

## Sprint 2 Submission

Team Four Real

- **Team Members**

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

- **GitHub Repository**

[https://github.com/zhaoluojia/cs5500\\_group\\_project](https://github.com/zhaoluojia/cs5500_group_project)

- **Project Board**

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

- **Project Name**

Exercise Manager

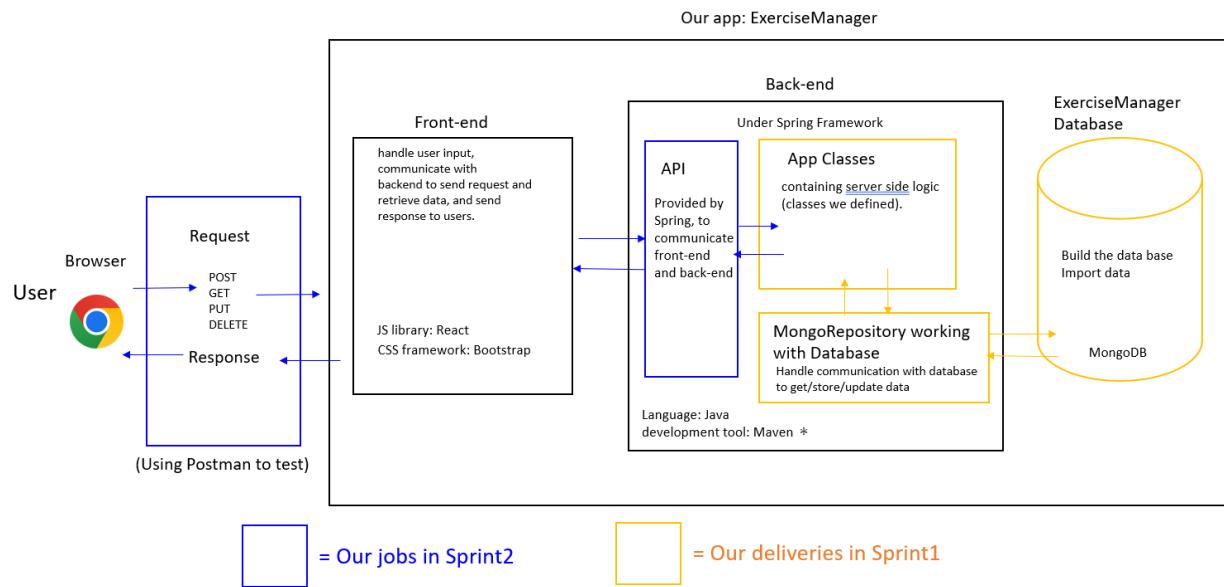
- **Delivery Against User Stories**

No	User Story	delivered	description
1	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.	yes	In UserController class, we have createExercise method which accepts user input (exerciseName, date, duration), and adds a new exercise with the calories calculated by exerciseName and user weight (acquired at user registration) to the user's exerciseList.
2	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.	yes	In UserController class, we have updateDurationGoal and updateCaloriesGoal methods, which accept user input (startDate, endDate, goal), and create a goal for the user. We also have getDurationTotalBetweenDates and getCaloriesTotalBetweenDates methods, which accept user input (startDate, endDate), and calculate the actual calories burned/actual duration run between a certain period.
3	As an exercise lover, I want to get a weekly report about the past 7 days' exercise activities, with charts showing each day's total exercise duration/calories, and advice on which day I should exercise more.	yes	In UserController class, we have getDailyDurationSumMap and getDailyCaloriesSumMap methods, which accept user input (startDate, endDate), and display each day's total exercise duration/calories in the weekly report. We also have getSmallestDurationBetweenDates and getSmallestCaloriesBetweenDates methods,

		which accept user input (startDate, endDate), and display the smallest duration run / calories burned in the past 7 days. According to that, our application can provide advice on the day that user can exercise more.
--	--	---

## ■ Functionality Delivered

In Sprint 2, we've implemented the REST API that allows user to query data as well as to add new data into the system through HTTP request.



\* Maven is used to manage the dependencies of a Spring application, such as the Spring framework itself and any other libraries that the application needs.

### ❖ Controller

We've implemented 22 controller methods for the backend. Please see below screenshots for details.

```
1 package com.exerisemgr.exercisemanager.controller;
2
3 import com.exerisemgr.exercisemanager.model.CaloriesGoal;
4 import com.exerisemgr.exercisemanager.model.DurationGoal;
5 import com.exerisemgr.exercisemanager.model.Exercise;
6 import com.exerisemgr.exercisemanager.model.User;
7 import com.exerisemgr.exercisemanager.service.UserService;
8
9 import java.util.Date;
10 import java.util.List;
11 import java.util.Map;
12
13 import org.springframework.beans.factory.annotation.Autowired;
14 import org.springframework.format.annotation.DateTimeFormat;
15 import org.springframework.http.HttpStatus;
16 import org.springframework.http.ResponseEntity;
17 import org.springframework.web.bind.annotation.*;
18
19 @RestController
20 @RequestMapping("/api/users")
21 public class UserController {
22
23     @Autowired
24     private UserService userService;
25
26     /**
27      * Create the User.
28      * @return: string
29      */
30     @RequestMapping(value="/register", method = RequestMethod.POST)
31     public String createUser(@RequestParam String userName,@RequestParam String password, @RequestParam Double weight) {
32         User user = new User(userName, password, weight);
33         userService.createUser(user);
34         return "Success";
35     }
36
37     /**
38      * Get the User by user information.
39      * @param username the userName.
40      * @param password the password.
41      * @return User object
42      */
43     @RequestMapping(value = "/login", method = RequestMethod.GET)
44     public User getUserByCredentials(@RequestParam("username") String username, @RequestParam("password") String password) {
45         return userService.getUserByCredentials(username, password);
46     }
47
48     /**
49      * Get All the Users by user information.
50      * @return All the Users
51      */
52     @RequestMapping(value = "/allUsers", method = RequestMethod.GET)
53     public ResponseEntity<List<User>> getAllUser(){
54         return new ResponseEntity<List<User>>(userService.getAllUser(), HttpStatus.OK);
55     }
56
57     /**
58      * Get the User by userId
59      * @param userId: the userId.
60      * @return: the User.
61      */
62     @RequestMapping(value="/{userId}", method = RequestMethod.GET)
63     public User getUserId(@PathVariable Long userId) {
64         return userService.getUserByUserId(userId);
65     }
66
67     /**
68      * Get the weight of the user by userId.
69      * @param userId userId.
```

```

70     * @return user's weight.
71     */
72     @RequestMapping(value = "/{userId}/weight", method = RequestMethod.GET)
73     public Double getWeightByUserId(@PathVariable Long userId) {
74         return userService.getWeightByUserId(userId);
75     }
76
77     /**
78      * Get the DurationGoal object by userId.
79      * @param userId userId.
80      * @return user's DurationGoal object.
81      */
82     @RequestMapping(value = "/{userId}/durationGoal", method = RequestMethod.GET)
83     public DurationGoal getDurationGoalByUserId(@PathVariable Long userId) {
84         return userService.getDurationGoalByUserId(userId);
85     }
86
87     /**
88      * Get the CaloriesGoal object by userId.
89      * @param userId userId.
90      * @return user's CaloriesGoal object.
91      */
92     @RequestMapping(value = "/{userId}/caloriesGoal", method = RequestMethod.GET)
93     public CaloriesGoal getCaloriesGoalByUserId(@PathVariable Long userId) {
94         return userService.getCaloriesGoalByUserId(userId);
95     }
96
97     /**
98      * Get a list of all Exercise objects of the user by userId
99      * @param userId userId.
100     * @return a list of Exercise objects.
101     */
102    @RequestMapping(value = "/{userId}/exercises", method = RequestMethod.GET)
103    public List<Exercise> getAllExerciseByUserId(@PathVariable Long userId) {
104        return userService.getAllExerciseByUserId(userId);
105    }
106
107    /**
108     * Get a map of each date to that day's duration sum by user id and start date, end date.
109     * @param userId userId.
110     * @param startDate the start date (Date).
111     * @param endDate the end date (Date).
112     * @return a Map mapping each day to its duration sum.
113     */
114    @RequestMapping(value = "/{userId}/dailyDurations", method = RequestMethod.GET)
115    public Map<Date, Double> getDailyDurationSumMap(@PathVariable Long userId, @RequestParam("startDate"
116        @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
117        @RequestParam("endDate")
118        @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate) {
119            return userService.getDailyDurationSumMap(userId, startDate, endDate);
120        }
121
122    /**
123     * Get a map of each date to that day's calories sum by user id and start date, end date.
124     * @param userId userId.
125     * @param startDate the start date (Date).
126     * @param endDate the end date (Date).
127     * @return a Map mapping each day to its calories sum.
128     */
129    @RequestMapping(value = "/{userId}/dailyCalories", method = RequestMethod.GET)
130    public Map<Date, Double> getDailyCaloriesSumMap(@PathVariable Long userId, @RequestParam("startDate"
131        @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
132        @RequestParam("endDate")
133        @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate) {
134            return userService.getDailyCaloriesSumMap(userId, startDate, endDate);
135        }
136
137    /**
138     * Get the total duration between the given start date and end date.
139     * @param userId userId.

