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

- **Project Board**
  https://github.com/users/zhaoluojia/projects/1/views/3

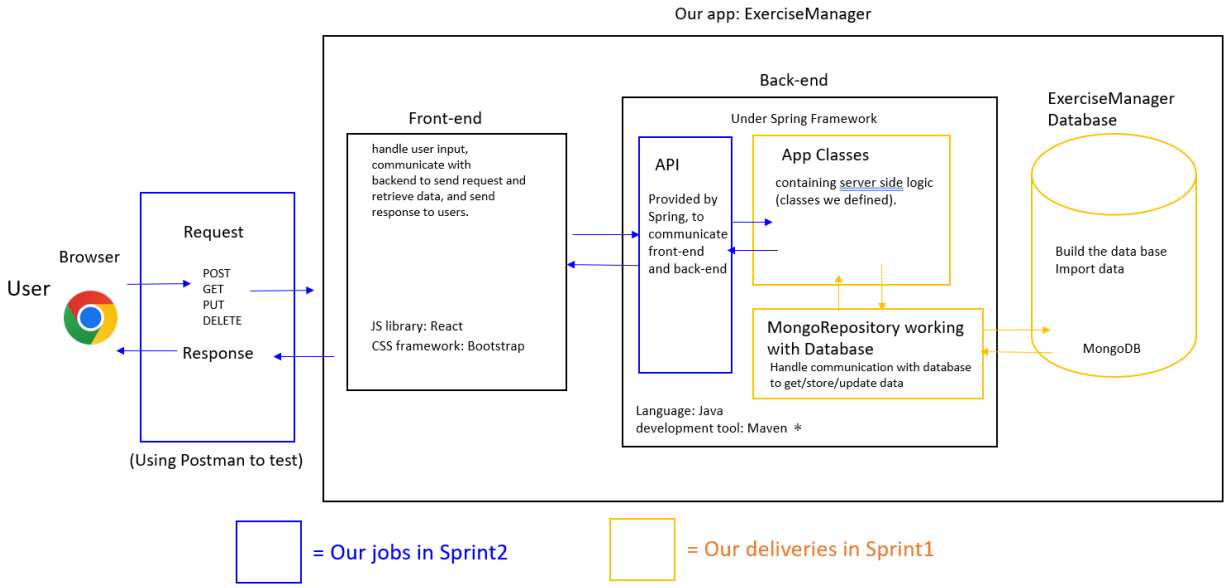
- **Project Name**
  Exercise Manager

- **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.

```java
package com.exerisemgr.exercisemanager.controller;

import com.exerisemgr.exercisemanager.model.CaloriesGoal;
import com.exerisemgr.exercisemanager.model.DurationGoal;
import com.exerisemgr.exercisemanager.model.Exercise;
import com.exerisemgr.exercisemanager.model.User;
import com.exerisemgr.exercisemanager.service.UserService;

import java.util.Date;
import java.util.List;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/users")
public class UserController {

  @Autowired
  private UserService userService;

  /**
   * Create the User.
   * @return: string
   */
  @RequestMapping(value="/register", method = RequestMethod.POST)
  public String createUser(@RequestParam String userName,@RequestParam String password, @RequestParam Double weight) {
    User user = new User(userName, password, weight);
    userService.createUser(user);
    return "success";
  }

  /**
   * Get the User by user information.
   * @param username the userName.
   * @param password the password.
   * @return User object
   */
  @RequestMapping(value = "/login", method = RequestMethod.GET)
  public User getUserByCredentials(@RequestParam("username") String username, @RequestParam("password") String password) {
    return userService.getUserByCredentials(username, password);
  }

  /**
   * Get All the Users by user information.
   * @return All the Users
   */
  @RequestMapping(value = "/allUsers", method = RequestMethod.GET)
  public ResponseEntity<List<User>> getAllUser(){
    return new ResponseEntity<List<User>>(userService.getAllUser(), HttpStatus.OK);
  }

  /**
   * Get the User by userId
   * @param userId: the userId.
   * @return: the User.
   */
  @RequestMapping(value="/{userId}", method = RequestMethod.GET)
  public User getUserById(@PathVariable Long userId) {
    return userService.getUserByUserId(userId);
  }

  /**
   * Get the weight of the user by userId.
   * @param userId userId.
```

```java
     * @return user's weight.
     */
    @RequestMapping(value = "/{userId}/weight", method = RequestMethod.GET)
    public Double getWeightByUserId(@PathVariable Long userId) {
      return userService.getWeightByUserId(userId);
    }

    /**
     * Get the DurationGoal object by userId.
     * @param userId userId.
     * @return user's DurationGoal object.
     */
    @RequestMapping(value = "/{userId}/durationGoal", method = RequestMethod.GET)
    public DurationGoal getDurationGoalByUserId(@PathVariable Long userId) {
      return userService.getDurationGoalByUserId(userId);
    }

    /**
     * Get the CaloriesGoal object by userId.
     * @param userId userId.
     * @return user's CaloriesGoal object.
     */
    @RequestMapping(value = "/{userId}/caloriesGoal", method = RequestMethod.GET)
    public CaloriesGoal getCaloriesGoalByUserId(@PathVariable Long userId) {
      return userService.getCaloriesGoalByUserId(userId);
    }

    /**
     * Get a list of all Exercise objects of the user by userId
     * @param userId userId.
     * @return a list of Exercise objects.
     */
    @RequestMapping(value = "/{userId}/exercises", method = RequestMethod.GET)
    public List<Exercise> getAllExerciseByUserId(@PathVariable Long userId) {
      return userService.getAllExerciseByUserId(userId);
    }

    /**
     * Get a map of each date to that day's duration sum by user id and start date, end date.
     * @param userId userId.
     * @param startDate the start date (Date).
     * @param endDate the end date (Date).
     * @return a Map mapping each day to its duration sum.
     */
    @RequestMapping(value = "/{userId}/dailyDurations", method = RequestMethod.GET)
    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) {
      return userService.getDailyDurationSumMap(userId, startDate, endDate);
    }

    /**
     * Get a map of each date to that day's calories sum by user id and start date, end date.
     * @param userId userId.
     * @param startDate the start date (Date).
     * @param endDate the end date (Date).
     * @return a Map mapping each day to its calories sum.
     */
    @RequestMapping(value = "/{userId}/dailyCalories", method = RequestMethod.GET)
    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) {
      return userService.getDailyCaloriesSumMap(userId, startDate, endDate);
    }

    /**
     * Get the total duration between the given start date and end date.
     * @param userId userId.
```

