ATLAN FRONTEND CHALLENGE DETAILED DOCUMENT

ON

SQL QUERY APP

Made By: Prabhakar Yadav

INDEX

Contents

Chapter 1: Introduction

1.1 Project Overview

The SQL Query App is a web-based application designed to assist users in executing SQL queries and visualizing the corresponding results in a user-friendly table format. The application aims to provide a convenient and efficient tool for data analysts and users working with SQL queries.

1.2 Purpose and Scope

The purpose of the SQL Query App is to enable users to run SQL queries without the need for a full-fledged backend or query engine. The application does not include syntax validation and backend processing, as its primary focus is on providing a smooth user experience for executing queries and viewing results.

1.3 Goals and Objectives

The main objectives of the SQL Query App are as follows:

1. Allow users to input SQL queries in the application.
2. Execute the queries and display the query results in a tabular format.
3. Implement additional features such as query bookmarking, history, and exporting.
4. Optimize performance for handling large data sets using virtualization.
5. Provide a user-friendly and responsive interface for ease of use.

The project aims to fulfill these objectives while ensuring simplicity, efficiency, and a smooth user experience.

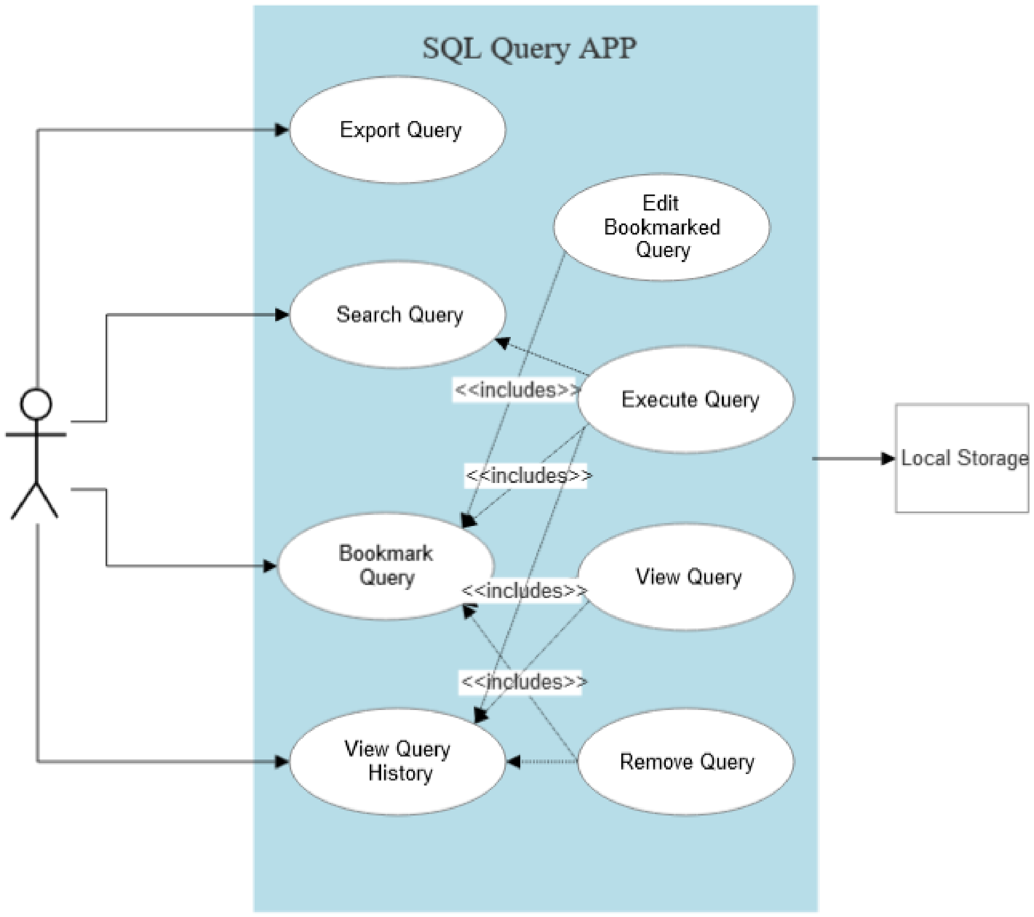
Chapter 2: Project Requirements

The SQL Query App has various requirements, including functional and non-functional aspects, to meet the project's goals effectively.

2.1 Functional Requirements

- **SQL Query Execution**: Users should be able to input SQL queries and execute them, displaying the corresponding results in a table format.
- **Custom Field Validations**: The application should perform custom field validations to ensure users input valid data and format their queries correctly.
- **Virtualization**: The app should efficiently handle large datasets by employing virtualization techniques for smooth rendering and handling of extensive query results.
- **Query Bookmarking**: Users should have the option to bookmark frequently used or important queries for quick access and execution.
- **Query History**: The application should maintain a history of executed queries, enabling users to review and re-run previous queries easily.
- **Query Export**: Users should be able to export query results in various formats, such as CSV or Excel, for further analysis or sharing with others.

2.2 Use-case Diagram



2.3 Non-functional Requirements

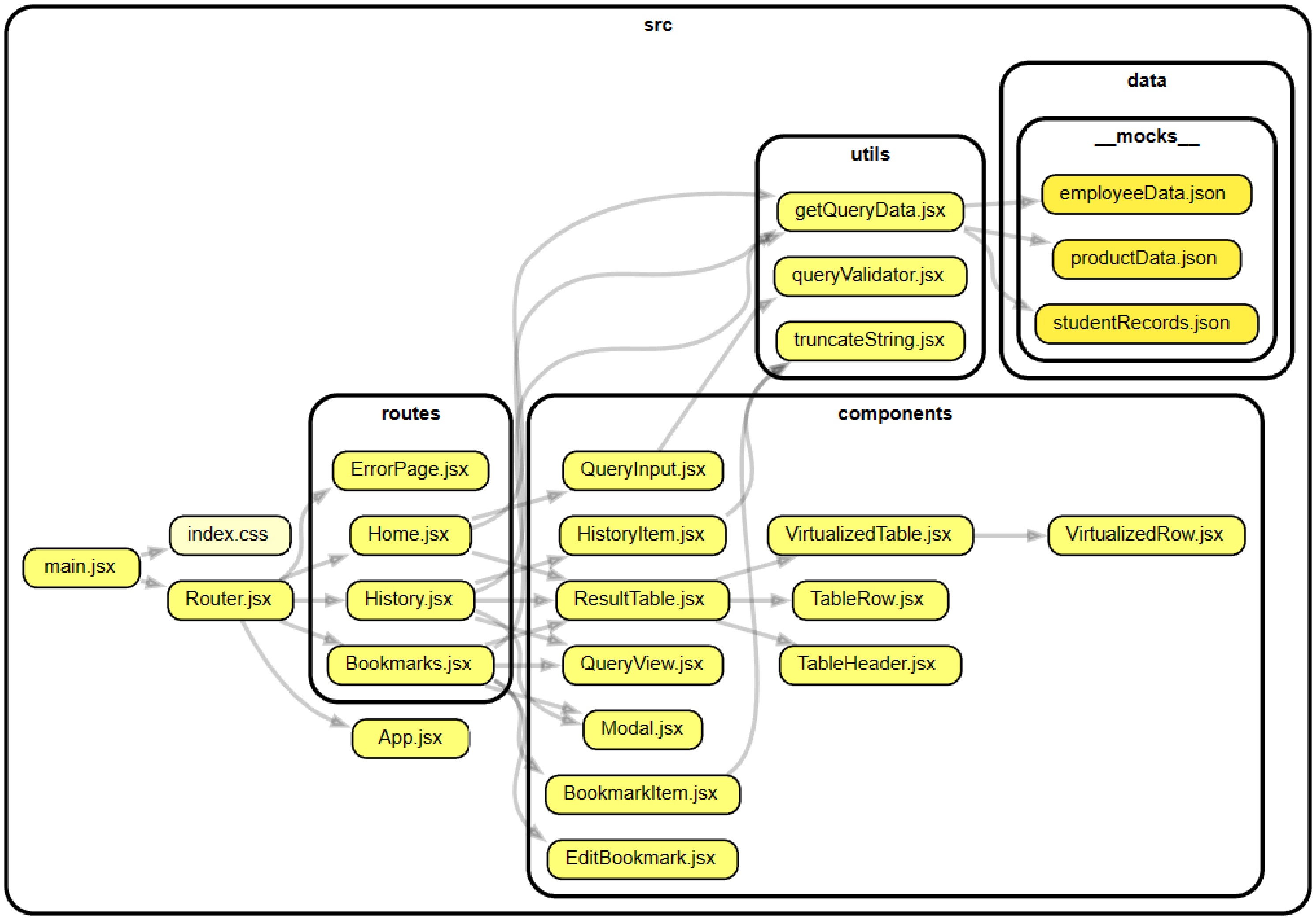
- **Performance**: The application should be highly performant, with quick response times and minimal loading delays.
- **User Experience**: The user interface should be intuitive, visually appealing, and user-friendly, providing a smooth and pleasant experience.
- **Responsiveness**: The app should be responsive and adaptable to various devices and screen sizes.

