FitQuest Design Document

Developer: Alexandru Mazilu

**1. System Overview**

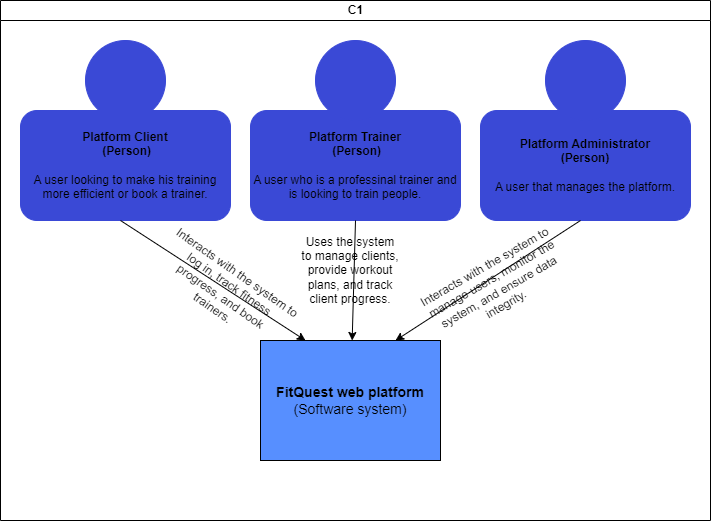
**FitQuest** is a scalable and secure fitness tracking application that enhances the interaction between users, trainers, and administrators. The system provides personalized workout plans and a comprehensive dashboard for each user role. The platform is built using modern web technologies such as **React.js** for the frontend, **Spring Boot** for the backend, and **MySQL** for the database, ensuring performance, security, and data integrity.

**2. Architecture Overview**

FitQuest is designed to be modular and scalable, divided into three main tiers: Frontend, Backend, and Database. The architecture supports multiple roles, including clients, trainers, and administrators, with each role having access to tailored features. The platform also supports real-time notifications via WebSockets.

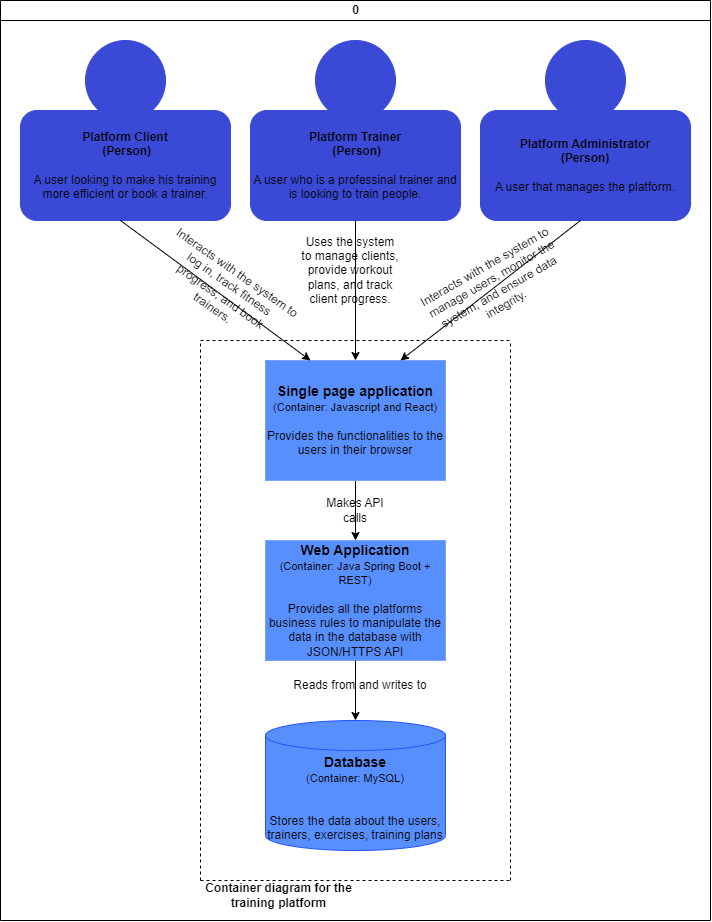
**2.1 C1 Diagram (System Context)**

The C1 diagram shows the overall system, highlighting interactions between users (clients, trainers, and admins) and the FitQuest platform.

