Assignment T2: Revised Project Proposal

Team JVM

Part 0: Team Basics

No changes.

Our team met with IA mentor Rennah Wang in class on 10/29 and 11/5

Part 1: Project Overview

We intend to develop a website that offers easy health tracking/management service to users. There are mainly four components of our service: diet tracking, exercise tracking, weight tracking, and health advice based on the records mentioned above.

We aim at users who would like to keep a record of their health-related metrics and seek advice on staying fit/building muscles. Users need to register and sign in to use our services. There will be a time out which logs the time of inactivity after a user signs in so that we can sign the user out if he/she/it has been inactive for a certain period of time (we will do 24 hours since there is no sensitive data in our service).

The final product will include a front-end website with GUI which spares users from interacting with our system using the command line.

We plan to host our server and store our data in the database using services provided by AWS so that the access to services and data is more flexible. We also plan to build a predictive model that predicts weights based on historical input data.

The project will need two major data sources, calories burn for common workouts like pushup and nutrition breakdown for food. We did not find any useful open-source dataset regarding workout, so collecting data manually is the only option. For nutrition data, we intend to take advantage of a publicly available API, FoodData Central API, which provides calories and nutrition for a good amount of food.

The API and Database designs are attached in the appendix.

Part 2: User Stories

1. As a user, I want to be able to log in the app so I can access my history from different devices if I am logged in.
   My conditions of satisfactions are:
   a. No other user has the same username as I do.
   b. My account is protected by a password that only I know
   c. The website remembers that I have logged into the device for some period. During this period, I do not need to log in again if I visit the website again.
   d. If I do not visit website for a long time (e.g. a day), the website should automatically log me out, so that other users that access the same device will not be able to use my account.
2. As a user, I want to record my daily diet in the app so that I can see my diet history and get some statistics about my past diet records.
   My conditions of satisfaction are:
   a. The app supports at least 5 types of food for me to choose from for each record
   b. I can modify or delete the created diet record in case I made a mistake
   c. The records are available on any device as long as I am logged in to the account.
   d. The app can provide me with accurate statistics on the food I eat
   e. I can see my history in a different time period (1 week, 1 month, 1 year)
3. As a user, I want to record my weight and on a daily basis so that I can track the changes in my weight throughout any period.
   My conditions of satisfaction are:
   a. I can record my weight in either pound or kilograms.
   b. I can view my weight in either pound or kilograms no matter which unit I used when entering it.
   c. I can modify or delete the created weight record in case I made a mistake
   d. The records are available on any device as long as I am logged in to the account.
4. As a user, I want to record my workout history so that I can keep track of the time and intensity of my past workouts.
   My conditions of satisfaction are:
   a. The app can recognize different types of workout from the records I input
   b. The app can provide me with statistics of my workout based on workout type and duration.
   c. I can modify or delete the created workout record in case I made a mistake
   d. I should receive a warning when I am trying to delete a record or modify sensitive attributes of the record like date
5. As a user, I want to see a visualized report of my past records so that I can understand the statistics better
   My conditions of satisfaction are:
   a. I should be able to select different durations to visualize (1 week, 1 month, 3 months, 6 months, 1 year)

    b. The display of data should be easy to understand. e.g. I can see the change of my weights in a curve chart.
    c. The overall display should be visually appealing.
    d. The statistics should always be up-to-date and reflect the most recent input
6. As an intensive user, I want to get some suggestions from the app about my life habit so that I can become healthier by following them.
   My conditions of satisfaction are:
    a. The suggestions are based specifically on the different records I input to the application, instead of just general health advice
    b. The suggestions should be easy to understand and follow
    c. The suggestions should be beneficial to my health instead of damaging it
    d. I can obtain suggestions based on my different personal needs (reduce fat, increase fitness, build shape)

