<Prochicken>

<ProChickenFitness>
Software Architecture Document

Version <1.1>

Revision History

Date	Version	Description	Author
<25/11/23>	<1.0>	Build-up the idea	Đạt, Đức, Hòa
<10/12/23>	<1.1>	Update content of section Deployment and section Impletation View	Đạt

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

The purpose of this document is to outline the software architecture for the development of the Fitness App. It serves as a reference for developers, designers, and other stakeholders involved in the project. The document provides a detailed overview of the system's architecture, ensuring a common understanding of design decisions and guiding future development efforts.

1.2 Scope

The Fitness App is designed to [provide a brief description of the app's main features and functionalities]. This document focuses on the high-level architecture, outlining the key components, their interactions, and the design principles that govern the development of the app.

1.3 Definitions, Acronyms, and Abbreviations

- User: An individual using the Fitness App.
- API: Application Programming Interface.
- UI: User Interface.
- DB: Database.

1.4 References

None

1.5 Overview

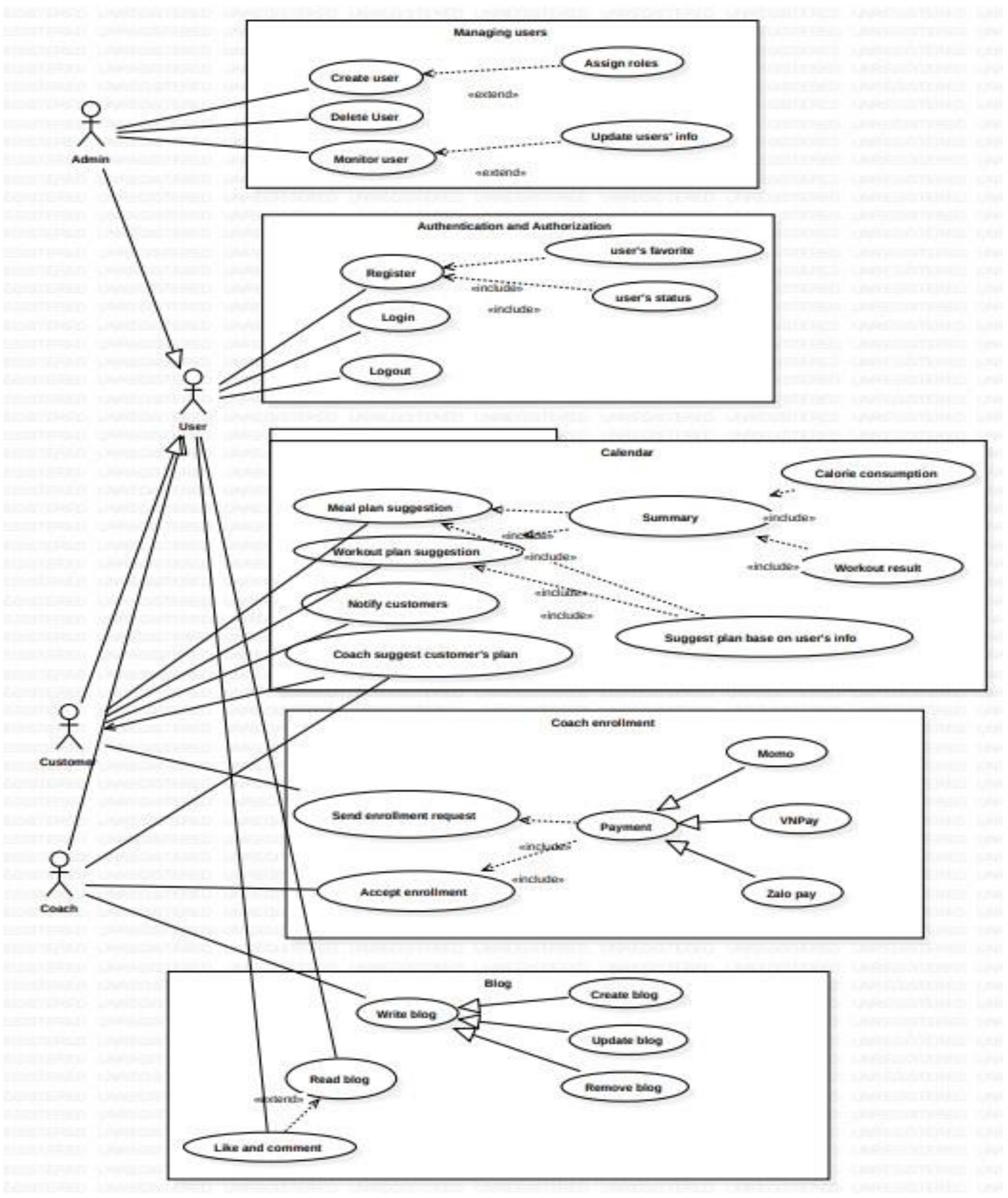
This Software Architecture Document includes:

- Architectural Goals and Constraints: The targets and requirements of the software architecture.
- Use-case Model: The use-case diagram of the software.
- Logical View: An in-depth view into how the system components work and the relationships between them.
- Deployment: How the components of the system are deployed and developed.
- Implementation View: Folder structure of the system and its components.

