SOFTWARE VERIFICATION, VALIDATION AND TESTING

TESTING DOCUMENTATION

Prepared by:
Delić Faris
Herenda Ajla

Proposed to:
Samed Jukić, Assist. Prof. Dr.
Aldin Kovačević, Teaching Assistant

19th of January, 2023

TABLE OF CONTENTS

1. Introduction

1.1. About the Project

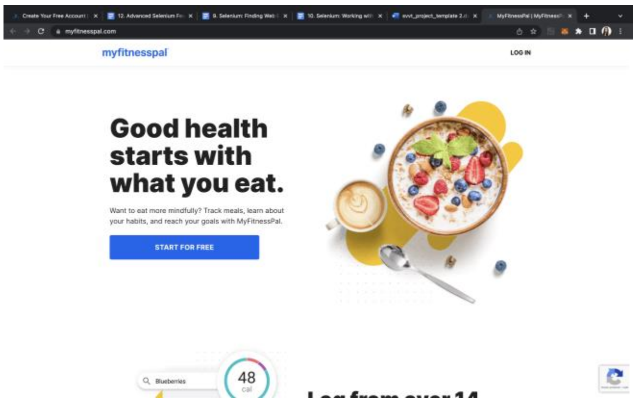
The tested website's name is "MyFitnessPal", as it can be concluded from the cover page and aids individuals with a goal in the fitness/health industry in achieving the aforementioned goal (loosing/gaining/maintaining weight/being healthy). Furthermore, it allows users to create a personalized profile (containing their personal details, focusing more on the information that describes their physical appearance and level of fitness (weight, height, activity level and etc.)

The link to the MyFitnessPal website: *https://www.myfitnesspal.com/*

1.2. Project Functionalities and Screenshots

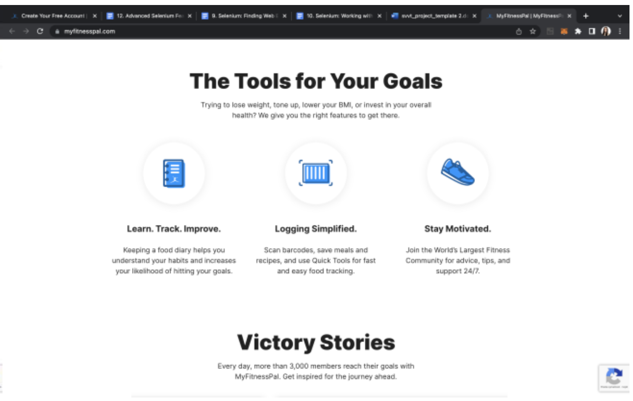
In order to corroborate the previously stated features of the website, we will provide a series of screenshots that will also deepen your understanding regarding those.

a) Account creation feature, which allows us to keep track of our data, which is used to describe our success rates on the road of achieving our goals.


myfitnesspal.com

Good health starts with what you eat.

Want to eat more mindfully? Track meals, learn about your habits, and reach your goals with MyFitnessPal.


myfitnesspal.com

The Tools for Your Goals

Trying to lose weight, tone up, lower your BMI, or invest in your overall health? We give you the right features to get there.

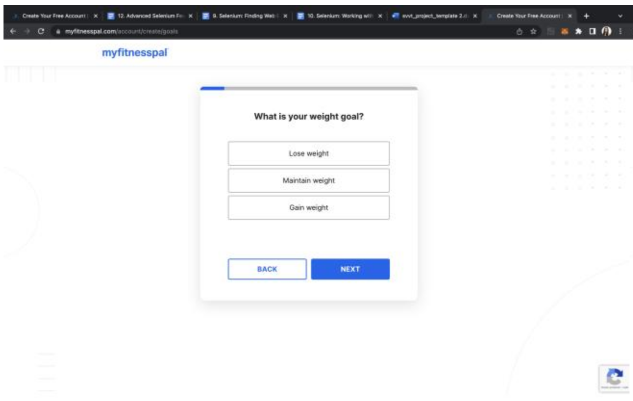
Learn. Track. Improve. Keeping a food diary helps you understand your habits and increases your likelihood of hitting your goals.