```

```

140     * @param startDate the start date (Date).
141     * @param endDate the end date (Date).
142     * @return the total Duration between start date and end date.
143     */
144     @RequestMapping(value = "/{userId}/durationTotalBetweenDates", method = RequestMethod.GET)
145     public Double getDurationTotalBetweenDates(@PathVariable Long userId, @RequestParam("startDate")
146     @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
147         @RequestParam("endDate")
148         @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate) {
149         return userService.getDurationTotalBetweenDates(userId, startDate, endDate);
150     }
151
152     /**
153     * Get the total calories between the given start date and end date.
154     * @param userId userId.
155     * @param startDate the start date (Date).
156     * @param endDate the end date (Date).
157     * @return the total calories between start date and end date.
158     */
159     @RequestMapping(value = "/{userId}/caloriesTotalBetweenDates", method = RequestMethod.GET)
160     public Double getCaloriesTotalBetweenDates(@PathVariable Long userId, @RequestParam("startDate")
161     @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
162         @RequestParam("endDate")
163         @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate) {
164         return userService.getCaloriesTotalBetweenDates(userId, startDate, endDate);
165     }
166
167     /**
168     * Get the smallest duration between the given start date and end date.
169     * @param userId userId.
170     * @param startDate the start date (Date).
171     * @param endDate the end date (Date).
172     * @return the smallest duration between start date and end date.
173     */
174     @RequestMapping(value = "/{userId}/smallestDurationBetweenDates", method = RequestMethod.GET)
175
176     public Double getSmallestDurationBetweenDates(@PathVariable Long userId,
177         @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
178         @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate) {
179         return userService.getSmallestDurationBetweenDates(userId, startDate, endDate);
180     }
181
182     /**
183     * Get the smallest calories between the given start date and end date.
184     * @param userId userId.
185     * @param startDate the start date (Date).
186     * @param endDate the end date (Date).
187     * @return the smallest calories between start date and end date.
188     */
189     @RequestMapping(value = "/{userId}/smallestCaloriesBetweenDates", method = RequestMethod.GET)
190     public Double getSmallestCaloriesBetweenDates(@PathVariable Long userId,
191         @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
192         @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate) {
193         return userService.getSmallestCaloriesBetweenDates(userId, startDate, endDate);
194     }
195
196     /**
197     * Update weight for the user.
198     * @param userId userId.
199     * @param weight the weight the user wants to change to.
200     */
201     @RequestMapping(value = "/{userId}/weight", method = RequestMethod.PUT)
202     public ResponseEntity<String> updateWeight(@PathVariable Long userId,
203         @RequestParam Double weight) {
204         userService.updateWeight(userId, weight);
205         return ResponseEntity.ok("Weight updated successfully");
206     }
207
208     /**
209     * Update password for the user.

```

```
210     * @param password the password the user want to change to.  
211     */  
212     @RequestMapping(value = "/{userId}/{password}", method = RequestMethod.PUT)  
213     public ResponseEntity<String> updatePassword(@PathVariable Long userId,  
214             @PathVariable String password) {  
215         userService.updatePassword(userId, password);  
216         return ResponseEntity.ok("Password updated successfully");  
217     }  
218  
219     /**  
220      * Update duration goal for the user.  
221      * @param userId userId.  
222      * @param startDate the starting date of the duration goal.  
223      * @param endDate the ending date of the duration goal.  
224      * @param durationGoal the duration goal the user want to change to.  
225      */  
226     @RequestMapping(value = "/{userId}/durationGoal", method = RequestMethod.PUT)  
227     public ResponseEntity<String> updateDurationGoal(@PathVariable Long userId,  
228             @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,  
229             @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate,  
230             @RequestParam Double durationGoal) {  
231         userService.updateDurationGoal(userId, startDate, endDate, durationGoal);  
232         return ResponseEntity.ok("Duration goal updated successfully");  
233     }  
234  
235     /**  
236      * Update calories goal for the user.  
237      * @param userId userId.  
238      * @param startDate the starting date of the calories goal.  
239      * @param endDate the ending date of the calories goal.  
240      * @param caloriesGoal the calories goal the user want to change to.  
241      */  
242     @RequestMapping(value = "/{userId}/caloriesGoal", method = RequestMethod.PUT)  
243     public ResponseEntity<String> updateCaloriesGoal(@PathVariable Long userId,  
244             @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,  
245             @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate,  
246             @RequestParam Double caloriesGoal) {  
247         userService.updateCaloriesGoal(userId, startDate, endDate, caloriesGoal);
```

```

248     return ResponseEntity.ok("Calories goal updated successfully");
249 }
250
251 /**
252 * Create new exercise in the exercise list for the user.
253 * @param userId userId.
254 * @param exerciseName the exercise the user want to add.
255 * @param date the date of the exercise.
256 * @param duration the duration of the exercise.
257 */
258 @RequestMapping(value = "/{userId}/exerciseList", method = RequestMethod.POST)
259 public ResponseEntity<String> createExercise(@PathVariable Long userId,
260     @RequestParam String exerciseName,
261     @RequestParam("date") @DateTimeFormat(pattern = "yyyy-MM-dd") Date date,
262     @RequestParam Double duration) {
263     userService.addExercise(userId, exerciseName, date, duration);
264
265     return ResponseEntity.ok("Exercise added successfully");
266 }
267
268 /**
269 * Delete the user.
270 * @param userId userId.
271 */
272 @RequestMapping(value = "/{userId}", method = RequestMethod.DELETE)
273 public ResponseEntity<String> deleteUser(@PathVariable Long userId) {
274     userService.deleteUser(userId);
275     return ResponseEntity.ok("User deleted successfully");
276 }
277
278 /**
279 * Delete the duration goal for the user.
280 * @param userId userId.
281 */
282 @RequestMapping(value = "/{userId}/durationGoal", method = {RequestMethod.DELETE})
283 public ResponseEntity<String> deleteDurationGoal(@PathVariable Long userId) {
284     userService.deleteDurationGoal(userId);
285     return ResponseEntity.ok("DurationGoal deleted successfully");
286 }
287
288 /**
289 * Delete the calories goal for the user.
290 * @param userId userId.
291 */
292 @RequestMapping(value = "/{userId}/caloriesGoal", method = RequestMethod.DELETE)
293 public ResponseEntity<String> deleteCaloriesGoal(@PathVariable Long userId) {
294     userService.deleteCaloriesGoal(userId);
295     return ResponseEntity.ok("CaloriesGoal deleted successfully");
296 }
297 }
```

## ❖ Controller Test Result

We've used Postman to create and send API requests to our backend system. Please see below our test results in detail.