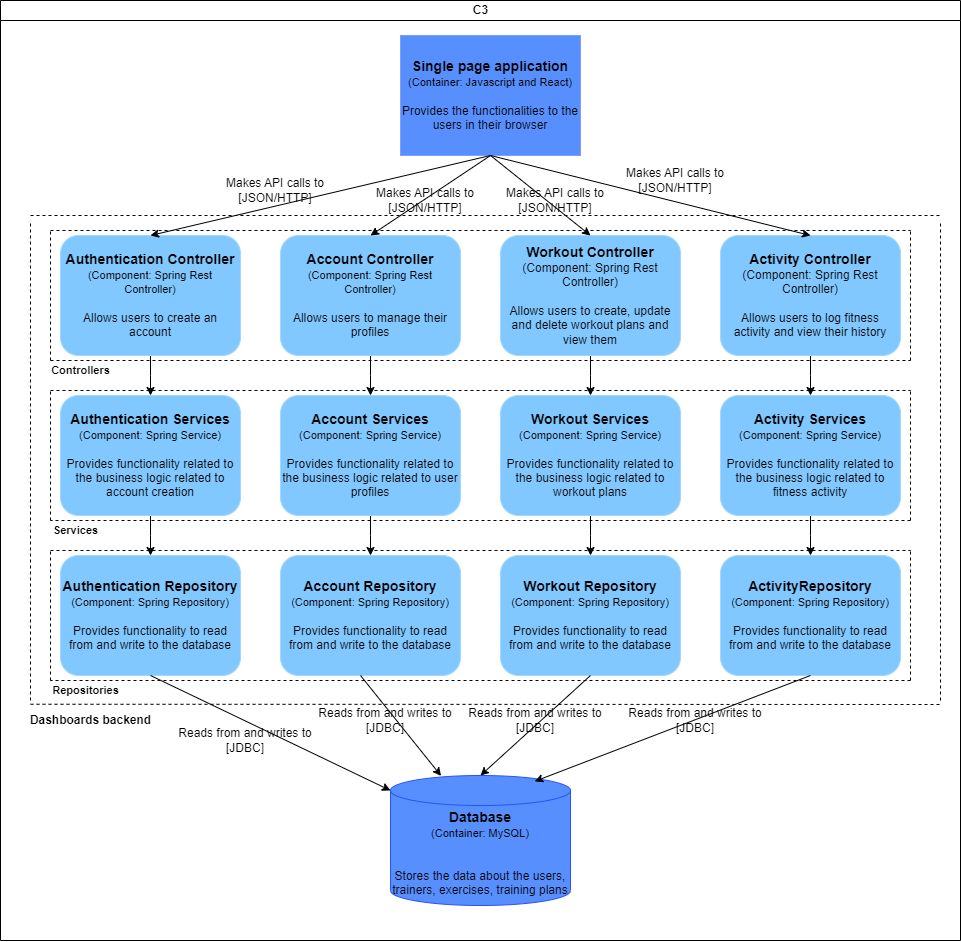


**2.2 C2 Diagram (Container Diagram)**

This diagram breaks down the system into key containers, showing how the web application (React.js + Springboot) and database (MySQL) communicate with each other.



**2.3 C3 Diagram (Component Diagram)**

The C3 diagram provides a more detailed view of the system’s internal structure, showing the individual components within the backend, including the authentication service, the profile system, the workout service, and the activity service.

**3. Design Decisions**

**3.1 Frontend: React.js**

**Chosen because**: React.js is a highly interactive, component-based framework, ideal for developing complex user interfaces like dashboards for different user roles (clients, trainers, admins). It allows efficient state management and reusable UI components.

**3.2 Backend: Spring Boot**

**Chosen because**: Spring Boot provides a fast setup and robust framework for developing secure and scalable backend services. It supports microservices architecture, which will be beneficial for future scaling and modularity.

**3.3 Database: MySQL**

**Chosen because**: MySQL is a widely-used relational database management system, ideal for handling structured data and offering excellent support for transactions, making it perfect for managing user data, workout logs, and system records.

**4. Architecture Constraints**

**4.1 Performance Constraints**

* **Real-time features**: The system must support real-time updates with minimal latency, particularly for trainer-client interactions and notifications.
* **Load Handling**: The application should support up to **1,000 simultaneous users** and handle **100 transactions per second**.

**4.2 Scalability Constraints**

* **Horizontal scaling**: The architecture must support horizontal scaling to handle increases in user traffic.
* **Future integrations**: The architecture should allow the integration of additional features without significant modifications.

**4.3 Security Constraints**

* **Data protection**: FitQuest must comply with **GDPR** and ensure the security of all user data, including encryption for sensitive information like passwords.
* **Role-based access**: Different roles (clients, trainers, admins) must have strict access controls to ensure that data is protected and accessible only to authorized users.

**4.4 Technology Constraints**

* **Open-source technologies**: The project must rely on open-source tools to stay within budget, which means avoiding proprietary licenses for software and cloud services unless absolutely necessary.

**5. Component Design**

**5.1 Authentication and Authorization**

* **Purpose**: The system must securely authenticate users (clients, trainers, and admins) and ensure appropriate access to resources based on roles.
* **Design**: FitQuest will use **Spring Security** with **JWT (JSON Web Tokens)** for handling authentication. Role-based access controls will be enforced through the backend.

**5.2 Dashboard Modules**

* **Client Dashboard**: Displays workout plans, progress tracking, and notifications. Users can log workouts and track their progress over time.
* **Trainer Dashboard**: Displays client lists, assigned workout plans, and notifications for real-time interactions.
* **Admin Dashboard**: Provides system monitoring capabilities, including user and exercise management.

**5.3 Real-Time Notifications**

* **Purpose**: Trainers and clients must receive notifications about workout updates or progress in real-time.
* **Design**: A **WebSocket server** will handle real-time data, pushing notifications directly to user dashboards. This service will also be scalable, ensuring minimal delay for high volumes of traffic.