# **Infosys Internship 4.0 Project Documentation**

## **Title: Project Documentation: Data Query and Visualization using LLM**

### **•Introduction:**

The "Data Query and Visualization using LLM" project aims to create an intelligent interface for querying a MySQL database using natural language. The primary objective is to simplify data retrieval for users by leveraging OpenAI's GPT-3.5-turbo model to translate natural language queries into SQL. The project overview includes database creation, SQL query generation using LLM, data visualization with Streamlit, and thorough documentation.

The objective is to create a system that enables users to query a database using natural language and visualize the results. We chose a football database for this project. Its significance lies in helping users query data easily in simple English rather than SQL, making it accessible for anyone to use.

These are some of my team members who collaborated on database creation, connecting with OpenAI, and visualizing the query results:

* Aryan Jadhav
* Akshay Reddy
* Harsha Sharma
* Kartikyae Dhapola
* Keshavi Sharma
* Tamanna

### **•Project Scope:**

**Inclusions:**

* Development of a web interface for natural language queries.
* Integration with OpenAI API for query translation.
* Database schema creation and population with synthetic data.
* Execution of translated SQL queries.
* Visualization of query results.

**Exclusions:**

* Real-time data updates.
* Handling of extremely complex SQL queries.
* Multi-language support beyond English.

**Limitations:**

* Dependency on the accuracy of OpenAI's model for SQL translation.
* Potential performance issues with large datasets.

### **•Requirements:**

**Functional Requirements:**

* Users can input a natural language query.
* The system converts the query to SQL.
* The system executes the SQL query and returns results.
* Visualization of results in various formats (Bar, Line, Scatter).

**Non-Functional Requirements:**

* The system should respond to queries within 5 seconds.
* The application should handle up to 100 concurrent users.
* Data visualization should be clear and intuitive.

**User Stories:**

* As a user, I want to see the results of my query in a visual format so that I can easily interpret the data.
* As a user, I want to query the database using natural language so that I can retrieve information without learning SQL.

### **•Technical Stack:**

Programming Languages: Python, SQL

Frameworks/Libraries: Streamlit, SQLAlchemy, Faker, OpenAI API, Pandas, Matplotlib

Databases: MySQL

Tools/Platforms: MySQL Workbench, Command Shell

### **•Architecture/Design:**

Overview of the system architecture:

### The system consists of a web interface, a backend server, and a MySQL database. The user interacts with the web interface, which sends the natural language query to the backend. The backend uses the OpenAI API to translate the query into SQL, executes the SQL query on the database, and returns the results to the web interface for display and visualization.

### UML Diagram:

A diagram of a server

Description automatically generated

**Design Decisions:**

* Chose Streamlit for its simplicity in building data apps.
* Used SQLAlchemy for ORM to facilitate database interactions.
* Selected OpenAI API for its advanced natural language processing capabilities.
* Implemented data visualization using Pandas and Matplotlib for their robust plotting functions.

**Trade-offs:**

* Streamlit was chosen for rapid development, though it might not be as performant as more robust frameworks like Flask or Django.

### **•Development:**

**Technologies and Frameworks:** Streamlit, SQLAlchemy, Faker, OpenAI API, Pandas, Matplotlib

**Coding Standards:** Followed PEP 8 guidelines for Python code. Used meaningful variable names and modularized code into functions.

**Challenges:**

* Ensuring accurate SQL translation by OpenAI.
* Handling SQL injection and other security concerns.
* Optimizing query execution for performance.

### **•Testing:**

**Approach:**

* Unit tests for individual functions.
* Integration tests for the interaction between components.
* System tests for end-to-end functionality.

**Results:**

* Identified and fixed issues with SQL query formatting.
* Improved error handling for database connections.
* Validated the accuracy of visualizations with sample queries.

### 

### **•Deployment:**

**Process:**

* Ensure that all dependencies are installed using a virtual environment or directly.
* Run the Streamlit application using the command shell.

**Instructions:**

* Clone the repository.
* Navigate to the project directory.
* Install the required dependencies: ‘pip install -r requirements.txt’
* Start the Streamlit application: ‘streamlit run app.py’

### **•User Guide:**

**Setup:**

* Ensure Python and required libraries are installed.
* Follow deployment instructions to start the application.

**Usage:**

* Open the web interface in a browser.
* Enter a natural language query (in normal English language) in the text area.
* Select a visualization type (if desired) and submit.
* View the results and visualization on the page.

**Troubleshooting:**

* If the application is slow, check the database performance.
* Ensure the OpenAI API key is valid and has sufficient quota.

### **•Conclusion:**

### **Project Outcomes:**

* Functional System: Successfully built a system that integrates database management, natural language processing, and data visualization.
* User Engagement: Created a user-friendly interface that allows non-technical users to query and visualize data effectively.

### **Achievements:**

* Effective Database Management: Successfully established a well-structured and accurate database, meeting the objective of efficient data management.
* Accurate SQL Query Generation: Implemented GPT-3.5 for natural language processing, resulting in precise and reliable SQL query generation.
* Creation of User-friendly Interface.

### **Lessons Learned:**

* Learned to create the database using MySQL.
* Understood how to connect the MySQL database with Python using MySQL-connector.
* Integrated OpenAI for query generation.
* Developed a user-friendly interface for visualization and query results in simple English.

### **Areas of Improvement:**

* The user interface can be modified to be more user-friendly.
* Complex database SQL query generation may still be an issue.
* The accuracy of SQL queries generated depends on the accuracy of the query generated by GPT-3.5; more advanced GPT versions could be explored for improvement.

### **\* Appendices:**

**Code Snippets:**

Creating Football database:

A computer screen shot of text

Description automatically generated

Function to generate SQL query:

A computer screen with text on it

Description automatically generated

Function to execute SQL query:

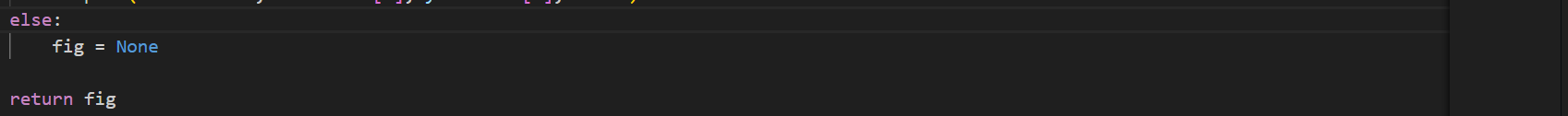
A black screen with text

Description automatically generated

For visualization:

A screenshot of a computer program

Description automatically generated



**Research References:**

* OpenAI API documentation-https://platform.openai.com/docs/introduction
* Streamlit documentation-https://docs.streamlit.io/