DB GUIvolution

Project Team: Yinru Li, Theodore Okamura, Sean Jung

Table of Contents

        - Simplified Sub-system to System interaction

5. System – Analysis Perspective – *Group responsibility*

1. Identify subsystems – analysis point of view
2. System (Tables and Description)
    1. Data analysis
        1. Data dictionary (Table - Name, Data Type, Description)
    2. Process models
3. Algorithm Analysis
    1. Big - O analysis of overall System and Sub-Systems

6. Project Scrum Report - *Group Responsibility*

1. Product Backlog (Table / Diagram)
2. Sprint Backlog (Table / Diagram)
3. Burndown Chart

7. Subsystems

7.1 Subsystem 1 – Name 1 - *Individual responsibility*

1. Initial design and model
    1. Illustrate with class, use-case, UML, sequence ..... diagrams
    2. Design choices
2. Data dictionary
3. If refined (changed over the course of project)
    1. Reason for refinement (Pro versus Con)
    2. Changes from initial model
    3. Refined model analysis
    4. Refined design (Diagram and Description)
4. Scrum Backlog (Product and Sprint - Link to Section 6)
5. Coding
    1. Approach (Functional, OOP)
    2. Language
6. User training
    1. Training / User manual (needed for final report)
7. Testing

7.2 Subsystem 2 – Name 2 - *Individual responsibility*

1. Initial design and model

    1. Illustrate with class, use-case, UML, sequence ..... diagrams
    2. Design choices
2. Data dictionary
3. If refined (changed over the course of project)
    1. Reason for refinement (Pro versus Con)
    2. Changes from initial model
    3. Refined model analysis
    4. Refined design (Diagram and Description)
4. Scrum Backlog (Product and Sprint - Link to Section 6)
5. Coding
    1. Approach (Functional, OOP)
    2. Language
6. User training
    1. Training / User manual (needed for final report)
7. Testing

7.3 Subsystem 3 – Name 3 - *Individual responsibility*

1. Initial design and model
    1. Illustrate with class, use-case, UML, sequence ..... diagrams
    2. Design choices
2. Data dictionary
3. If refined (changed over the course of project)
    1. Reason for refinement (Pro versus Con)
    2. Changes from initial model
    3. Refined model analysis
    4. Refined design (Diagram and Description)
4. Scrum Backlog (Product and Sprint - Link to Section 6)
5. Coding
    1. Approach (Functional, OOP)
    2. Language
6. User training
    1. Training / User manual (needed for final report)
7. Testing

7.4 Subsystem 4 – Name 4 - *Individual responsibility*

1. Initial design and model
    1. Illustrate with class, use-case, UML, sequence ..... diagrams
    2. Design choices
2. Data dictionary
3. If refined (changed over the course of project)

        1. Reason for refinement (Pro versus Con)
        2. Changes from initial model
        3. Refined model analysis
        4. Refined design (Diagram and Description)
    4. Scrum Backlog (Product and Sprint - Link to Section 6)
    5. Coding
        1. Approach (Functional, OOP)
        2. Language
    6. User training
        1. Training / User manual (needed for final report)
    7. Testing

8. Complete System – *Group responsibility*

1. Final software/hardware product
2. Source code and user manual – screenshots as needed - Technical report
    1. Github Link
3. Evaluation by client and instructor
4. Team Member Descriptions

This is just a guide, and use it to create/improve your report. Feel free to add sections. You are responsible for your own subsystem/s, not other members. You have to contribute to the team's goals and objectives, and develop your subsystem/s, write your documents and slides.

1. Project Definition
    a. Why (it is needed):
        i. Traditional database tools are powerful but intimidating for non-experts. There's a need for a user-friendly solution that simplifies the complex task of SQL querying.
    b. What (is the goal of the project):
        i. The goal is to democratize database querying by developing an intuitive application. It will feature a drag-and-drop interface and a natural language model to allow users to create SQL queries without needing to know SQL syntax.
    c. How (how will it be achieved):
        i. With a three-member team, each person will have a distinct role but will also cross-collaborate to ensure cohesiveness. One will focus on front-end and UI/UX design, another on back-end and SQL logic, and the third on implementing the natural language model. We'll use Agile principles to manage tasks, aiming for minimum viable products (MVPs) for iterative testing and improvement.

