Group 3: Jahlil Owens, Trishelle Leal, and Landon Strappazon

Written By Jahlil Owens

Dr. Ruth Lamprecht

CMSCI 349 A

September 13, 2024

## Nutrition and Fitness tracking with Test and Software Engineers

Within our Nutrition and fitness tracking project, both Test Engineers and Software Engineers have to work together to ensure a well-structured application that allows the purpose of users to track their nutrition and fitness throughout the week and reach their fitness goals. The application will allow the user to log their meals, track nutritional intake, and monitor their progress toward their dietary goals. The Test engineer will create the script for these tasks to be functional while the software Engineer will identify the major components of the application ensuring they function cohesively.

As a Test Engineer, they need to create test scripts that fit the requirements for the project. They may not be fully functional in the beginning but after multiple tests and communication with the rest of the team, the scripts should create a secure foundation that ensures that the application functions work as intended. The first test case will focus on the signup section where the user would click the signup button if they never created an account with the application and will input valid information such as their name, email, and password. If their information follows all of the requirements for a valid account, they will submit the form, and the system will generate a new account for the user. This will allow them to access all of the app's features to help start their journey on fitness and nutrition tracking. The second test case is the login section which focuses on the user inputting their correct login credentials to get access to your account within the system. If the information is incorrect then they will be prompted with a message that says the username or password is incorrect. Once the information is inputted correctly, they will be brought to the dashboard within the app to see their information and progress for the week. The third test case is Meal Logging which allows the users to search for food within the system database so they can then log the item for their nutrition. Once they select the food the app will take the data and update it to the front end of the UI where the user's meal logs and nutritional values exist in the dashboard. The fourth test case is the manual meal Logging which has the same process as the third case but instead of the system automatically inputting all of the nutritional values by a search the user is allowed to input each nutritional value for that food item to get a more accurate data towards their dietary goal within the week. The fifth test case is the Progress tracking which tracks the dietary goals that users need to view within the app. They will be prompted with visual representations like charts and graphs alongside the number of nutritional intakes to help display their progress toward their dietary goal. The final test case for a Test Engineer is the Goal setting case which displays the user's main goal that ties with their dietary goals. Once the user is logged in, they can click on the goals section and input a new goal if they do not like their previous goal if one exits. After this update, the goal will appear in the dashboard. The Test Engineer is useful because it allows the user to see and interact with the GUI elements within the system. They allow the user to take advantage of the unique features that assemble the application's purpose. Based on the six test cases that were established we can state that the roles and actors for this project are the User, the Food database, and the Admin. For the User, they would have a provided interface that allows them to log into the system and see the tracking system for their nutrition, diet, and exercise activities. Our Food Database will have information on different foods, condiments, sauces, and more that can be tracked within the user's dietary meals. Once the user inputs the information it can be stored in the system where it has the total tracking of the week for their dietary meal.

import java.util.HashMap;

import java.util.Map;

/\*

\* Group 3 Jahlil Owens, Trishelle Leal, and Landon Strappazon

\* code by Jahlil Owens

\* 9/23/24

\*/

public class Pseudocode {

// Pseudocode for Nutrition and Fitness Tracking Application

// Main actors: User, Food Database, Admin

// Components: Signup, Login, Meal Logging, Progress Tracking, Goal Setting

//sign up process

public static String signup(String name, String email, String password) {

//checks if the email and password are valid

if (isValidEmail(email) && isValidPassword(password)) {

//if the email and password are valid it creates a new account for the user

createNewAccount(name, email, password);

return "Account created successfully";

} else {

//else tells the user that the credentials are invalid

return "Invalid credentials";

}

}

public static boolean isValidEmail(String email) {

//checks if the email meets security criteria.

return email.matches("^[a-zA-Z0-9.\_%+-]+@[a-zA-Z");

}

public static boolean isValidPassword(String password) {

//checks if the password meets security criteria.

return password.matches("^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\\d)");

}

//the porcess of creating the new account for the user.

public static Map<String, String> createNewAccount(String name, String email, String password) {

Map<String, String> user = new HashMap<>();

//takes the credentials of the user and puts them nto the hasmap to be used later

user.put("name", name);

user.put("email", email);

user.put("password", password);

//store the user data into the data base then return the user to the map

//....

