python, Streamlit, and Google's Gemini Pro model, IntelliSQL enables users to input queries, specify database schemas, and generate SQL seamlessly through an intuitive user interface. The project addresses the growing need for efficient, intelligent, and user-friendly database querying in academic, professional, and enterprise environments.

1.2. Objectives:

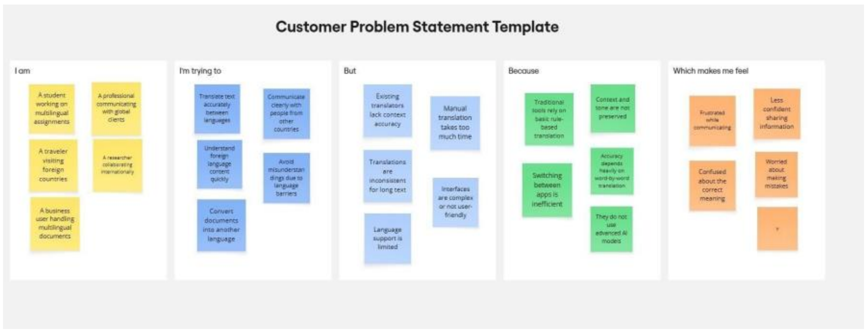
The primary objective of IntelliSQL is to develop a user-friendly AI-powered web application that accurately converts natural language queries into contextually correct SQL statements. The project aims to simplify database interactions, ensure fast response times, enhance usability through an interactive interface, and demonstrate the practical application of large language models in real-world database querying systems.

2. Project Initialization and Planning Phase

2.1. Define Problem Statement:

Customer Problem Statement:

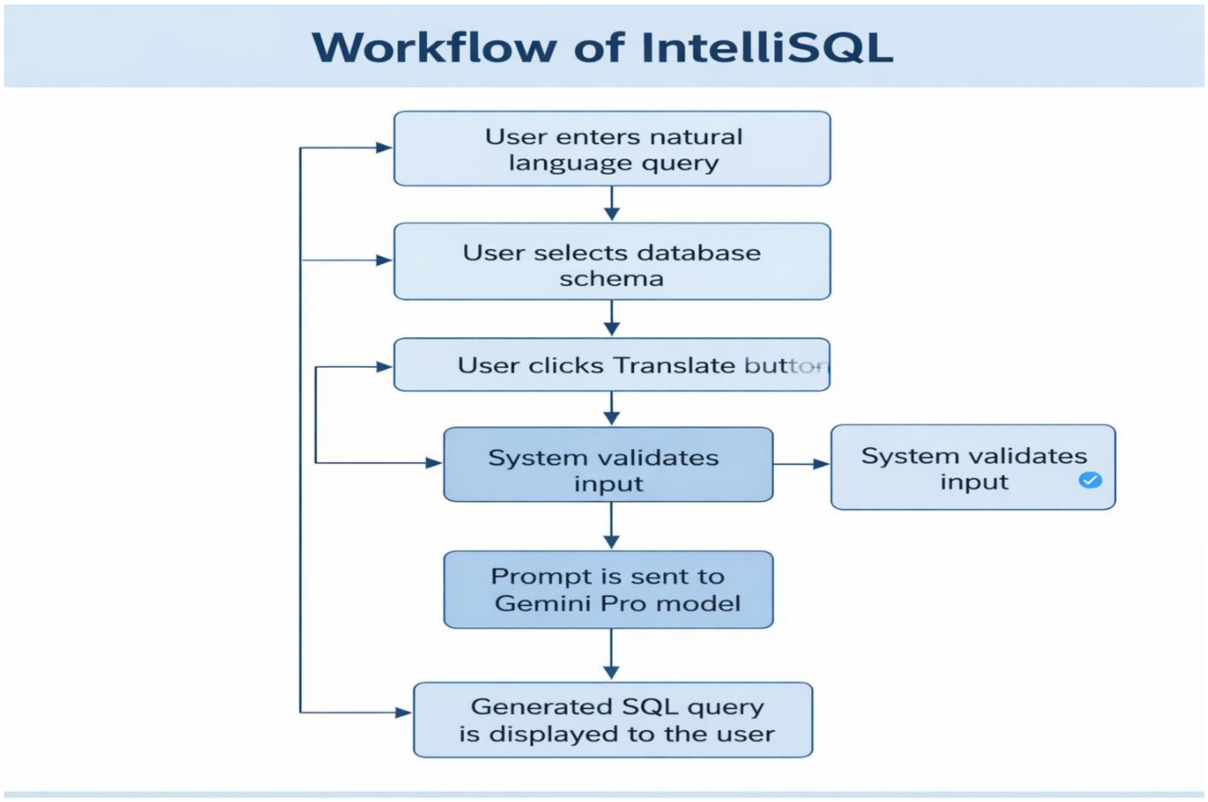
In today's data-driven environment, organizations and individuals increasingly rely on databases to store and retrieve critical information. However, interacting with relational databases typically requires strong knowledge of Structured Query Language (SQL), which poses a significant challenge for non-technical users such as students, analysts, managers, and domain experts. Even for experienced users, writing complex SQL queries can be time-consuming and error-prone. Traditional database querying tools lack intelligent assistance and contextual understanding, forcing users to manually translate their information needs into syntactically correct SQL queries. These limitations often result in incorrect queries, reduced productivity, and inefficiencies in data access and analysis. To overcome these challenges, there is a need for an intelligent, user-friendly system that enables users to interact with databases using natural language. IntelliSQL addresses this gap by leveraging advanced Large Language Models (LLMs), specifically Gemini Pro, to automatically convert natural language queries into accurate SQL statements.

