CA326
ChatSQL
Functional Specification Document

Georgijs Pitkevics	19355266
Chee Hin Choa	21100497

0. Table of contents

1. Introduction

1.1 Overview

The ChatSQL project emerges as a revolutionary solution aimed at transforming the way users interact with databases. Fueled by the need to simplify database interactions for individuals with minimal to none IT literacy, ChatSQL is a user-friendly assistant designed to facilitate natural language conversations about data and databases without requiring extensive knowledge of SQL.

1.2 Business Context

The ChatSQL project is designed to be the new advancement in user interaction with databases. The target users are those who have little-to-none IT literacy, this makes it a valuable asset for a wide range of users. The project allows users to converse with the application in regular english (natural language) about data and databases, eliminating the need for in-depth knowledge of SQL (Structured Query Language) - the traditional method of accessing databases via SQL prompts which requires 2-3 weeks to become familiar with the basics [1].

The tool is especially beneficial in scenarios where efficient and user-friendly database management and access is required for business and personal purposes. This makes ChatSQL an ideal solution for small businesses, educational institutions, and non-technical individuals in the many sectors that require database interactions without the learning curve of traditional SQL queries.

1.3 Glossary

Define and technical terms used in this document. Only include those with which the reader may not be familiar.

2. General Description

2.1 Product / System Functions

ChatSQL includes a range of functions aiming to simplify SQL database interactions for users with different levels of technical expertise. The core functionality include:

1. User Interaction and Experience:
    a. Database Selection: Users are free to interact with any database they desire.
    b. Natural Language Queries: Users can ask the chatbot questions related to the database using natural language. This entails, the user typing in a statement in the format of a question into a message form.
    c. Real-time feedback: Users are shown a progress indicator for queries that take longer to process, this enhances the transparency of the system's operations and visualizes the expected wait time.
    d. Error detection: If a query is flagged as dangerous by the system, (ie; user entered a query which attempts to delete data from the database, or modify existing data), ChatSQL alerts the user with a friendly notification. The system refuses to enact the SQL statement and then

continues to function as normal after the alert has been sent, asking the user for their next prompt.

2. Large Language Model (LLM):
    a. LLMs are utilized to understand and process the natural language and database tables. When a user gives the program a natural language sentence, the LLM analyzes the text to understand the intent and context.
    b. Once the user intent is understood the LLM translates the natural language coupled with the database table titles into a structured SQL query.
    c. The LLM utilized should be able to handle a wide range of query complexities, from simple data retrievals to intricate data comparison.
3. Data Retrieval:
    a. The system connected to a Large Language Model (LLM) to convert natural language queries into SQL statements.
    b. Responses are presented in natural language form, providing users with easily understandable information.

2.2 User Characteristics and Objectives

User Community:

ChatSQL is made for a diverse user community with varying levels of technical expertise. This includes individuals with minimal IT experience, such as project managers or stakeholders, who may not be familiar with SQL but require access to database information.

User Objectives:

Users aim to interact with databases effectively, extracting meaningful information from databases without diving into complex SQL queries.

User Requirements:

User would be required to upload database files and the ability to send messages, ensuring accessibility for individual users with limited technical knowledge.

Wish List:

Quick Learning Curve: Minimal effort should be required for users to start using the system effectively.
Intuitive User Interface: Users would desire an interface that is simple and easy to use, without any extensive training.

2.3 Operational Scenarios

Scenario 1: Database Query
User Action:
User uploaded and select the desired database.
User asks, "What are the total sales for the last year?"
System Response:
ChatSQL interprets the query using NLP.
Converts it into an SQL statement and runs it on SQL.
Responds with, "The total sales for the last year are $Int."

2.4 Constraints

Lists general constraints placed upon the design team, including speed requirements, industry protocols, hardware platforms, and so forth.

1. Security Measures: The system must prioritize data privacy and prevent any unauthorized modifications to the database.
2. Speed Requirements: The application should provide prompt responses to user queries, ensuring an enjoyable user experience.
3. Compatibility: ChatSQL should be compatible with different databases, considering differences in SQL syntax and structure.