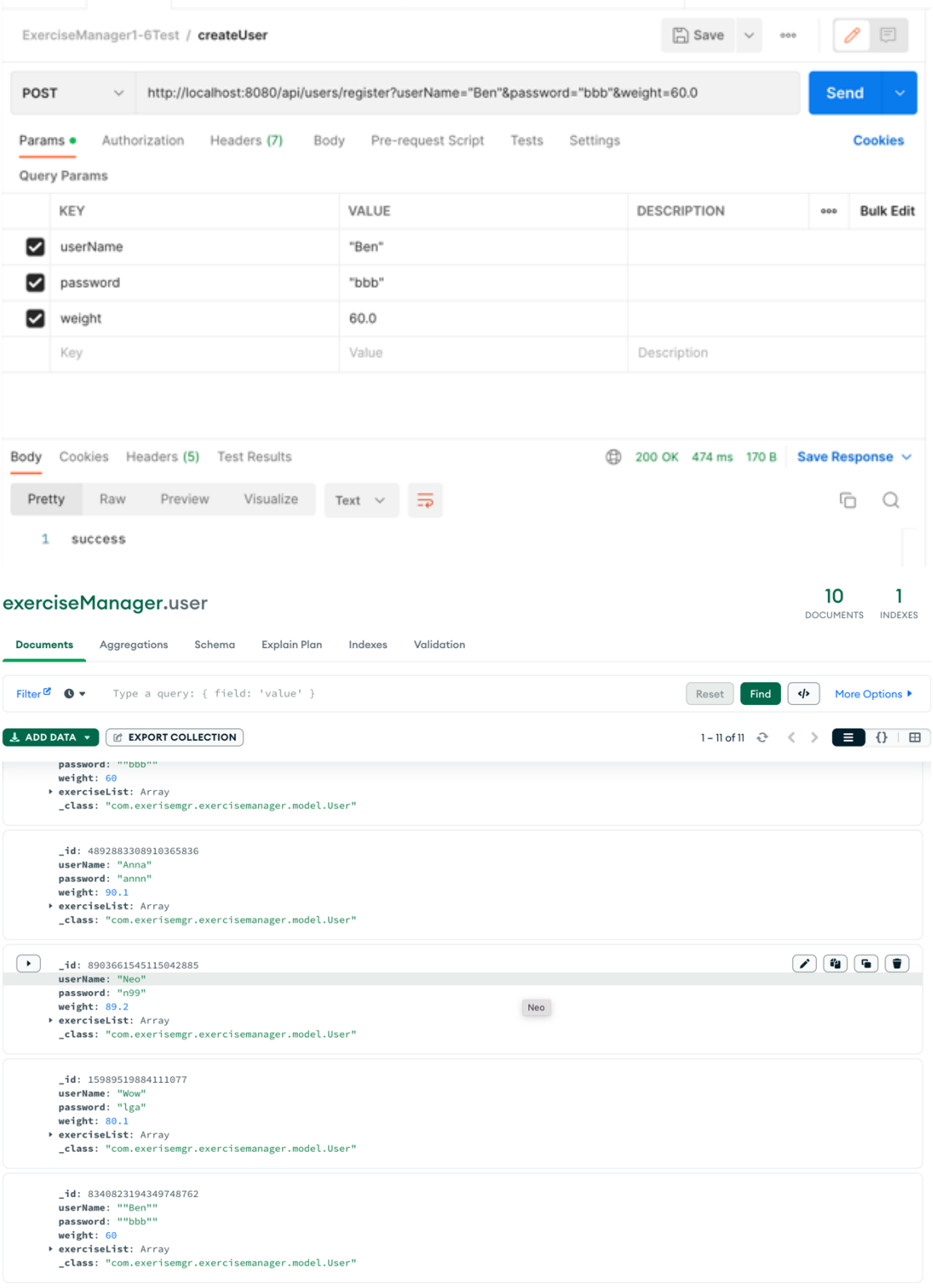
```
     * @param startDate the start date (Date).
     * @param endDate the end date (Date).
     * @return the total Duration between start date and end date.
     */
    @RequestMapping(value = "/{userId}/durationTotalBetweenDates", method = RequestMethod.GET)
    public Double getDurationTotalBetweenDates(@PathVariable Long userId, @RequestParam("startDate")
    @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
        @RequestParam("endDate")
        @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate) {
      return userService.getDurationTotalBetweenDates(userId, startDate, endDate);
    }

    /**
     * Get the total calories between the given start date and end date.
     * @param userId userId.
     * @param startDate the start date (Date).
     * @param endDate the end date (Date).
     * @return the total calories between start date and end date.
     */
    @RequestMapping(value = "/{userId}/caloriesTotalBetweenDates", method = RequestMethod.GET)
    public Double getCaloriesTotalBetweenDates(@PathVariable Long userId, @RequestParam("startDate")
    @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
        @RequestParam("endDate")
        @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate) {
      return userService.getCaloriesTotalBetweenDates(userId, startDate, endDate);
    }

    /**
     * Get the smallest duration between the given start date and end date.
     * @param userId userId.
     * @param startDate the start date (Date).
     * @param endDate the end date (Date).
     * @return the smallest duration between start date and end date.
     */
    @RequestMapping(value = "/{userId}/smallestDurationBetweenDates", method = RequestMethod.GET)
    public Double getSmallestDurationBetweenDates(@PathVariable Long userId,
        @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
        @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate) {
      return userService.getSmallestDurationBetweenDates(userId, startDate, endDate);
    }

    /**
     * Get the smallest calories between the given start date and end date.
     * @param userId userId.
     * @param startDate the start date (Date).
     * @param endDate the end date (Date).
     * @return the smallest calories between start date and end date.
     */
    @RequestMapping(value = "/{userId}/smallestCaloriesBetweenDates", method = RequestMethod.GET)
    public Double getSmallestCaloriesBetweenDates(@PathVariable Long userId,
        @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate,
        @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate) {
      return userService.getSmallestCaloriesBetweenDates(userId, startDate, endDate);
    }

    /**
     * Update weight for the user.
     * @param userId userId.
     * @param weight the weight the user wants to change to.
     */
    @RequestMapping(value = "/{userId}/weight", method = RequestMethod.PUT)
    public ResponseEntity<String> updateWeight(@PathVariable Long userId,
        @RequestParam Double weight) {
      userService.updateWeight(userId, weight);
      return ResponseEntity.ok("Weight updated successfully");
    }

    /**
     * Update password for the user.
     * @param userId userId.
```

```
     * @param password the password the user want to change to.
     */
    @RequestMapping(value = "/{userId}/{password}", method = RequestMethod.PUT)
    public ResponseEntity<String> updatePassword(@PathVariable Long userId,
        @PathVariable String password) {
      userService.updatePassword(userId, password);
      return ResponseEntity.ok("Password updated successfully");
    }

    /**
     * Update duration goal for the user.
     * @param userId userId.
     * @param startDate the starting date of the duration goal.
     * @param endDate the ending date of the duration goal.
     * @param durationGoal the duration goal the user want to change to.
     */
    @RequestMapping(value = "/{userId}/durationGoal", method = RequestMethod.PUT)
    public ResponseEntity<String> 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) {
      userService.updateDurationGoal(userId, startDate, endDate, durationGoal);
      return ResponseEntity.ok("Duration goal updated successfully");
    }

    /**
     * Update calories goal for the user.
     * @param userId userId.
     * @param startDate the starting date of the calories goal.
     * @param endDate the ending date of the calories goal.
     * @param caloriesGoal the calories goal the user want to change to.
     */
    @RequestMapping(value = "/{userId}/caloriesGoal", method = RequestMethod.PUT)
    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) {
      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   }
```

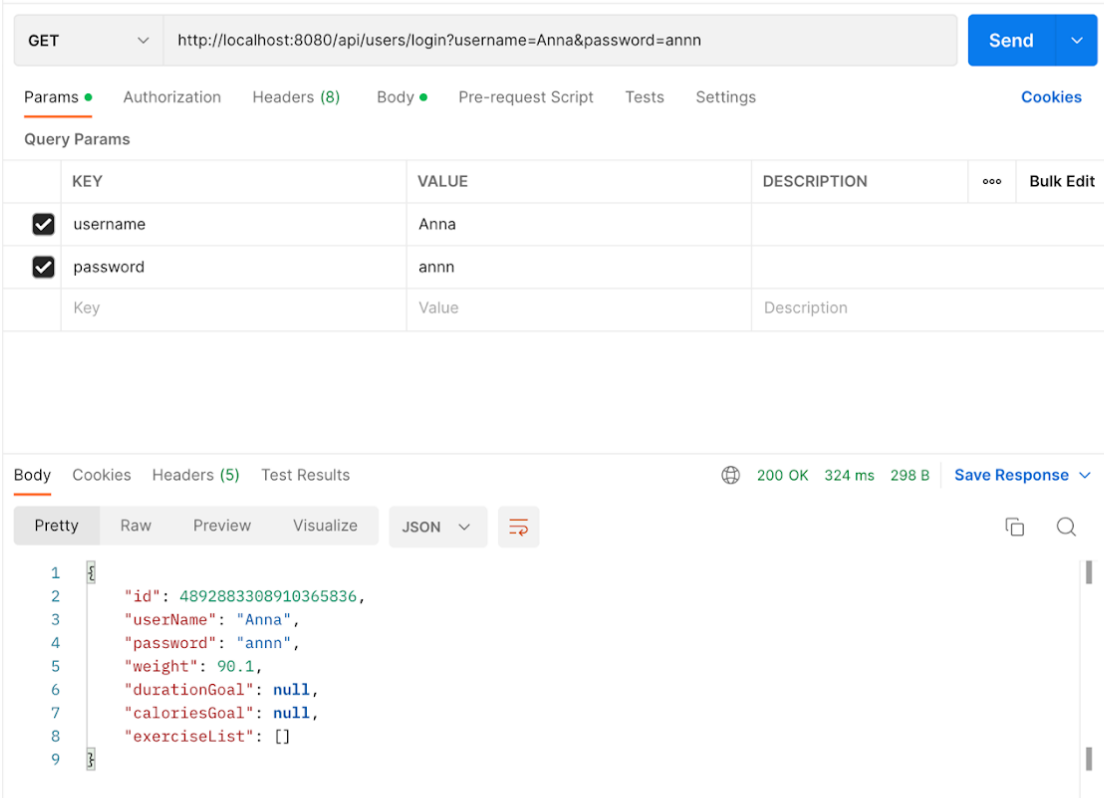
- **Controller Test Result**
  We've used Postman to create and send API requests to our backend system. Please see below our test results in detail.