Chapter 3: Architecture

The SQL Query App is built using a frontend-only architecture, as it does not require a backend server or database for query execution. The frontend is developed using React, a popular JavaScript library for building user interfaces. The application leverages various React components to create a seamless and interactive user experience.

The architecture follows a modular and component-based approach, where different functionalities of the app are encapsulated in individual components. This makes the codebase more maintainable and allows for easy scalability in the future.

3.1 Dependency Graph



3.2 Technologies Used

The SQL Query App is built using modern web technologies to ensure a robust and efficient application. The following technologies are utilized in the development:

- Vite
- React

- GitHub
- Netlify
- Local Storage
- Virtualization
- JavaScript (ES6+)
- CSS

3.3 Project Dependencies

The SQL Query App relies on several external libraries and packages to enhance its functionality and user experience. These dependencies are essential for streamlining development and providing efficient solutions to specific challenges within the application.

3.3.1 External Libraries and Packages

1. **React**: A fundamental JavaScript library for building user interfaces. React's component-based architecture forms the core structure of the SQL Query App, enabling efficient rendering and dynamic UI updates.
2. **react-csv**: This library facilitates data export to CSV format in React applications. It enables users of the SQL Query App to save query results and table data for further analysis or sharing.
3. **react-dom**: Provides DOM-specific methods for React components, crucial for efficiently updating and managing the application's user interface.
4. **react-router-dom**: A collection of navigation components for React applications, enabling multi-page functionality and smoother user flow within the SQL Query App.
5. **react-virtualized-auto-sizer**: A wrapper component that automatically resizes its child to fit the available width and height. It plays a vital role in implementing virtualization techniques for handling large datasets efficiently.
6. **react-window**: This library optimizes rendering performance when dealing with extensive lists and tabular data in React applications. The SQL Query App utilizes it to render the table with exceptional speed and smoothness.
7. **uuid**: A library for generating and working with universally unique identifiers (UUIDs). It assists in creating unique keys and IDs for various components and data elements in the application.

3.4 Project Setup

The SQL Query App is organized using a modular and structured approach. The project follows the React component architecture, with each component responsible for specific functionalities. The main files and directories in the project include:

- **src**: The source directory containing all the React components and application logic.
  - **components**: Contains reusable components such as QueryInput, ResultTable, BookmarkItem, etc.
  - **utils**: Holds utility functions like query validation and data handling.
  - **routs**: Includes .jsx files for differnet routes of the application. Like Home.jsx, History.jsx
  - **data**: Contains dataset for query execution
- **public**: Contains the index.html file, the entry point of the application.
- **package.json**: The file defining project dependencies and scripts.

Chapter 4: User Interface Design

The user interface of the SQL Query App is designed to be intuitive and user-friendly, providing a seamless experience for users. The layout of the app includes the following main components:

1. **Query Input**: A textarea where users can input their SQL queries.
2. **Execute Button**: A button to execute the query and display the results.
3. **Result Table**: A table to present the query results in a tabular format.
4. **Bookmarking**: Users can bookmark frequently used queries for quick access.
5. **History**: A section that maintains a history of executed queries for easy review and re-execution.
6. **Export Options**: Users can export query results in various formats such as CSV or Excel.
7. **Error Handling**: Error messages are displayed for invalid queries or input errors.

4.1 Design Considerations

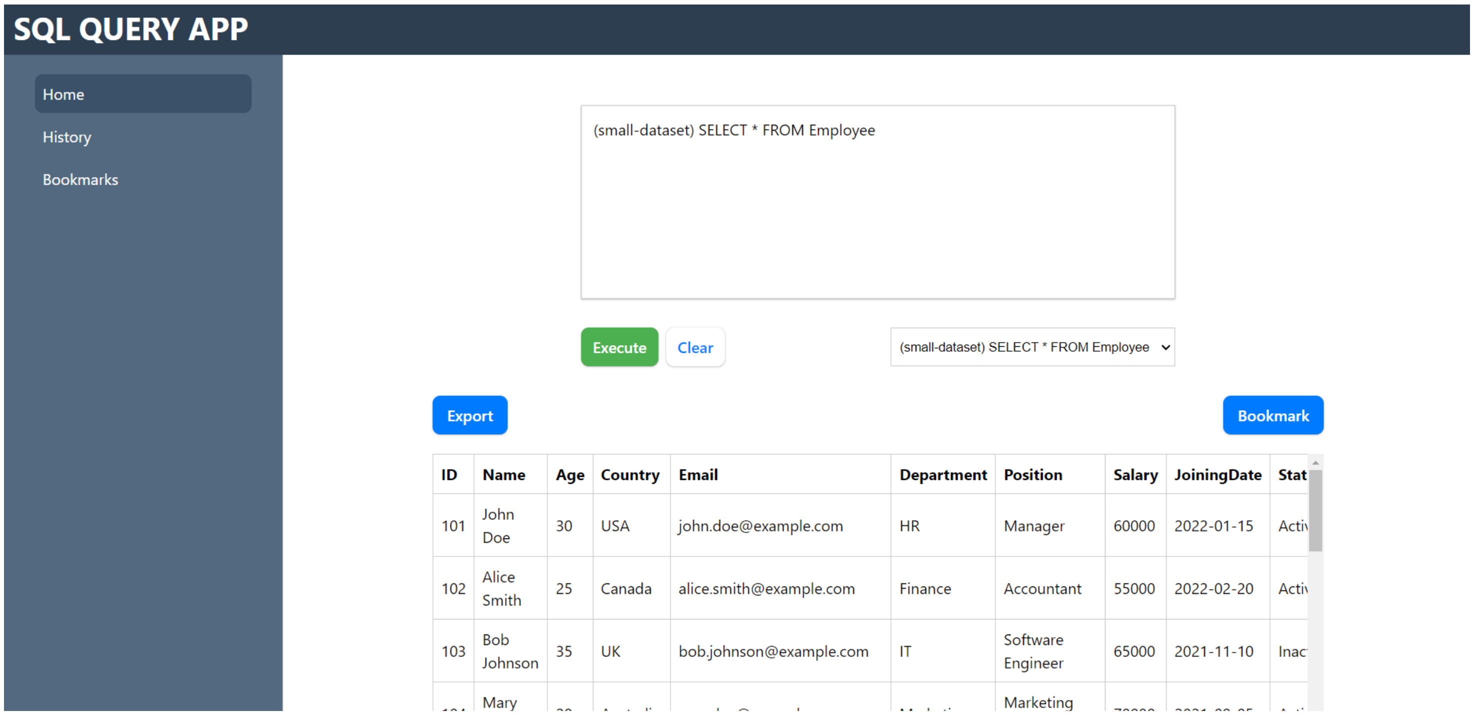
The SQL Query App focuses on providing a simple yet efficient solution for running SQL queries and displaying results. Design considerations include:

- **User Experience**: The user interface is designed to be intuitive and easy to navigate, ensuring a seamless experience for users.
- **Performance**: Virtualization and efficient data rendering techniques are employed to handle large datasets smoothly and reduce page load times.
- **Error Handling**: Custom field validations and error messages help users identify and correct invalid inputs.
- **Responsiveness**: The application is designed to be responsive, adapting to various screen sizes and devices for optimal user interaction.

Chapter 5: Features and Functionality

5.1 SQL Query Execution

Users can input SQL queries in the application, and the system will execute them, displaying the corresponding results in a table format.