Logging Simplified. Scan barcodes, save meals and recipes, and use Quick Tools for fast and easy food tracking.

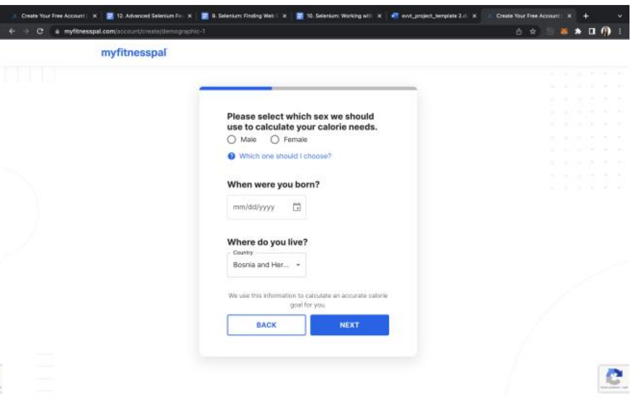
Stay Motivated. Join the World's Largest Fitness Community for advice, tips, and support 24/7.

Victory Stories

Every day, more than 3,000 members reach their goals with MyFitnessPal. Get inspired for the journey ahead.


myfitnesspal.com/account/create/goals

What is your weight goal?


myfitnesspal.com/account/create/demographic-1

Please select which sex we should use to calculate your calorie needs.

Male / Female

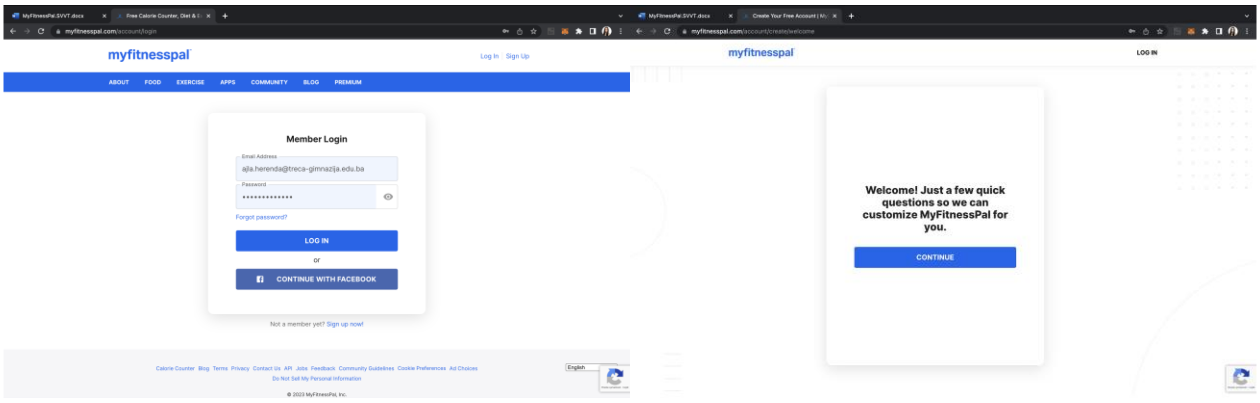
Which one should I choose?