3. Functional Requirements

3.1 User Interaction

3.1.1 Databases

- Description:
  This feature allows users to choose the specific database they intend to query. It is designed to provide flexibility and control, enabling users to interact with their database of choice for data retrieval or manipulation.
- Criticality:
  The ability to select different databases is of high importance, as it directly impacts the system's versatility and user accessibility. It is crucial for accommodating a diverse range of user needs and database environments.
- Technical Issues:
  One of the main challenges is ensuring that the system is compatible with a variety of database types and structures. This requires a robust and adaptable framework capable of interfacing with different database systems, ranging from traditional relational databases to more modern NoSQL databases.

Dependencies with Other Requirements: There are no direct dependencies on other requirements. However, the effectiveness of this feature can influence the system's overall performance and user satisfaction.

Others: Maintaining updated support for new database technologies and ensuring secure connections to different database systems are additional considerations for this requirement.

3.1.2 Natural Language Queries:

- Description:
  The ability of the system to comprehend and handle user inquiries made in natural language is the main focus of this criterion. Instead of using technical or structured query formats, users will be able to phrase queries in everyday English, enabling a natural and clear engagement.

- Criticality:

This feature is vital because it expands system accessibility and defines the fundamental user experience—especially for non-technical query language users. It is a crucial distinction that improves consumer ease.

- Technical Issues:
  Utilising Large Language Models (LLM) to precisely interpret and convert user-inputted natural language queries into useful database queries is the primary difficulty. Because of this, the LLM must possess extremely advanced processing skills to comprehend the variety of linguistic expressions and subtle contextual cues found in natural language.

- Dependencies with Other Requirements:
  While this feature can function on its own, it works best when combined with other features, such as database selection. The correctness of the system's response and error handling depend heavily on how precisely the LLM interprets natural language.

- Others:
  Maintaining a high degree of interpretation accuracy in the face of natural language variability, protecting user input privacy and security, and adjusting the LLM to changing language usage and expressions are some of the major issues.

3.2 Large Language Models (LLM)

1. Conversion to SQL:
   - Description:
     This requirement relates to the system's capacity to convert natural language queries entered by users into structured SQL statements. The system's ability to communicate with databases in response to user commands given in plain language is largely dependent on this procedure.

   - Criticality:
     Translating natural language to SQL is crucial because it creates the link between user input and useful database queries. This function is essential to the system's functionality and efficiency in responding to user queries.

   - Technical Issues:
     Creating a highly effective algorithm or utilising sophisticated Large Language Models (LLM) that can correctly comprehend natural language and translate it into SQL syntax is the main challenge. This entails mapping each query to the appropriate SQL commands and structures while also comprehending the purpose, context, and details of each query.

Dependencies with Other Requirements: This capability depends on the LLM's ability to comprehend natural language inputs operating effectively. For proper SQL conversion, it is essential that the LLM understands the user's purpose and the details of the query.

- Others:
  To ensure that the conversion algorithm or LLM stays up to date with changes in SQL standards and natural language usage, constant efforts are needed to increase its accuracy. Another important factor to keep in mind is striking a balance between the technical specificity needed for SQL statements and query input that is easy for users to understand.
-

2. Understanding Complex Queries:
- Description:
  ChatSQL can understand and execute intricate queries involving interactions with several database tables. The purpose of this feature is to make sophisticated data retrieval and manipulation within a database across various tables easier.

- Criticality:
  There is a medium level of criticality in answering complicated questions. Although it greatly improves the system's ability to handle complex user requirements, it is not as important as the core features of simple query processing.

- Technical Issues:
  The main difficulty is in properly planning and executing SQL joins and table relationships. When queries span multiple tables, the system must be able to recognise and build the appropriate joins and conditions to ensure accurate and effective data retrieval.