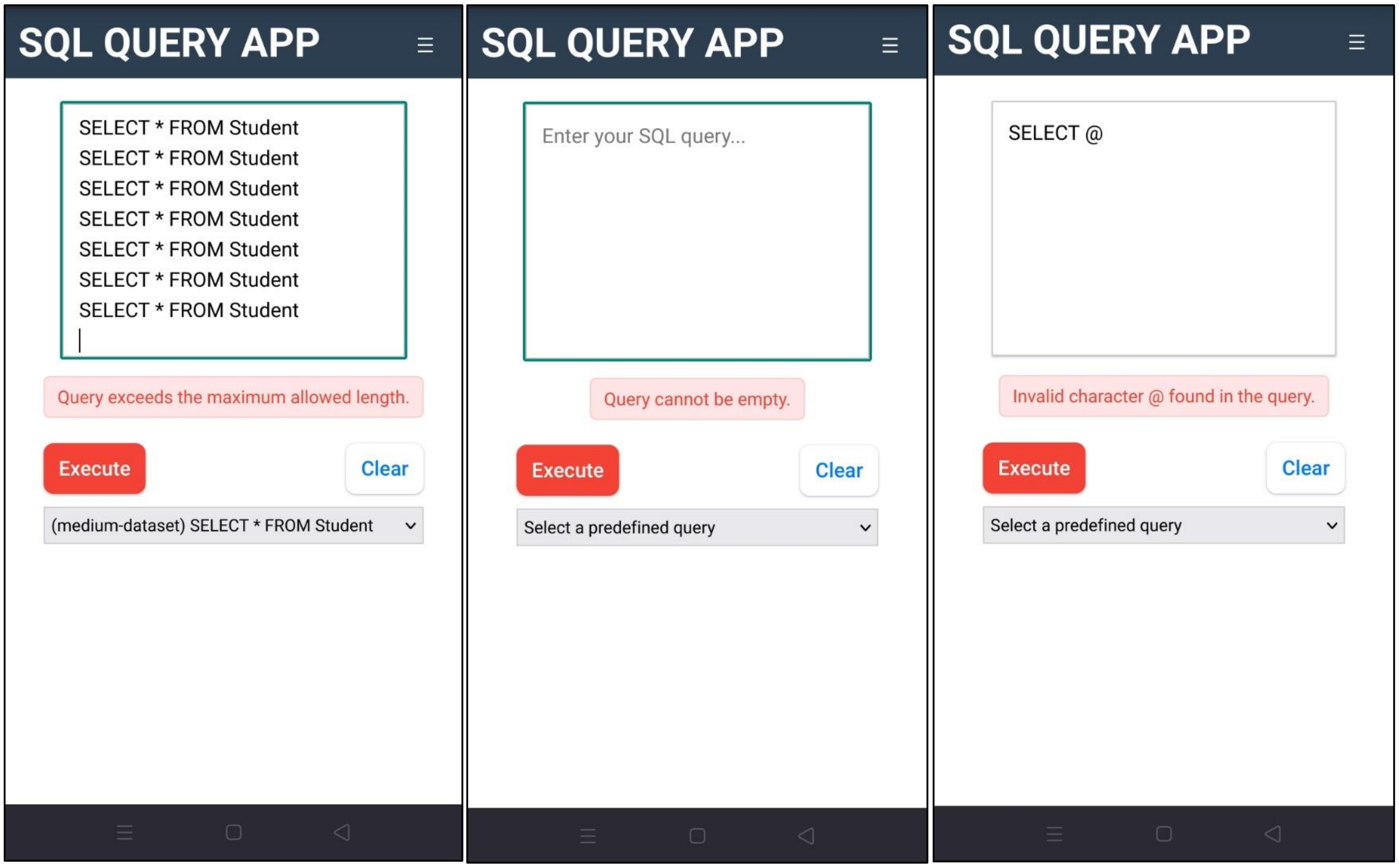


```
(small-dataset) SELECT * FROM Employee
```

ID	Name	Age	Country	Email	Department	Position	Salary	JoiningDate	Stat
101	John Doe	30	USA	john.doe@example.com	HR	Manager	60000	2022-01-15	Acti
102	Alice Smith	25	Canada	alice.smith@example.com	Finance	Accountant	55000	2022-02-20	Acti
103	Bob Johnson	35	UK	bob.johnson@example.com	IT	Software Engineer	65000	2021-11-10	Inac
[illegible]	Mary	[illegible]	[illegible]	[illegible]	[illegible]	Marketing	[illegible]	[illegible]	[illegible]

5.2 Custom Field Validations

The application performs custom field validations to ensure that users input valid data and format their queries correctly. Error messages are displayed for invalid inputs.



```
SELECT * FROM Student
SELECT * FROM Student
SELECT * FROM Student
SELECT * FROM Student
SELECT * FROM Student
SELECT * FROM Student
SELECT * FROM Student
```

Query exceeds the maximum allowed length.

Query cannot be empty.

```
SELECT @
```

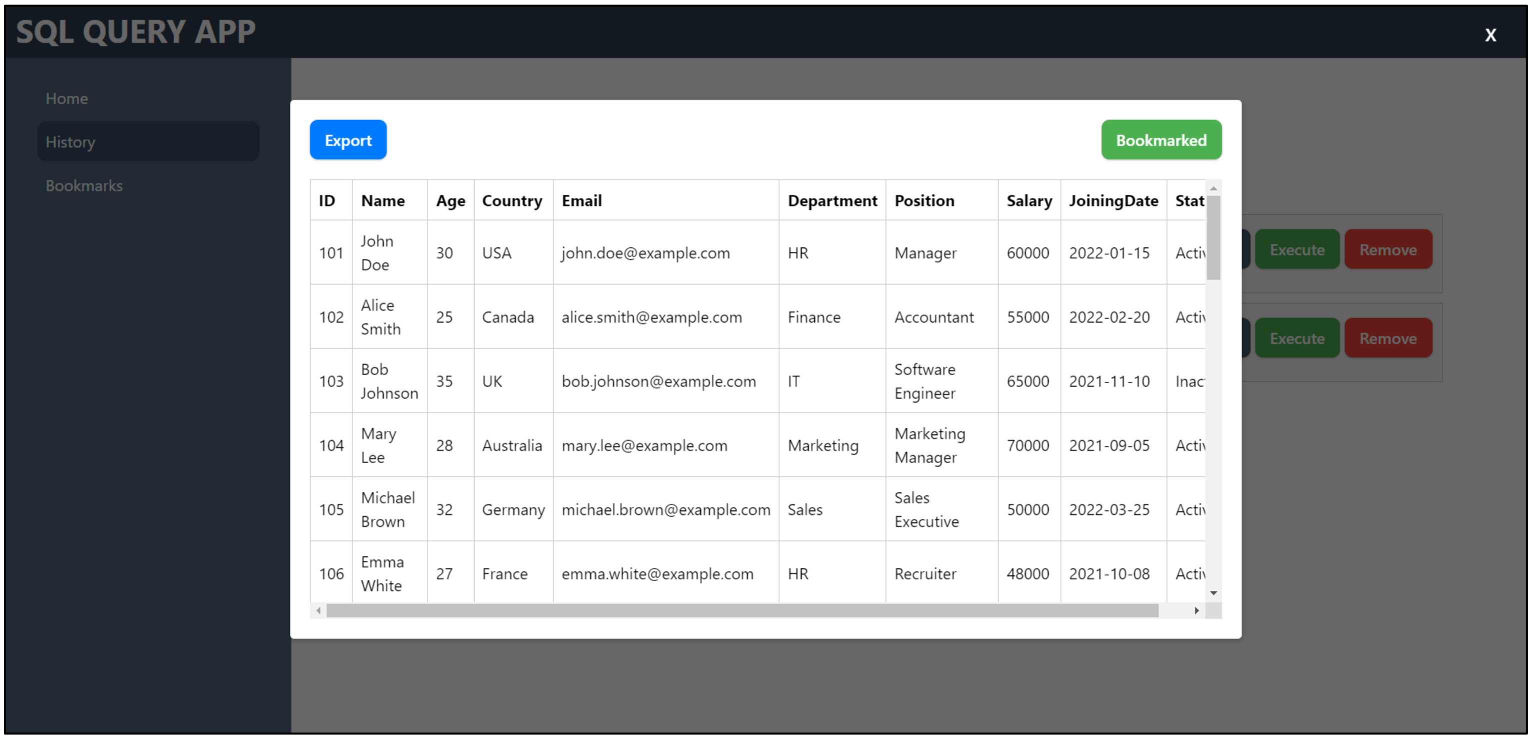
Invalid character @ found in the query.

