

Fitness Application

Name:Cristea Vasile-Cristian

Group:30236

Table of Contents

[Deliverable 1 3](#_Toc64843130)

[Project Specification 3](#_Toc64843131)

[Functional Requirements 3](#_Toc64843132)

[Use Case Model 3](#_Toc64843133)

[Use Cases Identification 3](#_Toc64843134)

[UML Use Case Diagrams 3](#_Toc64843135)

[Supplementary Specification 3](#_Toc64843136)

[Non-functional Requirements 3](#_Toc64843137)

[Design Constraints 3](#_Toc64843138)

[Glossary 3](#_Toc64843139)

[Deliverable 2 3](#_Toc64843140)

[Domain Model 3](#_Toc64843141)

[Architectural Design 4](#_Toc64843142)

[Conceptual Architecture 4](#_Toc64843143)

[Package Design 4](#_Toc64843144)

[Component and Deployment Diagram 4](#_Toc64843145)

[Deliverable 3 4](#_Toc64843146)

[Design Model 4](#_Toc64843147)

[Dynamic Behavior 4](#_Toc64843148)

[Class Diagram 4](#_Toc64843149)

[Data Model 4](#_Toc64843150)

[System Testing 4](#_Toc64843151)

[Future Improvements 4](#_Toc64843152)

[Conclusion 4](#_Toc64843153)

[Bibliography 4](#_Toc64843154)

# Deliverable 1

## Project Specification

The aim of the project is to design and develop a fitness application that allows the users to track their progress and access various resources such as workout plans, developed by professional trainers, recommendations given by them or even chatting with the trainers. The app will by design in Java, using Spring Boot, and React for UI and it will have two main actors: the normal user and the trainer, besides the admin. Normal users will be able to track their fitness activity, to generate monthly reports of their progress, to filter the workout records by the activity type and to follow plans developed by the trainers. The trainers will be able to develop daily plans for the users, with different types of activities and different goals. More than that, they will be able to update and to delete the plans and to get in touch with the users that follow their plans using an online chat.

## Functional Requirements

1. User registration and login: Users can create an account by providing their email address and password. The app must verify the user’s credentials and allow them to log in.
2. Personal profile: Users can create and edit their personal profile by providing their age, height and weight.
3. Workout plans: Users can access a database of workout plans with different levels of difficulty, goals and activity types. The app must allow users to select a plan and view the list of exercises included.
4. Fitness tracking: Users can track their fitness such as workouts, runs, walks and other physical activities. The app must allow users to enter the type of activity, duration and other metrics.
5. Trainers plans: Trainers can create, edit and delete workout plans with different levels of difficulty, goals and activity types.
6. Trainer registration: Trainers will be registered by the admin. The app must allow the admin to register a new trainer, or to delete one.

## Use Case Model 1

### Use Cases Identification

**Use-Case:** Exercise Tracking

**Level:** User goal

**Primary Actor:** Registered user

**Main success scenario**

1. The user presses the button that corresponds to the add workout feature.
2. The app presents a form with different types of activities such as running, walking and others.
3. The user selects the activity type they want to track.
4. The user enters the activity details and saves the activity record.
5. The app displays the activity record in the user’s workout history.

**Alternative sequence**:

1. If the user is not registered, the app prompts them a message.
2. If the user wants to edit or delete an activity record, the app provides options to edit or delete the record.

**Use-Case:** Plan Management

**Level:** Trainer goal

**Primary Actor:** Registered trainer

**Main success scenario**

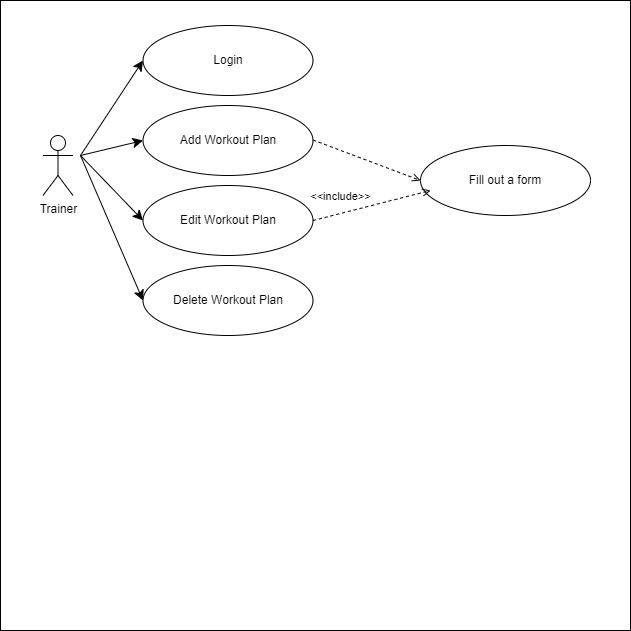
1. The user presses the corresponding button to add a workout plan.
2. The trainer selects the main activity of the plan.
3. The trainer set a goal to the plan.
4. The app presents a form in which the trainer can insert daily plans with exercises.
5. The trainer adds a description to the workout plan.
6. The trainer pressed the Add button.
7. The app displays the workout plan in the plans section created by the trainer.