### Test #1

**Test Method:** public User createUser(@RequestParam String userName, @RequestParam String password, @RequestParam Double weight)

**Operation:** POST

**Test URL:** <http://localhost:8080/api/users/register?userName=Ben&password=bbb&weight=60.0>

**Request Body:** N/A

**Expected Test Result:** New user with userName Ben added to the database

**Actual Test Result:**

ExerciseManager1-6Test / createUser

POST http://localhost:8080/api/users/register?userName=“Ben”&password=“bbb”&weight=60.0

**Params** • Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

**Query Params**

KEY	VALUE	DESCRIPTION	...	Bulk Edit
✓ userName	“Ben”			
✓ password	“bbb”			
✓ weight	60.0			
Key	Value	Description		

**Body** Cookies Headers (5) Test Results

200 OK 474 ms 170 B Save Response

Pretty Raw Preview Visualize Text

1 success

### exerciseManager.user

10 DOCUMENTS 1 INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT COLLECTION 1 - 11 of 11 Neo

- ```
password: "bbb"
weight: 60
exerciselist: Array
_class: "com.exerisemgr.exercisemanager.model.User"
```
- ```
_id: 4892883308910365836
userName: "Anna"
password: "annn"
weight: 90.1
exerciselist: Array
_class: "com.exerisemgr.exercisemanager.model.User"
```
- ```
_id: 8903661545115042885
userName: "Neo"
password: "n99"
weight: 89.2
exerciselist: Array
_class: "com.exerisemgr.exercisemanager.model.User"
```
- ```
_id: 15989519884111077
userName: "Wow"
password: "lga"
weight: 80.1
exerciselist: Array
_class: "com.exerisemgr.exercisemanager.model.User"
```
- ```
_id: 8340823194349748762
userName: "Ben"
password: "bbb"
weight: 60
exerciselist: Array
_class: "com.exerisemgr.exercisemanager.model.User"
```

**Test Result:** Passed

## Test #2

**Test Method:** public User getUserByCredentials(@RequestParam("username") String username, @RequestParam("password") String password)

**Operation:** GET

**Test URL:** http://localhost:8080/api/users/login?username=Anna&password=annn

**Request Body:** N/A

**Expected Test Result:** User with user Name Anna returned

**Actual Test Result:**

GET http://localhost:8080/api/users/login?username=Anna&password=annn Send

Params • Authorization Headers (8) Body • Pre-request Script Tests Settings Cookies

| KEY      | VALUE | DESCRIPTION | ... | Bulk Edit |
|----------|-------|-------------|-----|-----------|
| username | Anna  |             |     |           |
| password | annn  |             |     |           |
| Key      | Value | Description |     |           |

Body Cookies Headers (5) Test Results 200 OK 324 ms 298 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "id": 4892883308910365836,  
3   "userName": "Anna",  
4   "password": "annn",  
5   "weight": 90.1,  
6   "durationGoal": null,  
7   "caloriesGoal": null,  
8   "exerciseList": []  
9 }
```

**Test Result:** Passed

## Test #3

**Test Method:** public ResponseEntity<List<User>> getAllUser()

**Operation:** GET

**Test URL:** http://localhost:8080/api/users/allUsers

**Request Body:** N/A

**Expected Test Result:** All users in the database returned

**Actual Test Result:**

GET http://localhost:8080/api/users/allUsers

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

|      | KEY     | VALUE       | DESCRIPTION  | Bulk Edit |
|------|---------|-------------|--------------|-----------|
| Body | Cookies | Headers (5) | Test Results |           |

Pretty Raw Preview Visualize JSON

```

1  [
2   {
3     "id": 2719832509417932639,
4     "userName": "Jose",
5     "password": "seg",
6     "weight": 77.6,
7     "durationGoal": null,
8     "caloriesGoal": null,
9     "exerciseList": []
10    },
11    {
12      "id": 938887676302741085,
13      "userName": "Lin",
14      "password": "11222",
15      "weight": 98.2,
16      "durationGoal": {
17        "id": 8043518057942108978,
18        "userId": 938887676302741085,
19        "startDate": "2022-02-15T08:00:00.000+00:00",
20        "endDate": "2022-02-19T08:00:00.000+00:00",
21        "durationGoal": 30000.0
22      },
23      "caloriesGoal": {
24        "id": 234065553691790539,
25        "userId": 938887676302741085,
26        "startDate": "2022-02-15T08:00:00.000+00:00",
27        "endDate": "2022-02-19T08:00:00.000+00:00",
28        "caloriesGoal": 40000.0
29      },
30      "exerciseList": [
31        {
32          "id": 9082594620666430038,
33          "userId": 938887676302741085,
34          "exerciseName": "running",
35          "date": "2022-02-15T08:00:00.000+00:00",
36          "duration": 25.0,
37          "calories": 230.1
38        },
39        {
40          "id": 5075114236301758348,
41          "userId": 938887676302741085,
42          "exerciseName": "walking",
43          "date": "2022-02-17T08:00:00.000+00:00",
44          "duration": 250.0,
45          "calories": 1230.2
46        },
47        {
48          "id": 7932753669627465080,
49          "userId": 938887676302741085,
50          "exerciseName": "cycling",
51          "date": "2022-02-19T08:00:00.000+00:00",
52          "duration": 125.0,
53          "calories": 530.0
54        }
55      ],
56      {
57        "id": 357012774381374935,
58        "userId": 938887676302741085,
59        "exerciseName": "walking",
60        "date": "2022-02-19T08:00:00.000+00:00",
61        "duration": 250.0,
62        "calories": 730.0
63      },
64      {
65        "id": 8405409336368320359,
66        "userId": 938887676302741085,
67        "exerciseName": "kayaking",
68        "date": "2022-02-19T08:00:00.000+00:00",
69        "duration": 100.0,
70        "calories": 76.7
71      }
72    },
73    {
74      "id": 2962781115616742244,
75      "userName": "Mike",
76      "password": "m1000",
77      "weight": 90.8,
78      "durationGoal": {
79        "id": 3721681080220467204,
80        "userId": 2962781115616742244
81      }
82    }
83  ]

```

Status: 200 OK Time: 538 ms Size: 5.6 KB Save Response

Cookies Capture requests Runner Trash

```

81     "startDate": "2022-02-15T08:00:00.000+00:00",
82     "endDate": "2022-02-19T08:00:00.000+00:00",
83     "durationGoal": 30000.0
84   },
85   "caloriesGoal": {
86     "id": 1694549481149057157,
87     "userId": 2962781115616742244,
88     "startDate": "2022-02-15T08:00:00.000+00:00",
89     "endDate": "2022-02-19T08:00:00.000+00:00",
90     "caloriesGoal": 40000.0
91   },
92   "exerciseList": [
93     {
94       "id": 6392085278626758828,
95       "userId": 2962781115616742244,
96       "exerciseName": "running",
97       "date": "2022-02-15T08:00:00.000+00:00",
98       "duration": 25.0,
99       "calories": 230.1
100    },
101    {
102      "id": 7546541538780294098,
103      "userId": 2962781115616742244,
104      "exerciseName": "walking",
105      "date": "2022-02-17T08:00:00.000+00:00",
106      "duration": 250.0,
107      "calories": 1230.2
108    },
109    {
110      "id": 513813054268329336,
111      "userId": 2962781115616742244,
112      "exerciseName": "cycling",
113      "date": "2022-02-19T08:00:00.000+00:00",
114      "duration": 125.0,
115      "calories": 530.0
116    },
117    {
118      "id": 5859557227218487064,
119      "userId": 2962781115616742244,
120      "exerciseName": "walking",
121      "date": "2022-02-19T08:00:00.000+00:00",
122      "duration": 250.0,
123      "calories": 730.0
124    },
125    {
126      "id": 630378873627559366,
127      "userId": 2962781115616742244,
128      "exerciseName": "kayaking",
129      "date": "2022-02-19T08:00:00.000+00:00",
130      "duration": 100.0,
131      "calories": 76.7
132    }
133  ],
134 },
135 {
136   "id": 8757637119955584962,
137   "userName": "Alice",
138   "password": "alice123",
139   "weight": 70.5,
140   "durationGoal": {
141     "id": 3900944275542658672,
142     "userId": 8757637119955584962,
143     "startDate": "2022-02-15T08:00:00.000+00:00",
144     "endDate": "2022-02-19T08:00:00.000+00:00",
145     "durationGoal": 30000.0
146   },
147   "caloriesGoal": {
148     "id": 1923117620111689881,
149     "userId": 8757637119955584962,
150     "startDate": "2022-02-15T08:00:00.000+00:00",
151     "endDate": "2022-02-19T08:00:00.000+00:00",
152     "caloriesGoal": 40000.0
153   },
154   "exerciseList": [

```