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:



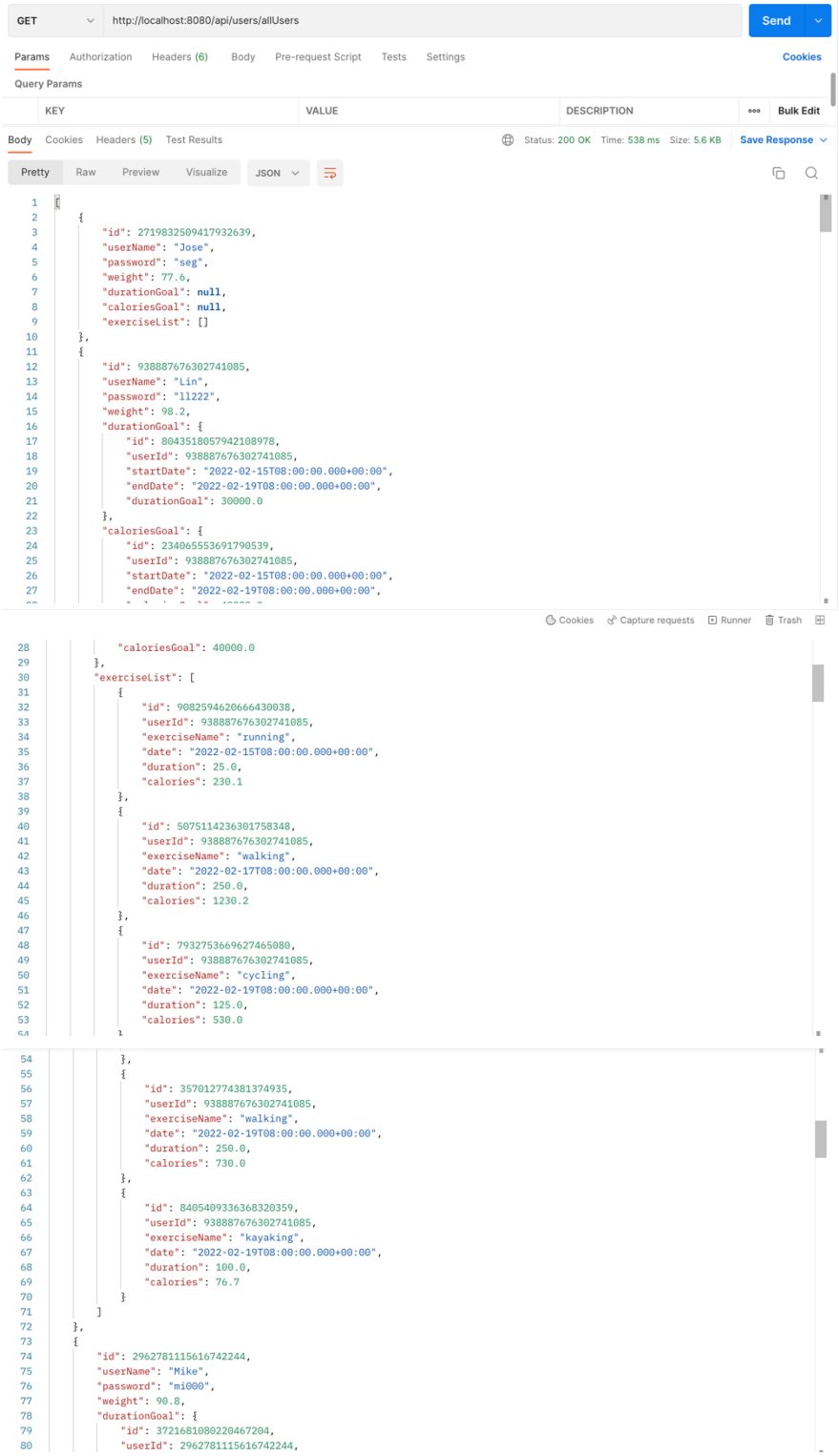
Test Result: Passed

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 userName Anna returned
Actual Test Result:



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
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 538 ms
Size: 5.6 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 2719832509417932639,
"userName": "Jose",
"password": "seg",
"weight": 77.6,
"durationGoal": null,
"caloriesGoal": null,
"exerciseList": []
},
{
"id": 938887676302741085,
"userName": "Lin",
"password": "ll222",
"weight": 98.2,
"durationGoal": {
"id": 8043518057942108978,
"userId": 938887676302741085,
"startDate": "2022-02-15T08:00:00.000+00:00",
"endDate": "2022-02-19T08:00:00.000+00:00",
"durationGoal": 30000.0
},
"caloriesGoal": {
"id": 234065553691790539,
"userId": 938887676302741085,
"startDate": "2022-02-15T08:00:00.000+00:00",
"endDate": "2022-02-19T08:00:00.000+00:00",
Cookies
Capture requests
Runner
Trash
"caloriesGoal": 40000.0
},
"exerciseList": [
{
"id": 9082594620666430038,
"userId": 938887676302741085,
"exerciseName": "running",
"date": "2022-02-15T08:00:00.000+00:00",
"duration": 25.0,
"calories": 230.1
},
{
"id": 5075114236301758348,
"userId": 938887676302741085,
"exerciseName": "walking",
"date": "2022-02-17T08:00:00.000+00:00",
"duration": 250.0,
"calories": 1230.2
},
{
"id": 7932753669627465080,
"userId": 938887676302741085,
"exerciseName": "cycling",
"date": "2022-02-19T08:00:00.000+00:00",
"duration": 125.0,
"calories": 530.0
},
{
"id": 3570127743813749 35,
"userId": 938887676302741085,
"exerciseName": "walking",
"date": "2022-02-19T08:00:00.000+00:00",
"duration": 250.0,
"calories": 730.0
},
{
"id": 8405409336368320359,
"userId": 938887676302741085,
"exerciseName": "kayaking",
"date": "2022-02-19T08:00:00.000+00:00",
"duration": 100.0,
"calories": 76.7
}
]
},
{
"id": 2962781115616742244,
"userName": "Mike",
"password": "mi000",
"weight": 90.8,
"durationGoal": {
"id": 3721681080220467204,
"userId": 2962781115616742244,

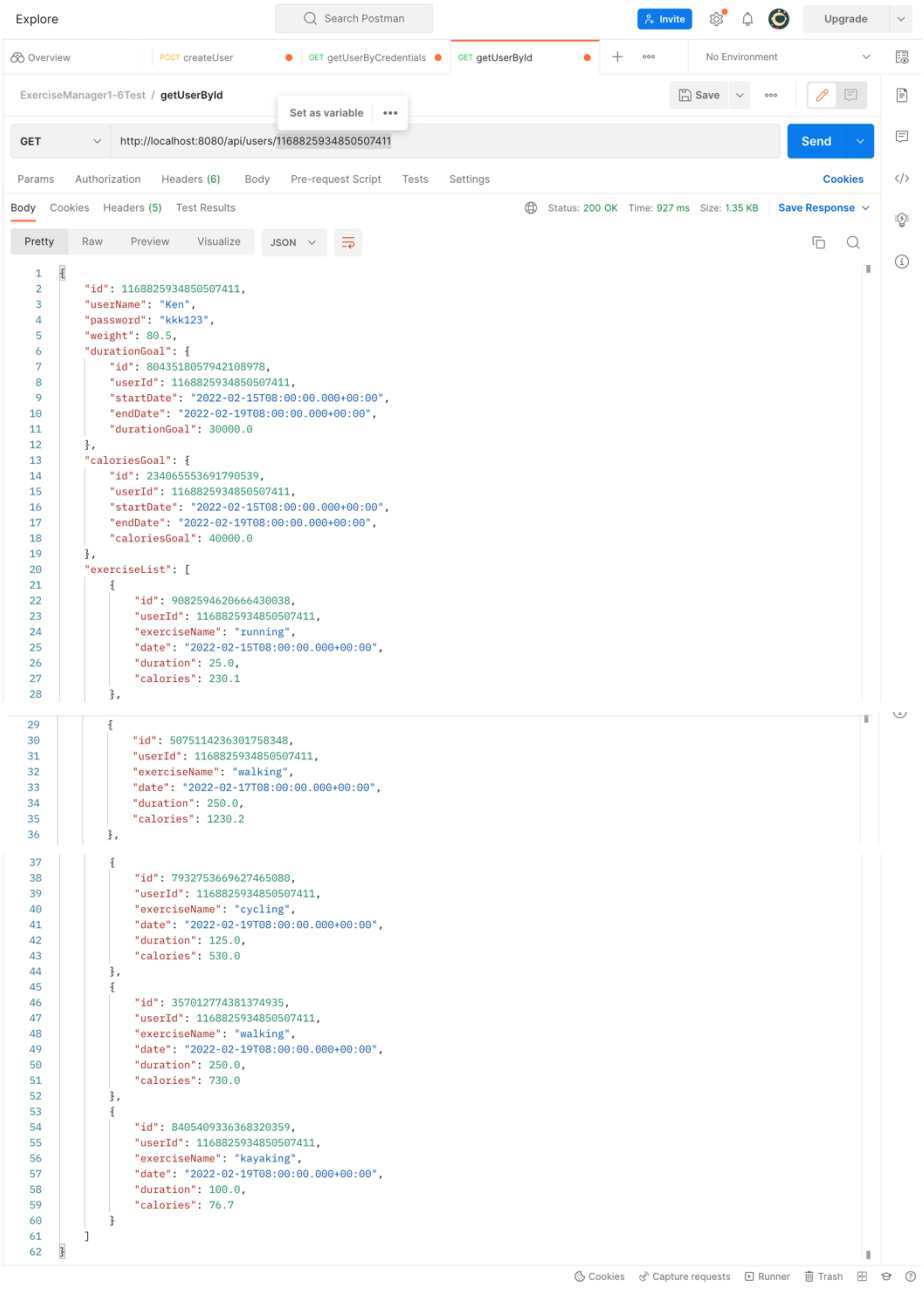
```
            "startDate": "2022-02-15T08:00:00.000+00:00",
            "endDate": "2022-02-19T08:00:00.000+00:00",
            "durationGoal": 30000.0
        },
        "caloriesGoal": {
            "id": 1694549481149057157,
            "userId": 2962781115616742244,
            "startDate": "2022-02-15T08:00:00.000+00:00",
            "endDate": "2022-02-19T08:00:00.000+00:00",
            "caloriesGoal": 40000.0
        },
        "exerciseList": [
            {
                "id": 6392085278626758828,
                "userId": 2962781115616742244,
                "exerciseName": "running",
                "date": "2022-02-15T08:00:00.000+00:00",
                "duration": 25.0,
                "calories": 230.1
            },
            {
                "id": 7546541538780204098,
                "userId": 2962781115616742244,
                "exerciseName": "walking",
                "date": "2022-02-17T08:00:00.000+00:00",
                "duration": 250.0,
                "calories": 1230.2
            },
            {
                "id": 513813054268329336,
                "userId": 2962781115616742244,
                "exerciseName": "cycling",
                "date": "2022-02-19T08:00:00.000+00:00",
                "duration": 125.0,
                "calories": 530.0
            },
            {
                "id": 58595572272184870 64,
                "userId": 2962781115616742244,
                "exerciseName": "walking",
                "date": "2022-02-19T08:00:00.000+00:00",
                "duration": 250.0,
                "calories": 730.0
            },
            {
                "id": 630378873627559366,
                "userId": 2962781115616742244,
                "exerciseName": "kayaking",
                "date": "2022-02-19T08:00:00.000+00:00",
                "duration": 100.0,
                "calories": 76.7
            }
        ]
    },
    {
        "id": 8757637119955584962,
        "userName": "Alice",
        "password": "alice123",
        "weight": 70.5,
        "durationGoal": {
            "id": 3900944275542658672,
            "userId": 8757637119955584962,
            "startDate": "2022-02-15T08:00:00.000+00:00",
            "endDate": "2022-02-19T08:00:00.000+00:00",
            "durationGoal": 30000.0
        },
        "caloriesGoal": {
            "id": 1923117620111689881,
            "userId": 8757637119955584962,
            "startDate": "2022-02-15T08:00:00.000+00:00",
            "endDate": "2022-02-19T08:00:00.000+00:00",
            "caloriesGoal": 40000.0
        },
        "exerciseList": [
```

```
        {
            "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": 581247988122216838,
            "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": 234065553691790539,
        "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": 9082594620666430038,
            "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
        }
    ]
},
{
    "id": 7294158642769053171,
    "userName": "\"Bill\"",
    "password": "\"bbb\"",
    "weight": 60.0,
    "durationGoal": null,
    "caloriesGoal": null,
    "exerciseList": []
},
{
    "id": 4892883308910365836,
    "userName": "Anna",
    "password": "annn",
    "weight": 90.1,
    "durationGoal": null,
    "caloriesGoal": null,
    "exerciseList": []
},
{
    "id": 8903661545115042885,
    "userName": "Neo",
    "password": "n99",
    "weight": 89.2,
    "durationGoal": null,
    "caloriesGoal": null,
    "exerciseList": []
},
{
    "id": 8340823194349748762,
    "userName": "\"Ben\"",
    "password": "\"bbb\"",
    "weight": 60.0,
    "durationGoal": null,
    "caloriesGoal": null,
    "exerciseList": []
}
]
```