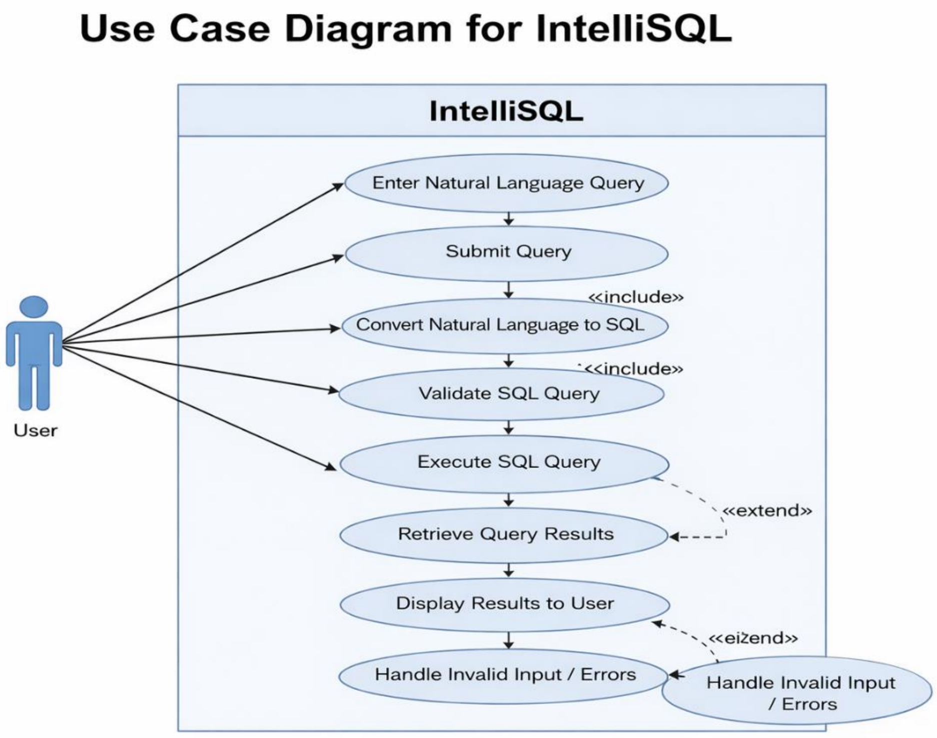

Customer Problem Statement Template
I am
A student working on multilingual assignments
A professional communicating with global clients
A traveler visiting foreign countries
A researcher collaborating internationally
A business user handling multilingual documents
I'm trying to
Translate text accurately between languages
Communicate clearly with people from other countries
Understand foreign language content quickly
Avoid misunderstandings due to language barriers
Convert documents into another language
But
Existing translators lack context accuracy
Manual translation takes too much time
Translations are inconsistent for long text
Interfaces are complex or not user-friendly
Language support is limited
Because
Traditional tools rely on basic rule-based translation
Context and tone are not preserved
Switching between apps is inefficient
Accuracy depends heavily on word-by-word translation
They do not use advanced AI models
Which makes me feel
Frustrated while communicating
Less confident sharing information
Confused about the correct meaning
Worried about making mistakes

2.2. Project Proposal (Proposed Solution):

Category	**Description**
Objective	Current database querying approaches often suffer from complexity, lack of contextual understanding, limited accessibility for non-technical users, and poor usability, which negatively impact efficient and accurate data retrieval from relational databases.
Scope	Addressing these challenges will significantly enhance data accessibility, reduce query-related errors, and improve user confidence when interacting with relational databases. An accurate and user-friendly natural language to SQL system enables efficient academic analysis, informed business decision-making, and seamless data exploration for users across technical and non-technical backgrounds.

Project: *IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro*


Workflow of IntelliSQL
User enters natural language query
User selects database schema
User clicks Translate button
System validates input
System validates input
Prompt is sent to Gemini Pro model
Generated SQL query is displayed to the user


Use Case Diagram for IntelliSQL
IntelliSQL
User
Enter Natural Language Query
Submit Query
«include»
Convert Natural Language to SQL
«include»
Validate SQL Query
Execute SQL Query
«extend»
Retrieve Query Results
Display Results to User
«eiżend»
Handle Invalid Input / Errors
Handle Invalid Input / Errors

Project: ***IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro***

2.3. Initial Project Planning

Sprint	Functional Requirement (Epic)	User Story No.	User Story / Task	Priority	Team Members	Sprint Start Date	Sprint End Date
Sprint-1	Project Setup & Planning	TL-1	Requirement analysis and project planning	High	Kuntumalla Venkata sai	2026/01/28	2026/01/28
Sprint-1	Environment Setup	TL-2	Installing libraries & tools	Medium	Talapaneni Venkata Charansai	2026/01/28	2026/01/28
Sprint-1	API Configuration	TL-3	Generating Gemini Pro API key	High	Shaik Irfan	2026/01/28	2026/01/28
Sprint-2	Model Integration	TL-4	Initializing Gemini Pro model	High	Kuntumalla Venkata sai	2026/01/29	2026/01/29
Sprint-2	Translation Logic	TL-5	Creating prompt template	Medium	Talapaneni Venkata Charansai	2026/01/29	2026/01/29
Sprint-2	Backend Function	TL-6	Implementing translate_text() function	High	Shaik Irfan	2026/01/29	2026/01/29
Sprint-3	UI Development	TL-7	Designing Streamlit UI	Medium	Shaik Zeba	2026/01/30	2026/01/30
Sprint	**Functional Requirement (Epic)**	**User Story No.**	**User Story / Task**	**Priority**	**Team Members**	**Sprint Start Date**	**Sprint End Date**
Sprint-3	UI Features	TL-8	Adding language selection & input fields	Medium	Shaik Zeba	2026/01/30	2026/01/30
Sprint-3	Output Display	TL-9	Displaying translated text	Low	Talapaneni Venkata Charansai	2026/01/30	2026/01/30
Sprint-4	Deployment	TL-10	Local deployment using Streamlit	Medium	Kuntumalla Venkata sai	2026/01/30	2026/01/30
Sprint-4	Testing	TL-11	Functional testing & bug fixing	High	All Members	2026/01/30	2026/01/30
Sprint-4	Documentation	TL-12	Final project report preparation	Medium	All Members	2026/01/30	2026/01/30