Part 3: Acceptance Test

Test cases w.r.t. user stories and conditions of satisfaction

1. Users.
    a. Register with a not existing username. The test passes if the register succeeds and information is logged in the database, and otherwise fails.
    b. Register with an existing username. The test passes if the register does not go through due to a non-unique username, and otherwise fails.
    c. Log in using a correct combination of username and password. The test passes if the log in succeeds and a token is stored in the database and otherwise fails.
    d. Log in using an incorrect combination of username and password. The test passes if the log in does not go through and otherwise fails.
    e. Log out the user after logging in. The test passes if user correctly logs out of the system
    f. Log in the user, then visit the website again after a short period of time (e,g, three hours). The test passes if the system remembers the user as logged in, and does not require user to log in again..
    g. Log in the user, then do not access the website for a designated length of time (e,g, one day). The test passes if the system automatically logs the user out.
2. Diet record.
    a. Choose 1, 2, 3, 4, and 5 types of foods and log the record. The test passes if all the information is correctly recorded in the database, and otherwise fails.
    b. Modify the record of a day by changing the amount of food, remove a type of food, add a type of food, and delete the whole record. The test passes if all the information is correctly recorded in the database, and otherwise fails.
    c. After entering the record of a whole month of a user, retrieve the record over 1 day, 1 week, and 1 month. The test passes if the curve matches the record in the database, and otherwise fails.
3. Weight record.
    a. Enter weight in the pound and then kilogram. The test passes if all the information is correctly recorded in the database, and otherwise fails.
    b. Retrieve a record from the database and display in both pound and kilogram. The test passes if all the two displayed values is equal after unit conversion and otherwise fails.
    c. Modify the record of a day by changing the value of the weight and delete the whole record. The test passes if all the information is correctly recorded in the database, and otherwise fails.
4. Exercise record.
    a. Exhaustively enter each type of exercise and enter the record. The test passes if all the information is correctly recorded in the database, and otherwise fails.

    b. Modify the record of a day by changing the duration of exercise, type of exercise, add/remove an exercise, and delete the whole record. The test passes if all the information is correctly recorded in the database, and otherwise fails.
    c. Modify each of the entries classified as sensitive. The test passes if a warning is given, and otherwise fails.
5. Visualization.
    a. Display the data up-to-date. The test passes if the displayed record includes all the up-to-date data, and otherwise fails.
6. Health suggestions.
    a. Manually create accounts with a single of the following characteristics (a) there is few recorded exercises, (2) the diet is extremely unbalanced, (3) weight goes up/down quickly while keeping the rest normal. The test passes if the given advice can help address the problem based on human judgments and otherwise fails.
7. General
    a. Access the record of diet, weight and exercise from chrome, Firefox, and Safari on Windows, macOS, and Linux. The test passes if records can be displayed correctly, and otherwise fails.

Part 4: Tech Stack

IDE: IntelliJ

Build Tool: Maven

Style Checker: CheckStyle-IDEA

Unit Testing: Mockito

Testing Coverage Tracking: Coverage(built-in)

Bug Finder: SpotBug

Frontend: React

Backend: Java, Spring,

Authentication: OAuth

Database: AWS MySQL

Appendix A: API design

API	URL	Request Body	Response

Login	GET /user/login	{
		"username": String,
		"password": String
		}
			HTTP Status Code: 200 Ok
			
			Redirect to application
			
			{
			"code": 200,
			"message": Successfully login.
			}
			
			HTTP Status Code: 400 Bad Request
			
			{
			"errorCode": 1001,
			"errorMessage": Wrong username or password.
			}

Register	POST /user/register	{
		"userName": String,
		"password": String
		}
		
		Verify the username is not registered
			HTTP Status Code: 200 Ok
			
			Redirect to application html
			
			{
			"code": 200,
			"message": Successfully Registered.
			}
			
			HTTP Status Code: 400 Bad Request
			
			Check if the username is registered upon submission
			
			{
			"code": 1002,
			"message": Username Already Exists.
			}

Logout	GET /user/logout		HTTP Status Code: 200 Ok Redirect to login html
Get all diet tags	GET /diet/tags		{ "code": 200,

			"message": Success, "types": [{"typeID": x, "dietType": y}, ...] }

Create a new diet record	POST /diet/records	{
		"dietType": String,
		"dietId": Integer,
		"weight": Double,
		"unit": gram/pound,
		"nutrientList": [
		{
		"energy": Double,
		"protein": Double,
		"carbs": Double,
		"fat": Double
		}
		]
		}
			HTTP Status Code: 200 Ok
			{
			"code": 200,
			"message": Successfully record.
			}
			
			HTTP Status Code: 401 Unauthorized

Get diet records history	GET /diet/records?period=(all/week/month/year)&length=Integer	{
		"userId": Integer,
		}
			HTTP Status Code: 200 Ok
			{
			"code": 200,
			"message": Successfully retrieve history.
			"recordsList": [
			{
			"recordId": Integer,
			"dietType": String,
			"weight": Double,
			"unit": gram/pound,
			"time": timestamp,
			"nutrientList": [
			{
			"energy": Double,
			"protein": Double,
			"carbs": Double,
			"fat": Double
			}
			]
			}
			]
			}
			