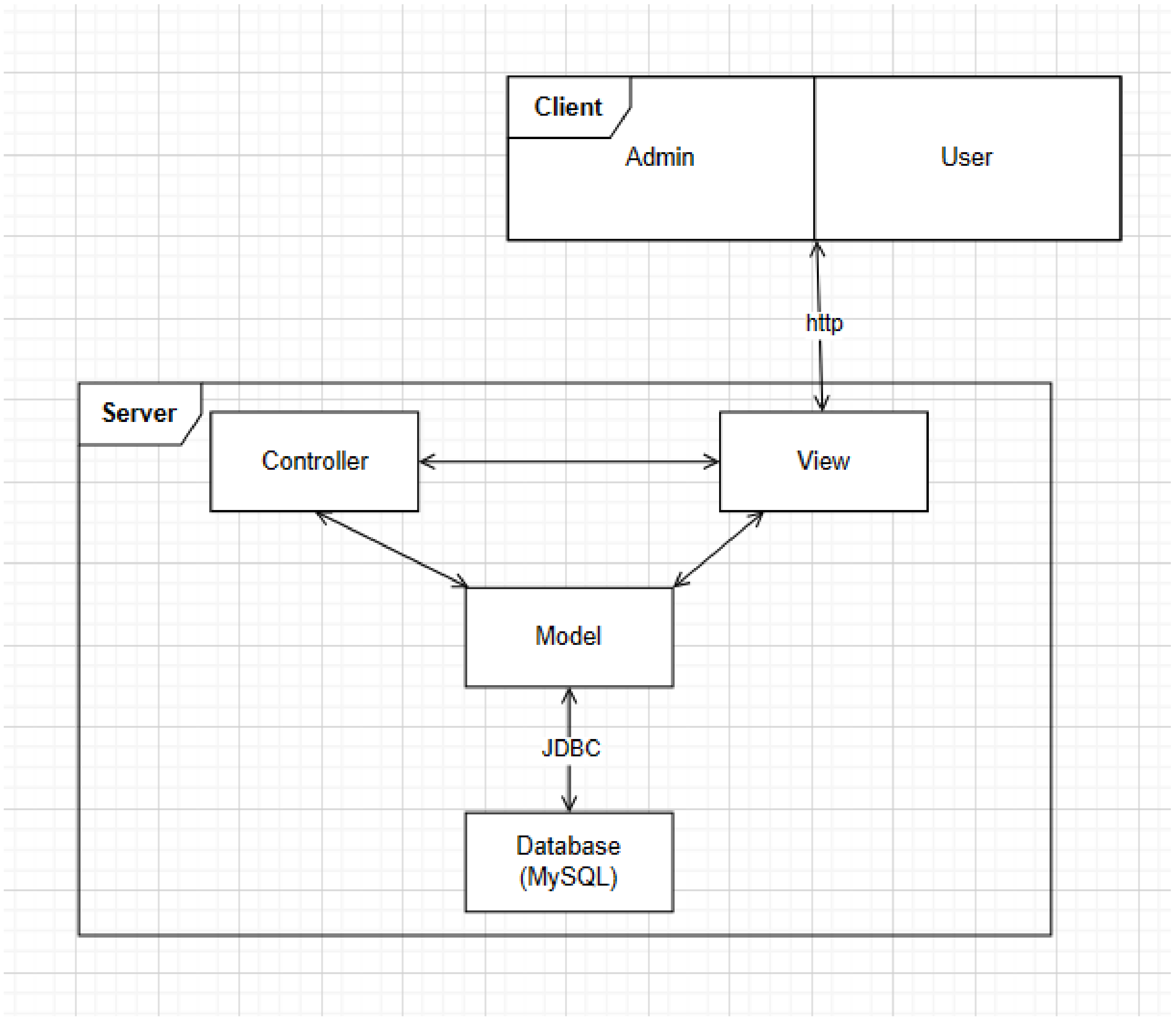
2. Architectural Goals and Constraints

- **Performance:** The app should aim for a response time of under 2 seconds for user interactions, ensuring a smooth and responsive experience.
- **Security:** The app should implement industry-standard encryption algorithms and secure authentication mechanisms to protect user data. It should comply with relevant privacy regulations and undergo regular security audits.
- **Scalability:** The app should be able to handle a significant increase in users and data without a noticeable decrease in performance. It should be able to scale up to accommodate at least a 50% increase in user base.
- **Maintainability:** The app should have a modular and well-documented codebase, allowing for easy updates and bug fixes. It should aim for a code maintainability score of at least 80%.