System.out.println("New acount created for: "+name);

return user;

}

// Test Case 1: Signup Test

public static Object testSignup() {

String result = signup("John Doe", "john.doe@example.com", "password123");

assert(result == "Account created successfully");

return result;

}

// this is the seeing if the login was successful or not

public static String login(String email, String password) {

if (isCorrectCredentials(email, password)) {

return "Login successful";

} else {

return "Incorrect username or password";

}

}

// checks to see if the credentials for the user are correct or incorrect

public static boolean isCorrectCredentials(String email, String password) {

return email.equals("john.doe@example.com") && password.equals("password123");

//return true;

}

// Test Case 2: Login Test

public static Object testLogin() {

String result = login("john.doe@example.com", "password123");

assert(result == "Login successful");

return result;

}

// this logs all of the meals from the user.

public static String logMeal(String foodItem) {

// the data is to be searched in the food database.

String foodData = searchFoodDatabase(foodItem);

// if the food data exist within the database update the users meal log.

if (foodData != null) {

updateMealLog(foodData);

return "Meal logged successfully";

} else {

// otherwise say food is not found

return "Food not found";

}

}

//this looks for the food within the database

public static String searchFoodDatabase(String foodItem) {

//if the food exist then return its properties

if (foodItem.equalsIgnoreCase("Apple")) {

return "Apple: 95 calories";

} else {

//otherwise return nothing

return null;

}

}

//updates the meal log

public static String updateMealLog(String foodData) {

System.out.println("Logged meal: " + foodData);

return foodData;

}

// Test Case 3: Meal Logging Test

public static Object testMealLogging() {

String result = logMeal("Apple");

assert(result == "Meal logged successfully");

return result;

}

// allows the user to input a specific food item and its details

public static String manualLogMeal(String foodItem, Map<String, Integer> nutritionalData) {

updateMealLogWithManualData(foodItem, nutritionalData);

return "Meal logged with manual nutritional data";

}

// updates the meal log based on the manual input

public static String updateMealLogWithManualData(String foodItem, Map<String, Integer> nutritionalData) {

System.out.println("Logged " + foodItem + " with nutrition: " + nutritionalData.toString());

return foodItem;

}

// Test Case 4: Manual Meal Logging Test

public static Object testManualMealLogging() {

Map<String, Integer> nutritionalData = new HashMap<>();

//all of the properties for the food that are important for the users dietary diet

nutritionalData.put("calories", 150);

nutritionalData.put("protein", 5);

nutritionalData.put("fat", 10);

nutritionalData.put("Carbs", 50);

String result = manualLogMeal("Custom Food", nutritionalData);

assert(result == "Meal logged with manual nutritional data");

return result;

}

// keeps hold of the progress that is being tracked

public static Map<String, Object> trackProgress() {

Map<String, Object> progressData = calculateProgress();

return progressData;

}

// calaculates the progress that the user has done throughout the day and week

public static Map<String, Object> calculateProgress() {

Map<String, Object> progress = new HashMap<>();

progress.put("caloriesConsumed", 1500);

progress.put("proteinConsumed", 100);

progress.put("goalAchieved", false);

return progress;

}

// Test Case 5: Progress Tracking Test

public static Object testProgressTracking() {

Map<String, Object> expectedProgressData = new HashMap<>();

// this shows how the progress data can be examined.

expectedProgressData.put("caloriesConsumed", 1500);

expectedProgressData.put("proteinConsumed", 100);

expectedProgressData.put("goalAchieved", false);

//track the users actual progress

Map<String, Object> result = trackProgress();

//do the results match the expected progress

assert(result.equals(expectedProgressData));

return expectedProgressData;

}

// allows the user to set their dietary goals

public static String setGoal(String newGoal) {

updateGoal(newGoal);

return "Goal updated successfully";

}

// the process of updating the users new dietary goals

public static String updateGoal(String newGoal) {

System.out.println("Goal updated to: " + newGoal);

return newGoal;

}

// Test Case 6: Goal Setting Test

public static Object testGoalSetting() {

String result = setGoal("Lose 5 lbs");

assert(result == "Goal updated successfully");

return result;

}

//this would be loading the complete food database to the system

public static Object loadFoodDatabase() {

//this is a placeholder for the actual database loading process

System.out.println("Food database loaded.");

return new Object();

}

//this monitors the uers progress throughout the daya nd week

public static Object monitorUserProgress() {

//this is a placeholder for the actual progress monitoring process

System.out.println("User progress monitored.");

return new Object();

}

// Main flow of the application

public static void main(String[] args) {

// User interactions

Object userSignup = testSignup();

Object userLogin = testLogin();

Object mealLogging = testMealLogging();

Object manualMealLogging = testManualMealLogging();

Object progressTracking = testProgressTracking();

Object goalSetting = testGoalSetting();

// System components

Object foodDatabase = loadFoodDatabase();

Object userTracking = monitorUserProgress();

}

}