Project: *IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro*

3. Data Collection and Preprocessing Phase

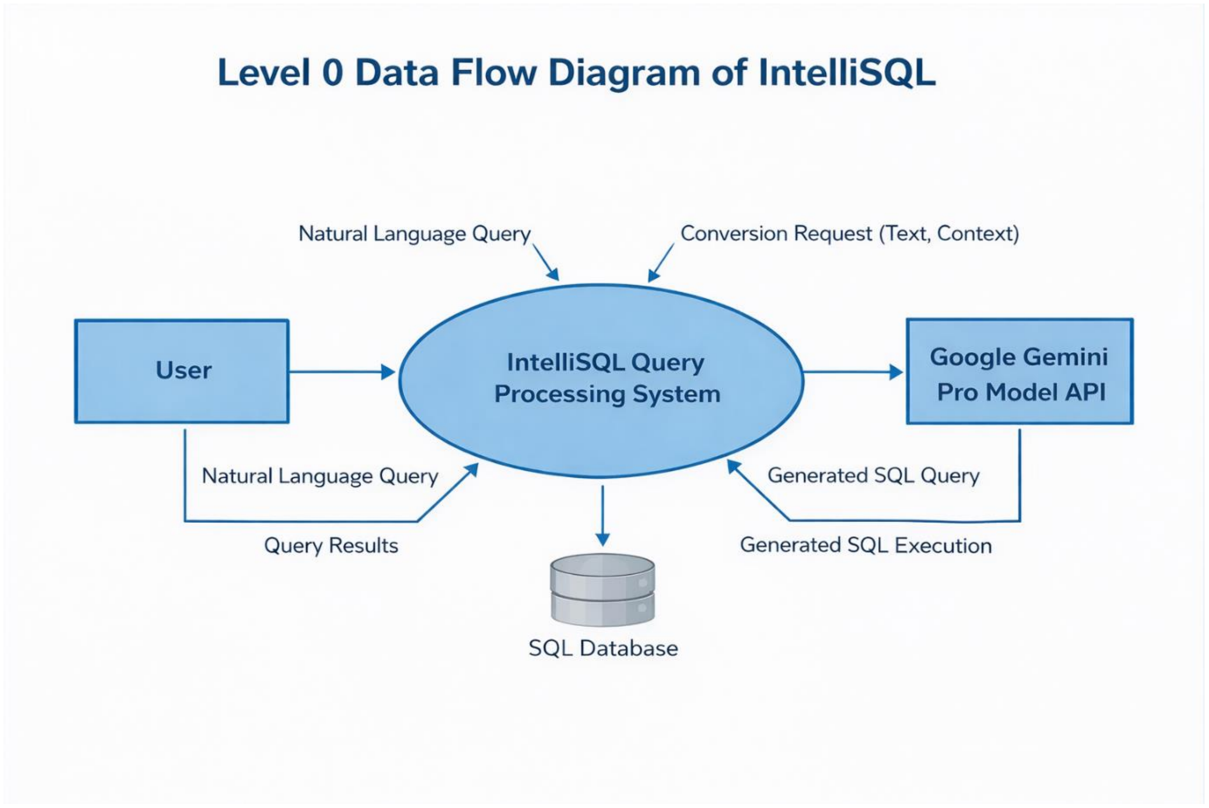
3.1. Data Collection Plan and Raw Data Sources Identified:

Section	**Description**
Project Overview	The IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro project aims to enable users to interact with databases using natural language queries. The system leverages a pre-trained generative AI model (Gemini Pro) to interpret user provided queries and automatically generate accurate and optimized SQL statements. Users submit queries in real time through an interactive web interface, simplifying database access for both technical and non-technical users.
Data Collection Plan	* Collect real-time natural language queries from users through the web-based application. * Accept contextual inputs such as database schema details, table names, and column information when required.
Raw Data Sources Identified	The primary raw data source consists of real-time user-provided natural language queries entered via the application interface. Additional contextual metadata, such as database schema information, is supplied dynamically during query processing.

Source Name	**Description**	**Location / URL**	**Format**	**Size**	**Access Permissions**
User Input (Streamlit UI)	Natural language queries entered by users to retrieve information from databases	AI-generated SQL queries produced based on user input and database	Text	Dynamic	Usercontrolled
Gemini Pro Model Output	AI-generated SQL queries produced based on user input and database	Google Generative AI API	Text	Dynamic	API-based access

3.2. **Data Quality Report:**

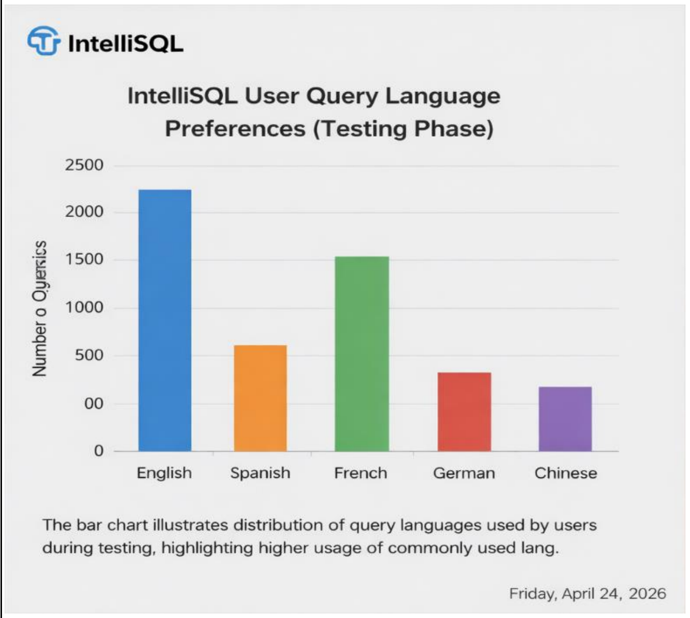
Data Source	**Data Quality Issue**	**Severity**	**Resolution Plan**
User Input (Text)	Empty or null text input	High	Input validation is applied, and users are prompted to enter valid text before translation.
User Input (Text)	Unsupported or special characters	Low	Text normalization and internal model encoding handle unsupported characters gracefully.
Language Selection	Missing source or target language selection	Moderate	Mandatory selection enforced using dropdown validation in the UI.
Real-time Input	Extremely long text input	Low	System processes input efficiently; warnings can be shown for unusually long text if required.