- **Robustness:** The app should be able to handle unexpected or erroneous inputs gracefully, preventing crashes or unexpected behavior. It should include error handling mechanisms to detect and recover from errors, ensuring the system remains stable and functional.
- **Fault Tolerance:** The app should be designed to withstand and recover from failures or disruptions, such as hardware failures or network outages. It should include mechanisms like redundancy, backup systems, and fault recovery procedures to minimize downtime and data loss.

3. Use-Case Model



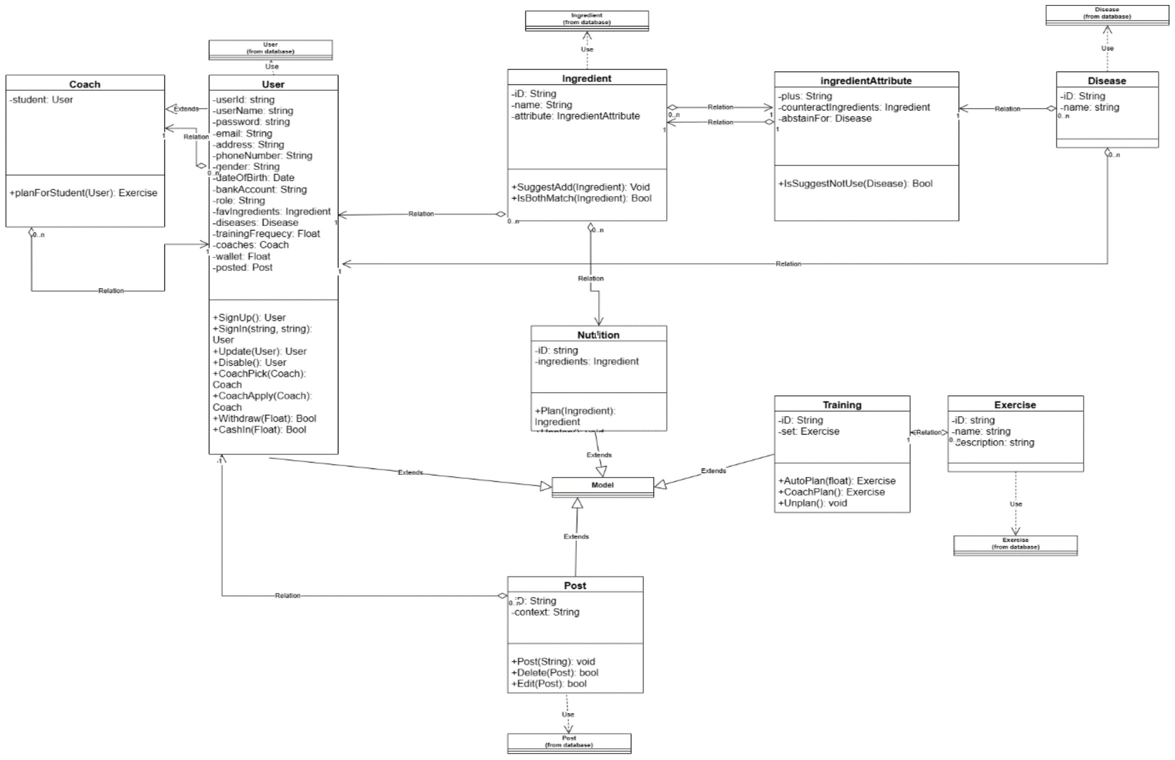
4. Logical View



4.1 Component: Model

Description: The component model includes the following classes that represent data structure properties when the application communicates with the database server:

- User: Model represents the user as trainer on the system.
- Nutrition: Model represents the nutrition plan for user.
- Training: Model represents the training plan for user.
- Post: Model represents the experience share through posts.