When were you born? mm/dd/yyyy

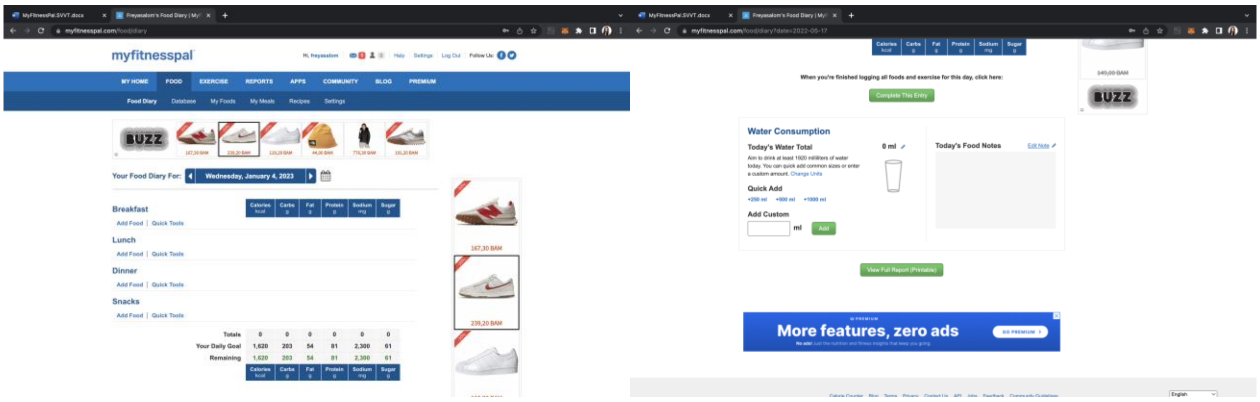
Where do you live? Country: Bosnia and Her...

We use this information to calculate an accurate calorie goal for you.

b) Login and Sign-Up feature, which are prerequisites for the features to follow (daily calorie/water intake/exercise tracker).

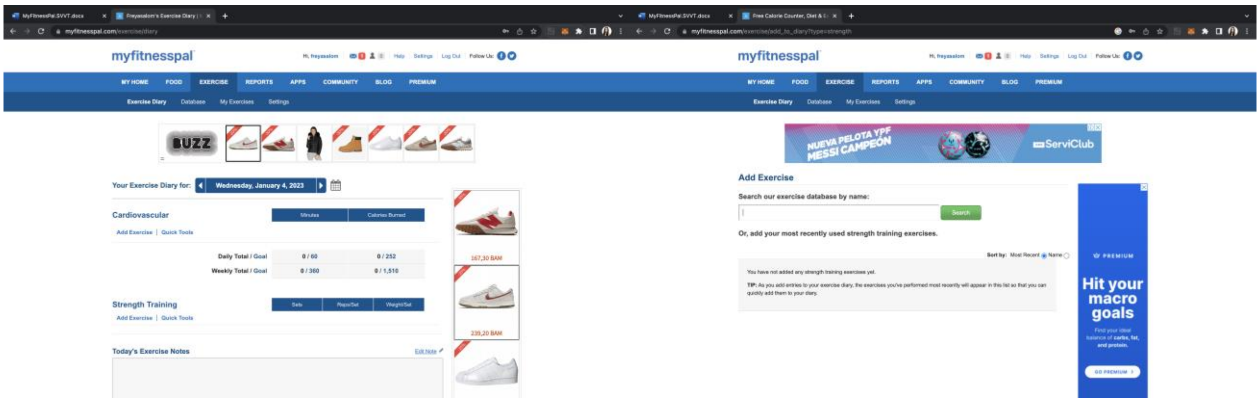

myfitnesspal
Log In | Sign Up
ABOUT FOOD EXERCISE APPS COMMUNITY BLOG PREMIUM
Member Login
Email Address
ajla.herenda@treca-gimnazija.edu.ba
Password
Forgot password?
LOG IN
or
CONTINUE WITH FACEBOOK
Not a member yet? Sign up now!
LOG IN
Welcome! Just a few quick questions so we can customize MyFitnessPal for you.
CONTINUE

c) Daily calorie intake tracker for breakfast, lunch, dinner and snack time.