```

155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
    {
        "id": 3381584537267358921,
        "userId": 8757637119955584962,
        "exerciseName": "running",
        "date": "2022-02-15T08:00:00.000+00:00",
        "duration": 25.0,
        "calories": 230.1
    },
    {
        "id": 3759168729245370287,
        "userId": 8757637119955584962,
        "exerciseName": "walking",
        "date": "2022-02-17T08:00:00.000+00:00",
        "duration": 250.0,
        "calories": 1230.2
    },
    {
        "id": 5612479889122216838,
        "userId": 8757637119955584962,
        "exerciseName": "cycling",
        "date": "2022-02-19T08:00:00.000+00:00",
        "duration": 125.0,
        "calories": 530.0
    },
    {
        "id": 6329918108370553085,
        "userId": 8757637119955584962,
        "exerciseName": "walking",
        "date": "2022-02-19T08:00:00.000+00:00",
        "duration": 250.0,
        "calories": 730.0
    },
    {
        "id": 7690744618914038691,
        "userId": 8757637119955584962,
        "exerciseName": "kayaking",
        "date": "2022-02-19T08:00:00.000+00:00",
        "duration": 100.0,
        "calories": 76.7
    }
],
{
    "id": 1168825934850507411,
    "userName": "Ken",
    "password": "kkk123",
    "weight": 80.5,
    "durationGoal": {
        "id": 7248139580008515573,
        "userId": 7330243107283945319,
        "startDate": "2022-02-15T08:00:00.000+00:00",
        "endDate": "2022-02-21T08:00:00.000+00:00",
        "durationGoal": 30000.0
    },
    "caloriesGoal": {
        "id": 23406555291790539,
        "userId": 1168825934850507411,
        "startDate": "2022-02-15T08:00:00.000+00:00",
        "endDate": "2022-02-19T08:00:00.000+00:00",
        "caloriesGoal": 40000.0
    },
    "exerciseList": [
        {
            "id": 908259462666430038,
            "userId": 1168825934850507411,
            "exerciseName": "running",
            "date": "2022-02-15T08:00:00.000+00:00",
            "duration": 25.0,
            "calories": 230.1
        },
        {
            "id": 5075114236301758348,
            "userId": 1168825934850507411,
            "exerciseName": "walking",
            "date": "2022-02-17T08:00:00.000+00:00",
            "duration": 250.0,
            "calories": 1230.2
        },
        {

```

```

234     "id": 7932753669627465080,
235     "userId": 1168825934850507411,
236     "exerciseName": "cycling",
237     "date": "2022-02-19T08:00:00.000+00:00",
238     "duration": 125.0,
239     "calories": 530.0
240   },
241   {
242     "id": 357012774381374935,
243     "userId": 1168825934850507411,
244     "exerciseName": "walking",
245     "date": "2022-02-19T08:00:00.000+00:00",
246     "duration": 250.0,
247     "calories": 730.0
248   },
249   {
250     "id": 8405409336368320359,
251     "userId": 1168825934850507411,
252     "exerciseName": "kayaking",
253     "date": "2022-02-19T08:00:00.000+00:00",
254     "duration": 100.0,
255     "calories": 76.7
256   }
257 ],
258 },
259 }

260   {
261     "id": 7294158642769053171,
262     "userName": "\"Bill\"",
263     "password": "\"bbb\"",
264     "weight": 60.0,
265     "durationGoal": null,
266     "caloriesGoal": null,
267     "exerciseList": []
268   },
269   {
270     "id": 4892883308910365836,
271     "userName": "Anna",
272     "password": "annn",
273     "weight": 90.1,
274     "durationGoal": null,
275     "caloriesGoal": null,
276     "exerciseList": []
277   },
278   {
279     "id": 890361545115042885,
280     "userName": "Neo",
281     "password": "n99",
282     "weight": 89.2,
283     "durationGoal": null,
284     "caloriesGoal": null,
285     "exerciseList": []
286   },
287   {
288     "id": 8340823194349748762,
289     "username": "\"Ben\"",
290     "password": "\"bbb\"",
291     "weight": 60.0,
292     "durationGoal": null,
293     "caloriesGoal": null,
294     "exerciseList": []
295   }
296 ]

```

**Test Result:** Passed

#### Test #4

**Test Method:** public User getUserById(@PathVariable Long userId)

**Operation:** GET

**Test URL:** <http://localhost:8080/api/users/1168825934850507411>

**Request Body:** N/A

**Expected Test Result:** User with userId 1168825934850507411 returned

**Actual Test Result:**

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62

```

The screenshot shows the Postman interface with a successful API call to `/api/users/1168825934850507411`. The response body is a JSON object containing user information, exercise goals, and a list of exercises.

```

{
  "id": 1168825934850507411,
  "userName": "Ken",
  "password": "kkk123",
  "weight": 80.5,
  "durationGoal": {
    "id": 8043518057942108978,
    "userId": 1168825934850507411,
    "startDate": "2022-02-15T08:00:00.000+00:00",
    "endDate": "2022-02-19T08:00:00.000+00:00",
    "durationGoal": 30000.0
  },
  "caloriesGoal": {
    "id": 23406553691790539,
    "userId": 1168825934850507411,
    "startDate": "2022-02-15T08:00:00.000+00:00",
    "endDate": "2022-02-19T08:00:00.000+00:00",
    "caloriesGoal": 40000.0
  },
  "exerciseList": [
    {
      "id": 9882594620664306938,
      "userId": 1168825934850507411,
      "exerciseName": "running",
      "date": "2022-02-15T08:00:00.000+00:00",
      "duration": 25.0,
      "calories": 230.1
    },
    {
      "id": 5075114236301758348,
      "userId": 1168825934850507411,
      "exerciseName": "walking",
      "date": "2022-02-17T08:00:00.000+00:00",
      "duration": 250.0,
      "calories": 1230.2
    },
    {
      "id": 7932753669627465080,
      "userId": 1168825934850507411,
      "exerciseName": "cycling",
      "date": "2022-02-19T08:00:00.000+00:00",
      "duration": 125.0,
      "calories": 530.0
    },
    {
      "id": 357012774381374935,
      "userId": 1168825934850507411,
      "exerciseName": "walking",
      "date": "2022-02-19T08:00:00.000+00:00",
      "duration": 250.0,
      "calories": 730.0
    },
    {
      "id": 8405409336368320359,
      "userId": 1168825934850507411,
      "exerciseName": "kayaking",
      "date": "2022-02-19T08:00:00.000+00:00",
      "duration": 100.0,
      "calories": 76.7
    }
  ]
}

```

**Test Result:** Passed

### Test #5

**Test Method:** public Double getWeightByUserId(@PathVariable Long userId)

**Operation:** GET

**Test URL:** <http://localhost:8080/api/users/1168825934850507411/weight>

**Request Body:** N/A

**Expected Test Result:** Weight with userId 1168825934850507411 returned

**Actual Test Result:**

The screenshot shows the Postman interface with a test named "ExerciseManager1-6Test / getWeightByUserId". The request method is GET, and the URL is <http://localhost:8080/api/users/1168825934850507411/weight>. The response status is 200 OK, and the body contains the value "1 80.5".