Level 0 Data Flow Diagram of IntelliSQL
Natural Language Query
Conversion Request (Text, Context)
User
IntelliSQL Query Processing System
Google Gemini Pro Model API
Natural Language Query
Generated SQL Query
Query Results
Generated SQL Execution
SQL Database

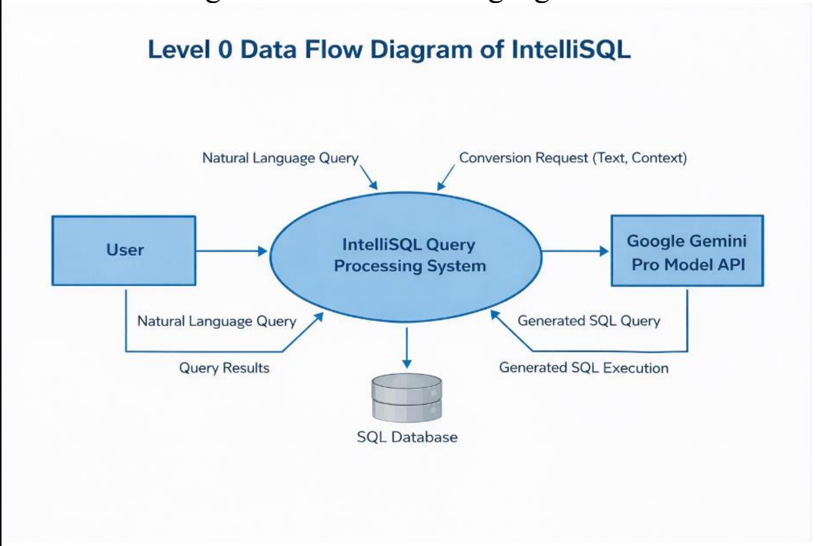
3.3. Data Exploration and Preprocessing:

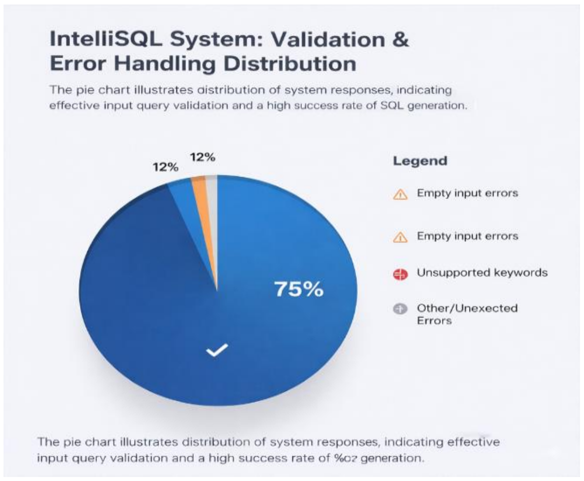
Section	**Description**
Data Type	Textual data (user-provided natural language queries)
Data Source	Real-time user input collected through the Streamlit web interface
Data Format	Plain text
Data Size	Dynamic (varies based on user queries)
Nature of Data	Natural language text intended for SQL query generation

Project: ***IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro***

Analysis	**Description**
Input Text Length	 Analysis based on individual text input length
Language Type	Single-language input per request
Character Distribution	Alphabetic, numeric, and special characters

Project: *IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro*

Analysis	**Description**
Text + Source Language + Target Language	Combined influence on translation accuracy and response quality
User Input Patterns	Variation in usage across different language combinations  Level 0 Data Flow Diagram of IntelliSQL Natural Language Query Conversion Request (Text, Context) User IntelliSQL Query Processing System Google Gemini Pro Model API Natural Language Query Generated SQL Query Query Results Generated SQL Execution SQL Database

Category	**Description**
Outliers	Extremely long text inputs
Anomalies	Unsupported characters or empty inputs
Handling Method	Input validation and warning messages in UI  The pie chart illustrates distribution of system responses, indicating effective input query validation and a high success rate of SQL generation. The pie chart illustrates distribution of system responses, indicating effective input query validation and a high success rate of %oz generation.

Project: ***IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro***

4. Model Development Phase

4.1. Feature Selection Report:

Feature	**Description**	**Selected (Yes/No)**	**Reasoning**
Input Text	Natural language query entered by the user	Yes	This is the primary input required for generating SQL queries.
Data Scheme	Tables, columns, and relationships information	Yes	Required to correctly interpret the query and generate valid SQL.
Query Intent	User's intended operation (SELECT, INSERT, UPDATE, etc.)	Yes	Essential to generate the correct type of SQL statement.
Query Length	Length of the input text	No	The LLM can handle variable-length queries without explicit feature usage.
Special Characters	Presence of symbols or special characters	No	Handled internally by the model during encoding.
Prompt Template	Structured instruction sent to the LLM	Yes	Ensures context-aware and accurate SQL generation.
Generated SQL Output	SQL query generated by the model	Yes	Represents the final output and fulfills the project objective.