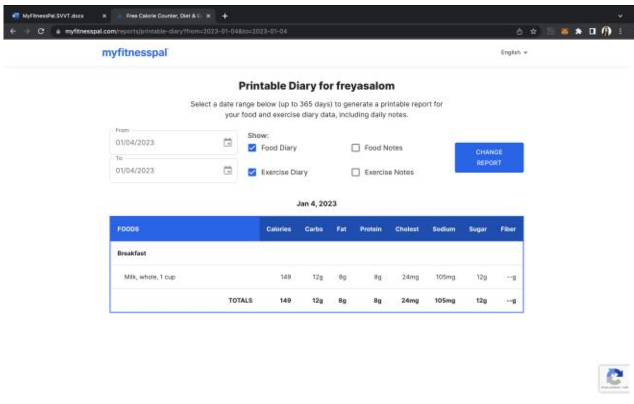

myfitnesspal
MY HOME FOOD EXERCISE REPORTS APPS COMMUNITY BLOG PREMIUM
Food Diary Database My Foods My Meals Recipes Settings
Your Food Diary For: Wednesday, January 4, 2023
Breakfast
Lunch
Dinner
Snacks
Add Food | Quick Tools
Totals
Your Daily Goal
Remaining
Calories Carbs Fat Protein Sodium Sugar
When you're finished logging all foods and exercise for this day, click here:
Complete This Entry
Water Consumption
Today's Water Total
0 ml
Quick Add
Add Custom
Today's Food Notes
View Full Report (Printable)
More features, zero ads

	Calories kcal	Carbs g	Fat g	Protein g	Sodium mg	Sugar g
Totals	0	0	0	0	0	0
Your Daily Goal	1,620	203	54	81	2,300	61
Remaining	1,620	203	54	81	2,300	61

d) Exercise tracker.


myfitnesspal
MY HOME FOOD EXERCISE REPORTS APPS COMMUNITY BLOG PREMIUM
Exercise Diary Database My Exercises Settings
Your Exercise Diary for: Wednesday, January 4, 2023
Cardiovascular
Minutes Calories Burned
Add Exercise | Quick Tools
Daily Total / Goal 0 / 60 0 / 252
Weekly Total / Goal 0 / 360 0 / 1,510
Strength Training
Sets Reps/Set Weight/Set
Today's Exercise Notes
Add Exercise
Search our exercise database by name:
Search
Or, add your most recently used strength training exercises.
You have not added any strength training exercises yet.
Hit your macro goals

e) Features related to the nutritional value of the foods we had for that day.


myfitnesspal.com/reports/printable-diary?from=2023-01-04&to=2023-01-04

Printable Diary for freyasalom

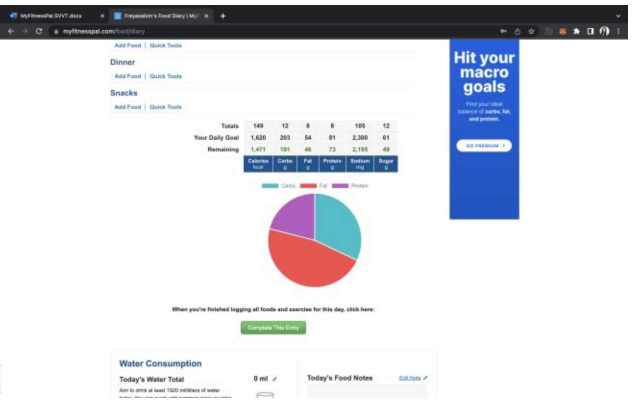
Select a date range below (up to 365 days) to generate a printable report for your food and exercise diary data, including daily notes.