5.3 Virtualization

Virtualization techniques are implemented to optimize performance when dealing with large data sets, ensuring smooth rendering and handling of extensive query results.

5.4 Query Bookmarking

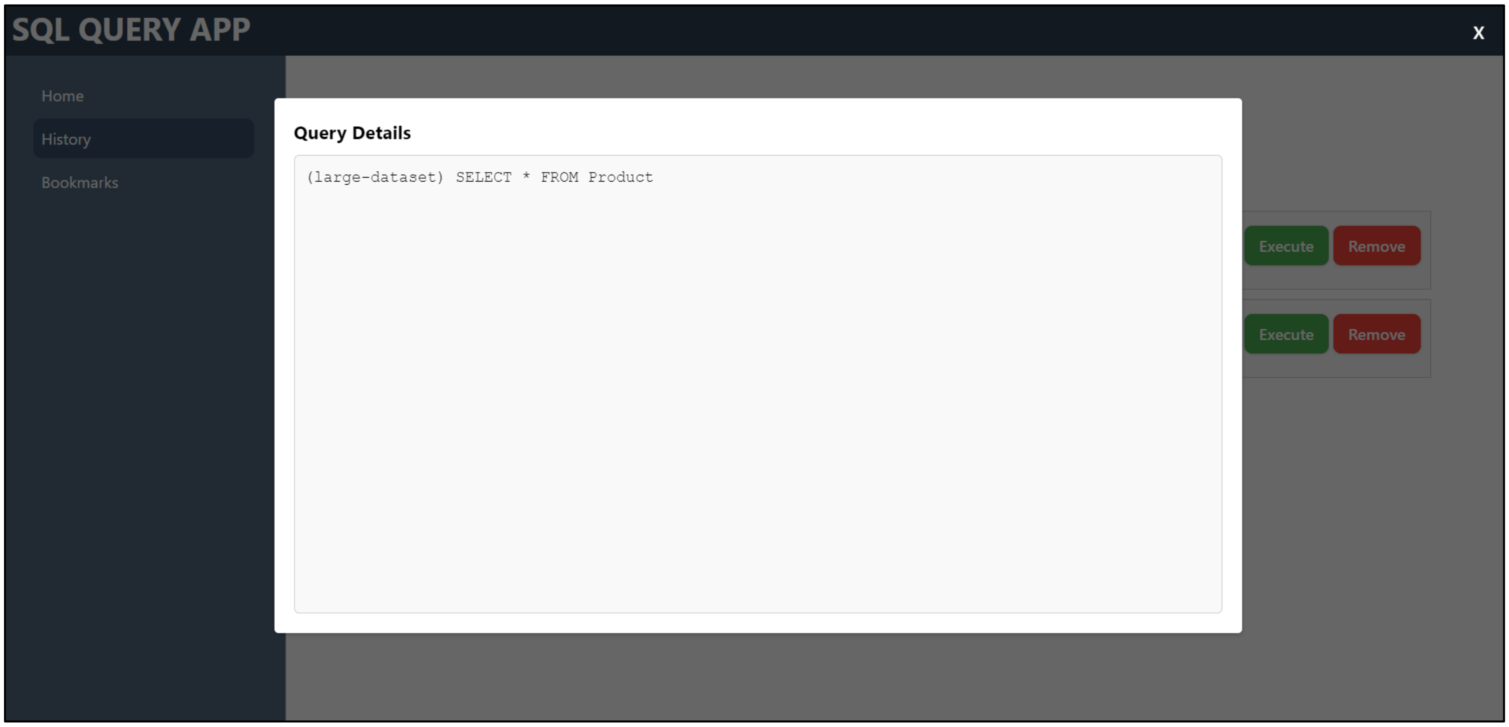
Users can bookmark frequently used or important queries, allowing them to access and execute them quickly without typing the entire query again.



ID	Name	Age	Country	Email	Department	Position	Salary	JoiningDate	Stat
101	John Doe	30	USA	john.doe@example.com	HR	Manager	60000	2022-01-15	Acti
102	Alice Smith	25	Canada	alice.smith@example.com	Finance	Accountant	55000	2022-02-20	Acti
103	Bob Johnson	35	UK	bob.johnson@example.com	IT	Software Engineer	65000	2021-11-10	Inac
104	Mary Lee	28	Australia	mary.lee@example.com	Marketing	Marketing Manager	70000	2021-09-05	Acti
105	Michael Brown	32	Germany	michael.brown@example.com	Sales	Sales Executive	50000	2022-03-25	Acti
106	Emma White	27	France	emma.white@example.com	HR	Recruiter	48000	2021-10-08	Acti

5.5 Query History

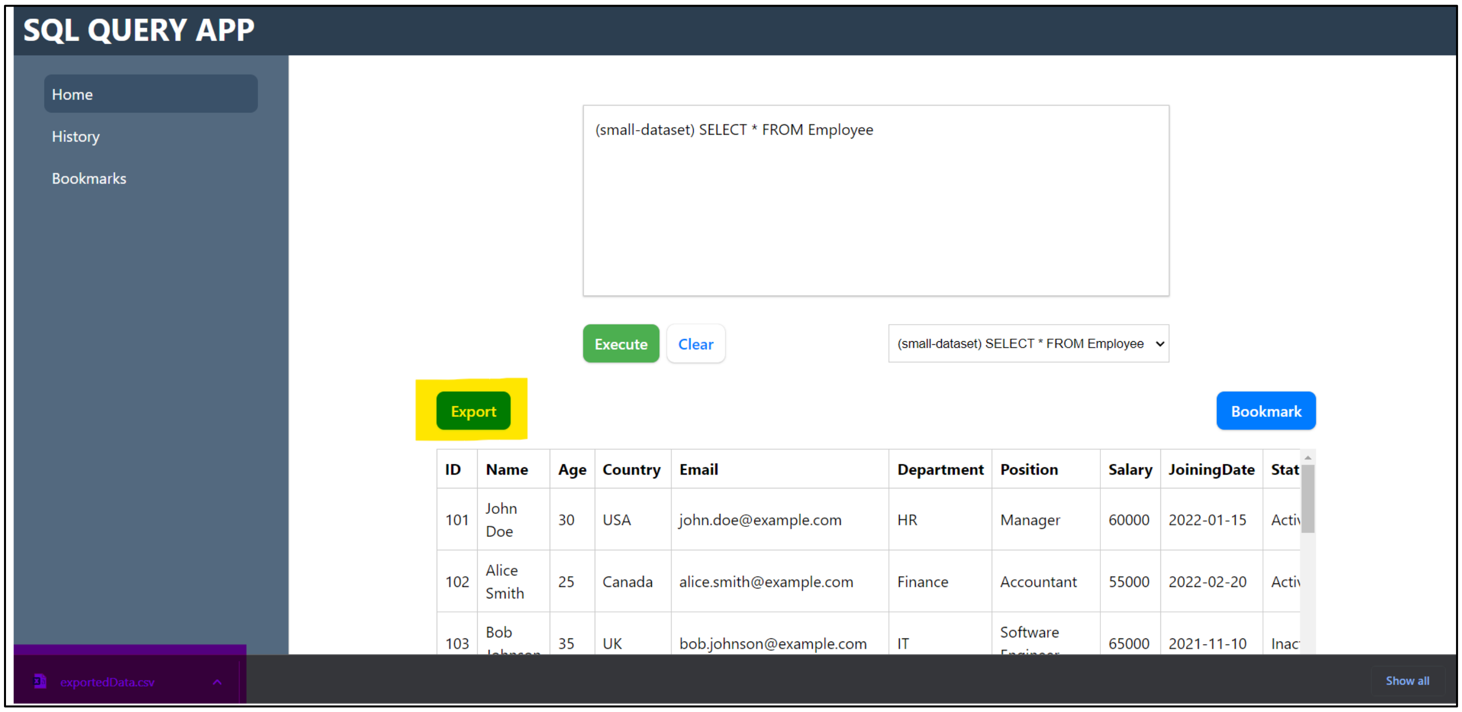
The application maintains a history of executed queries, enabling users to review and re-run previous queries easily.



```
(large-dataset) SELECT * FROM Product
```

5.6 Query Export

Users have the option to export query results in various formats, such as CSV or Excel, for further analysis or sharing with others.



