# Design document

## **Introduction**

This document provides an overview of the architectural design and technical decisions for the fitness application. The project aims to deliver a personalized fitness and diet planning platform for users and trainers, with features like workout and diet suggestions, progress tracking, and trainer management.

## **Architectural Overview**

### 2.1 System Architecture

The system follows a typical client-server architecture, with the front end developed using React and the backend built with Spring Boot. MySQL is used as a relational database to store user, trainer, and workout data. The system is designed to ensure scalability, security, and ease of maintenance.

**Technologies Used:**

Frontend: React (JavaScript/TypeScript)

Backend: Spring Boot (Java)

Database: MySQL

Version Control: Git

CI/CD: GitLab CI for Continuous Integration

Containerization: Docker

# **Architecture Constraints and Design Decisions**

### 3.1 Choice of Spring Boot (Backend)

Reason: Spring Boot was chosen for its simplicity in creating production-ready applications with minimal configuration. It provides built-in support for building RESTful APIs, which are critical for communicating between the frontend and backend.

Alternative Consideration: While Django could have been an option, Spring Boot provides stronger integration with Java, which is essential due to the team’s expertise in Java-based frameworks.

### 3.2 Choice of React (Frontend)

Reason: React’s component-based architecture allows for easier state management and scalable UI development. It works seamlessly with REST APIs, making it an ideal choice for a dynamic web application.

Alternative Consideration: Angular was considered, but React was chosen due to its faster learning curve and large community support.

### 3.3 Choice of MySQL (Database)

Reason: MySQL is a widely-used relational database, offering robust ACID properties and easy integration with Spring Boot’s JPA for data management.

Alternative Consideration: NoSQL databases were considered, but MySQL was selected because it best suits the structured nature of the project’s data requirements (e.g., user and trainer profiles, workout plans).

## **C4 Model Diagrams**

### 4.1 Level 1: System Context Diagram

This diagram shows how external actors (users and trainers) interact with the system. Users access the application via a web browser, while trainers manage client profiles via the same interface. All communications go through the API gateway hosted in Spring Boot.

### 4.2 Level 2: Container Diagram

This diagram breaks down the system into its main containers:

Frontend (React): User interface where users and trainers interact with the system.

Backend (Spring Boot): API layer that handles business logic and communicates with the database.

Database (MySQL): Stores user, trainer, and workout information.

### 4.3 Level 3: Component Diagram

The components within the backend are divided into Controllers, Services, and Repositories:

Controller Layer: Handles incoming HTTP requests (e.g., for user registration, and workout plan creation).

Service Layer: Implements the business logic, such as generating personalized workout plans.

Repository Layer: Interfaces with the MySQL database for CRUD operations.