**Test Result:** Passed

### Test #6

**Test Method:** public DurationGoal getDurationGoalById(@PathVariable Long userId)  
**Operation:** GET

**Test URL:** <http://localhost:8080/api/users/1168825934850507411/durationGoal>

**Request Body:** N/A

**Expected Test Result:** DurationGoal with userId 1168825934850507411 returned

**Actual Test Result:**

The screenshot shows the Postman interface with a test named "ExerciseManager1-6Test / getDurationGoalById". The request method is GET, and the URL is <http://localhost:8080/api/users/1168825934850507411/durationGoal>. The response status is 200 OK, and the body is a JSON object representing a DurationGoal.

```

1 {
2   "id": 8043518057942108978,
3   "userId": 1168825934850507411,
4   "startDate": "2022-02-15T08:00:00.000+00:00",
5   "endDate": "2022-02-19T08:00:00.000+00:00",
6   "durationGoal": 30900.0
7 }
  
```

**Test Result:** Passed

### Test #7

**Test Method:** public CaloriesGoal getCaloriesGoalById(@PathVariable Long userId)  
**Operation:** GET

**Test URL:** <http://localhost:8080/api/users/1168825934850507411/caloriesGoal>

**Request Body:** N/A

**Expected Test Result:** CaloriesGoal with userId 1168825934850507411 returned

**Actual Test Result:**

GET http://localhost:8080/api/users/1168825934850507411/caloriesGoal

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

| KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description |     |           |

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON ↻

Status: 200 OK Time: 58 ms Size: 328 B Save Response ↻

```
1 {  
2     "id": 7100385009237836066,  
3     "userId": 1168825934850507411,  
4     "startDate": "2022-02-15T08:00:00.000+00:00",  
5     "endDate": "2022-02-19T08:00:00.000+00:00",  
6     "caloriesGoal": 40000.0  
7 }
```

**Test Result:** Passed

### Test #8

**Test Method:** public List<Exercise> getAllExerciseByUserId(@PathVariable Long userId)

**Operation:** GET

**Test URL:** http://localhost:8080/api/users/1168825934850507411/exercises

**Request Body:** N/A

**Expected Test Result:** All exercises with userId 1168825934850507411 returned

**Actual Test Result:**

GET http://localhost:8080/api/users/1168825934850507411/exercises

**Params** Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

**Query Params**

| KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
| Key | Value | Description | ... |           |

**Body** Cookies Headers (5) Test Results

Status: 200 OK Time: 28 ms Size: 935 B | Save Response ▾

Pretty Raw Preview Visualize JSON

```

1 [
2   {
3     "id": 8032906545372087743,
4     "userId": 1168825934850507411,
5     "exerciseName": "running",
6     "date": "2022-02-15T08:00:00.000+00:00",
7     "duration": 25.0,
8     "calories": 230.1
9   },
10  {
11    "id": 4967259014025789512,
12    "userId": 1168825934850507411,
13    "exerciseName": "walking",
14    "date": "2022-02-17T08:00:00.000+00:00",
15    "duration": 250.0,
16    "calories": 1230.2
17  },
18  {
19    "id": 7381590578320000328,
20    "userId": 1168825934850507411,
21    "exerciseName": "cycling",
22    "date": "2022-02-19T08:00:00.000+00:00",
23    "duration": 125.0,
24    "calories": 530.0
25  },
26  {
27    "id": 5032002845671703977,
28    "userId": 1168825934850507411,
29    "exerciseName": "walking",
30    "date": "2022-02-19T08:00:00.000+00:00",
31    "duration": 250.0,
32    "calories": 730.0
33  },
34  {
35    "id": 9066046195560303288,
36    "userId": 1168825934850507411,
37    "exerciseName": "kayaking",
38    "date": "2022-02-19T08:00:00.000+00:00",
39    "duration": 100.0,
40    "calories": 76.7
41  }
]

```

**Test Result:** Passed

### Test #9

**Test Method:** public Map<Date, Double> getDailyDurationSumMap(@PathVariable Long userId, @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate, @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate)

**Operation:** GET

**Test URL:**

http://localhost:8080/api/users/1168825934850507411/dailyDurations?startDate=2022-02-15T08:00:00.000+00:00&endDate=2022-02-21T08:00:00.000+00:00

**Request Body:** N/A

**Expected Test Result:** The daily duration sum map from 2023-02-15 to 2023-02-21 with userId 1168825934850507411 returned

**Actual Test Result:**

GET http://localhost:8080/api/users/1168825934850507411/dailyDurations?startDate=2022-02-15T08:00:00.000+00:00&endDate=2022-02-21T08:00:00.000+00:00

Params • Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

| KEY                                           | VALUE                         | DESCRIPTION | ... | Bulk Edit |
|-----------------------------------------------|-------------------------------|-------------|-----|-----------|
| <input checked="" type="checkbox"/> startDate | 2022-02-15T08:00:00.000+00:00 |             |     |           |
| <input checked="" type="checkbox"/> endDate   | 2022-02-21T08:00:00.000+00:00 |             |     |           |
| Key                                           | Value                         | Description |     |           |

Body Cookies Headers (5) Test Results Status: 200 OK Time: 257 ms Size: 422 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "2022-02-19T08:00:00.000+00:00": 475.0,  
2 "2022-02-16T08:00:00.000+00:00": 0.0,  
3 "2022-02-21T08:00:00.000+00:00": 0.0,  
4 "2022-02-18T08:00:00.000+00:00": 0.0,  
5 "2022-02-20T08:00:00.000+00:00": 0.0,  
6 "2022-02-15T08:00:00.000+00:00": 25.0,  
7 "2022-02-17T08:00:00.000+00:00": 250.0  
8  
9 }
```

**Test Result:** Passed

**Test #10**

**Test Method:** public Map<Date, Double> getDailyCaloriesSumMap(@PathVariable Long userId, @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate, @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate)

**Operation:** GET

**Test URL:**

http://localhost:8080/api/users/1168825934850507411/dailyCalories?startDate=2022-02-15T08:00:00.000+00:00&endDate=2022-02-21T08:00:00.000+00:00

**Request Body:** N/A

**Expected Test Result:** The daily calories sum map from 2023-02-15 to 2023-02-21 with userId 1168825934850507411 returned

**Actual Test Result:**

GET http://localhost:8080/api/users/1168825934850507411/dailyCalories?startDate=2022-02-15T08:00:00.000+00:00&endDate=2022-02-21T08:00:00.000+00:00

**Params**

| KEY                                           | VALUE                         | DESCRIPTION | ... | Bulk Edit |
|-----------------------------------------------|-------------------------------|-------------|-----|-----------|
| <input checked="" type="checkbox"/> startDate | 2022-02-15T08:00:00.000+00:00 |             |     |           |
| <input checked="" type="checkbox"/> endDate   | 2022-02-21T08:00:00.000+00:00 |             |     |           |
| Key                                           | Value                         | Description |     |           |

**Body**

