CSCI 5448 Project - 6

Status Summary:

Title: Calorie Tracker
Members: Gaurav Roy, Varun Bhaskara

1. Work Done:

   Our goal is to build a web application that allows users to track their calories based on their meals and activities and see how they stand against the Calorie or Macro goal they set for themselves. In order to be able to do this, we created a Spring Boot application in the backend to create and expose Routes and a React application as the frontend to allow users to interact with the application.

   Backend:

   Varun:
   - Setup the Spring Boot application using the generated template from https://start.spring.io/
   - Setup MongoDB Atlas and Spring Mongo to store all user related data
   - Defined Models and the relationship between them.
   - Created Repository methods to be able to access Models
   - Implemented two design patterns,
     - Template - To calculate maintenance calories of a user based on Gender, Age, Height and Weight
     - Factory - To allow users to add a predefined set of exercises and calculate calories based on the number of minutes the exercise was performed.
   - Added routes for adding, removing and updating exercise's.
   - Added routes for querying the API and getting details about nutritions per given food.

   Gaurav:
   - Implemented JWT authentication using Spring Security to allow users to access protected routes.
   - Evaluated the feasibility between using Nutritionix API and USDA Food database and decided on Nutritionix due to ease of use and availability of Common and Branded foods.

- Created Service implementations for Routes defined in Controller's.
- Added route's for user login and signup
- Added route's for fetching and updating user goals.
- Added route's for adding and removing consumptions
- Tested routes using Postman

Frontend:

Gaurav:

- Setup React project with Redux, Bootstrap and Routing.
- Implemented Login/Signup screen.
- Implemented user goal's fetch and update screen.
- Implemented the home screen to render foods consumed on that given day along with the exercise performed.

Varun:

- Added reducers and actions in React-Redux to perform API call from the frontend.
- Added the component to search and add foods.
- Added the component to create and update exercises.

2. Changes:
    - Since Project-5 the overall idea has remained the same, but we had to make some changes to how we store our data. Initially, we thought of using MySQL, but we realized that a non-relational data store like MongoDB would be a better fit for our needs. MongoDB offers a cloud-hosted service called MongoDB Atlas, which provides us with a reliable and easy-to-use database solution. Additionally, using MongoDB with Spring Boot is simple and efficient thanks to the Spring Mongo library. By switching to MongoDB, we have the advantage of having a more flexible data model.
    - We also decided to useNutritionix API over USDA API since it provides more detailed nutrition information, including macronutrient and micronutrient data. We also found that Nutritionix API has a much better support for natural language search queries, allowing our users to search for food items using more intuitive search terms.

3. Patterns
    - **Template**: We use the template pattern to calculate maintenance calories for a user based on their height, weight, age, gender and exercise

frequency. Calculation of maintenance calories requires the calculation of basal metabolic rate which varies from one gender to another.By using the template pattern, we were able to implement two subclasses (Male and Female) that provide specific implementations for the BMR calculation. This results in code reusability and maintainability, as the common code for calculating maintenance calories is abstracted into a superclass, and the subclass implementations can be easily updated without affecting the overall algorithm.

- **Factory**: We use factory method to allow users to create different types of exercises and calculate calories burned doing that exercise based on approximate estimation of Metabolic Equivalent of Task of that exercise. With this we ensure the encapsulation of the creation of different types of exercises and their corresponding calorie calculation algorithms. This allows for a flexible and extensible design, where new types of exercises can be added without modifying the existing code.

- **Singleton:** In Spring Boot, Singleton's are used extensively to manage the lifecycle of Beans, which are objects managed by spring Inversion of Control container. When a Spring Boot application starts up, the Spring IoC container creates a single instance of each bean defined in the application context. These beans are then shared by all components that require them, ensuring that there is only one instance of each bean in the application (Reference: https://www.baeldung.com/spring-boot-singleton-vs-beans)

- **Proxy:** In Spring Boot, the proxy pattern is extensively used, wherein one object the proxy control's access to another object, in our code the repository classes directly access the DB via model classes which are the proxies.