- Dependencies with Other Requirements:
  Dependencies with Additional Requirements: The reliability and successful completion of the fundamental processes for converting natural language to SQL are prerequisites for this advanced functionality. The underlying interpretation and conversion mechanisms are expanded upon to handle more complex query structures.

- Others:
  Other factors to take into account are making sure that complex query processing is still easy to use and accessible for people without a lot of SQL expertise, updating the system frequently to accommodate changing database schemas, and optimising query execution for complex queries to prevent performance issues.

3.3 **Data Retrieval**

1. Query Large Language Model (LLM):

- Description: To facilitate the conversion of natural language queries into SQL statements, this requirement calls for the use of a Large Language Model (LLM) in ChatSQL. System and LLM interaction must be efficient for the system to parse user input and generate matching SQL queries.
- Criticality: Accessing the LLM is essential because it is the means by which user queries are converted from natural language into a format that the database can comprehend and process. The operation of the system depends on this feature.
- Technical Issues: Problems: Effectively integrating with the LLM's API and controlling the processing of big datasets are two major obstacles. The system must be able to ensure effective data handling and response times while delivering sophisticated natural language queries to the LLM and receiving SQL statements in return.
- Dependencies with Other Requirements: The fundamental skills of translating natural language inputs into SQL are necessary for this process. These underlying functionalities are directly related to how well the LLM can be queried.
- Others:Others: To adjust to any modifications in the LLM's functionality or API, the integration must be continuously monitored and updated. Maintaining the system's scalability and performance is also a crucial continuous concern, particularly when handling complicated or large-scale queries.

2. Presentation in Natural Language:

- Description: ChatSQL should provide the user with results in natural language following the execution of SQL queries. This entails converting the structured data that was pulled from the database into a user-readable and easily understood format.
- Criticality: Since it immediately affects the user experience, the way query results are presented in natural language is extremely important. It is crucial to make sure users, particularly those who are unfamiliar with reading raw database outputs, can understand the information that is retrieved from the database with ease.
- Technical Issues: Making sure that query results are converted into natural language while maintaining coherence and readability is the main challenge. This calls for complex algorithms or LLM implementations that can understand and present database data in an understandable way.
- Dependencies along with Additional Criteria: SQL queries must run successfully in order for results to be presented in plain language. As the last phase in the user interaction flow, it depends on the precision and efficiency of the steps that came before it.
- Others: Ongoing tasks include refining the natural language generation to enhance clarity and conciseness, adapting to various data formats and types for coherent presentation, and ensuring that the language used aligns with the user's level of technical expertise. Additionally, maintaining a balance between detailed data representation and user-friendly language is crucial for this requirement.

3.4 Data Privacy and Security

1. Read-Only Operations:

Description: ChatSQL should only execute read-only operations on the database.
Criticality: High
Technical Issues: Implement security measures to prevent modifications.
Dependencies: None

2. User Authentication:

Description: Implement a secure user authentication system.
Criticality: Medium
Technical Issues: Design a user authentication mechanism.
Dependencies: Successful completion of security measures.

3.5 Additional Features

1. User-Friendly Interface:

Description: Design an intuitive and user-friendly interface.
Criticality: High
Technical Issues: Incorporate responsive design and user experience principles.
Dependencies: None

2. Error Handling:

Description: Implement effective error handling and provide clear error messages.
Criticality: Medium
Technical Issues: Identify potential error scenarios and develop appropriate responses.
Dependencies: None

4. System Architecture

The architecture of ChatSQL is designed to be modular, flexible, and scalable, emphasizing simplicity and efficiency in its components. Here is a high-level overview of the anticipated system architecture:

4.1 User Interface (UI) Module:

Description: The UI module handles user interactions, providing a seamless and intuitive interface for users to communicate with ChatSQL.

Components:

HTML/CSS for layout and styling.
Javascript for dynamic updates without full page reloads.
TailWindCSS for streamlined styling.
Interaction:

Receives user queries and displays results.
Communicates with the Backend Module for data retrieval.