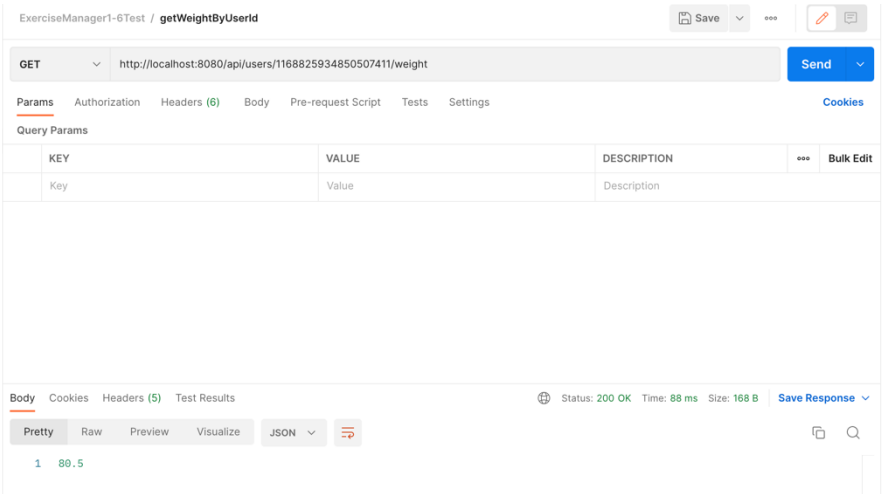

Test Result: Passed

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:



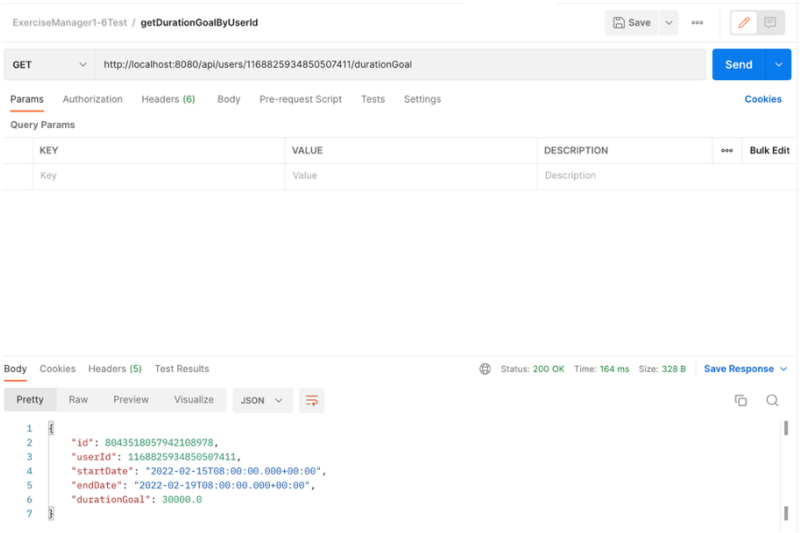
Test Result: Passed

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:



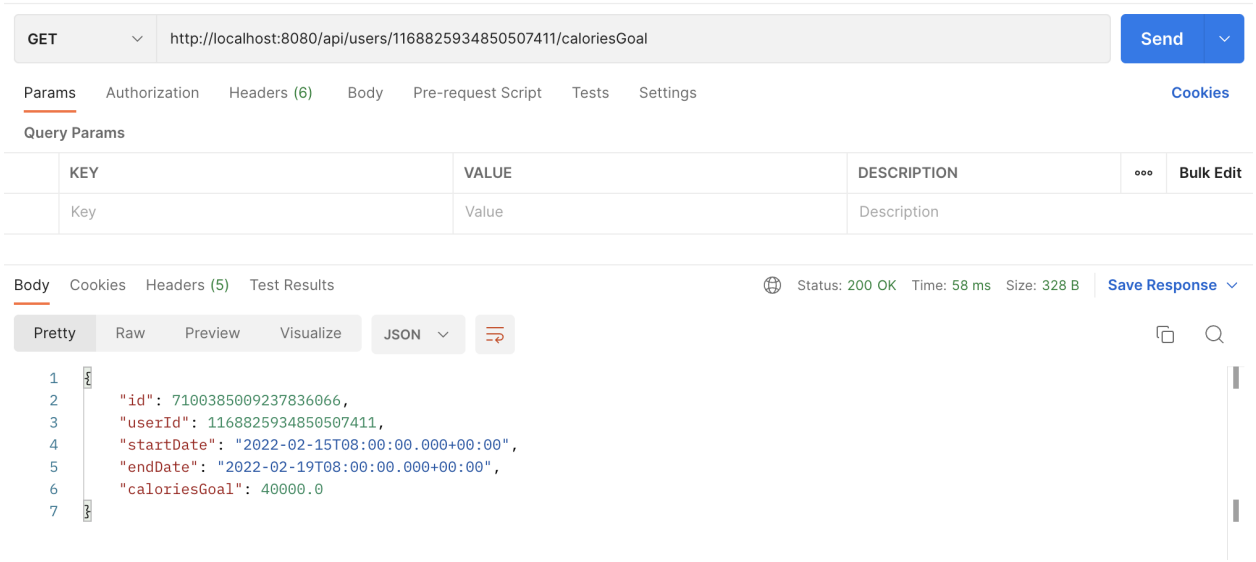
Test Result: Passed

Test #6
Test Method: public DurationGoal getDurationGoalByUserId(@PathVariable Long userId)
Operation: GET
Test URL: http://localhost:8080/api/users/116882593485050741l/durationGoal
Request Body: N/A
Expected Test Result: DurationGoal with userId 116882593485050741l returned
Actual Test Result:



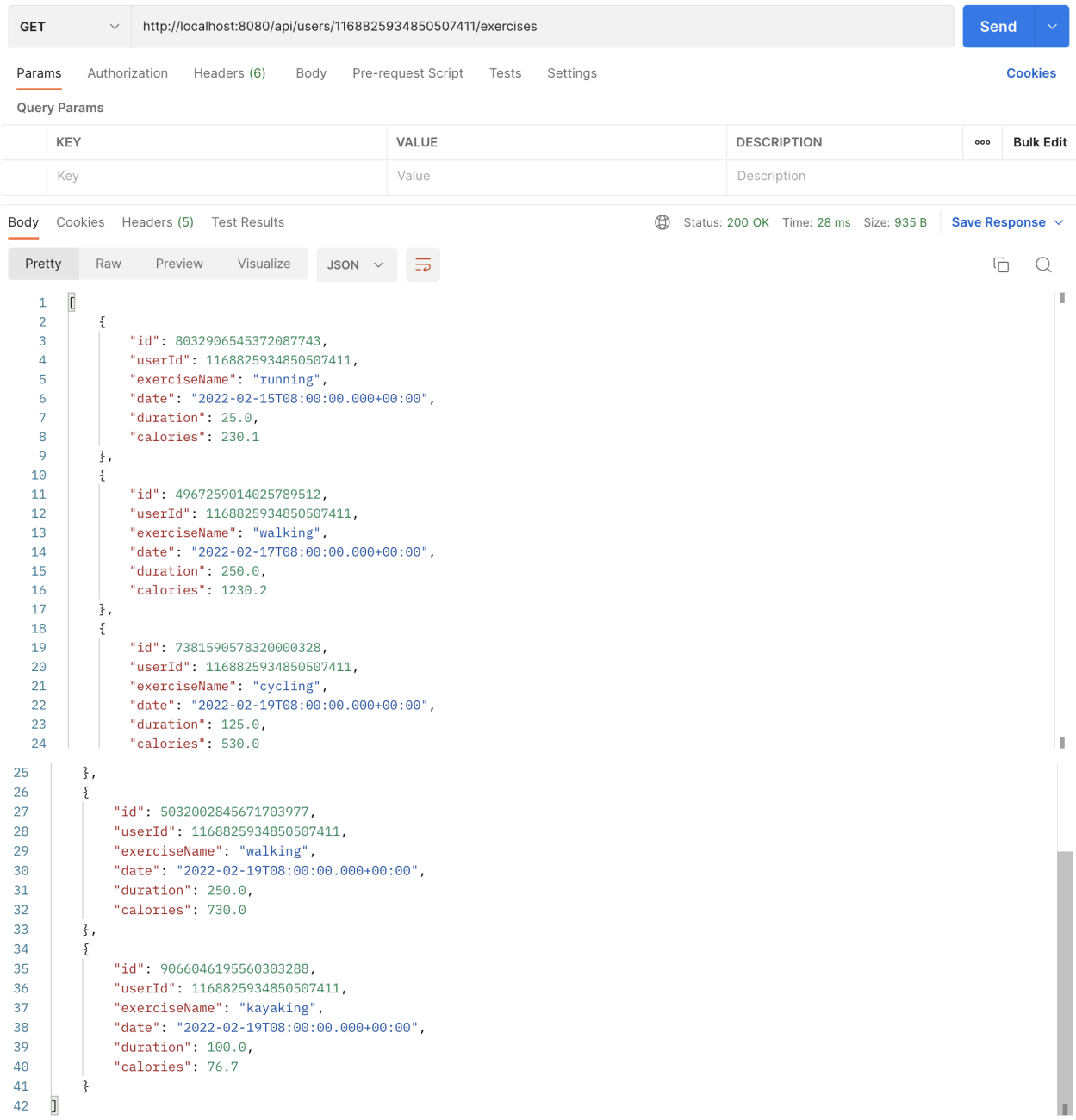
Test Result: Passed

Test #7
Test Method: public CaloriesGoal getCaloriesGoalByUserId(@PathVariable Long userId)
Operation: GET
Test URL: http://localhost:8080/api/users/116882593485050741l/caloriesGoal
Request Body: N/A
Expected Test Result: CaloriesGoal with userId 116882593485050741l returned
Actual Test Result:



Test Result: Passed

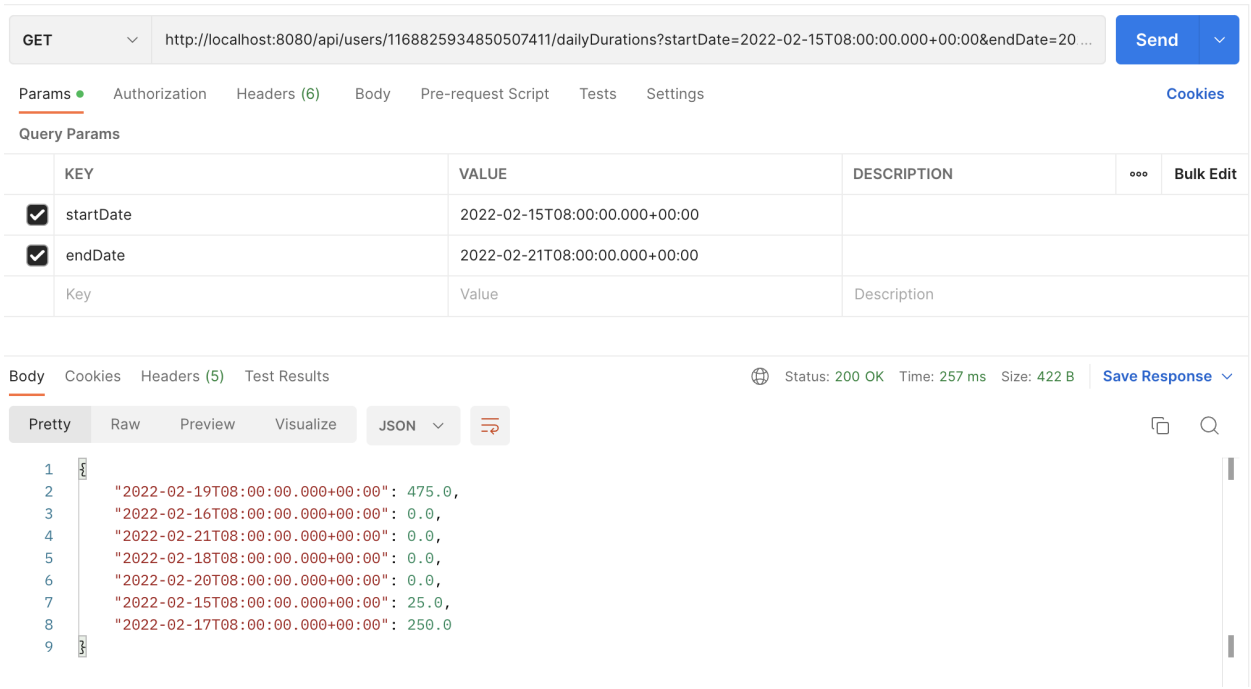
Test #8
Test Method: public List<Exercise> getAllExerciseByUserId(@PathVariable Long userId)
Operation: GET
Test URL: http://localhost:8080/api/users/1168825934850507411/exercises
Request Body: N/A
Expected Test Result: All exercises with userId 1168825934850507411 returned
Actual Test Result:



Test Result: Passed

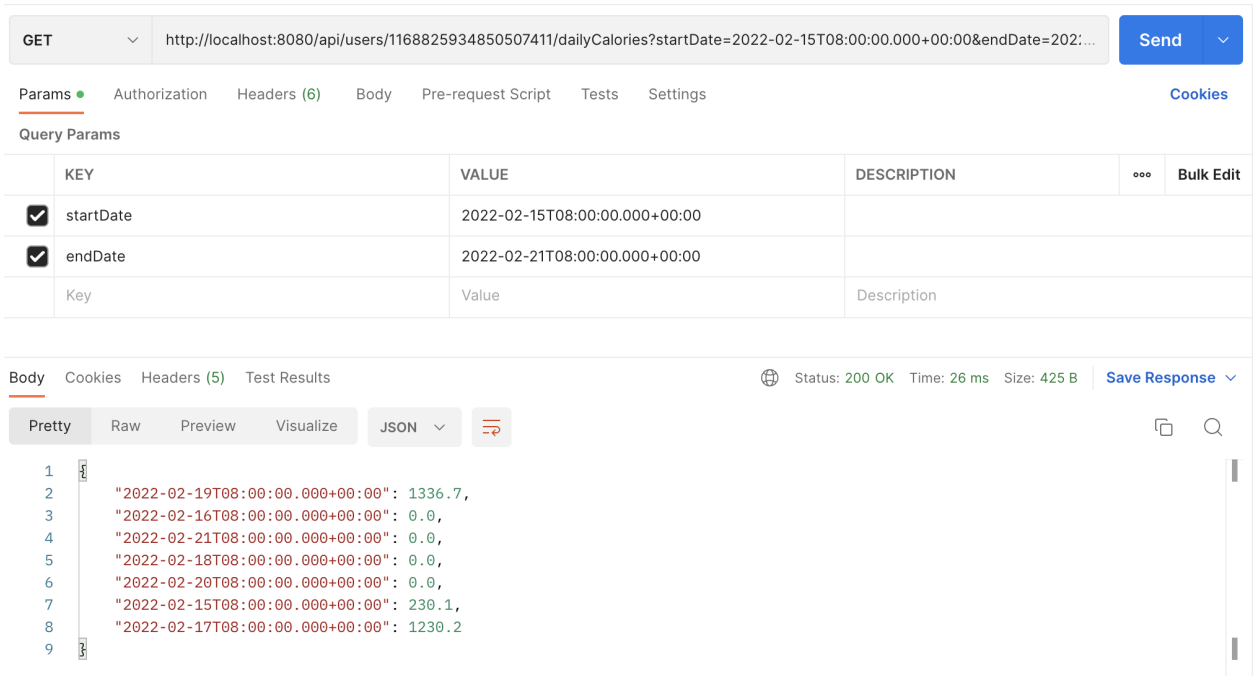
Test #9
Test Method: public Map<Date, Double> getDailyDurationSumMap(@PathVariable Long userId, @RequestParam("startDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date startDate, @RequestParam("endDate") @DateTimeFormat(pattern = "yyyy-MM-dd") Date endDate)
Operation: GET
Test URL:
http://localhost:8080/api/users/1168825934850507411/dailyDurations?startDate=2022-02-15T08:00:00.000+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:



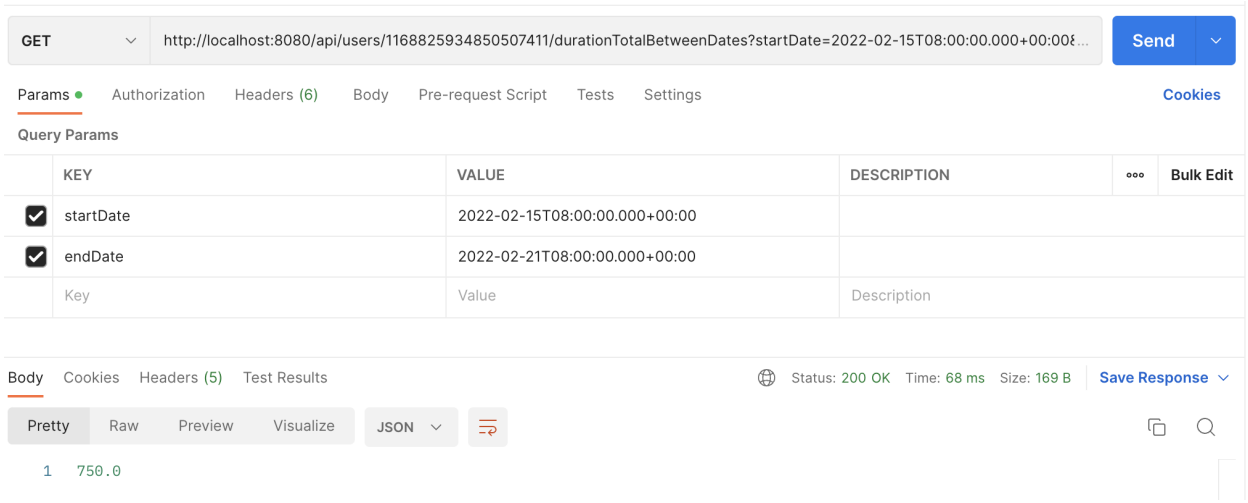
Test Result: Passed

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:



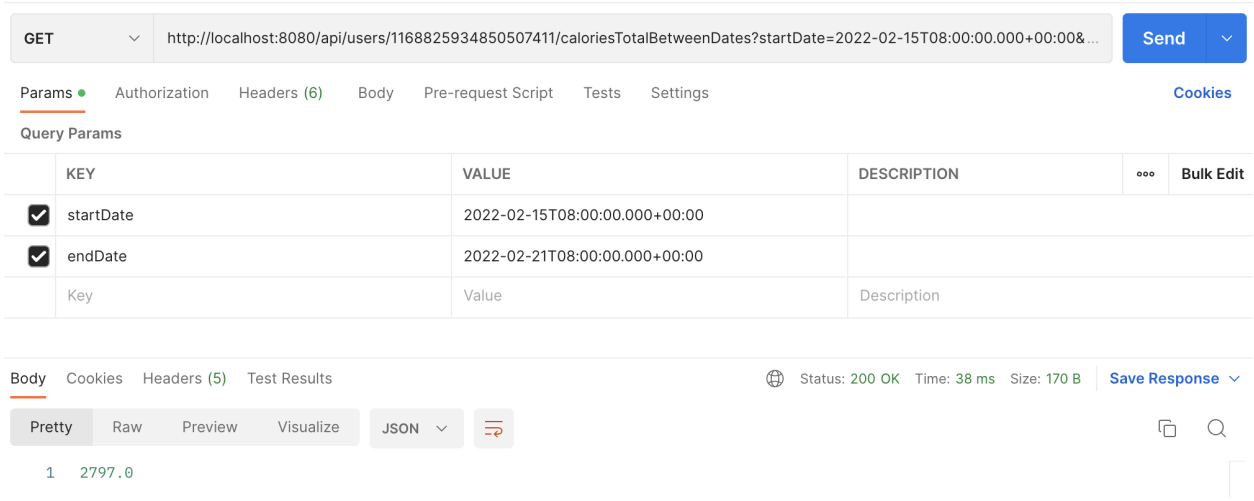
Test Result: Passed

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:



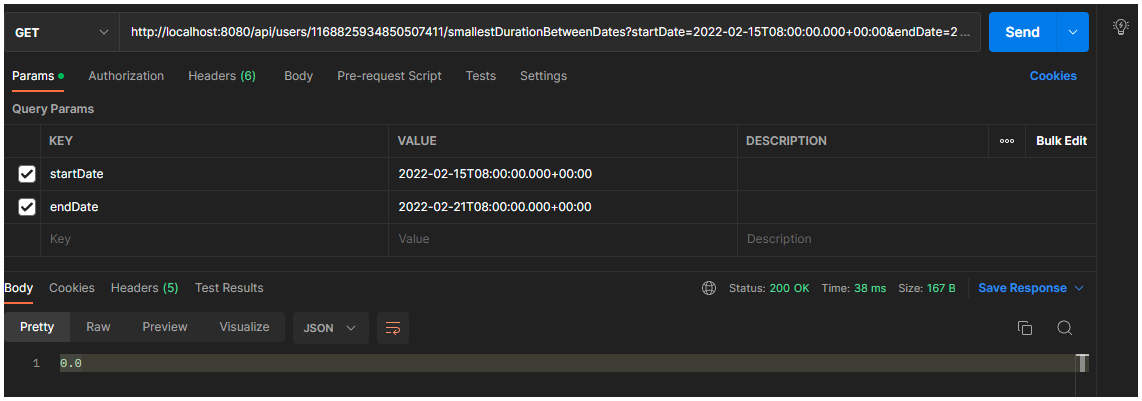
Test Result: Passed

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:



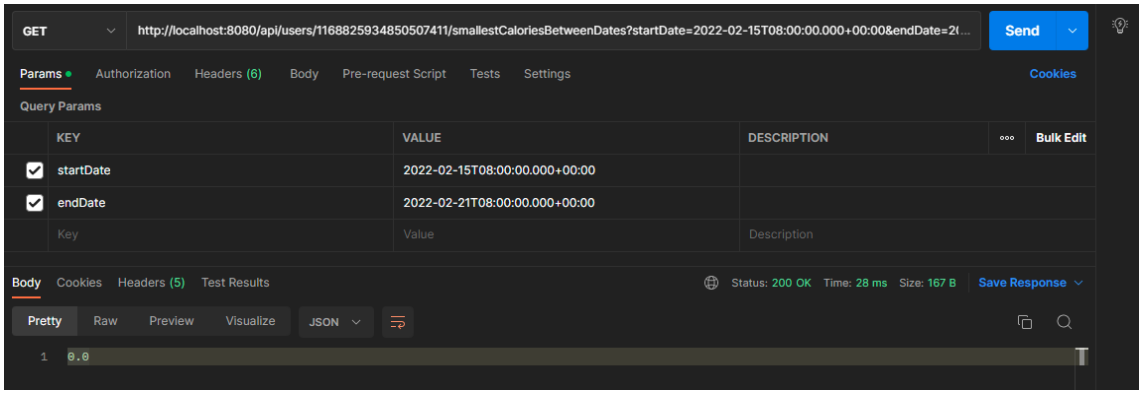
Test Result: Passed

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:



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

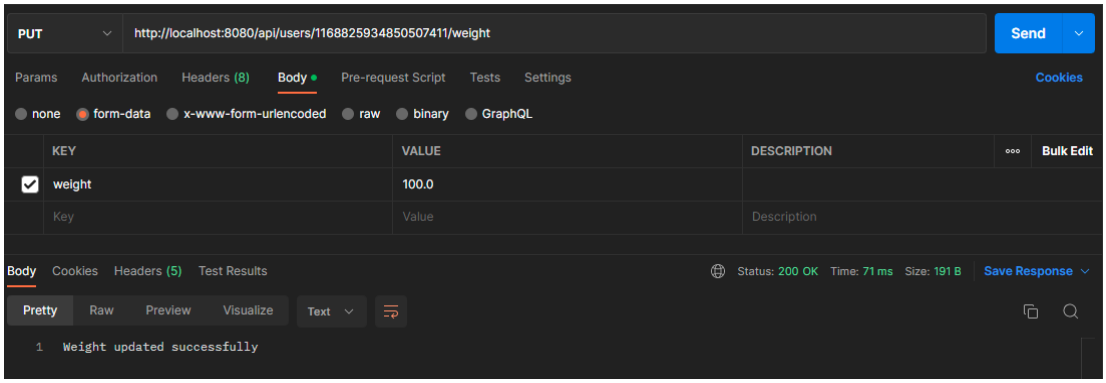
Test URL:

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

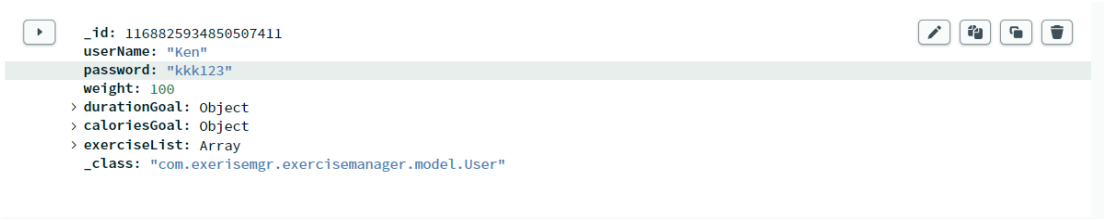
Request Body: 100.0 (weight)

Expected Test Result: update userId 1168825934850507411 weight to 100.0

Actual Test Result:



Database check:



Test Result: Passed

Test #16

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

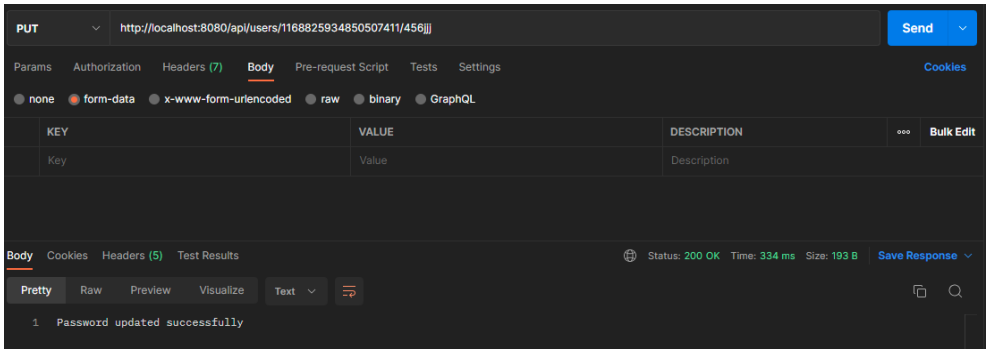
Operation: PUT

Test URL:

http://localhost:8080/api/users/1168825934850507411/456jjj

Request Body: N/A

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

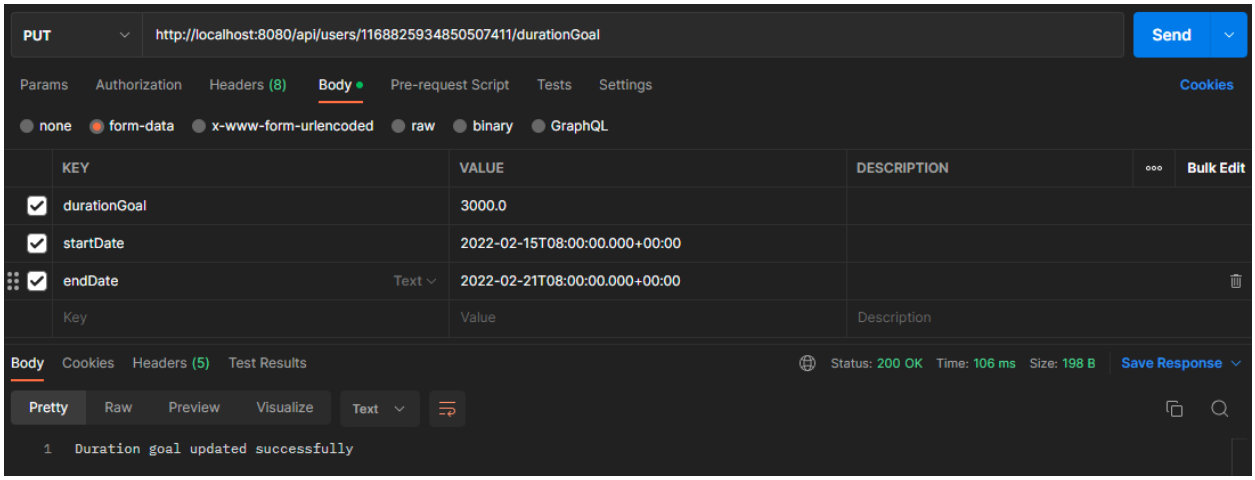


Database Check


```
_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("startDate") @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:



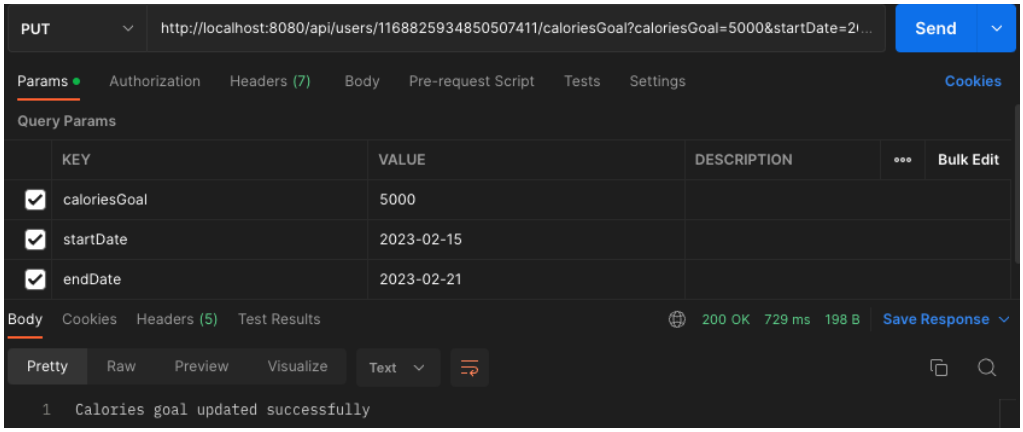
Database Check


```
_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("startDate") @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:



Database Check

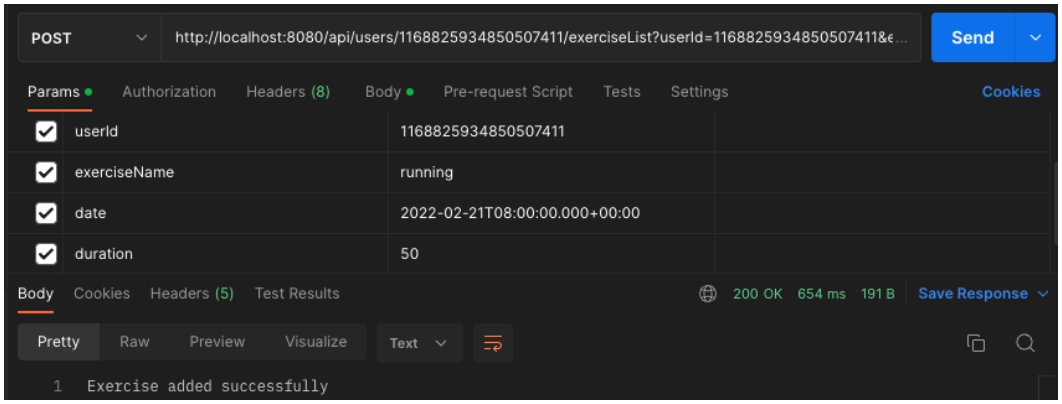

```
_id: 1168825934850507411
userName: "Ken"
password: "kkk123"
weight: 80.5
▸ durationGoal: Object
▾ caloriesGoal: Object
    _id: 2153866899858344386
    caloriesGoal: 5000
    userId: 3250401998014466736
    startDate: 2023-02-15T00:00:00.000+00:00
    endDate: 2023-02-21T00:00:00.000+00:00
▸ exerciseList: 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:



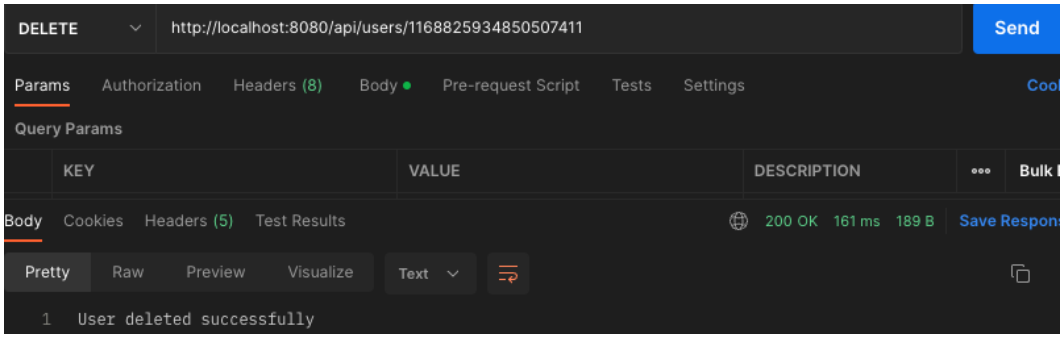
Database Check