(small-dataset) SELECT * FROM Employee

(small-dataset) SELECT * FROM Employee

ID	Name	Age	Country	Email	Department	Position	Salary	JoiningDate	Stat
101	John Doe	30	USA	john.doe@example.com	HR	Manager	60000	2022-01-15	Acti
102	Alice Smith	25	Canada	alice.smith@example.com	Finance	Accountant	55000	2022-02-20	Acti
103	Bob	35	UK	bob.johnson@example.com	IT	Software	65000	2021-11-10	Inac

Chapter 6: Performance Optimization

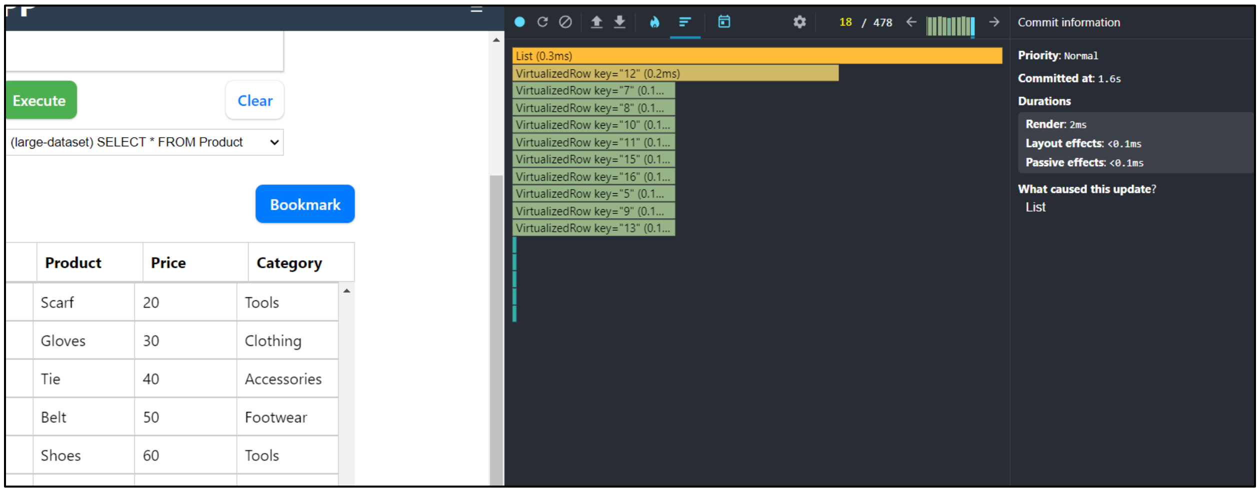
To ensure the SQL Query App's optimal performance, various measures have been taken to enhance speed and efficiency.

6.1 Virtualized Table Rendering

One of the key performance optimizations is the use of react-window to implement virtualization for the large dataset table. By rendering only, the visible rows at any given time, virtualization significantly reduces the rendering load, leading to smoother scrolling and improved performance. This approach ensures that even with extensive data, the application maintains high responsiveness.

6.2 Rendering Time Analysis

To analyze the rendering performance of the application, React Profiler was utilized. This tool allowed for the measurement of rendering time for different components, including the large dataset table. As a result, it was observed that the entire table is rendered in just 2ms, with each virtual row rendering in an impressive 0.1-0.2ms. This exceptional rendering efficiency ensures a seamless user experience, even with substantial data.



Product	Price	Category
Scarf	20	Tools
Gloves	30	Clothing
Tie	40	Accessories
Belt	50	Footwear
Shoes	60	Tools

```
List (0.3ms)
VirtualizedRow key="12" (0.2ms)
VirtualizedRow key="7" (0.1...
VirtualizedRow key="8" (0.1...
VirtualizedRow key="10" (0.1...
VirtualizedRow key="11" (0.1...
VirtualizedRow key="15" (0.1...
VirtualizedRow key="16" (0.1...
VirtualizedRow key="5" (0.1...
VirtualizedRow key="9" (0.1...
VirtualizedRow key="13" (0.1...
```

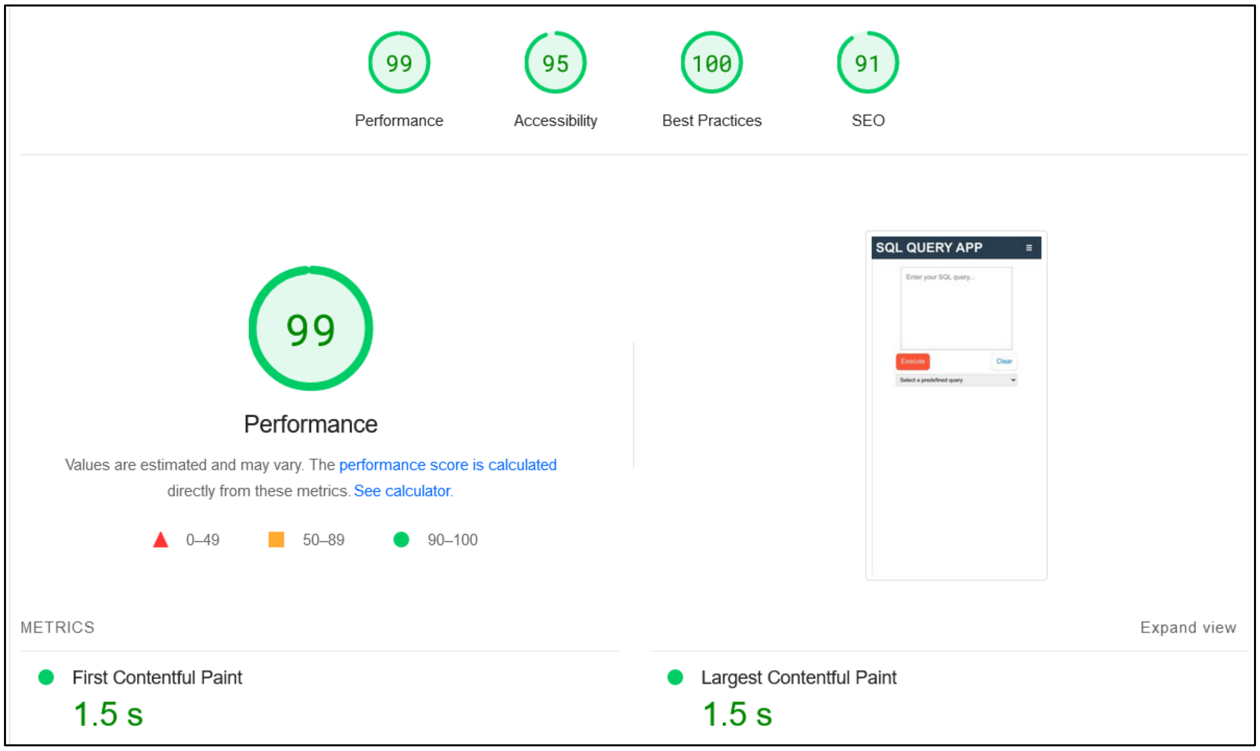
Priority: Normal
Committed at: 1.6s
Durations
Render: 2ms
Layout effects: <0.1ms
Passive effects: <0.1ms
What caused this update?
List

6.3 Page Load Time

The SQL Query App's page load time was assessed using Google PageSpeed Insights. The tool provided valuable metrics to gauge the application's performance. Notably, the key performance metrics were as follows:

- **First Contentful Paint (FCP)**: 1.5 seconds
- **Largest Contentful Paint (LCP)**: 1.5 seconds
- **Total Blocking Time (TBT)**: 30 milliseconds
- **Cumulative Layout Shift (CLS)**: 0
- **Speed Index**: 3.1 seconds

These metrics demonstrate the application's quick load times and efficient performance, meeting the goal of providing users with a smooth and responsive experience.



99	95	100	91
Performance	Accessibility	Best Practices	SEO

99

Performance

Values are estimated and may vary. The performance score is calculated directly from these metrics. See calculator.