```

1   "2022-02-19T08:00:00.000+00:00": 1336.7,
2   "2022-02-16T08:00:00.000+00:00": 0.0,
3   "2022-02-21T08:00:00.000+00:00": 0.0,
4   "2022-02-18T08:00:00.000+00:00": 0.0,
5   "2022-02-20T08:00:00.000+00:00": 0.0,
6   "2022-02-15T08:00:00.000+00:00": 230.1,
7   "2022-02-17T08:00:00.000+00:00": 1230.2
8
9

```

Status: 200 OK Time: 26 ms Size: 425 B Save Response

**Test Result:** Passed

### Test #11

**Test Method:** 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)

**Operation:** GET

**Test URL:**

http://localhost:8080/api/users/1168825934850507411/durationTotalBetweenDates?startDate=2022-02-15T08:00:00.000+00:00&endDate=2022-02-21T08:00:00.000+00:00

**Request Body:** N/A

**Expected Test Result:** The duration total from 2023-02-15 to 2023-02-21 with userId 1168825934850507411 returned

**Actual Test Result:**

GET http://localhost:8080/api/users/1168825934850507411/durationTotalBetweenDates?startDate=2022-02-15T08:00:00.000+00:00...&endDate=2022-02-21T08:00:00.000+00:00

**Params**

| KEY                                           | VALUE                         | DESCRIPTION | ... | Bulk Edit |
|-----------------------------------------------|-------------------------------|-------------|-----|-----------|
| <input checked="" type="checkbox"/> startDate | 2022-02-15T08:00:00.000+00:00 |             |     |           |
| <input checked="" type="checkbox"/> endDate   | 2022-02-21T08:00:00.000+00:00 |             |     |           |
| Key                                           | Value                         | Description |     |           |

**Body**

```
1   750.0
```

Status: 200 OK Time: 68 ms Size: 169 B Save Response

## Test Result: Passed

### Test #12

**Test Method:** 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)

**Operation:** GET

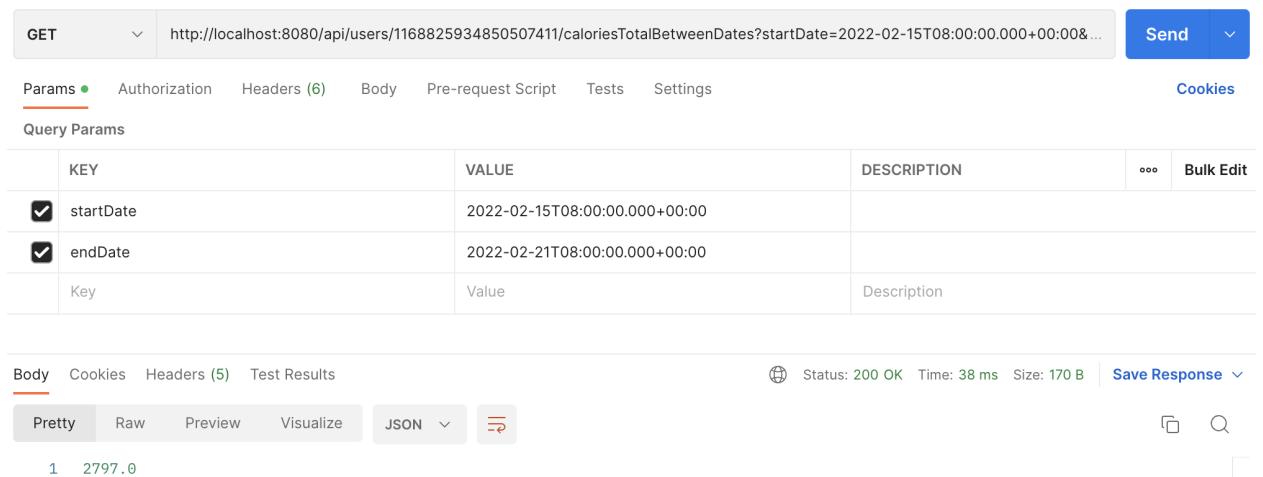
**Test URL:**

http://localhost:8080/api/users/1168825934850507411/caloriesTotalBetweenDates?startDate=2022-02-15T08:00:00.000+00:00&endDate=2022-02-21T08:00:00.000+00:00

**Request Body:** N/A

**Expected Test Result:** The calories total from 2023-02-15 to 2023-02-21 with userId 1168825934850507411 returned

**Actual Test Result:**



The screenshot shows a Postman test result for a GET request. The URL is http://localhost:8080/api/users/1168825934850507411/caloriesTotalBetweenDates?startDate=2022-02-15T08:00:00.000+00:00&endDate=2022-02-21T08:00:00.000+00:00. The request method is GET. The response status is 200 OK, with a time of 38 ms and a size of 170 B. The response body is 2797.0.

## Test Result: Passed

### Test #13

**Test Method:** public Double getSmallestDurationBetweenDates(@PathVariable Long userId, @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate, @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate)

**Operation:** GET

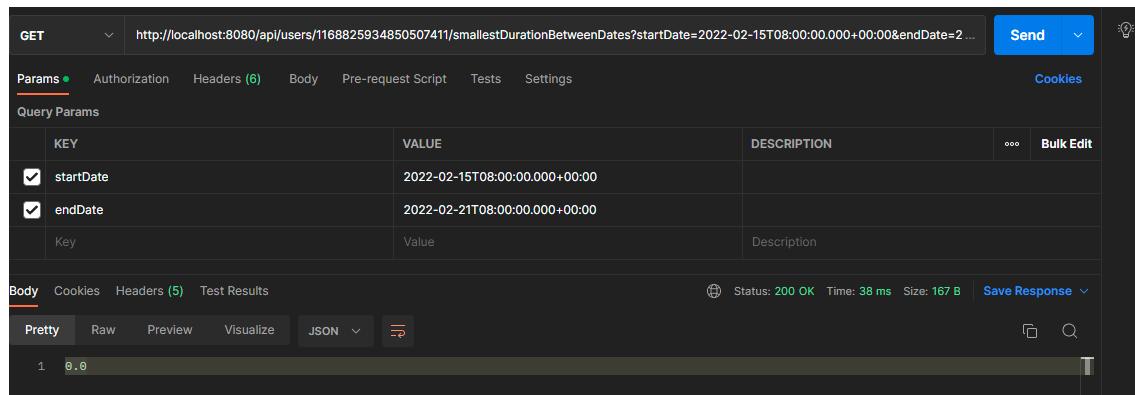
**Test URL:**

<http://localhost:8080/api/users/1168825934850507411/smallestDurationBetweenDates?startDate=2022-02-15T08:00:00.000+00:00&endDate=2022-02-21T08:00:00.000+00:00>

**Request Body:** N/A

**Expected Test Result:** The smallest duration from 2023-02-15 to 2023-02-21 with userId 1168825934850507411 returned

**Actual Test Result:**



The screenshot shows the Postman test interface for the getSmallestDurationBetweenDates API. The request method is set to GET, and the URL is http://localhost:8080/api/users/1168825934850507411/smallestDurationBetweenDates?startDate=2022-02-15T08:00:00.000+00:00&endDate=2022-02-21T08:00:00.000+00:00. The 'Params' tab is selected, showing two query parameters: 'startDate' with value '2022-02-15T08:00:00.000+00:00' and 'endDate' with value '2022-02-21T08:00:00.000+00:00'. The response status is 200 OK, time taken is 38 ms, and size is 167 B.

**Test Result:** Passed

### Test #14

**Test Method:** public Double getSmallestCaloriesBetweenDates(@PathVariable Long userId, @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate, @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate)

**Operation:** GET

**Test URL:**

<http://localhost:8080/api/users/1168825934850507411/smallestCaloriesBetweenDates?startDate=2022-02-15T08:00:00.000+00:00&endDate=2022-02-21T08:00:00.000+00:00>

**Request Body:** N/A

**Expected Test Result:** The smallest calories from 2023-02-15 to 2023-02-21 with userId 1168825934850507411 returned

**Actual Test Result:**

Test Result: Passed

### Test #15

**Test Method:** public void updateWeight(@PathVariable Long userId, @RequestParam Double weight)

**Operation:** PUT

**Test URL:**

<http://localhost:8080/api/users/1168825934850507411/weight>

**Request Body:** 100.0 (weight)

**Expected Test Result:** update userId 1168825934850507411 weight to 100.0

**Actual Test Result:**

1 Weight updated successfully

**Database check:**