			HTTP Status Code: 400 Bad Request
			
			{
			"errorCode": 2002,
			"errorMessage": User id not exists.
			}

			HTTP Status Code: 401 Unauthorized

Edit a diet record	PATCH /diet/records/{recordId}	{
		"dietType": String,
		"dietId": Integer,
		"weight": Double,
		"unit": gram/pound,
		"nutrientList": [
		{
		"energy": Double,
		"protein": Double,
		"carbs": Double,
		"fat": Double
		}
		]
		}
			HTTP Status Code: 200 Ok
			{
			"code": 200,
			"message": Successfully update.
			}
			
			HTTP Status Code: 400 Bad Request
			
			{
			"errorCode": 2001,
			"errorMessage": Record id not exists.
			}
			
			HTTP Status Code: 401
			Unauthorized

Create a new weight record	POST /weight/records	{
		"weight": Double,
		"unit": gram/pound
		}
			HTTP Status Code: 200 Ok
			{
			"code": 200,
			"message": Successfully record.
			}
			
			HTTP Status Code: 401
			Unauthorized

Get weight record history	GET /weight/records?period=(all/week/month/year)&length=Integer	{
		"userId": Integer,
		}
			HTTP Status Code: 200 Ok
			{
			"code": 200,
			"message": Successfully retrieve history.
			"recordsList": [
			{
			"recordId": Integer,
			"weight": Double,
			"unit": kilogram/pound,
			"time": timestamp
			}
			]
			}
			
			HTTP Status Code: 400 Bad Request
			
			{
			"errorCode": 3002,

			"errorMessage": User id not exists. } HTTP Status Code: 401 Unauthorized

Edit a weight record	PATCH /weight/records/{weightId}	{
		"weight": Double,
		"unit": gram/pound
		}
			HTTP Status Code: 200 Ok
			{
			"code": 200,
			"message": Successfully update.
			}
			
			HTTP Status Code: 400 Bad Request
			
			{
			"errorCode": 3001,
			"errorMessage": Record id not exists.
			}
			
			HTTP Status Code: 401 Unauthorized

Create a new exercise record	POST /exercise/records	{
		"exerciseType": String,
		"duration": Double, *// minutes*
		}
			HTTP Status Code: 200 Ok
			{
			"code": 200,
			"message": Successfully record.
			}
			
			HTTP Status Code: 401 Unauthorized

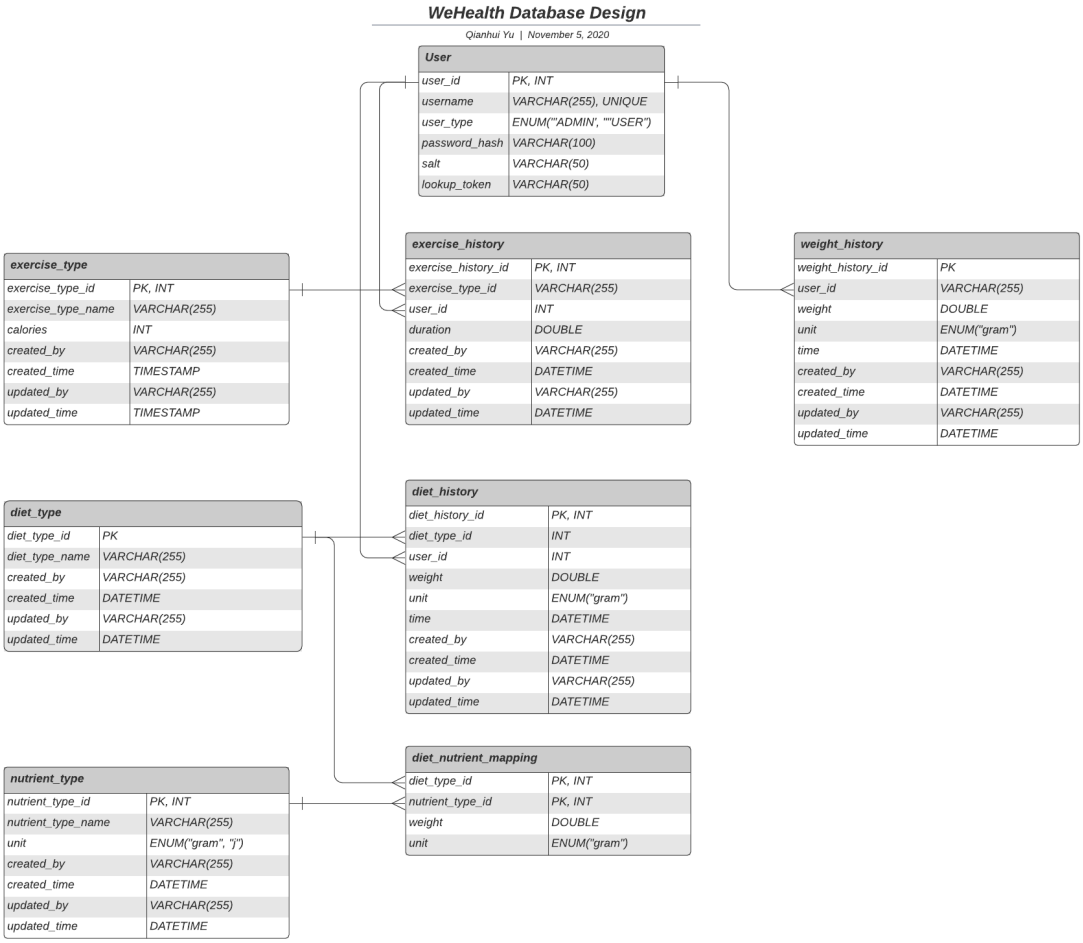
Get exercise records history	GET /exercise/records?period=(all/week/month/year)&length=Integer	{
		"userId": Integer,
		}
			HTTP Status Code: 200 Ok
			{
			"code": 200,
			"message": Successfully retrieve history.
			"recordsList": [
			{
			"recordId": Integer,
			"exerciseType": String,
			"duration": Double,
			"time": timestamp
			}
			]
			}
			