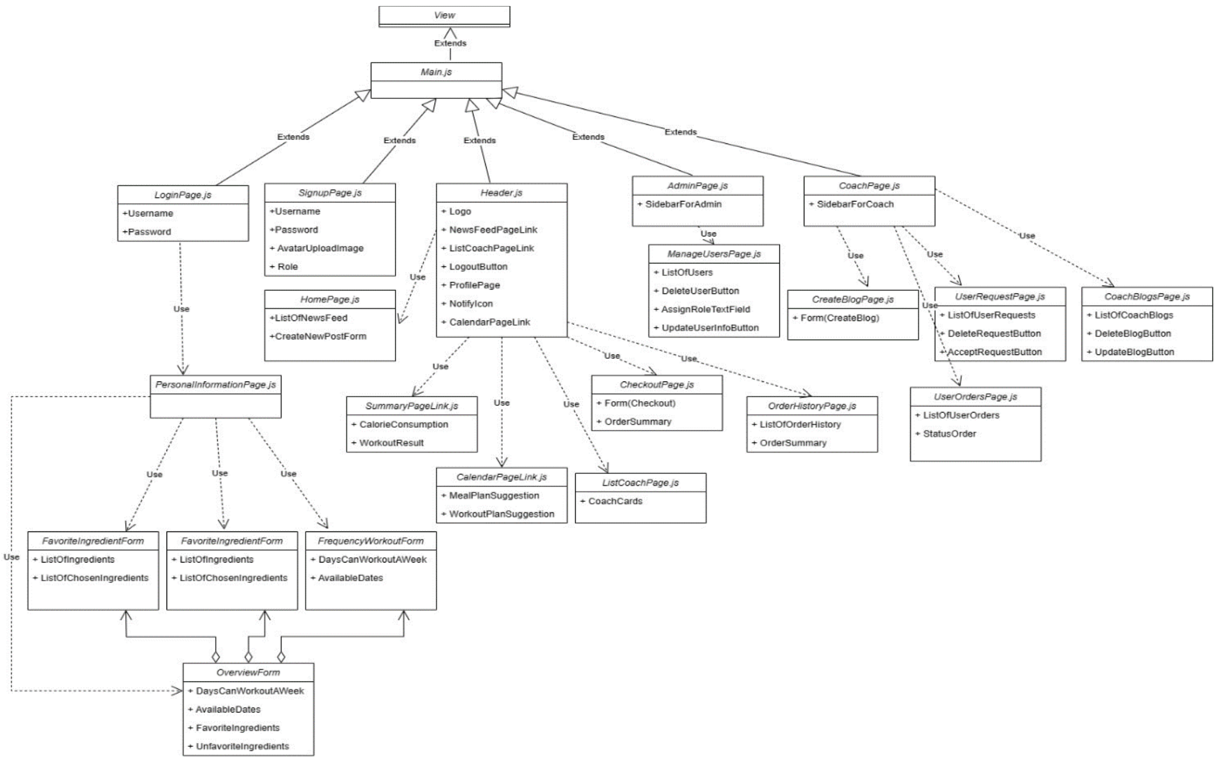


Key-class explanation:

- *User:*
  - userId: Id of user
  - userName: Full name or nickname of user
  - password: For user validation
  - email: Contact of user as email
  - address: Address of user.
  - phoneNumber: Contanct of user as phone number.
  - gender: Gender of user.
  - dateOfBirth: Date of birth of user.
  - bankAccount: Banking information for cash in/ cash out wallet.
  - role: Coach or trainer user.
  - favIngredients: Favorite ingredients of user.
  - diseases: User's diseases history.
  - trainingFrequency: User training frequency by time.
  - coaches: List of coaches that user has picked.
  - wallet: Current number in user wallet on server.
  - posted: Post history by user
- *Nutrition:*
  - iD: key of this nutrition plan
  - ingredients: Ingredients set information
- *Training:*
  - iD: key of this training plan
  - set: Set of exercises for user to practice
- *Post:*
  - iD: key of this post
  - context: custom context created by user has posted to server.