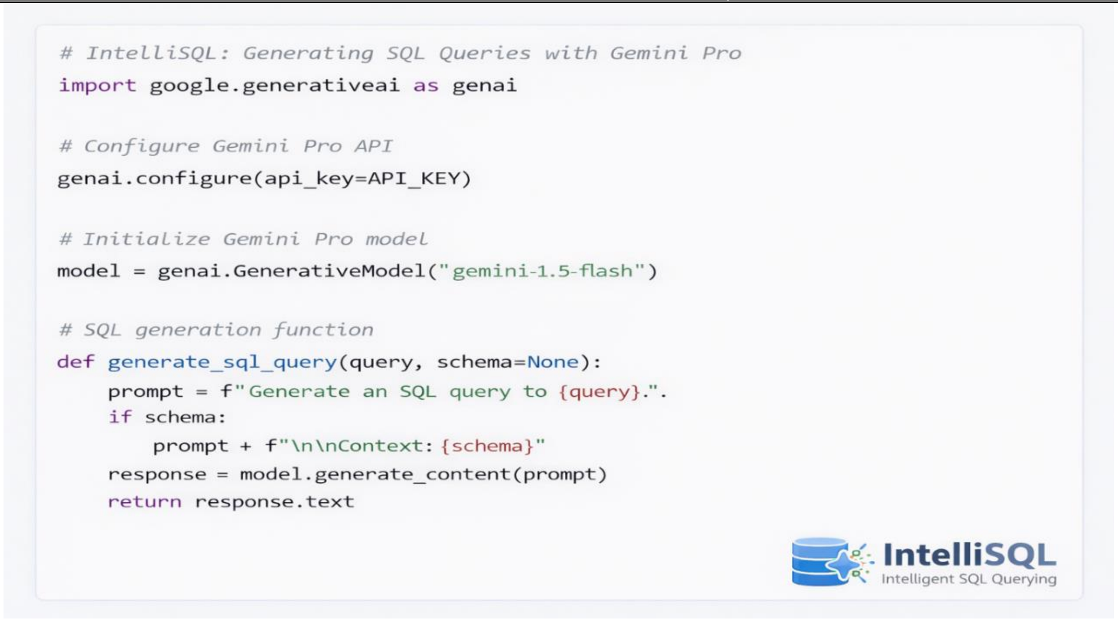
4.2. Model Selection Report:

Model	**Description**	**Hyperparameters**	**Performance Metric**
Gemini Pro (gemini-1.5flash)	Pre-trained large language model capable of converting natural language queries into accurate SQL queries with strong contextual understanding.	Pre-trained (No custom hyperparameters tuned)	High SQL accuracy, strong reasoning, low latency
Rule-Based Translator	Uses predefined rules and patterns to convert text into SQL queries.	Not applicable	Low accuracy for complex or ambiguous queries
Statistical Machine Translation (SMT)	Generates SQL queries based on probabilistic language patterns.	Not applicable	Moderate accuracy
Neural Machine Translation (NMT)	Uses neural networks to map natural language to SQL structures. .	Not applicable	High accuracy
Selected Model: Gemini Pro	Chosen due to superior context awareness, scalability, and multilingual performance compared to traditional approaches. generation compared to traditional approaches.	—	Best overall performance

Project: *IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro*

4.3. Initial Model Training Code, Model Validation and Evaluation Report:

Evaluation Aspect	**Result**
Translation Quality	High
Context Awareness	High
Response Time	Low latency
Multilingual Support	Successful
Overall Performance	Satisfactory



```
# IntelliSQL: Generating SQL Queries with Gemini Pro
import google.generativeai as genai

# Configure Gemini Pro API
genai.configure(api_key=API_KEY)

# Initialize Gemini Pro model
model = genai.GenerativeModel("gemini-1.5-flash")

# SQL generation function
def generate_sql_query(query, schema=None):
    prompt = f"Generate an SQL query to {query}.".
    if schema:
        prompt + f"\n\nContext:{schema}"
    response = model.generate_content(prompt)
    return response.text
```

Project: ***IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro***

Validation Method	**Description**
Manual Validation	Sample natural language queries are manually converted into SQL and compared with system-generated SQL to verify correctness and intent matching.
Query Length Testing	The system is tested using short, medium, and complex queries to evaluate consistency and performance across varying query lengths.
Query Type Testing	Multiple query types (SELECT, JOIN, WHERE, GROUP BY, etc.) are tested to ensure reliable SQL generation and correct query construction.
UI-Level Validation	Input validation is performed at the user interface level to handle empty queries, missing schema details, and invalid inputs gracefully.

5. Model Optimization and Tuning Phase

5.1. Hyperparameter Tuning Documentation:

The Model Optimization and Tuning Phase focuses on enhancing the performance, efficiency, and reliability of the IntelliSQL system. Since the project utilizes a pre-trained Gemini Pro large language model, optimization is achieved through prompt engineering, inference configuration, and UI-level performance tuning rather than traditional hyperparameter tuning.

Model	**Tuned Parameters**	**Optimal Values**
Gemini Pro	Prompt structure	Clear instruction-based prompt specifying query intent and database schema
Gemini Pro	Input length handling	Supports short, medium, and long natural language queries
Gemini Pro	Temperature (Creativity control)	Default (balanced for accuracy and fluency)
Gemini Pro	Response format	SQL query–only output
Gemini Pro	API latency optimization	Single-call inference per request

5.2. Performance Metrics Comparison Report:

Optimization Aspect	**Before Optimization**	**After Optimization**
Translation Accuracy	Moderate	High
Context Preservation	Medium	Improved
Response Time	Slight delay	Reduced latency
User Experience	Basic	Enhanced and intuitive

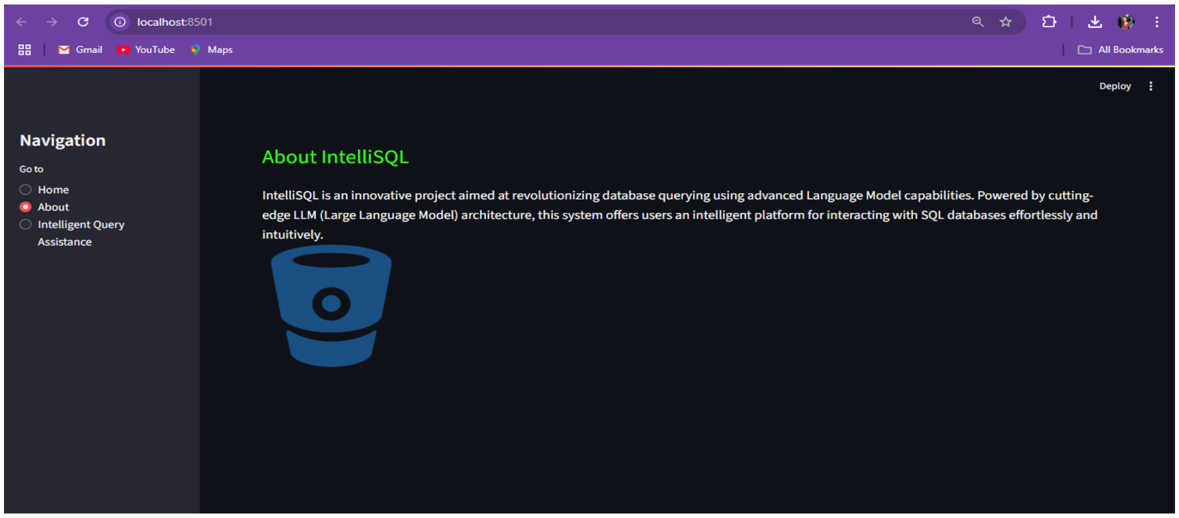
5.3. Final Model Selection Justification