2. Project Requirments
    a. Functionalities
        i. **GUI-Based Query Interface:**
        Objective: Enable users to construct SQL queries visually.
        Description:
            1. Develop a graphical user interface (GUI) where users can drag and drop shapes representing SQL keywords.
            2. The order and connections between shapes will define the SQL query's structure.
            3. Provide an intuitive and user-friendly interface for query construction.

        ii. **Query Validation**:
        Objective: Ensure the generated queries are valid.
        Description:
            1. Implement a parser and validator to check the syntax and semantics of user-created queries.
            2. Clearly communicate errors and issues with the query to the user.

iii. **GUI-Based Query Result Interface:**
Objective: Present query results in a user-friendly manner.
Description:
1. Display query results in a standard table format.
2. Allow users to trace the source of fields in the result back to their respective tables.
3. Provide specifications and metadata for each field.

iv. **Natural Language Processing (NLP) for Querying:**
Objective: Allow users to enter queries in natural language.
Description:
1. Implement NLP algorithms and language models to interpret and convert natural language queries into SQL.
2. Ensure compatibility with the GUI-based query result interface.

v. **SQL Flavor Support:**
Objective: Adapt to different SQL database systems.
Description:
1. Enable users to choose their desired SQL flavor (e.g., SQLite, MySQL, PostgreSQL).
2. Develop adapters or connectors to handle variations in SQL syntax and functionality.

b. Usability

i. **User Interface**
The usability of our SQL Query and Database Management System is a top priority, as we aim to provide an intuitive and efficient user experience for individuals with varying levels of SQL knowledge. Our user interface design focuses on the following key aspects:
1. Intuitive Drag-and-Drop Interface: Our graphical user interface (GUI) allows users to construct SQL queries using a simple drag-and-drop mechanism. This approach eliminates the need for users to have in-depth SQL knowledge, making it accessible to a broader audience.
2. User-Friendly Visual Elements: We have carefully designed shapes and symbols that represent SQL keywords, ensuring they are easily distinguishable and understandable. Users can seamlessly connect these elements to build complex queries.

3. Real-Time Feedback: The system provides real-time feedback to users during query construction. It highlights any syntax errors or potential issues, allowing users to correct mistakes immediately.
4. Error Handling: We have implemented robust error handling to guide users in rectifying query errors. Error messages are clear and concise, aiding users in understanding and resolving issues effectively.
5. Customization and Personalization: Users can customize the interface to match their preferences, including choosing color schemes and layouts. This personalization enhances the overall user experience.
6. Accessibility: Our interface adheres to accessibility standards, ensuring that it is usable by individuals with disabilities. Features such as keyboard navigation and voice interpreter are possible accessibility features.
7. User Assistance and Tutorials: We offer comprehensive tutorials and tooltips within the interface to help users become proficient in constructing queries and utilizing advanced features.

ii. Performance:
The performance of our SQL Query and Database Management System is optimized to deliver fast and efficient query processing, ensuring users can access and analyze data without delays or bottlenecks. Key performance aspects include:

1. Query Execution Speed: Our system is designed to execute queries swiftly, even when dealing with large datasets. We employ query optimization techniques and indexing to enhance retrieval times.
2. Scalability: The system is built to handle increasing workloads and database sizes. It can efficiently scale up to accommodate the needs of growing organizations and databases.
3. Resource Efficiency: We have optimized resource utilization to minimize memory and CPU usage. This ensures that the system operates smoothly without causing strain on the underlying hardware.
4. Real-Time Updates: For applications requiring real-time data, our system provides near-instant updates, ensuring