4.2 Backend Module:

Description: The Backend Module serves as the core of ChatSQL, managing the conversion of natural language to SQL, querying the Large Language Model (LLM), and interacting with the database.

Components:

Flask framework for Python, handling HTTP requests and responses.
Large Language Model (LLM) for natural language to SQL conversion.
Database connector for communication with selected databases.
Interaction:

Receives user queries from the UI Module.
Utilizes the LLM for natural language to SQL conversion.
Executes SQL queries on the chosen database.
Sends results back to the UI Module.
4.3 Natural Language Processing (NLP) Module:
Description: The NLP Module focuses on interpreting and understanding user queries in natural language.

Components:

Natural Language Processing algorithms and libraries.
Interaction:

Integrates with the Backend Module to provide accurate conversion from natural language to SQL.
Handles complex queries and table relationships.
4.4 Security Module:
Description: The Security Module ensures data privacy and protects against unauthorized modifications.

Components:

User authentication mechanisms.
Security protocols to restrict database operations.
Interaction:

Integrates with the Backend Module to enforce read-only operations.
Validates user access rights.
4.5 Database Module:
Description: The Database Module manages the connection to various databases and executes SQL queries.

Components:

Database connector libraries.
Database-specific drivers.

Interaction:

Communicates with the Backend Module to execute SQL queries.
Retrieves data from the database.
4.6 External Services:
Description: External services, if any, may include third-party tools or APIs that enhance functionality.

Components:

Potential integration with Prisma DB services for visualization.
LLM API for natural language to SQL conversion.
Interaction:

May involve API calls for additional functionalities.
Enhances system capabilities through external tools.
4.7 Reused or 3rd Party Components:
HTMX: Utilized in the UI Module for dynamic updates without full page reloads.
TailWindCSS: Employed for streamlined styling in the UI Module.
Flask: Utilized as the web framework for the Backend Module.
Large Language Model (LLM): Integrated into the NLP Module for natural language to SQL conversion.
Third-party database connectors: Used in the Database Module for compatibility with various database types.
4.8 Communication Flow:
User Query:

User interacts with the UI Module, posing queries in natural language.
NLP Conversion:

The UI Module sends user queries to the NLP Module for conversion.
SQL Generation:

The NLP Module converts queries into SQL and sends them to the Backend Module.
Database Interaction:

The Backend Module interacts with the Database Module to execute SQL queries.
Security Checks:

The Security Module ensures read-only operations and validates user access rights.
Result Presentation:

The Backend Module sends query results back to the UI Module for presentation to the user.

5. High-Level Design

This section should set out the high-level design of the system. It should include one or more system models showing the relationship between system components and the systems and its environment. These might be object-models, DFD, etc.

5.1 System Model

The high-level design of ChatSQL is represented through a system model that illustrates the relationships between key components and their interactions with the system and its environment.

System Model

Components:

User Interface (UI):

Receives user queries.
Displays results to the user.
Natural Language Processing (NLP) Module:

Interprets user queries in natural language.
Converts queries to SQL.
Backend Module:

Orchestrates the overall system functionality.
Manages communication between UI, NLP, Security, and Database Modules.
Executes SQL queries.
Security Module:

Ensures data privacy and security.
Validates user access rights.
Database Module:

Connects to various databases.
Executes SQL queries.
Retrieves and stores data.
External Services:

Integrates with third-party tools or APIs for enhanced functionality.
5.2 Data Flow Diagram (DFD)

- The ChatSQL system's Data Flow Diagram (DFD) shows how data moves between different parts, from user input to query execution and result display. The main elements and how they work together are explained as follows:

- User: The person with whom the interaction began. Through the User Interface (UI), users enter their queries, including the names of database tables.

- The user interface, or UI, acts as the front end where users type natural language queries. The user is also presented with query results and system feedback.