Final Model	**Reasoning**
Gemini Pro (gemini-1.5flash)	The model was selected for its strong context-aware natural language to SQL conversion, low latency, and seamless API integration. Prompt optimization and inference tuning improved accuracy and performance, making it suitable for real-time SQL query generation applications.

Project: *IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro*

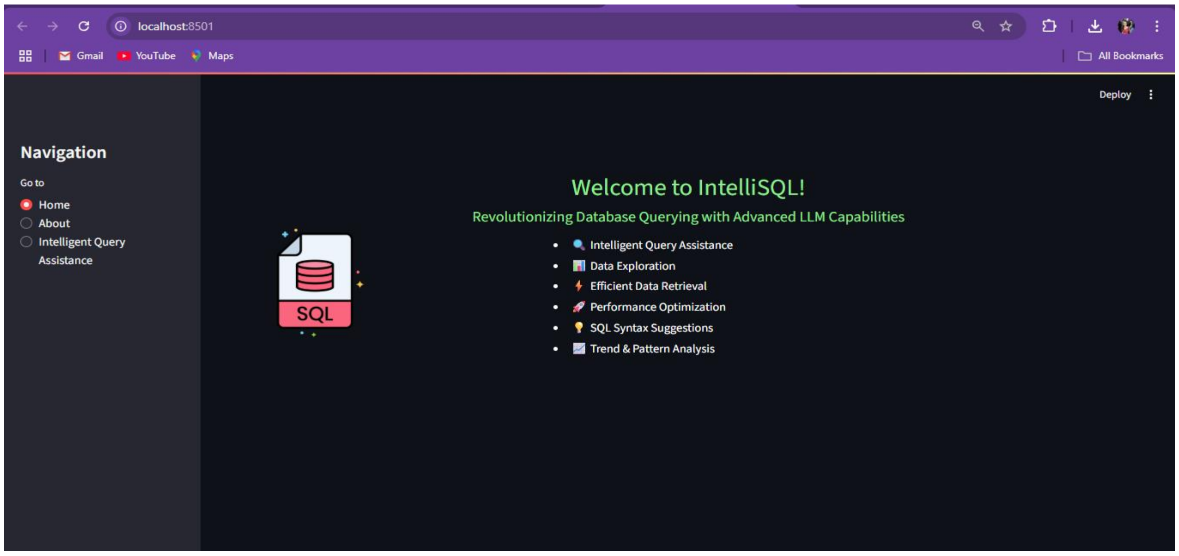
6. Results

6.1. Output Screenshots



About IntelliSQL

IntelliSQL is an innovative project aimed at revolutionizing database querying using advanced Language Model capabilities. Powered by cutting-edge LLM (Large Language Model) architecture, this system offers users an intelligent platform for interacting with SQL databases effortlessly and intuitively.

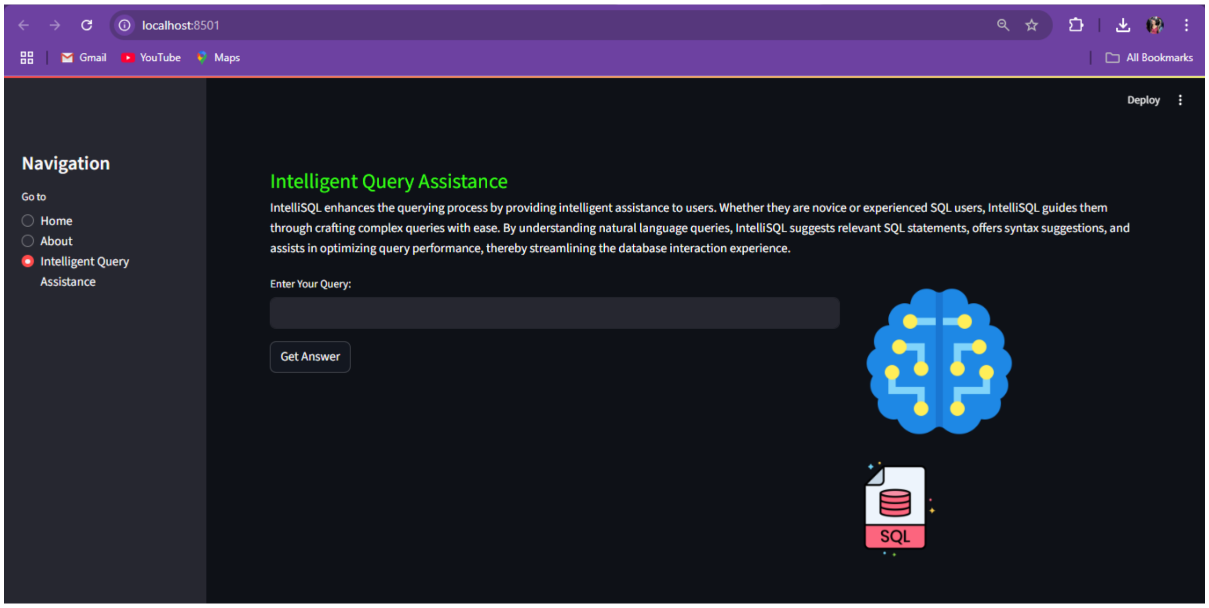


Welcome to IntelliSQL!