From: 01/04/2023
To: 01/04/2023

Show: [✓ Food Diary] Food Notes [✓ Exercise Diary] Exercise Notes

Jan 4, 2023

FOODS	Calories	Carbs	Fat	Protein	Cholest	Sodium	Sugar	Fiber
Breakfast								
Milk, whole, 1 cup	149	12g	8g	8g	24mg	105mg	12g	--g
TOTALS	149	12g	8g	8g	24mg	105mg	12g	--g


myfitnesspal.com/food/diary

Dinner
Add Food | Quick Tools

Snacks
Add Food | Quick Tools

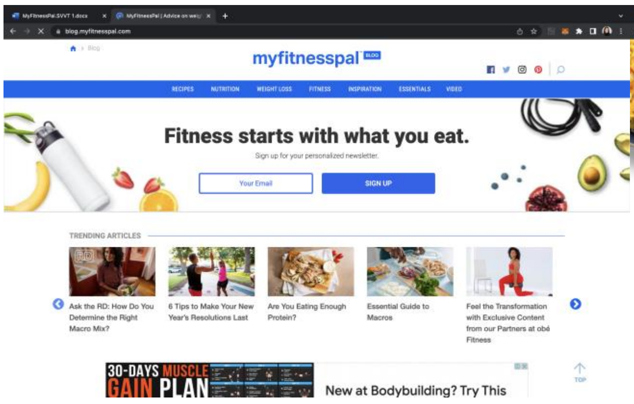
	Calories kcal	Carbs g	Fat g	Protein g	Sodium mg	Sugar g
Totals	149	12	8	8	105	12
Your Daily Goal	1,620	203	54	81	2,300	61
Remaining	1,471	191	46	73	2,195	49

When you're finished logging all foods and exercise for this day, click here:

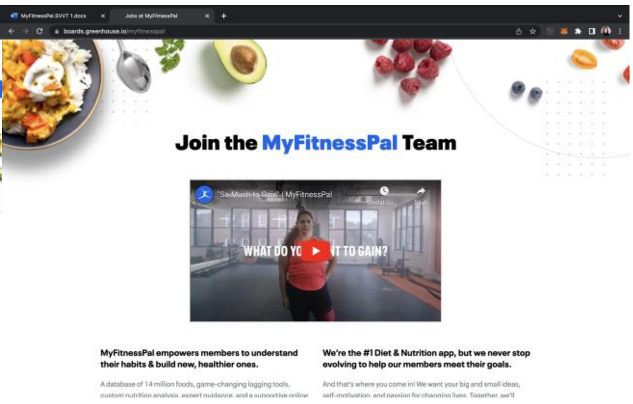
Water Consumption
Today's Water Total 0 ml
Aim to drink at least 1920 milliliters of water

Today's Food Notes

f) Features that provide additional support in terms of fitness information (blog) and employment opportunities.



blog.myfitnesspal.com
myfitnesspal BLOG
RECIPES
NUTRITION
WEIGHT LOSS
FITNESS
INSPIRATION
ESSENTIALS
VIDEO
Fitness starts with what you eat.
Sign up for your personalized newsletter.
Your Email
SIGN UP
TRENDING ARTICLES
Ask the RD: How Do You Determine the Right Macro Mix?
6 Tips to Make Your New Year's Resolutions Last
Are You Eating Enough Protein?
Essential Guide to Macros
Feel the Transformation with Exclusive Content from our Partners at obé Fitness
30-DAYS MUSCLE GAIN PLAN
New at Bodybuilding? Try This
TOP


boards.greenhouse.io/myfitnesspal

Join the MyFitnessPal Team

MyFitnessPal empowers members to understand their habits & build new, healthier ones.

A database of 14 million foods, game-changing logging tools, custom nutrition analysis, expert guidance, and a supportive online

We're the #1 Diet & Nutrition app, but we never stop evolving to help our members meet their goals.

And that's where you come in! We want your big and small ideas, self-motivation, and passion for changing lives. Together, we'll

2. Test Plan

2.1. Scope

Due to the complexity of the familiar website and the shortage of human workforce we are dealing with, since the team is comprised from the two of us, we will be attempting to test only certain features, which are crucial for the user experience. The previously mentioned features include the actions related to the start for free/login/sign up process, all of which are closely related to the personalized account. In addition, we will test the features related to tracking an individual's daily calorie intake/exercise level/daily steps and etc. It is important to be mentioned, that we will also write several tests for the "Jobs" and "Blog" section, that can be found on this website. What we can ensure that it will be skipped in our testing, is testing performed on the account's settings, like privacy and security.