```
_id: 1168825934850507411
userName: "Ken"
password: "kkk123"
weight: 100
> durationGoal: Object
> caloriesGoal: Object
> exerciseList: Array
_class: "com.exerisemgr.exercisemanager.model.User"
```

**Test Result:** Passed

### Test #16

**Test Method:** public void updatePassword(@PathVariable Long userId, @PathVariable String password)

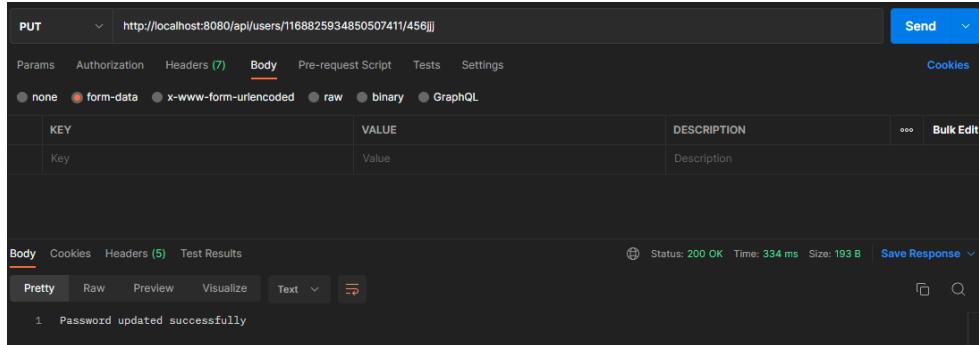
**Operation:** PUT

**Test URL:**

<http://localhost:8080/api/users/1168825934850507411/456iji>

**Request Body:** N/A

**Expected Test Result:** update userId 1168825934850507411 password to 456jjj  
**Actual Test Result:**



The screenshot shows a Postman interface with a 'PUT' request to 'http://localhost:8080/api/users/1168825934850507411/456jjj'. The 'Body' tab is selected, showing a table with one row: 'Key' (Value) and 'Value' (Description). The response status is 200 OK, time 334 ms, size 193 B. The response body is '1 Password updated successfully'.

### Database Check

```
_id: 1168825934850507411
userName: "Ken"
password: "456jjj"
weight: 80.5
> durationGoal: Object
> caloriesGoal: Object
> exerciseList: Array
_class: "com.exerisemgr.exercisemanager.model.User"
```

**Test Result:** Passed

### Test #17

**Test Method:** public void updateDurationGoal(@PathVariable Long userId, @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate, @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate, @RequestParam Double durationGoal)

**Operation:** PUT

**Test URL:**

<http://localhost:8080/api/users/1168825934850507411/durationGoal>

**Request Body:** 3000.0 (durationGoal)

2023-02-15 (startDate)

2023-02-21 (endDate)

**Expected Test Result:** update userId 1168825934850507411 durationGoal to 3000.0 from 2023-02-15 to 2023-02-21

**Actual Test Result:**

The screenshot shows a Postman interface with the following details:

- Method:** PUT
- URL:** <http://localhost:8080/api/users/1168825934850507411/durationGoal>
- Body Type:** form-data
- Body Fields:**

| KEY          | VALUE                         | DESCRIPTION |
|--------------|-------------------------------|-------------|
| durationGoal | 3000.0                        |             |
| startDate    | 2022-02-15T08:00:00.000+00:00 |             |
| endDate      | 2022-02-21T08:00:00.000+00:00 |             |
- Status:** 200 OK
- Time:** 106 ms
- Size:** 198 B
- Response Body:** "Duration goal updated successfully"

## Database Check

```

_id: 1168825934850507411
userName: "Ken"
password: "durationGoal"
weight: 80.5
durationGoal: Object
  _id: 8889932433995023845
  durationGoal: 3000
  userId: 7502546917618566409
  startDate: 2023-02-15T00:00:00.000+00:00
  endDate: 2023-02-21T00:00:00.000+00:00
  caloriesGoal: Object
  exerciseList: Array
  _class: "com.exerisemgr.exercisemanager.model.User"

```

**Test Result:** Passed

## Test #18

**Test Method:** public ResponseEntity<String> updateCaloriesGoal(@PathVariable Long userId, @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate, @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate, @RequestParam Double caloriesGoal)

**Operation:** PUT

**Test URL:**

<http://localhost:8080/api/users/1168825934850507411/caloriesGoal>

**Request Body:** 5000.0 (caloriesGoal)

2023-02-15 (startDate)

2023-02-21 (endDate)

**Expected Test Result:** update userId 1168825934850507411 caloriesGoal to 5000.0  
from 2023-02-15 to 2023-02-21

**Actual Test Result:**

The screenshot shows a Postman interface with a 'PUT' method selected. The URL is `http://localhost:8080/api/users/1168825934850507411/caloriesGoal?caloriesGoal=5000&startDate=2023-02-15T00:00:00.000+00:00&endDate=2023-02-21T00:00:00.000+00:00`. The 'Params' tab is active, showing three query parameters: 'caloriesGoal' (value: 5000), 'startDate' (value: 2023-02-15), and 'endDate' (value: 2023-02-21). The response status is 200 OK, with a time of 729 ms and a size of 198 B. The response body is: "1 Calories goal updated successfully".

### Database Check

```

_id: 1168825934850507411
userName: "Ken"
password: "kkk123"
weight: 80.5
durationGoal: Object
caloriesGoal: Object
_id: 2153866899858344386
caloriesGoal: 5000
userId: 3250401998014466736
startDate: 2023-02-15T00:00:00.000+00:00
endDate: 2023-02-21T00:00:00.000+00:00
exerciseList: Array
_class: "com.exerisemgr.exercisemanager.model.User"

```

**Test Result:** Passed

### Test #19

**Test Method:** public ResponseEntity<String> createExercise(@PathVariable Long userId, @RequestParam String exerciseName, @RequestParam Date date, @RequestParam Double duration)

**Operation:** POST

**Test URL:**

<http://localhost:8080/api/users/1168825934850507411/exerciseList>

**Request Body:** running (exerciseName)  
2022-02-21T00:00:00.000+00:00 (date)  
50 (duration)

**Expected Test Result:** Add exercise to userId 1168825934850507411 exerciseList with exercise {exerciseName: running, date: 2022-02-21T00:00:00.000+00:00, duration: 50}

**Actual Test Result:**

The screenshot shows a Postman interface with a POST request to `http://localhost:8080/api/users/1168825934850507411/exerciseList?userId=1168825934850507411&e...`. The Params tab contains four entries: `userId` (value: 1168825934850507411), `exerciseName` (value: running), `date` (value: 2022-02-21T08:00:00.000+00:00), and `duration` (value: 50). The response status is 200 OK, with 654 ms latency and 191 B size. The response body is: "Exercise added successfully".

## Database Check