4. Class diagram:

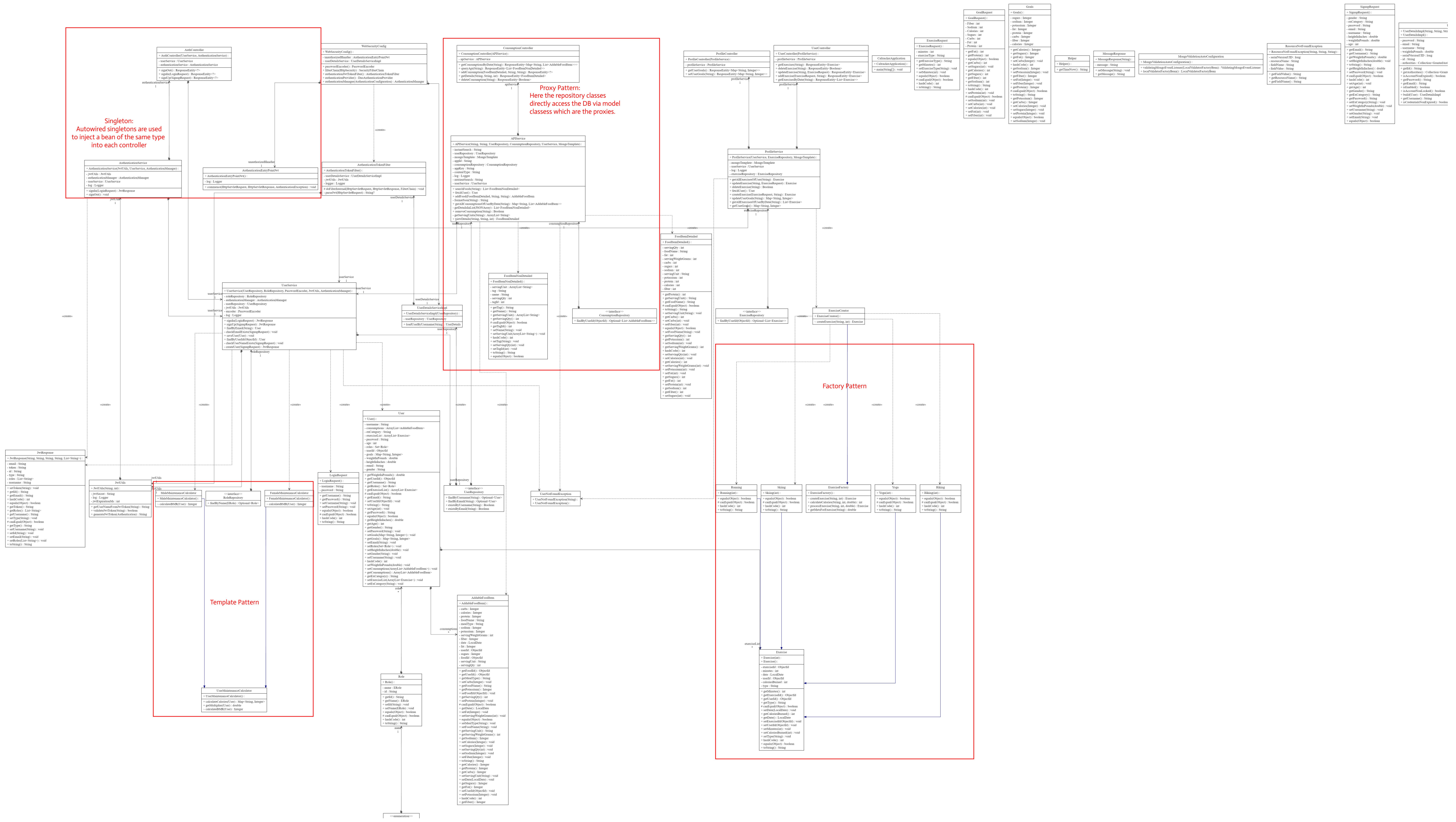
   Attached in the end

5. Plan for next Iteration:

   - By Next Iteration we will have a detailed view of the Analysis page, that would give the details about the Macros consumed and charts.

- It will also have a tracker or display to show how far someone is from their daily calorie Goals.
- And to make UI improvements and put more validations in Forms and API Handling.

If the above mentioned points are met, all of our design and product requirements will be achieved and by 5/3 we'll have our product working end to end.


Singleton:
Autowired singletons are used to inject a bean of the same type into each controller
Proxy Pattern:
Here the repository classes directly access the DB via model classess which are the proxies.
Template Pattern
Factory Pattern
AuthController
WebSecurityConfig
ConsumptionController
ProfileController
UserController
CaltrackerApplication
ExerciseRequest
GoalRequest
Goals
Helper
MessageResponse
MongoValidationAutoConfiguration
ResourceNotFoundException
SignupRequest
UserDetailsImpl
AuthenticationService
AuthenticationEntryPointJwt
AuthenticationTokenFilter
APIService
ProfileService
FoodItemDetailed
FoodItemNonDetailed
ConsumptionRepository
ExerciseRepository
ExerciseCreator
UserService
UserDetailsServiceImpl
JwtResponse
JwtUtils
MaleMaintenanceCalculator
RoleRepository
FemaleMaintenanceCalculator
LoginRequest
User
UserRepository
UserNotFoundException
Running
Skiing
ExerciseFactory
Yoga
Hiking
UserMaintenanceCalculator
Role
AddableFoodItem
Exercise
ERole