4.2 Component: View

Class Diagram

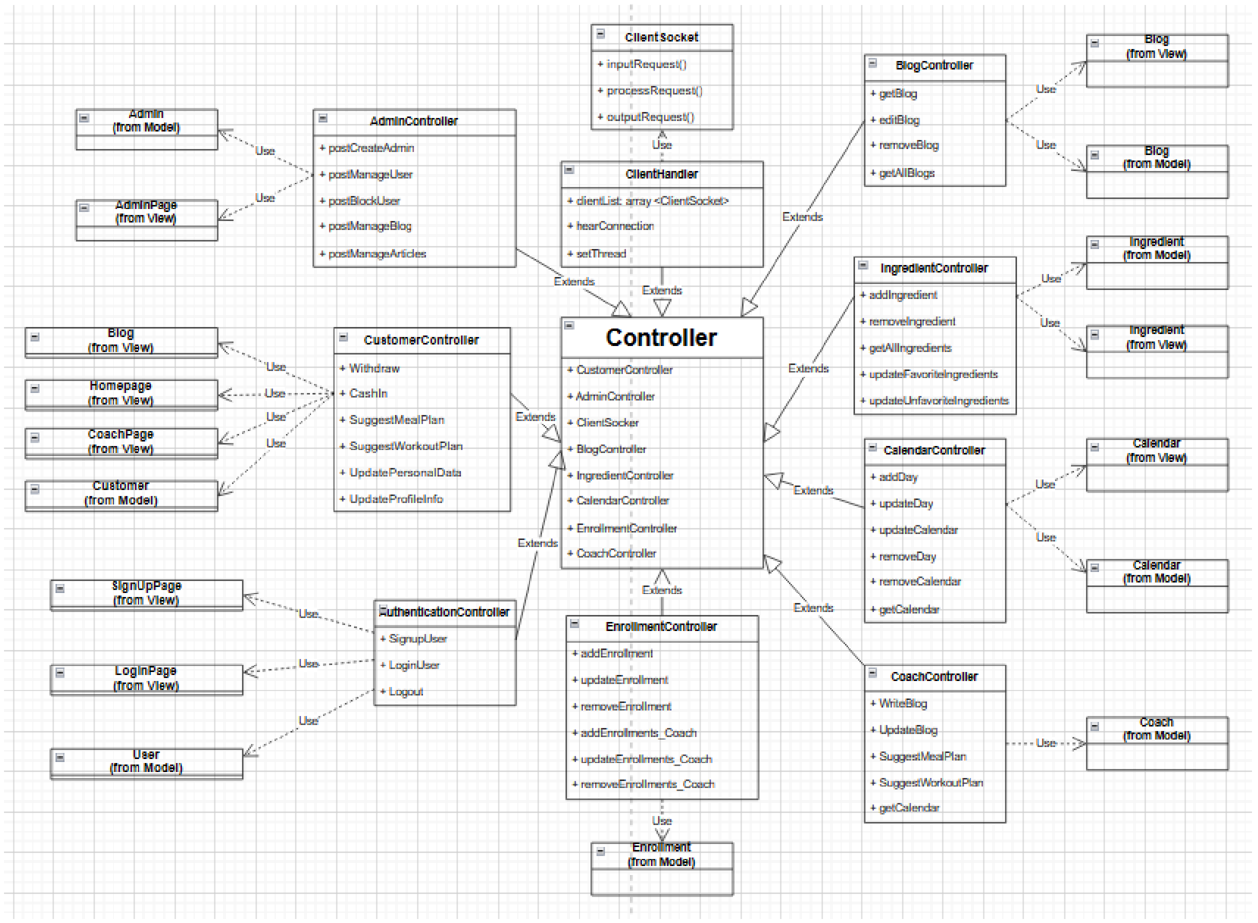


Description

- The component view displays the architecture of a website. It shows how these components in a website interact with each other.
- SignupPage.js: This page will be displayed to users when they first access the website. It contains a form that has fields such as username, password, a button for uploading user avatar.
- LoginPage.js: This page will be displayed to users when they've registered and want to use the website. It contains a form that has fields such as username and password.
- Header.js: This component contains navigation bars that help user move around different pages like news feed page, list of coach page, profile page, order history page, calendar page. It also contains notify icon that provides information when a coach accepts a user request.
- AdminPage.js: This page will be used by coaches. It has a sidebar that helps coaches navigate between pages created specifically for coaches like creating blogs, view user requests, update/delete their blogs, view user's orders.
- CoachPage.js: This page is similar to AdminPage.js but will be used by coaches.
- HomePage.js: This page is the default page when users logged in. This page will display a list of news feed from users.
- PersonalInformationPage.js: This page will include four forms that will help in the process of retrieving user's favorite/unfavorite ingredients.
- SummaryPage.js: This page displays the total calories consumption of a user, the exercises that have been completed that day.
- OrderHistoryPage.js: This page displays historical information about the orders that users have paid.
- ListCoachPage.js: This page displays a list of coaches that are currently available and can be requested to coach a user.