```

_id: 1168825934850507411
userName: "Ken"
password: "456jjj"
weight: 80.5
durationGoal: Object
caloriesGoal: Object
exerciseList: Array
  0: Object
  1: Object
  2: Object
  3: Object
  4: Object
  5: Object
    _id: 7576025435215709196
    userId: 1168825934850507411
    exerciseName: "running"
    date: 2022-02-21T00:00:00.000+00:00
    duration: 50
    calories: 0
  class: "com.exerisemgr.exercisemanager.model.User"

```

**Test Result:** Passed

### Test #20

**Test Method:** @RequestMapping(value = "/{userId}", method = RequestMethod.DELETE)  
**public ResponseEntity<String> deleteUser(@PathVariable Long userId)**

**Operation:** DELETE

**Test URL:**

<http://localhost:8080/api/users/1168825934850507411>

**Request Body:**

**Expected Test Result:** delete user with userId 1168825934850507411

**Actual Test Result:**

The screenshot shows a Postman interface with a DELETE request to the URL `http://localhost:8080/api/users/1168825934850507411`. The response status is 200 OK, and the body contains the message "User deleted successfully".

## Database Check

|                                                                                                                                                                |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>_id: 2719832509417932639 userName: "Jose" password: "seg" weight: 77.6 &gt; exerciseList: Array _class: "com.exerisemgr.exercisemanager.model.User"</pre> |
| <pre>_id: 2115626923314204356 userName: "Jose" password: "seg" weight: 77.6 &gt; exerciseList: Array _class: "com.exerisemgr.exercisemanager.model.User"</pre> |
| <pre>_id: 6227000740225632374 userName: "Jose" password: "seg" weight: 77.6 &gt; exerciseList: Array _class: "com.exerisemgr.exercisemanager.model.User"</pre> |

**Test Result:** Passed

### Test #21

**Test Method:** @RequestMapping(value = "/{userId}/durationGoal", method = {RequestMethod.DELETE})  
**public ResponseEntity<String> deleteDurationGoal(@PathVariable Long userId)**

**Operation:** DELETE

**Test URL:**

<http://localhost:8080/api/users/1168825934850507411/durationGoal>

**Request Body:**

**Expected Test Result:** delete the durationGoal of userId 1168825934850507411

**Actual Test Result:**

DELETE http://localhost:8080/api/users/1168825934850507411/durationGoal

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

| KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
|     |       |             |     |           |

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

1 DurationGoal deleted successfully

## Database Check

```
_id: 1168825934850507411
userName: "Ken"
password: "456jjj"
weight: 80.5
> caloriesGoal: Object
> exerciseList: Array
  _class: "com.exerisemgr.exercisemanager.model.User"
```

**Test Result:** Passed

### Test #22

**Test Method:** @RequestMapping(value = "/{userId}/caloriesGoal", method = RequestMethod.DELETE)  
**public ResponseEntity<String> deleteCaloriesGoal(@PathVariable Long userId)**

### Operation: DELETE

**Test URL:**

<http://localhost:8080/api/users/1168825934850507411/caloriesGoal>

**Request Body:**

**Expected Test Result:** delete caloriesGoal with userId 1168825934850507411

**Actual Test Result:**

DELETE http://localhost:8080/api/users/1168825934850507411/caloriesGoal

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

| KEY | VALUE | DESCRIPTION | ... | Bulk Edit |
|-----|-------|-------------|-----|-----------|
|     |       |             |     |           |

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize Text

1 CaloriesGoal deleted successfully

## Database Check

```
_id: 1168825934850507411
userName: "Ken"
password: "456jjj"
weight: 80.5
> exerciseList: Array
  _class: "com.exerisemgr.exercisemanager.model.User"
```

---

**Test Result:** Passed

## ■ **Code Quality Assessment**

**We utilized CodeMR to assess our code quality.**

**Summary:** Overall, our program is evaluated with good quality in most perspectives, with only a small portion of trivial-medium level problematic classes in terms of complexity (39%) and cohesion (11%). We provide our interpretation focusing on the possible reasons and solutions to improve it in the near future.

**Interpretation:**

**1. Lack of cohesion:** According to the report, the problem was found in the User class.

- ❖ Possible reasons: the presence of exerciseList, which can be separated from the user object conceptually.
- ❖ To improve it: we can refactor the exerciseList into a different ExerciseList class and then link User with ExerciseList classes instead. By doing so, we let the User class focus on user-specific to strengthen the cohesion of our code.

**2. Complexity:** According to the report, the problem is found in the UserServiceImpl class in the service package and the ResourceNotFoundException class in the expectation package.

- ❖ Possible reasons:  
For UserServiceImpl, the use of hardcoded values can add difficulties for future code change and maintenance (despite the fact that we make the number of MET for each exercise constant already, we still hardcoded the activity type in calculateCalories, for example).

For the ResourceNotFoundException class, the lack of context can be confusing for debugging because it is hard to know why the resource is not found (we are not providing any information at this moment).

- ❖ To improve it:  
For UserServiceImpl class, we can define constants for activity types as well, and move these constants to the top of the class.  
For the ResourceNotFoundException class, we can provide more information, such as the name of the resource or the place where the resource is not found, in order to make debugging easier.

## **Report**

See the screenshots of our CodeMR report below.

## Analysis of ExerciseManager

General Information

**Total lines of code: 525**

**Number of classes: 11**

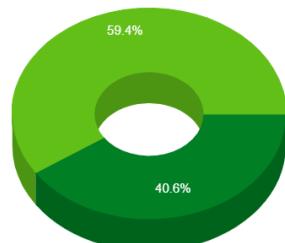
**Number of packages: 6**

**Number of external packages: 15**

**Number of external classes: 60**

**Number of problematic classes: 1**

**Number of highly problematic classes: 0**

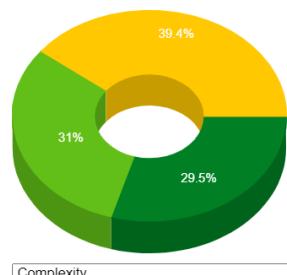


Coupling

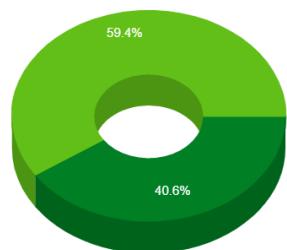
- Very High
- High
- Medium-high
- Low-medium
- Low

## Distribution of Quality Attributes

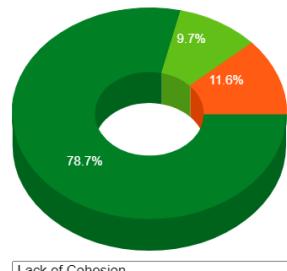
Complexity, Coupling, Cohesion, and Size



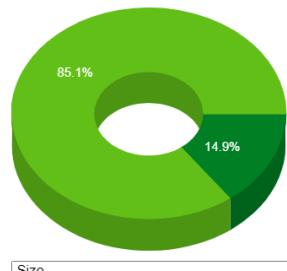
Complexity



Coupling



Lack of Cohesion



Size

## List of all classes (#11)

| ID | CLASS                     | COUPLING | COMPLEXITY | LACK OF COHESION | SIZE | LOC         | COMPLEXITY | COUPLING | LACK OF COHESION | SIZE       |
|----|---------------------------|----------|------------|------------------|------|-------------|------------|----------|------------------|------------|
| 1  | UserServiceImpl           | █        | █          | █                | 204  | medium-high | low-medium | low      | low              | low-medium |
| 2  | UserController            | █        | █          | █                | 108  | low-medium  | low-medium | low      | low              | low-medium |
| 3  | ResourceNotFoundException | █        | █          | █                | 3    | medium-high | low        | low      | low              | low        |
| 4  | UserService               | █        | █          | █                | 23   | low-medium  | low        | low      | low              | low-medium |
| 5  | DurationGoal              | █        | █          | █                | 16   | low-medium  | low        | low      | low              | low        |
| 6  | CaloriesGoal              | █        | █          | █                | 16   | low-medium  | low        | low      | low              | low        |
| 7  | User                      | █        | █          | █                | 61   | low         | low        | high     | low              | low-medium |
| 8  | Exercise                  | █        | █          | █                | 51   | low         | low        | low      | low              | low-medium |
| 9  | Goal                      | █        | █          | █                | 36   | low         | low        | low      | low              | low        |
| 10 | ExerciseManagerAp...      | █        | █          | █                | 5    | low         | low        | low      | low              | low        |
| 11 | UserRepository            | █        | █          | █                | 2    | low         | low        | low      | low              | low        |