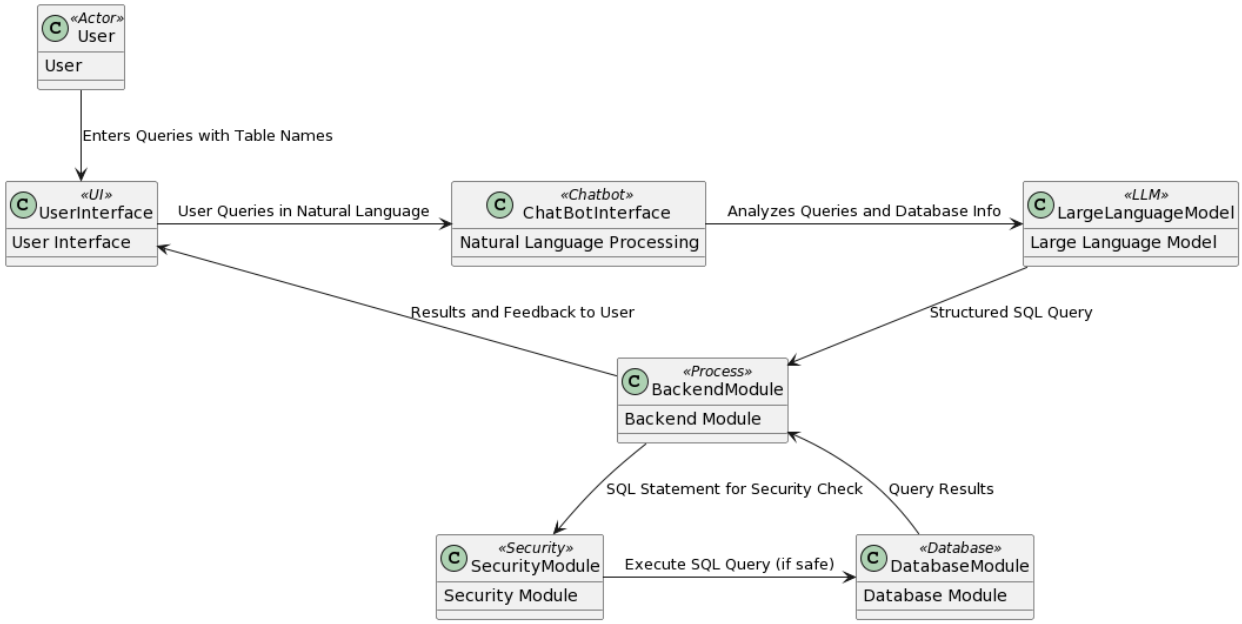
- Natural Language Processing Chatbot Interface: This part handles the user's natural language inquiries. It comprehends and deciphers the query's context as well as the user's intent.

- Large Language Model (LLM): The Chatbot Interface sends the interpreted queries to the LLM. It converts these queries into structured SQL queries by analysing them and the supplied database table names. This translation is essential for transforming natural language that is easy to use into a format that the database system can comprehend and use.

- Backend Module: The Backend Module receives the structured SQL query that the LLM produced. This module serves as a go-between for the database system and the language processing components.