▲ 0–49 ■ 50–89 ● 90–100

METRICS Expand view

● First Contentful Paint
1.5 s

● Largest Contentful Paint
1.5 s

Chapter 7: Data Storage and Management

In the SQL Query App, data storage and management are handled through the utilization of the browser's local storage feature. Local storage is a client-side storage mechanism that allows the application to store data persistently on the user's device. This approach offers several benefits, including easy data retrieval, reduced server load, and improved application performance.

7.1 Local Storage Implementation

The SQL Query App employs local storage to store two main types of data:

1. **Query History**: When users execute queries, the application stores the query history locally. This enables users to access and review their previous queries at any time, even after closing the application. The query history provides users with a convenient way to recall and re-run queries they have used before.
2. **Query Bookmarks**: Users can bookmark frequently used or important queries using the application's bookmark feature. These bookmarks are also stored in the local storage, allowing users to quickly access and execute their saved queries without having to re-enter them.

7.2 Benefits of Local Storage

By utilizing local storage for data storage and management, the SQL Query App gains several advantages:

- **Offline Access**: Since local storage is available even when the user is offline, the application can still function and display previously executed queries and bookmarks.
- **Quick Retrieval**: Data stored in local storage can be quickly retrieved and displayed within the application, reducing loading times and enhancing user experience.
- **Reduced Server Load**: By storing data locally, the application reduces the need to make frequent requests to the server, leading to decreased server load and improved overall system performance.
- **User Privacy**: Local storage data is accessible only to the user on their device, ensuring data privacy and security.

Chapter 8: Deployment and Hosting

The SQL Query App is deployed and hosted using Netlify, a popular platform that simplifies the process of deploying modern web applications and static websites. Netlify offers a straightforward deployment methodology and a reliable hosting environment, ensuring that the application is accessible to users across the globe.

8.1 Deployment Methodology

The deployment process follows a continuous integration and continuous deployment (CI/CD) workflow. Whenever changes are pushed to the main branch of the GitHub repository, Netlify automatically triggers a build process. This build process compiles the React code, optimizes assets, and creates a production-ready version of the SQL Query App.

Netlify's integration with GitHub streamlines the deployment process, allowing for rapid updates and continuous improvements to the application. The deployment methodology ensures that the latest version of the SQL Query App is always accessible to users without manual intervention.

8.2 Hosting Environment

Netlify provides a scalable and reliable hosting environment for the SQL Query App. The application is hosted on Netlify's global content delivery network (CDN), which ensures fast loading times and low latency for users worldwide. The CDN caches static assets and content, reducing server load and improving overall performance.

Netlify's robust hosting infrastructure guarantees high availability and uptime, minimizing the risk of downtime and ensuring that users can access the SQL Query App at any time. Additionally, Netlify's SSL certificate management ensures secure and encrypted communication between users and the application.

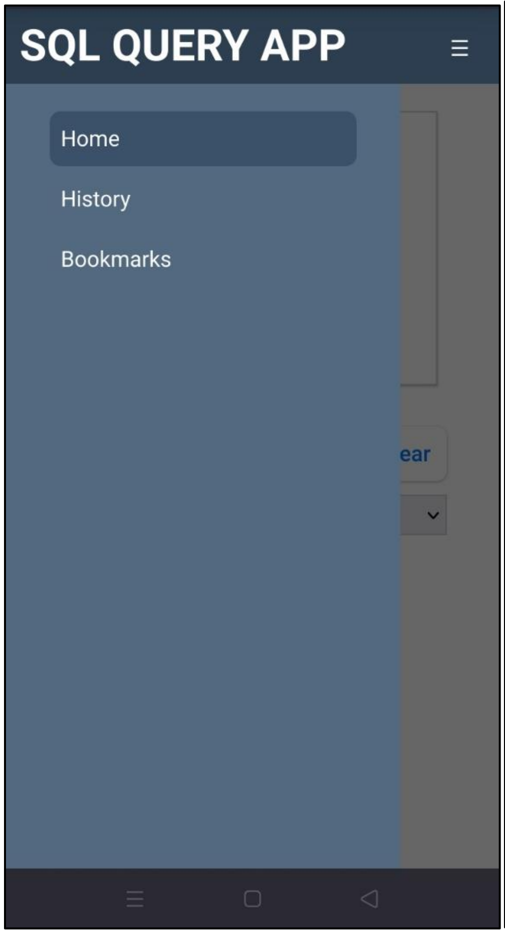
Chapter 9: Future Enhancements

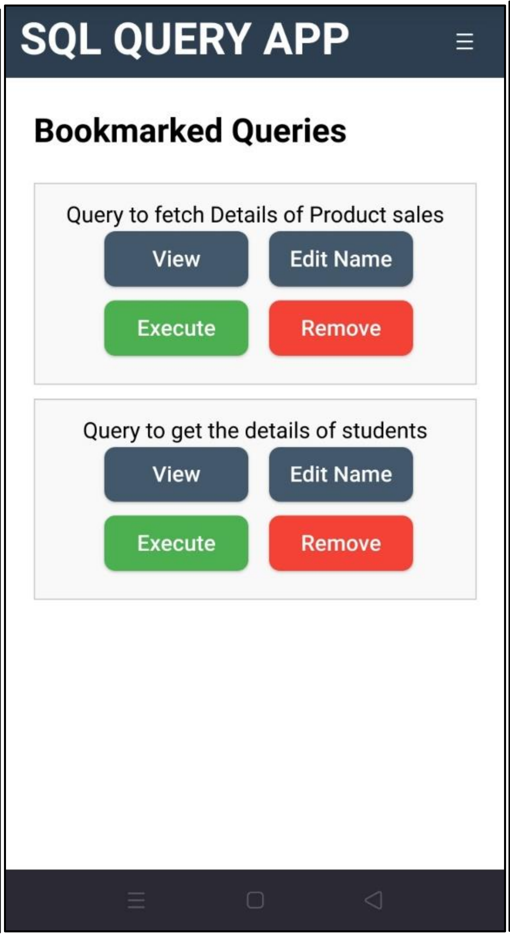
As the SQL Query App evolves and gains feedback from users, several planned features and improvements are envisioned to enhance its functionality and user experience. The development team is committed to implementing these enhancements in future updates to further enrich the application's capabilities. Some of the planned features and improvements include:

1. **Syntax Highlighting**: Implementing syntax highlighting for SQL queries within the input field, aiding users in identifying query elements and improving the overall query-writing experience.
2. **Query Suggestions**: Providing intelligent query suggestions as users type, based on historical queries and frequently used patterns, to expedite query creation.
3. **Query Execution History**: Enhancing the query execution history by displaying detailed information, such as execution time and result count, to help users analyze query performance.
4. **Export Customization**: Allowing users to customize the export format and layout of query results, tailoring exports to meet specific reporting and analysis requirements.
5. **User Authentication**: Introducing user authentication and account management, enabling users to save preferences, share queries with teammates, and securely access their query history across devices.
6. **Query Performance Insights**: Introducing performance insights and recommendations, highlighting potential query optimization opportunities and suggesting ways to improve query execution times.
7. **Collaboration Features**: Enabling collaborative query editing and sharing among multiple users, facilitating team collaboration on complex SQL queries and analyses.
8. **Dark Mode**: Introducing a dark mode option to provide users with a visually comfortable interface during extended usage periods and in low-light environments.
9. **Custom Themes**: Allowing users to customize the application's visual appearance, empowering them to create personalized themes and adapt the interface to their preferences.
10. **Real-Time Query Results**: Implementing real-time updates for query results, enabling users to monitor and analyze continuously updating data sources in real time.

