

Sprint 3 Scoping

Team Four Real

- **Team Members**

Yu-Hsuan Lin, Yi-Jie Chou, LuoJia Zhao, Chienchia Chiu

- **GitHub Repository**

https://github.com/zhaoluoJia/cs5500_group_project

- **Project Board**

<https://github.com/users/zhaoluoJia/projects/1/views/3>

- **Project Name**

Exercise Manager

- **Sprint 3 Scope**

In Sprint 2, we'll focus on creating the UI, deploying the app to the cloud, and setting up a CI pipeline to ensure code quality and correctness.

- **Sprint 3 Tasks**

1. **Create UI**

We'll use the React library to develop three different web pages that will display data showcasing three different user stories. These pages will be built using the REST API that we designed and implemented in previous sprints. Wireframes and specifications for each page have already been created as below.

User story1:

As an exercise lover, I want to get the calories burned in my activities by inputting the duration/type (walking/jogging/cycling/kayaking) of exercise and weight.

- **Webpage description:**

A page asking for input of the exercise info and output the calories and a success message:

- ☐ **Input area:**

Username(String) / Date(yyyy-mm-dd) / Duration(hh:mm) / Exercise Type.

- ☐ **Output area:**

If input valid: Showing the calories burned with message "exercise added successfully. You burned xxx calories in this exercise!".

If input invalid: Showing empty/wrong next to the input column is contains invalid (empty) input.

- **REST API to be used:**

```
public ResponseEntity<String> createExercise(@PathVariable Long userId,
    @RequestParam String exerciseName,
    @RequestParam("date") @DateTimeFormat(pattern = "yyyy-MM-dd") Date date,
    @RequestParam Double duration)
```

The following two REST APIs should be created:

```
public Double getCaloriesOfExercise (
    @RequestParam String exerciseName,
```

```

    @RequestParam Double duration) {
        return userService.calculateCalories(weight, exerciseName, duration);
    }

    public User getUserByUsername(
        @RequestParam("username") String username) {
        return userService.getUserByCredentials(username);
        // this corresponding service “.getUserByCredentials(username)” is to be created too.
    }

```

■ Wireframes

Initial page (asking for input)

Record an exercise and check calories burned!

User name:

Date:

e.g. 2023/03/25

Duration:

e.g. 01:12 for 1 hour and 12 minutes

Exercise Type:

running ▼

Page for a valid input

Record an exercise and check calories burned!

User name:

Date:

e.g. 2023/03/25

Duration:

e.g. 01:12 for 1 hour and 12 minutes

Exercise Type:

running ▼

Page for invalid/empty inputs

Record an exercise and check calories burned!

User name:

Date:

Duration:

Exercise Type:

empty/invalid

e.g. 2023/03/25

e.g. 01:12 for 1 hour and 12 minutes

User Story2:

As a person on a diet, I want to set a goal and compare the actual calories burned/actual duration run between a certain period.

■ Webpage description:

A page asking for input of the goal info and output the progress with message:

Step 1 (Part A):

Asking the user to input the username, if valid, Part B would show, if not, there would show an error message “empty/invalid”.

Step 2 (Part B):

Asking the user to select a goal, which is the calories goal or duration goal. After selecting the button, Part C would show.

Step 3 (Part C):

Asking the user to input the goal, if the user selected the calories goal earlier, the user could input the calories goal; if the user selected the duration goal earlier, the user could input the duration goal. If the input valid, Part D would show, if not, there would show an error message “empty/invalid”.

Step 4 (Part D):

Final page: Showing the total calories burn / total duration, a progress bar, and a message that shows how many percentages the user completed.

■ REST API to be used:

```
public Map<Date, Double> getDurationTotalBetweenDates(@PathVariable Long userId,
@RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
@RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate)
```

```
public Map<Date, Double> getCaloriesTotalBetweenDates(@PathVariable Long userId,
@RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
```

@RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate)

■ Wireframes:

The wireframe shows a page titled "Welcome to Exercise Manager". It is divided into four main sections:

- Part A (Red border):** A login section with a "UserName:" label, a text input field, and a "Send" button.
- Part B (Green border):** A "Select a goal" section with two buttons: "Calories Goal" and "Duration Goal".
- Part C (Yellow border):** A form for goal details. It includes a label "Calories Goal / Duration Goal:" followed by a text input field. Below this are "Start Date:" and "End Date:" labels, each followed by a date input field with a placeholder "yyyy/mm/dd". A "Send" button is at the bottom.
- Part D (Blue border):** A summary section showing "Total Calories Burn / Total Duration: XXXXXX", a "Your Progress:" label with a progress bar (30% blue, 70% grey), and the text "You are at 30% of your goal".

Samples of invalid input:

This sample shows the "Welcome to Exercise Manager" page with an invalid input. The "UserName:" label is followed by an empty text input field. To the right of the input field, the text "empty/invalid" is displayed in red. The "Send" button is visible below the input field.

Welcome to Exercise Manager

UserName:

Select a goal

Calories Goal:

empty/invalid

Start Date:

empty/invalid

End Date:

empty/invalid

Sample of final page:

Welcome to Exercise Manager

UserName:

Select a goal

Calories Goal:

Start Date:

End Date:

Total Calories Burn: 600

Your Progress:

You are at 30% of your goal

User Story3:

As an exercise lover, I want to get a weekly report about the past 7 days' exercise activities , with charts showing me each day's total exercise duration /calories , and advice on which day I should exercise more.

Analysis of the smallest calories/exercise duration between two dates is implemented in UserService.

Planned to use API like Spring MVC to visualize each day's duration/ calories by charts when creating frontend.

■ Webpage description:

A page showing charts with weekly summary of calories and duration, and suggestions on which days should exercise more:

- ☐ Charts: Showing a daily calories burn chart and a daily duration chart
- ☐ Summary: Showing suggestions

■ REST API to be used:

```
public Double getDurationTotalBetweenDates(@PathVariable Long userId,
@RequestParam("startDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) Date
startDate, @RequestParam("endDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
Date endDate)
```

```
public Double getCaloriesTotalBetweenDates(@PathVariable Long userId,
@RequestParam("startDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) Date
startDate, @RequestParam("endDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
Date endDate)
```

```
public Double getSmallestDurationBetweenDates(@PathVariable Long userId,
@RequestParam("startDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) Date
startDate, @RequestParam("endDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
Date endDate)
```

```
public Double getSmallestCaloriesBetweenDates(@PathVariable Long userId,
@RequestParam("startDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE) Date
startDate, @RequestParam("endDate") @DateTimeFormat(iso = DateTimeFormat.ISO.DATE)
Date endDate)
```

■ Wireframes



2. Deploy the app to the cloud

Our database has already been deployed using MongoDB Atlas, and we'll now deploy our frontend and backend to AWS EC2.

3. Setup CI pipeline

To ensure code quality and correctness on every push to main, we'll set up a Continuous Integration pipeline using Github Actions.