4.3 Component: Controller

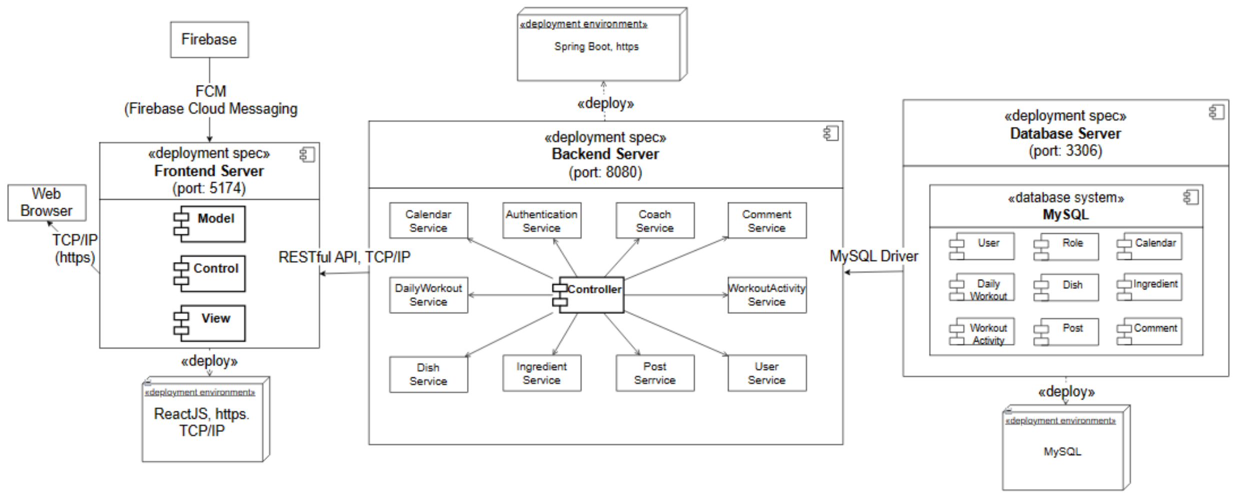
Class Diagram



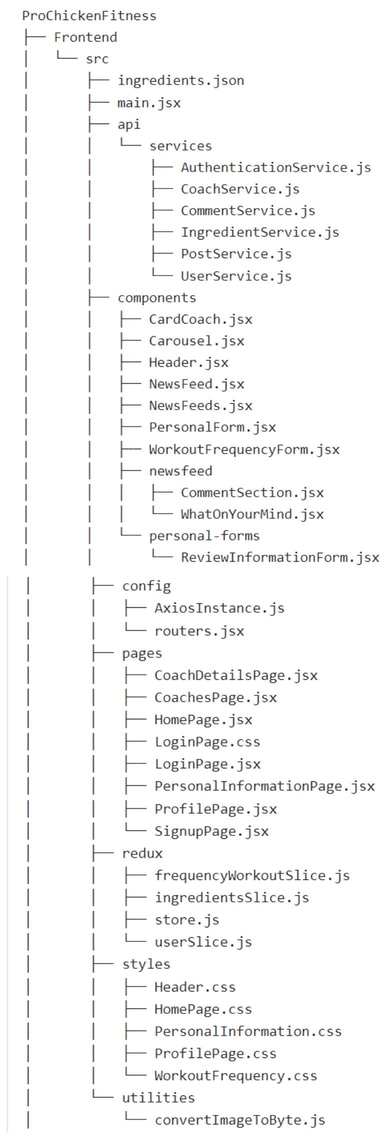
Description

- **CustomerController:** Manages customer/user behavior within the system, handles requests related to customer actions and interactions. Retrieves customer-related data and updates customer data from the Model. Sends user-related data to the views for rendering.
- **CoachController:** Retrieves coach and coaching session data from the Coach model, updates the model with coaching-related information. Sends coaching data to the views for rendering, manages user inputs from the views for scheduling coaching sessions or tracking progress.
- **AdminController:** Manages administrative tasks and behaviors within the system, typically responsible for handling requests related to administrative functionalities, user management, and system configuration. Presents administrative interfaces to the views for configuring the system.
- **ClientSocket:** Manages communication via sockets, possibly for client-server interactions, handles the establishment and maintenance of socket connections.
- **BlogController:** Retrieves blog posts and related data from the Blog model, updates the model with new blog posts, edits, or comments. Sends blog-related data to the views for rendering, receives user inputs from the views for creating or commenting on blog posts.
- **IngredientController:** Retrieves ingredient data from the Ingredient model, updates the model with changes in ingredient information. Sends ingredient-related data to the views for rendering, handles user input from the views for adding, updating, or removing ingredients.
- **CalendarController:** Retrieves event and scheduling data from the Calendar model, updates the model with changes in scheduled events. Sends scheduling data to the views for rendering, Processes user inputs from the views for creating or modifying events.
- **EnrollmentController:** Manages user enrollment data in the Enrollment model, updates the model based on user enrollment or withdrawal. Presents enrollment-related information to the views, receives user inputs from the views for enrollment processes.