Chapter 10: Screenshots and Visual

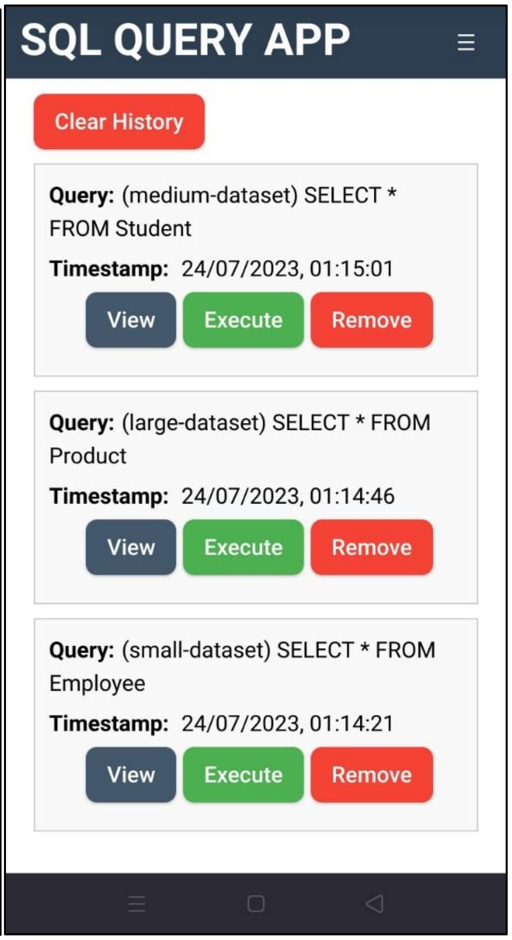
10.1 Mobile Menu and Responsive interface





Query to fetch Details of Product sales

Query to get the details of students

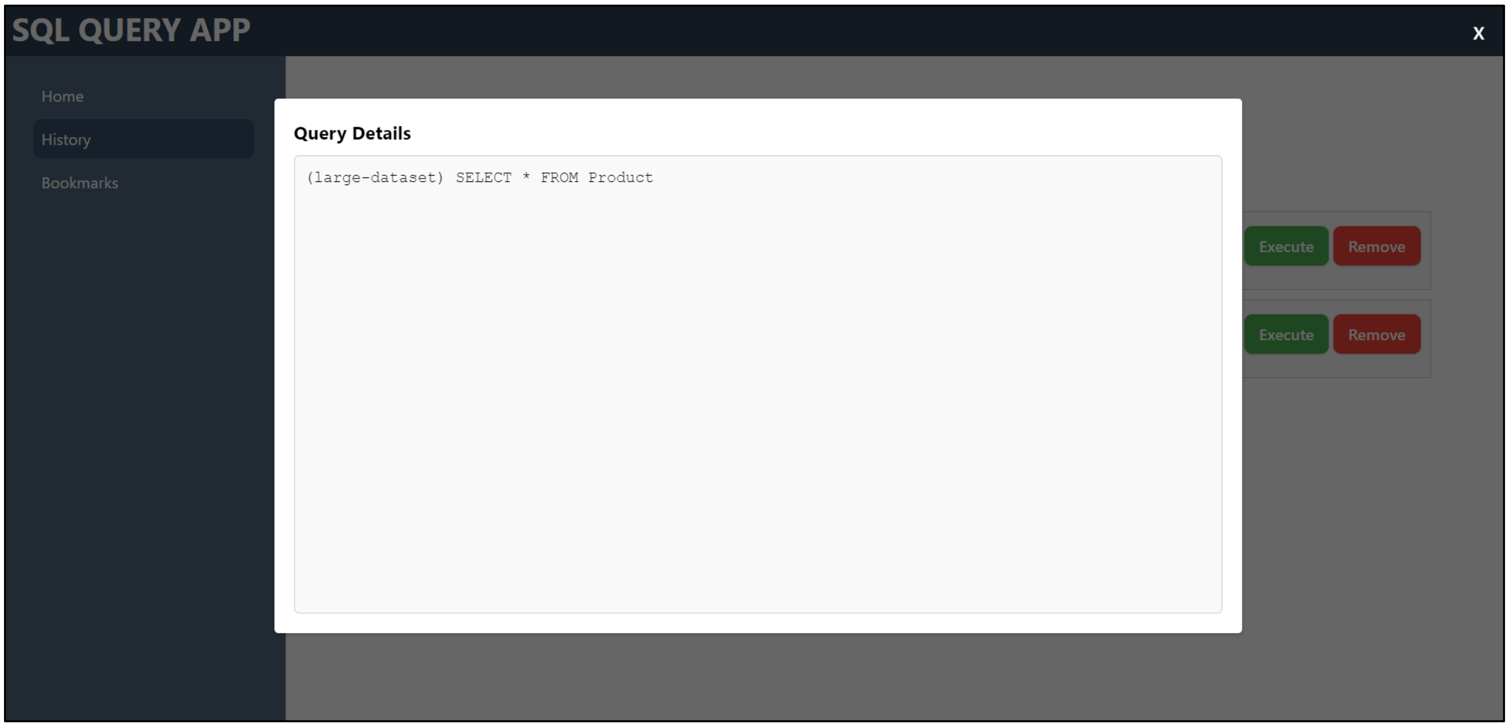


Query: (medium-dataset) SELECT * FROM Student
Timestamp: 24/07/2023, 01:15:01

Query: (large-dataset) SELECT * FROM Product
Timestamp: 24/07/2023, 01:14:46

Query: (small-dataset) SELECT * FROM Employee
Timestamp: 24/07/2023, 01:14:21

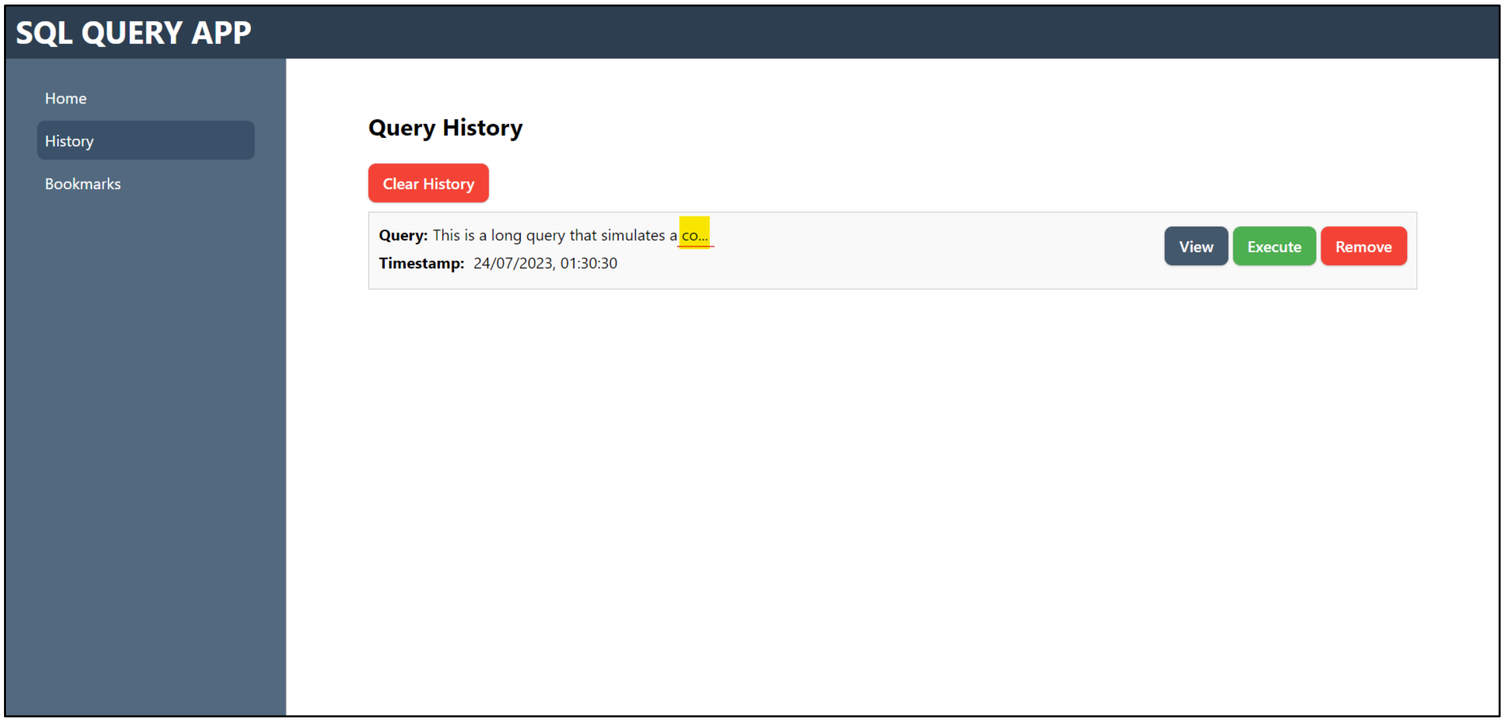
10.2 View Queries in history and bookmarks



