# Qzinsights - End-to-End Flow Documentation

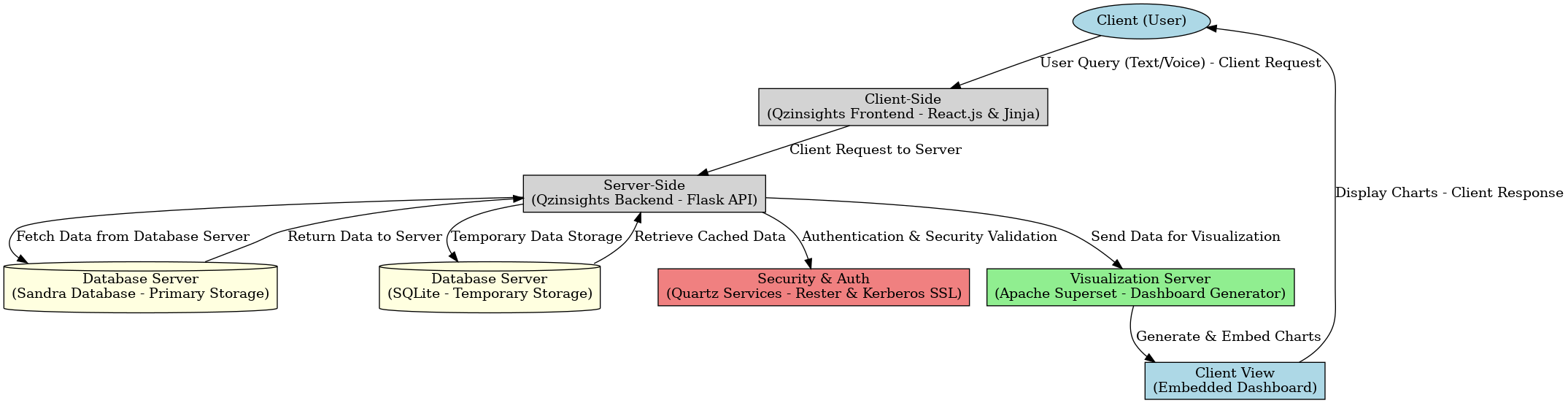
## 1. Introduction

Qzinsights is a web application that allows users to query a database via text or voice input. The system fetches data from the Sandra database (Quartz platform) and dynamically generates charts using Apache Superset. The final visualizations are embedded into the web application for user-friendly data exploration.

## 2. System Architecture

Qzinsights follows a \*\*Client-Server Microservices architecture\*\*. The main components include:   
- \*\*Client (User):\*\* Interacts with the system via a web interface.  
- \*\*Client-Side:\*\* React.js for Apache Superset and Jinja for Qzinsights UI.  
- \*\*Server-Side:\*\* Flask handles request processing and database interactions.  
- \*\*Database Server:\*\* Sandra (primary) for persistent storage, SQLite for temporary data caching.  
- \*\*Security & Authentication:\*\* Quartz services (Rester, Kerberos SSL) ensure secure authentication.  
- \*\*Visualization Server:\*\* Apache Superset dynamically generates charts and dashboards.  
- \*\*Client View:\*\* Embedded Dashboard for user-friendly interaction.

The following architecture diagram illustrates the client-server interaction and data flow:



## 3. Workflow

The end-to-end workflow of Qzinsights is as follows:

- User (Client) visits the Qzinsights web application.

- User enters a query using text input or voice search.

- Client-Side (React.js & Jinja) forwards the request to the Server-Side (Flask API).

- Server-Side processes the request and fetches relevant data from Sandra Database.

- If needed, temporary data is stored in SQLite.

- Server-Side communicates with Quartz for authentication and validation.

- Server-Side sends the retrieved data to Apache Superset for visualization.

- Superset generates the charts and embeds them into the Dashboard.

- Client View (Dashboard) displays the charts to the user.

## 4. Technology Stack

- \*\*Client-Side:\*\* React.js (for Apache Superset), Jinja (for Qzinsights UI)

- \*\*Server-Side:\*\* Flask (Python-based web framework)

- \*\*Database Server:\*\* Sandra (primary), SQLite (temporary storage)

- \*\*Security & Authentication:\*\* Quartz services (Rester, Kerberos SSL for secure authentication)

- \*\*Visualization Server:\*\* Apache Superset (for dynamic chart creation)

- \*\*Client View:\*\* Embedded Dashboard (React-based UI)

## 5. Security Considerations

Qzinsights follows a strict security model by implementing \*\*Kerberos SSL authentication\*\* to ensure secure database access and communication between microservices. Additionally, all user queries and results are processed securely to prevent unauthorized data access.

## 6. Deployment & Maintenance

The system is deployed using \*\*containerized microservices\*\* to ensure scalability and reliability. The backend and frontend services run independently, and regular updates are pushed via CI/CD pipelines.