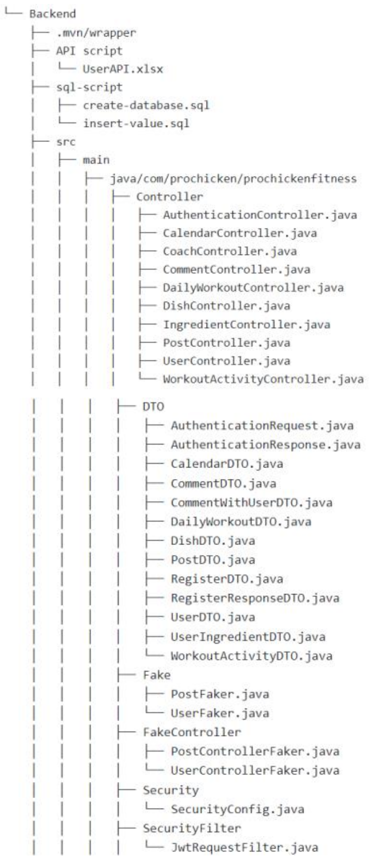
5. Deployment



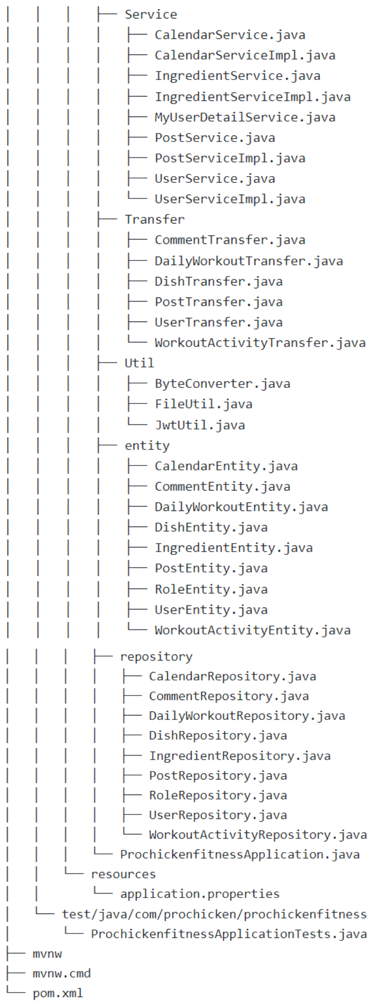
6. Implementation View



```
ProChickenFitness
├── Frontend
│   └── src
│       ├── ingredients.json
│       ├── main.jsx
│       ├── api
│       │   └── services
│       │       ├── AuthenticationService.js
│       │       ├── CoachService.js
│       │       ├── CommentService.js
│       │       ├── IngredientService.js
│       │       ├── PostService.js
│       │       └── UserService.js
│       ├── components
│       │   ├── CardCoach.jsx
│       │   ├── Carousel.jsx
│       │   ├── Header.jsx
│       │   ├── NewsFeed.jsx
│       │   ├── NewsFeeds.jsx
│       │   ├── PersonalForm.jsx
│       │   ├── WorkoutFrequencyForm.jsx
│       │   ├── newsfeed
│       │   │   ├── CommentSection.jsx
│       │   │   └── WhatOnYourMind.jsx
│       │   └── personal-forms
│       │       └── ReviewInformationForm.jsx
│       ├── config
│       │   ├── AxiosInstance.js
│       │   └── routers.jsx
│       ├── pages
│       │   ├── CoachDetailsPage.jsx
│       │   ├── CoachesPage.jsx
│       │   ├── HomePage.jsx
│       │   ├── LoginPage.css
│       │   ├── LoginPage.jsx
│       │   ├── PersonalInformationPage.jsx
│       │   ├── ProfilePage.jsx
│       │   └── SignupPage.jsx
│       ├── redux
│       │   ├── frequencyWorkoutSlice.js
│       │   ├── ingredientsSlice.js
│       │   ├── store.js
│       │   └── userSlice.js
│       ├── styles
│       │   ├── Header.css
│       │   ├── HomePage.css
│       │   ├── PersonalInformation.css
│       │   ├── ProfilePage.css
│       │   └── WorkoutFrequency.css
│       └── utilities
│           └── convertImageToByte.js
```



```
└── Backend
    ├── .mvn/wrapper
    ├── API script
    │   └── UserAPI.xlsx
    ├── sql-script
    │   ├── create-database.sql
    │   └── insert-value.sql
    ├── src
    │   ├── main
    │   │   ├── java/com/prochicken/prochickenfitness
    │   │   │   ├── Controller
    │   │   │   │   ├── AuthenticationController.java
    │   │   │   │   ├── CalendarController.java
    │   │   │   │   ├── CoachController.java
    │   │   │   │   ├── CommentController.java
    │   │   │   │   ├── DailyWorkoutController.java
    │   │   │   │   ├── DishController.java
    │   │   │   │   ├── IngredientController.java
    │   │   │   │   ├── PostController.java
    │   │   │   │   ├── UserController.java
    │   │   │   │   └── WorkoutActivityController.java
    │   │   │   ├── DTO
    │   │   │   │   ├── AuthenticationRequest.java
    │   │   │   │   ├── AuthenticationResponse.java
    │   │   │   │   ├── CalendarDTO.java
    │   │   │   │   ├── CommentDTO.java
    │   │   │   │   ├── CommentWithUserDTO.java
    │   │   │   │   ├── DailyWorkoutDTO.java