Query Details

```
(large-dataset) SELECT * FROM Product
```

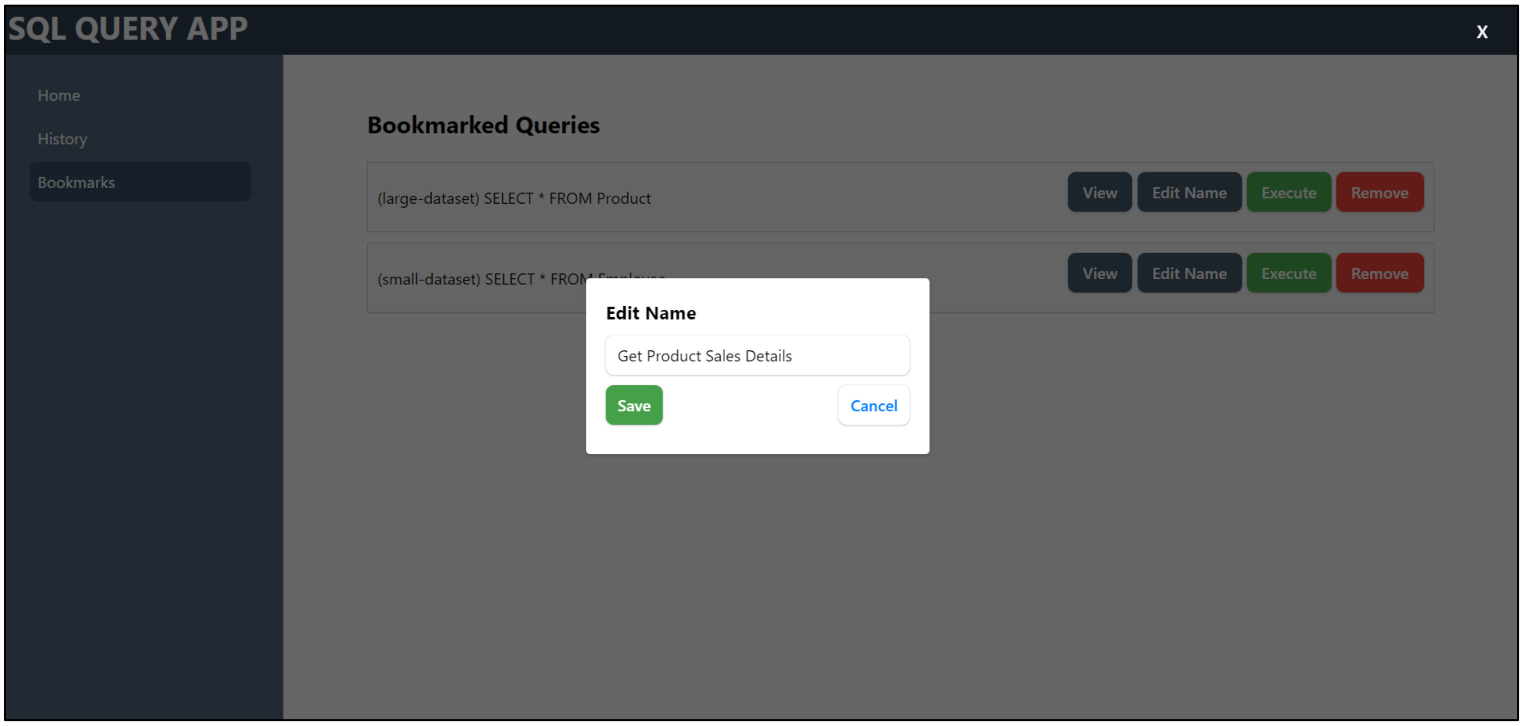
10.3 Large Query Size Trim in History:



Query History

Query: This is a long query that simulates a co...
Timestamp: 24/07/2023, 01:30:30

10.4 Bookmark Rename



Bookmarked Queries

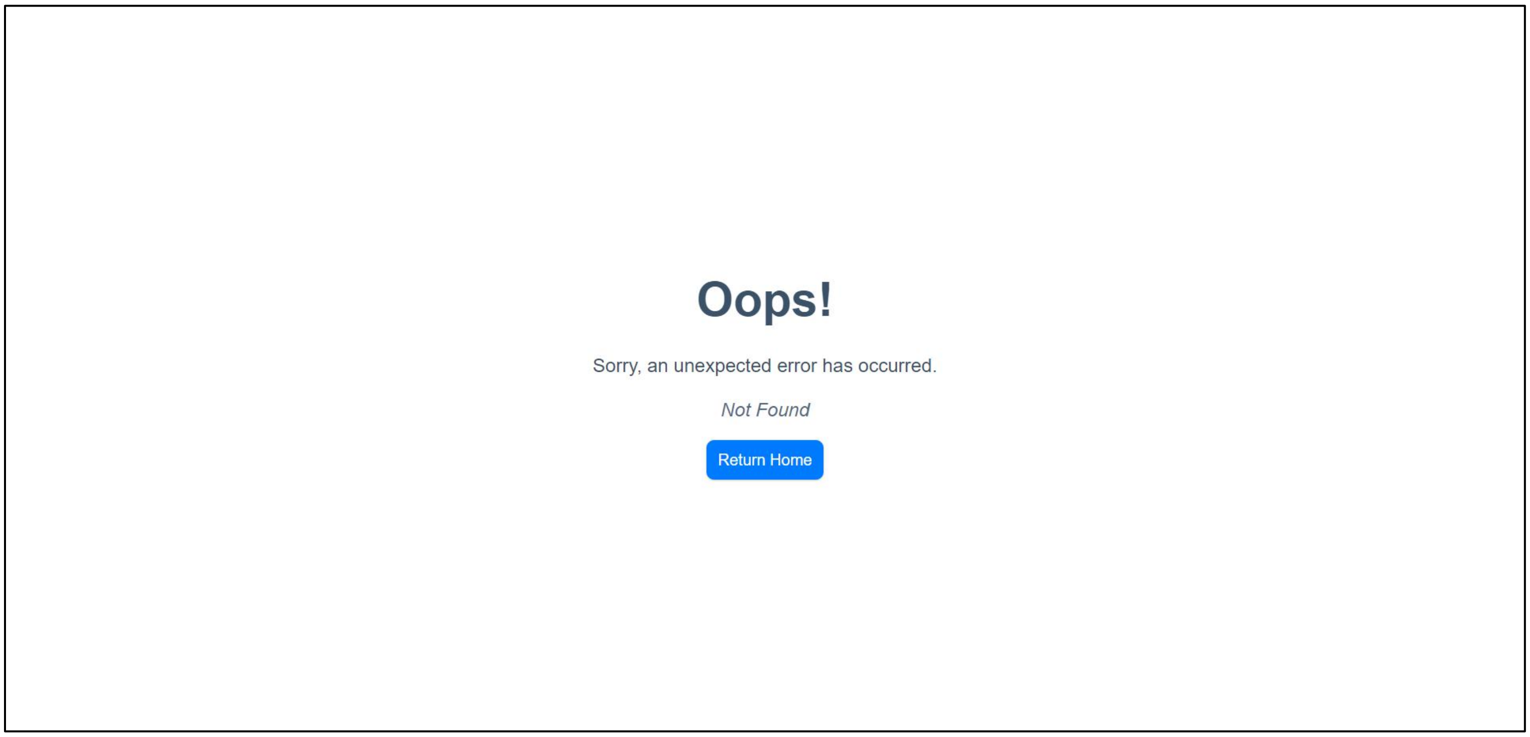
(large-dataset) SELECT * FROM Product

(small-dataset) SELECT * FROM Employee

Edit Name

Get Product Sales Details

10.5 Error Page



Oops!

Sorry, an unexpected error has occurred.

Not Found

Chapter 11: Conclusion

The SQL Query App is a user-friendly and efficient web-based application that facilitates the execution and display of SQL queries in a table format. With features like custom field validations, query bookmarking, and query exporting, the application offers a comprehensive tool for data analysts and developers.

The project's use of modern technologies, virtualization, and optimized UI design ensures a smooth and responsive user experience, even with extensive datasets.