**Alternative sequence**:

1. If the trainer is not registered, the app prompts them a message.
2. If the trainer wants to edit or delete an workout plan , the app provides options to edit or delete the plan.

**UML Use Case Diagrams**

Diagram

Description automatically generated

## Supplementary Specification

### Non-functional Requirements

1. Performance: The app must be fast and responsive, with minimal loading times and smooth transitions between screens.
2. Usability: The app must be easy to use, with an intuitive and user-friendly interface.
3. Security: The app must be secure, with robust encryption and authentication mechanisms to protect user data and prevent unauthorized access.
4. Maintainability: The app must be maintainable, with clean and well-documented code, and a robust testing and debugging process.

### Design Constraints

1. Programming languages: The app must be implemented using Java, to ensure compatibility and robustness.
2. Development framework: The app must be built using a specific development framework, such as ReactJS, to ensure consistency and ease of development.
3. User interface design: The app must have a friendly user interface, using a consistent variety of colors and fonts.

## Glossary

* Fitness App: A web application that allows users to track their exercise activities, set fitness goals.
* Registered user: A user who has created a profile and logged into the Fitness App.
* Workout Record: A record that contains details about a user’s exercise activity, such as activity type, duration and intensity.
* Workout history: A record that contains a user’s previous exercise activity type, duration, distance and kilocalories.

# Deliverable 2

## Domain Model

*Entities*:

* User: a person who uses the fitness app.
* Workout record: represents a physical activity that user performs, such as running, cycling, walking, gym or swmming.
* Workout Plan: represents a set of activities defined by a trainer for users to follow it.
* Plan Day: represents the information about a day in a workout plan.
* Plan Entry: represents a single exercise from a workout day, with its information.
* User plan: represents a plan that it is followed by a user.

*Attributes:*

* User: id, username, name, password, email, gender, height, weight, role.
* Workout Record: id, user, activity type, date, distance, duration, calories.
* Workout Plan: id, trainer, name, main activity, goal, level, description.
* Plan Day: id, plan, day number.
* Plan Entry: id, plan day, order number, duration, no of repetitions, description, rest between sets, number of sets.
* User Plan: id, user, plan, current day.

*Relationships:*

* Users can have many workout records, user plans or workout plans.
* Workout record belongs to one user.
* Workout plan belongs to one user and can have many plan days.
* Plan day belongs to one workout plan and can have many plan entries.
* Plan entry belongs to one plan day.
* User plan belongs to one user and one plan.

Diagram

Description automatically generated

## Architectural Design

### Conceptual Architecture

The architectural style used for the system is REST (Representational State Transfer) and the architectural pattern is MVC (Model-View-Controller).

The REST architectural style is well suited for designing web-based systems that are scalable, flexible, and easy to integrate with other systems and services. REST defines a set f constraints and principles for creating stateless, client-server systems that communicate using a uniform interface.

The MVC pattern is a widely used pattern in web development that separates the application into three main components: the Model, the View and the Controller. The Model represents the data and the business logic of the application, the View represents the user interface, and the Controller handles user input and updates the Model and View accordingly.

In the case of my application, the REST architecture is used to create a set of resources that can be accessed and manipulated using HTTP requests. For example, the system has resources for tracking the progress made by users. Clients can access these resources usgin HTTP requests, and the server can respond with the appropriate representations of the resources.

### Package Design

Diagram

Description automatically generated

### Component and Deployment Diagram

Diagram

Description automatically generated

# Deliverable 3

## Design Model

### Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

### Class Diagram

[Create the UML class diagram; apply GoF patterns and motivate your choice]

## Data Model

[Create the data model for the system.]

# System Testing

[Describe the testing methides and some test cases.]

# Future Improvements

[Present some features that apply to the application scope.]

# Conclusion

# Bibliography