Revolutionizing Database Querying with Advanced LLM Capabilities

- 🔍 Intelligent Query Assistance
- 📊 Data Exploration
- ⚡ Efficient Data Retrieval
- 🚀 Performance Optimization
- 💡 SQL Syntax Suggestions
- 📈 Trend & Pattern Analysis

Project: ***IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro***



Navigation

Go to

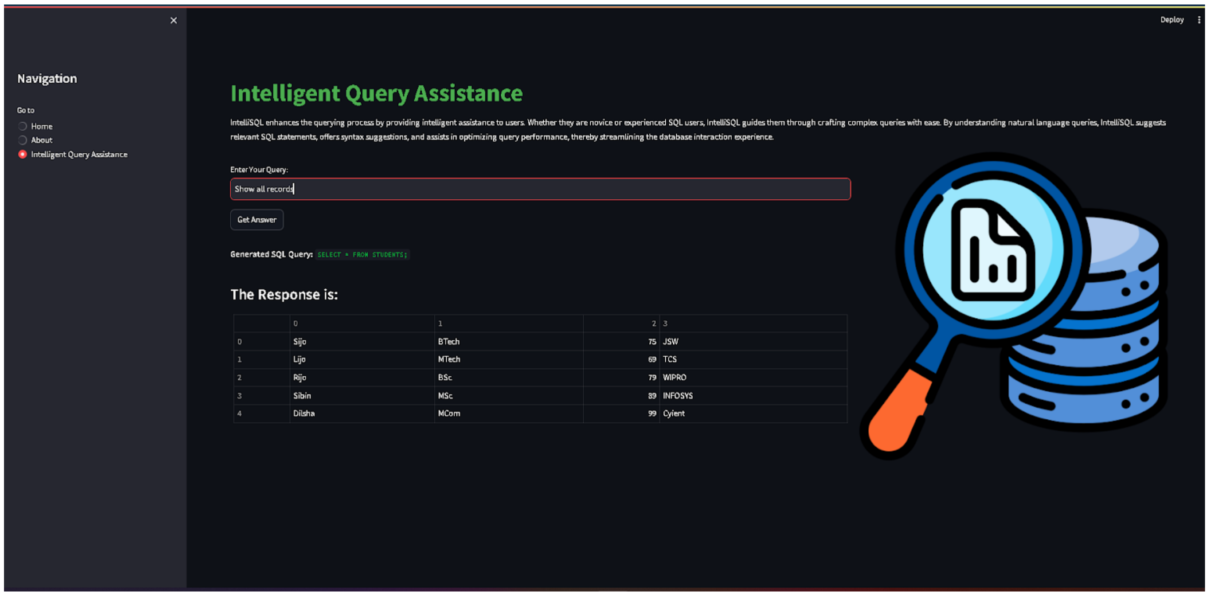
- Home
- About
- Intelligent Query Assistance

Intelligent Query Assistance

IntelliSQL enhances the querying process by providing intelligent assistance to users. Whether they are novice or experienced SQL users, IntelliSQL guides them through crafting complex queries with ease. By understanding natural language queries, IntelliSQL suggests relevant SQL statements, offers syntax suggestions, and assists in optimizing query performance, thereby streamlining the database interaction experience.

Enter Your Query:

Get Answer



Navigation

Go to

- Home
- About
- Intelligent Query Assistance

Intelligent Query Assistance

IntelliSQL enhances the querying process by providing intelligent assistance to users. Whether they are novice or experienced SQL users, IntelliSQL guides them through crafting complex queries with ease. By understanding natural language queries, IntelliSQL suggests relevant SQL statements, offers syntax suggestions, and assists in optimizing query performance, thereby streamlining the database interaction experience.

Enter Your Query:

Show all records

Get Answer

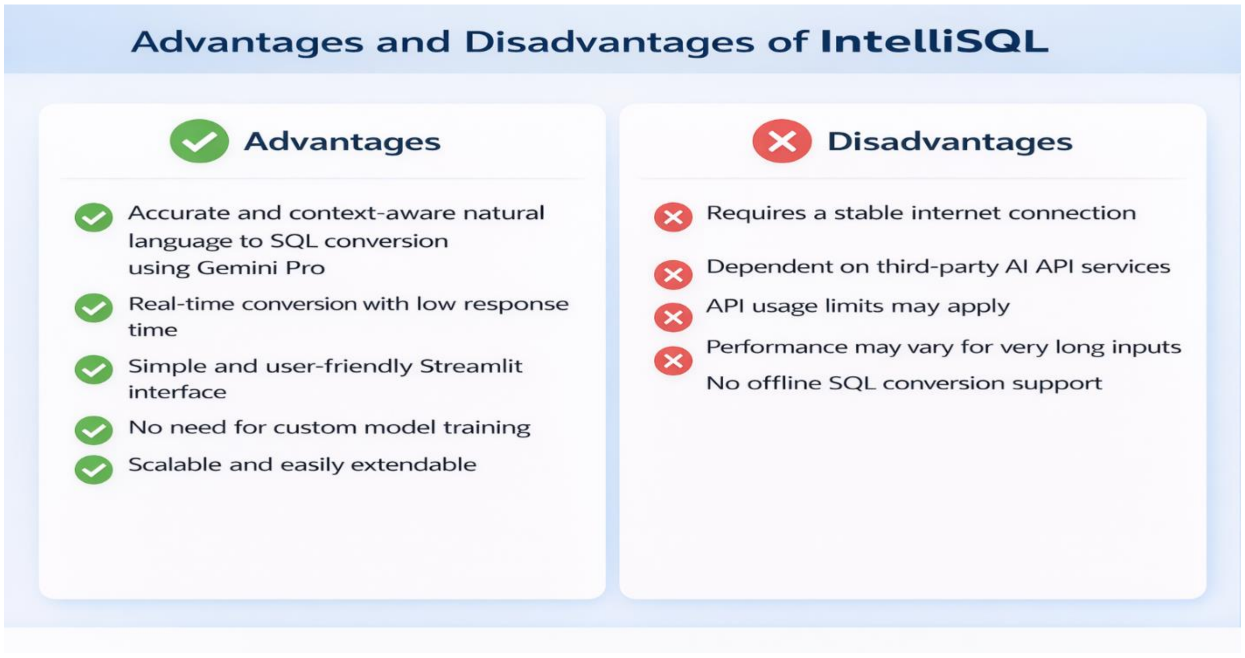
Generated SQL Query: `SELECT * FROM STUDENTS;`