			HTTP Status Code: 400 Bad Request
			
			{

			"errorCode": 4002, "errorMessage": User id not exists. } HTTP Status Code: 401 Unauthorized

Edit an exercise record	PATCH /exercise/records/{recordId}	{
		"exerciseType": String,
		"duration": Double, *// minutes*
		}
			HTTP Status Code: 200 Ok
			{
			"code": 200,
			"message": Successfully update.
			}
			
			HTTP Status Code: 400 Bad Request
			
			{
			"errorCode": 4001,
			"errorMessage": Record id not exists.
			"advice": String
			}
			
			HTTP Status Code: 401 Unauthorized

Get advice	GET /advice	{
		"userId": Integer,
		}
			HTTP Status Code: 200 Ok
			{
			"code": 200,
			"message": Successfully update.
			}
			
			HTTP Status Code: 401 Unauthorized

Appendix B: Database design




```
CREATE TABLE `User` (
  `user_id` PK, INT,
  `username` VARCHAR(255), UNIQUE ,
  `user_type` ENUM("'ADMIN', ""USER"),
  `password_hash` VARCHAR(100),
  `salt` VARCHAR(50),
  `lookup_token` VARCHAR(50)
);

CREATE TABLE `diet_nutrient_mapping` (
  `diet_type_id` PK, INT,
  `nutrient_type_id` PK, INT,
  `weight` DOUBLE,
  `unit` ENUM("gram")
);

CREATE TABLE `weight_history` (
  `weight_history_id` PK,
  `user_id` VARCHAR(255),
  `weight` DOUBLE,
  `unit` ENUM("gram"),
  `time` DATETIME,
  `created_by` VARCHAR(255),
  `created_time` DATETIME,
  `updated_by` VARCHAR(255),
  `updated_time` DATETIME
);

CREATE TABLE `exercise_type` (
  `exercise_type_id` PK, INT,
  `exercise_type_name` VARCHAR(255),
  `calories` INT,
  `created_by` VARCHAR(255),
  `created_time` TIMESTAMP,
  `updated_by` VARCHAR(255),
  `updated_time` TIMESTAMP
);

CREATE TABLE `diet_type` (
  `diet_type_id` PK,
```

```
  `diet_type_name` VARCHAR(255),
  `created_by` VARCHAR(255),
  `created_time` DATETIME,
  `updated_by` VARCHAR(255),
  `updated_time` DATETIME
);

CREATE TABLE `nutrient_type` (
  `nutrient_type_id` PK, INT,
  `nutrient_type_name` VARCHAR(255),
  `unit` ENUM("gram", "j"),
  `created_by` VARCHAR(255),
  `created_time` DATETIME,
  `updated_by` VARCHAR(255),
  `updated_time` DATETIME
);

CREATE TABLE `exercise_history` (
  `exercise_history_id` PK, INT,
  `exercise_type_id` VARCHAR(255),
  `user_id` INT,
  `duration` DOUBLE,
  `created_by` VARCHAR(255),
  `created_time` DATETIME,
  `updated_by` VARCHAR(255),
  `updated_time` DATETIME
);

CREATE TABLE `diet_history` (
  `diet_history_id` PK, INT,
  `diet_type_id` INT,
  `user_id` INT,
  `weight` DOUBLE,
  `unit` ENUM("gram"),
  `time` DATETIME,
  `created_by` VARCHAR(255),
  `created_time` DATETIME,
  `updated_by` VARCHAR(255),
  `updated_time` DATETIME
);
```