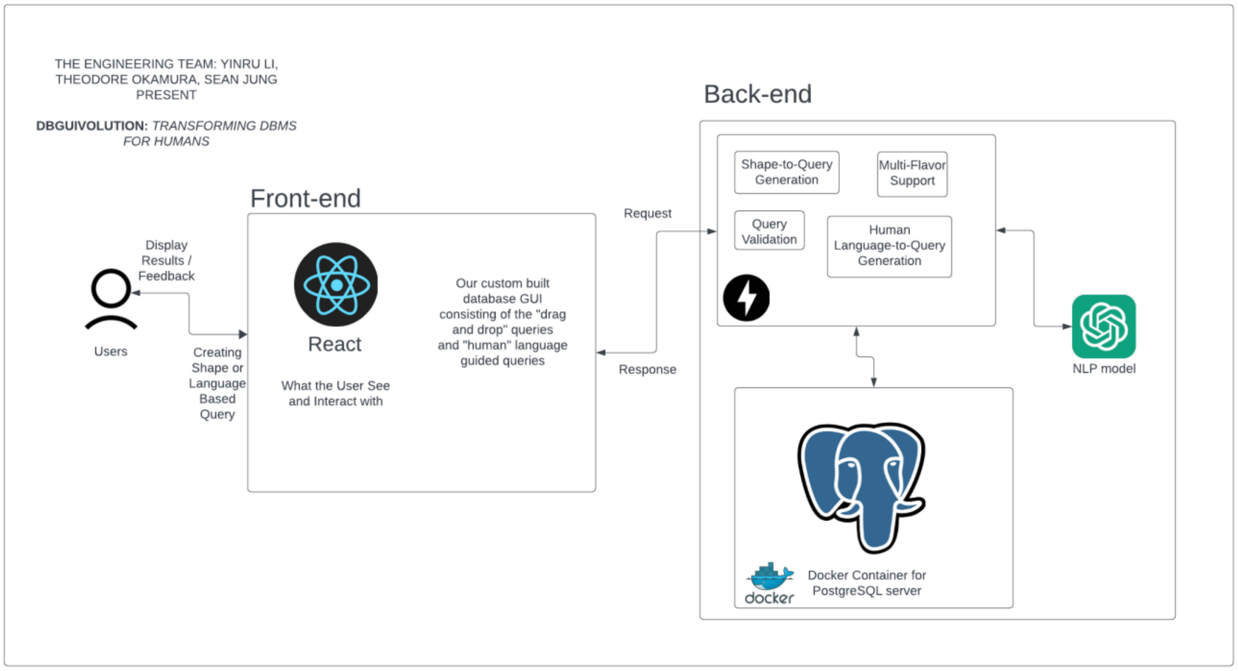
that users receive the most current information without lag.

   iii. By prioritizing both usability and performance, our SQL Query and Database Management System aims to deliver a seamless and efficient experience for users, whether they are novice database users or seasoned professionals. This dual focus ensures that users can interact with their data effectively while enjoying a user-friendly interface and optimal performance.

c. System
   i. Software
      1. WebApp
   ii. Database
      1. MySQL
d. Security
   i. User data will be isolated, queries from one user shall remain classified to other users.

3. Project Specification
   a. Focus / Domain / Area
      i. Develop an intuitive SQL Query and Database Management System emphasizing usability and performance optimization.
   b. Libraries / Frameworks / Development Environment
      i. Our app will be compiled in a node.js environment using React to build the front end. Our backend will be implemented using FastAPI to serve our internal endpoints. We shall use MongoDB to persistently save user data. Additional libraries may be used as the project development is undergone.
   c. Platform (Mobile, Desktop, Gaming, Etc)
      i. Web application
   d. Genre (Game, Application, etc)
      i. Application
   e. Teamwork Division
      i. GUI Based Query Interface – Li
      ii. GUI Based Query Results Interface – Li / Theo
      iii. Query Validation - Theo
      iv. Natural Language Processing - Sean
      v. SQL flavor support – Sean

4. Project Architecture Diagram:


THE ENGINEERING TEAM: YINRU LI, THEODORE OKAMURA, SEAN JUNG PRESENT
DBGUIVOLUTION: *TRANSFORMING DBMS FOR HUMANS*
Front-end
Back-end
Users
Display Results / Feedback
Creating Shape or Language Based Query
React
What the User See and Interact with
Our custom built database GUI consisting of the "drag and drop" queries and "human" language guided queries
Request
Response
Shape-to-Query Generation
Multi-Flavor Support
Query Validation
Human Language-to-Query Generation
NLP model
Docker Container for PostgreSQL server
docker