The Response is:

	0	1	2	3
0	Sijo	BTech	75	JSW
1	Lijo	MTech	69	TCS
2	Rijo	BSc	79	WIPRO
3	Sibin	MSc	89	INFOSYS
4	Dilsha	MCom	99	Cyient

Project: *IntelliSQL: Intelligent SQL Querying with LLMs Using Gemini Pro*

7. Advantages & Disadvantages



Advantages and Disadvantages of IntelliSQL

Advantages

- ✓ Accurate and context-aware natural language to SQL conversion using Gemini Pro
- ✓ Real-time conversion with low response time
- ✓ Simple and user-friendly Streamlit interface
- ✓ No need for custom model training
- ✓ Scalable and easily extendable

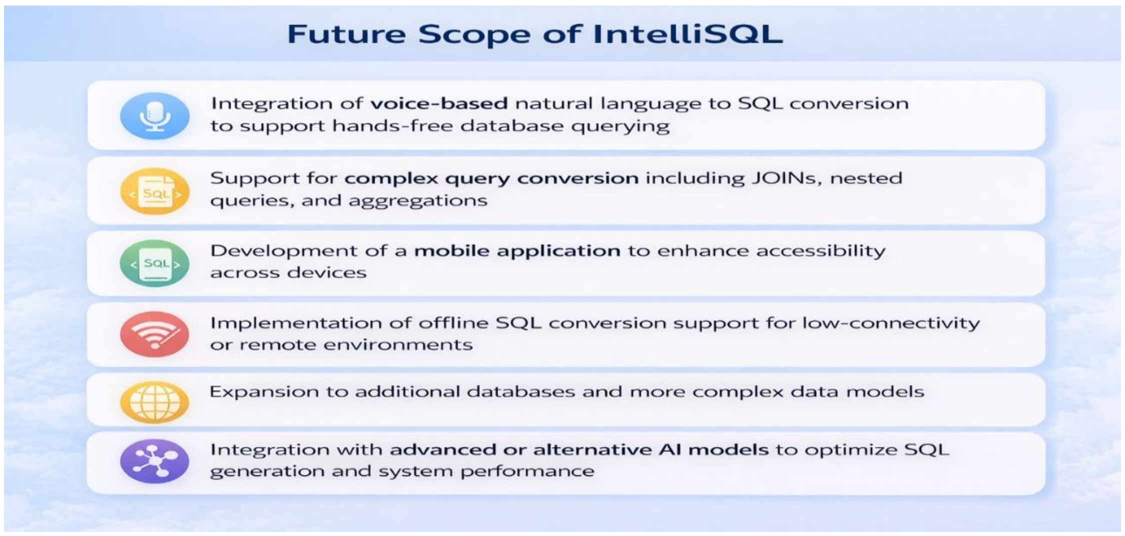
Disadvantages

- ✗ Requires a stable internet connection
- ✗ Dependent on third-party AI API services
- ✗ API usage limits may apply
- ✗ Performance may vary for very long inputs
- No offline SQL conversion support

8. Conclusion:

IntelliSQL successfully demonstrates the application of large language models in solving real-world database querying challenges. By integrating Gemini Pro with a Streamlit-based interface, the project enables accurate and context-aware conversion of natural language into SQL queries while maintaining simplicity and efficiency. The system effectively bridges the gap between non-technical users and structured database systems across academic and professional domains.

Future Scope



Future Scope of IntelliSQL

- Integration of **voice-based** natural language to SQL conversion to support hands-free database querying
- Support for **complex query conversion** including JOINs, nested queries, and aggregations
- Development of a **mobile application** to enhance accessibility across devices
- Implementation of offline SQL conversion support for low-connectivity or remote environments
- Expansion to additional databases and more complex data models
- Integration with **advanced or alternative AI models** to optimize SQL generation and system performance

9. Appendix

9.1. Source Code

```
import streamlit as st
import google.generativeai as genai
from dotenv import load_dotenv
import os

# Load environment variables
load_dotenv()
API_KEY = os.getenv("GOOGLE_API_KEY")

# Configure Gemini API
genai.configure(api_key=API_KEY)

# Initialize Gemini Pro model
model = genai.GenerativeModel("gemini-1.5-flash")

# Function to convert natural language to SQL
def generate_sql_query(user_query, database_schema):
    prompt = f"""
    You are an SQL expert.
    Given the following database schema:
    {database_schema}

    Convert the following natural language query into an SQL query:
    {user_query}

    Provide only the SQL query as output.
    """
    response = model.generate_content(prompt)
    return response.text
# Streamlit UI
st.set_page_config(page_title="IntelliSQL", page_icon="🗄️")
st.title("🗄️ IntelliSQL – Natural Language to SQL Generator")

database_schema = st.text_area("Enter database schema")
user_query = st.text_area("Enter your natural language query")

if st.button("Generate SQL"):
    if not database_schema or not user_query:
        st.warning("Please provide both schema and query")
    else:
        sql_query = generate_sql_query(user_query, database_schema)
        st.subheader("Generated SQL Query")
        st.code(sql_query, language="sql")
```

9.2. GitHub & Project Demo Link

GitHub : https://github.com/venkatasaikuntumalla/IntelliSQL-Intelligent-SQL-Querying-with-LLMs-Using-Gemini-Pro

Demo Link : https://drive.google.com/file/d/1a8a2gkF3y9ePA5p9HrAJHr9Upky7357u/view?usp=sharing