    │   │   │   │   ├── DishDTO.java
    │   │   │   │   ├── PostDTO.java
    │   │   │   │   ├── RegisterDTO.java
    │   │   │   │   ├── RegisterResponseDTO.java
    │   │   │   │   ├── UserDTO.java
    │   │   │   │   ├── UserIngredientDTO.java
    │   │   │   │   └── WorkoutActivityDTO.java
    │   │   │   ├── Fake
    │   │   │   │   ├── PostFaker.java
    │   │   │   │   └── UserFaker.java
    │   │   │   ├── FakeController
    │   │   │   │   ├── PostControllerFaker.java
    │   │   │   │   └── UserControllerFaker.java
    │   │   │   ├── Security
    │   │   │   │   └── SecurityConfig.java
    │   │   │   ├── SecurityFilter
    │   │   │   │   └── JwtRequestFilter.java
```



```
    │   │   │   ├── Service
    │   │   │   │   ├── CalendarService.java
    │   │   │   │   ├── CalendarServiceImpl.java
    │   │   │   │   ├── IngredientService.java
    │   │   │   │   ├── IngredientServiceImpl.java
    │   │   │   │   ├── MyUserDetailService.java
    │   │   │   │   ├── PostService.java
    │   │   │   │   ├── PostServiceImpl.java
    │   │   │   │   ├── UserService.java
    │   │   │   │   └── UserServiceImpl.java
    │   │   │   ├── Transfer
    │   │   │   │   ├── CommentTransfer.java
    │   │   │   │   ├── DailyWorkoutTransfer.java
    │   │   │   │   ├── DishTransfer.java
    │   │   │   │   ├── PostTransfer.java
    │   │   │   │   ├── UserTransfer.java
    │   │   │   │   └── WorkoutActivityTransfer.java
    │   │   │   ├── Util
    │   │   │   │   ├── ByteConverter.java
    │   │   │   │   ├── FileUtil.java
    │   │   │   │   └── JwtUtil.java
    │   │   │   ├── entity
    │   │   │   │   ├── CalendarEntity.java
    │   │   │   │   ├── CommentEntity.java
    │   │   │   │   ├── DailyWorkoutEntity.java
    │   │   │   │   ├── DishEntity.java
    │   │   │   │   ├── IngredientEntity.java
    │   │   │   │   ├── PostEntity.java
    │   │   │   │   ├── RoleEntity.java
    │   │   │   │   ├── UserEntity.java
    │   │   │   │   └── WorkoutActivityEntity.java
    │   │   │   ├── repository
    │   │   │   │   ├── CalendarRepository.java
    │   │   │   │   ├── CommentRepository.java
    │   │   │   │   ├── DailyWorkoutRepository.java
    │   │   │   │   ├── DishRepository.java
    │   │   │   │   ├── IngredientRepository.java
    │   │   │   │   ├── PostRepository.java
    │   │   │   │   ├── RoleRepository.java
    │   │   │   │   ├── UserRepository.java
    │   │   │   │   └── WorkoutActivityRepository.java
    │   │   │   └── ProchickenfitnessApplication.java
    │   │   └── resources
    │   │       └── application.properties
    │   └── test/java/com/prochicken/prochickenfitness
    │       └── ProchickenfitnessApplicationTests.java
    ├── mvnw
    ├── mvnw.cmd
    └── pom.xml
```