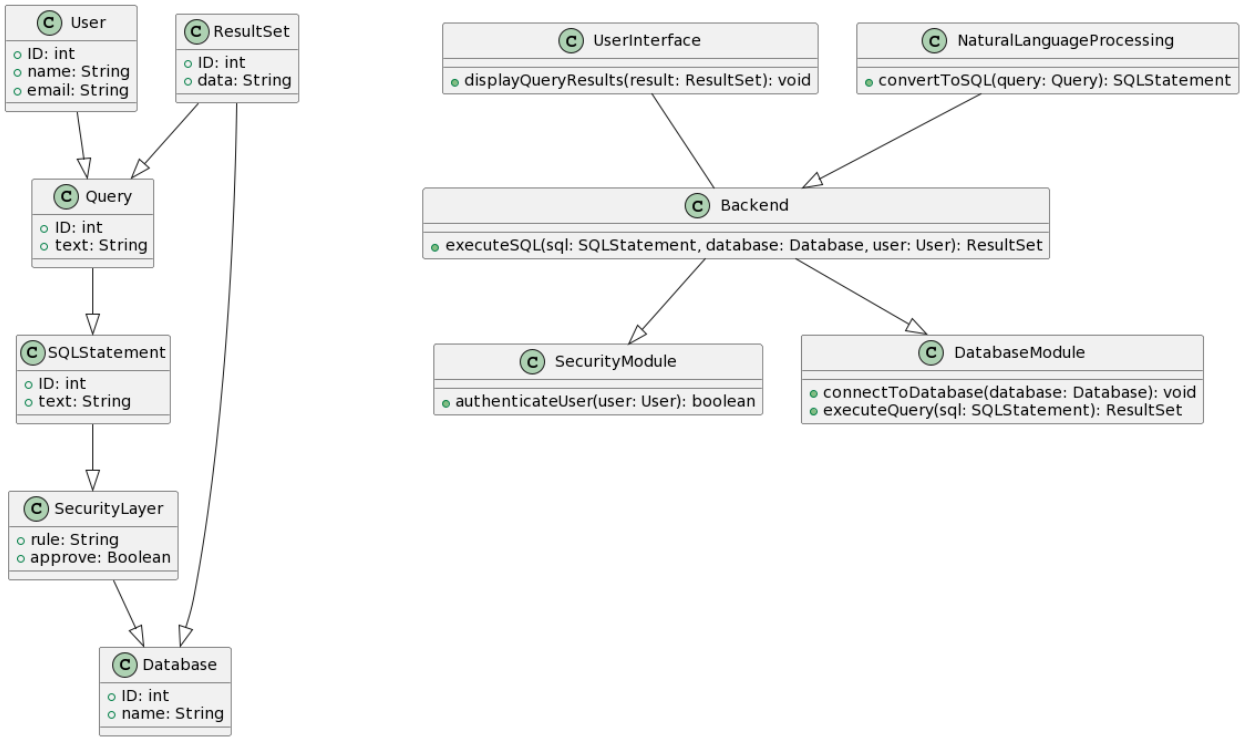
- Security Module: The SQL query passes through a security check in the Security Module prior to being executed. This step is crucial to ensuring that there is no security risk associated with the SQL statement, such as attempts to modify or delete data without authorization.

- Database Module: The Database Module runs the SQL query following the successful completion of the security check. In order to retrieve or modify data in accordance with the query, this module communicates with the real database.
- Data Flow: The figure shows how data moves from the first user input through several processing phases (translation from SQL to natural language, security check), until the query is run and the user is presented with the results via the user interface.



5.3 Object Model

The Object Model provides an overview of the key objects and their relationships within the system.



Object Model

Key Objects:

User:

Represents individuals interacting with the system.
Query:

Represents user queries in natural language.
SQL Statement:

Represents the SQL generated from user queries.
Database:

Represents the various databases available for querying.
Result Set:

Represents the data retrieved from the database in response to user queries.
Security Token:

Represents authentication and user access rights.

5.4 Interaction Flow

The Interaction Flow diagram outlines the step-by-step flow of actions and communication between system components during a typical user interaction.

Interaction Flow

Interaction Steps:

User Query:

User interacts with the UI, posing a query.
NLP Conversion:

The UI sends the user query to the NLP Module.
SQL Generation:

The NLP Module converts the query into an SQL statement.
Database Interaction:

The Backend Module communicates with the Database Module to execute the SQL.
Security Checks:

The Security Module validates user access rights.
Result Presentation:

The Backend Module sends query results to the UI for presentation to the user.

6. Preliminary Schedule

This section provides an initial version of the project plan, including the major tasks to be accomplished, their interdependencies, and their tentative start/stop dates. The plan also includes information on hardware, software, and wetware resource requirements.

The project plan should be accompanied by one or more PERT or GANTT charts.

##Link to compare different between pert chart and gantt chart
I recommend doing pert chart
https://www.forbes.com/advisor/business/pert-chart-vs-gantt-chart/

7. Appendices

Specifies other useful information for understanding the requirements.