```
_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"
```

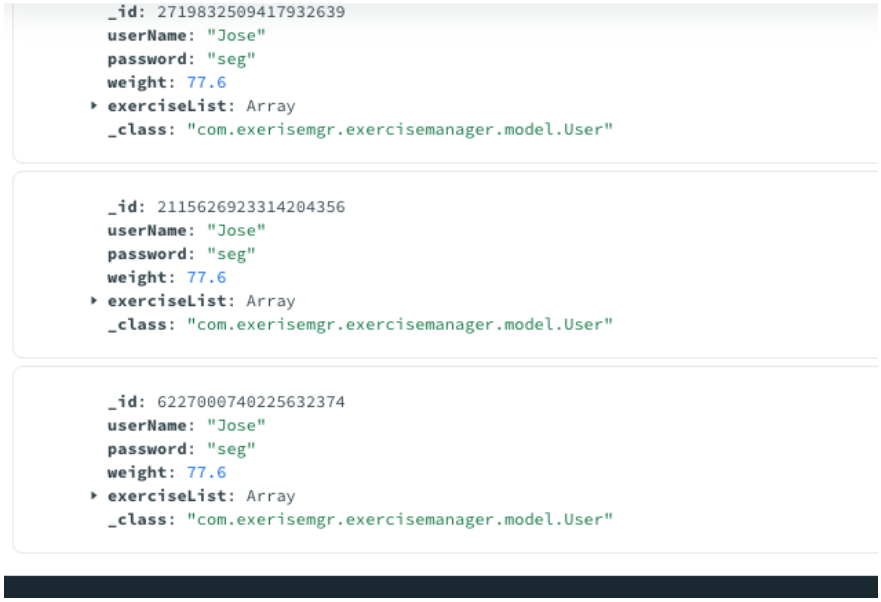

Test Result: Passed

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:



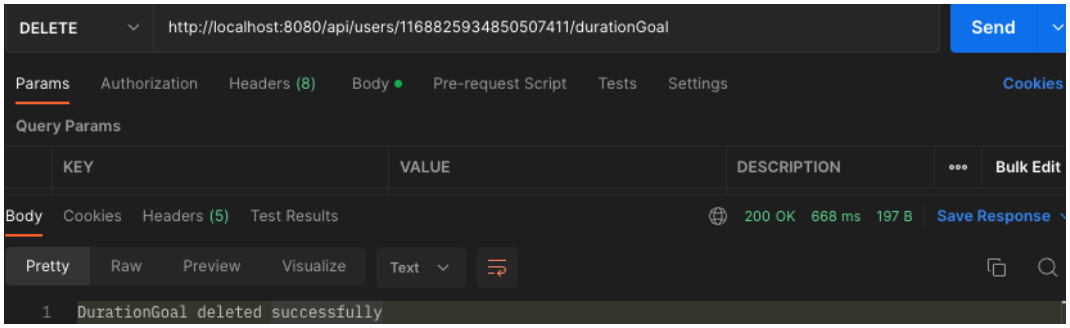
Database Check



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:



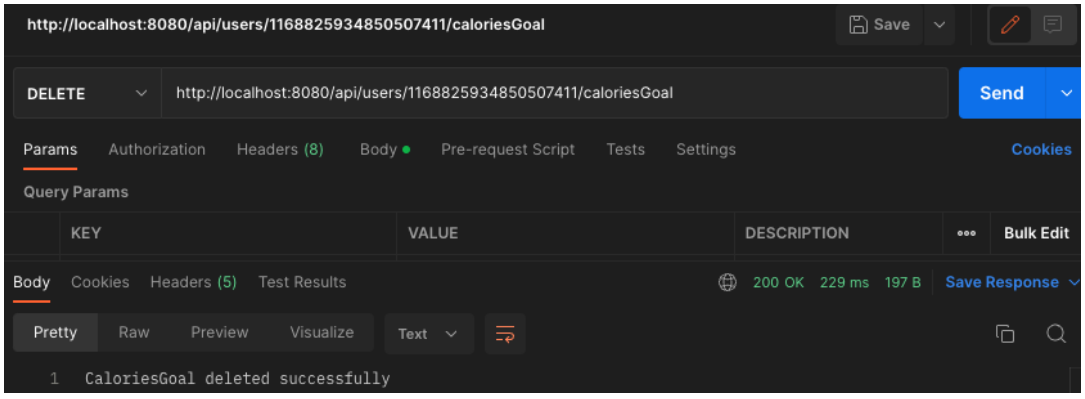
Database Check


```
_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:



Database Check

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

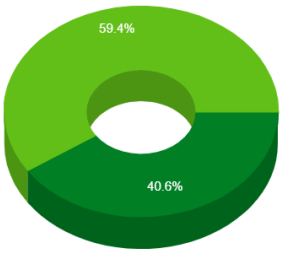
Number of packages: 6

Number of external packages: 15

Number of external classes: 60

Number of problematic classes: 1

Number of highly problematic classes: 0

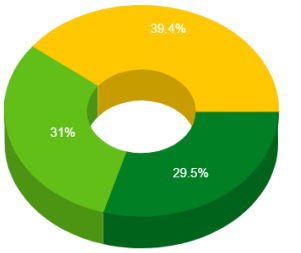


Coupling

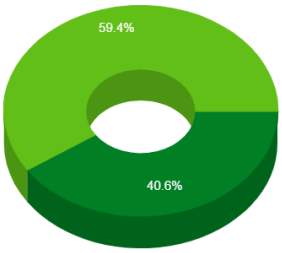
- Very High
- High
- Medium-high
- Low-medium
- Low

Distribution of Quality Attributes

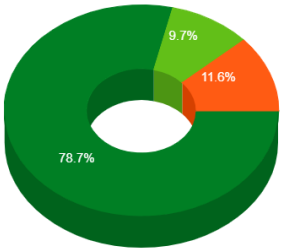
Complexity, Coupling, Cohesion, and Size



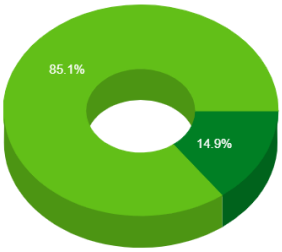
Complexity



Coupling



Lack of Cohesion



Size

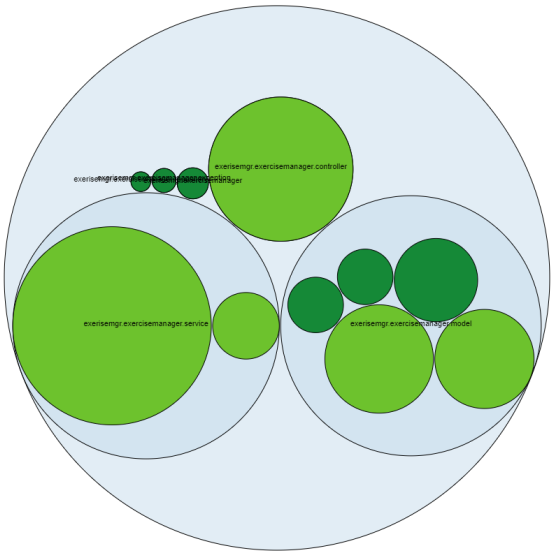
List of all classes (#11)

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

Metric Selection

Size

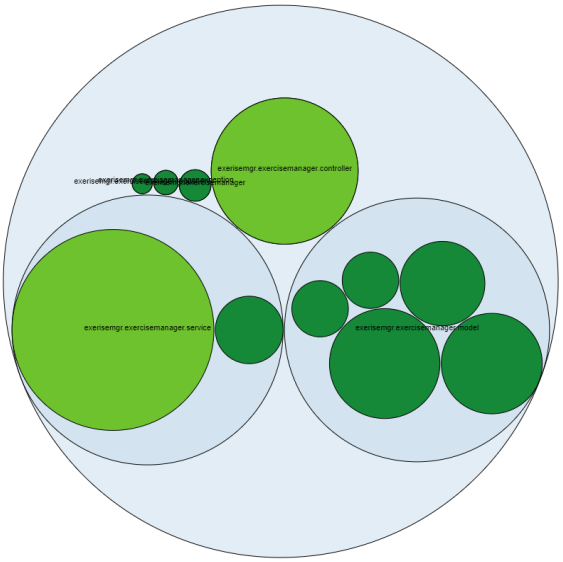
click to zoom, for detailed metrics, press ctrl or ⌘ while hovering



Metric Selection

Coupling

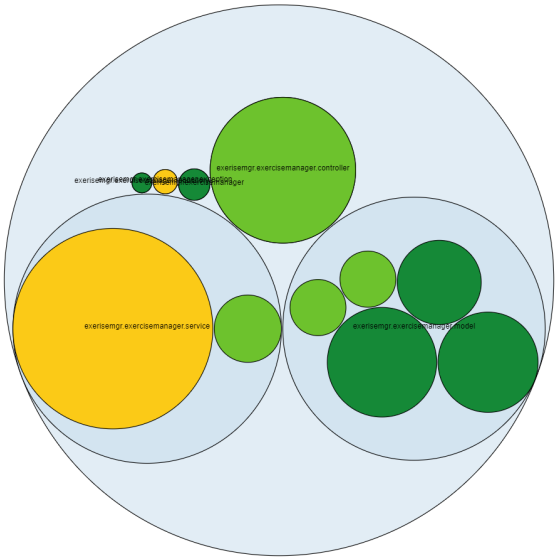
click to zoom, for detailed metrics, press ctrl or ⌘ while hovering



Metric Selection

Complexity

click to zoom, for detailed metrics, press ctrl or ⌘ while hovering



Metric Selection

Lack of Cohesion

click to zoom, for detailed metrics, press ctrl or ⌘ while hovering