2.2. Testing Environment and Tools

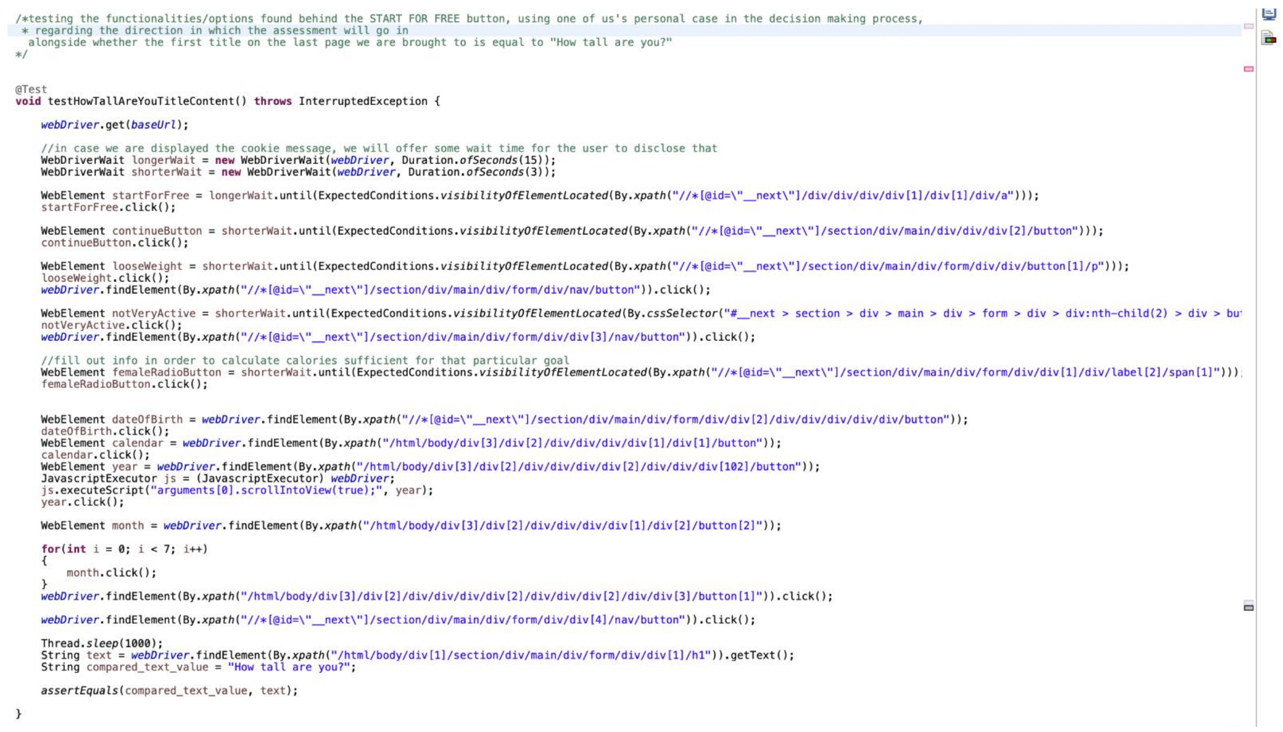
The following applies to all of our tests, further meaning that all of them are written in the Java programming language, as part of which we have implemented Selenium as a dependency in order for it to be successfully executed in Eclipse. The test cases make use of the JRE Library is J2SE 1.5, belong to the group of JUnit 5 test cases and are implemented in a Maven project in the Eclipse IDE (Version: 2021-09 (4.21.0)).

3. Test Execution

3.1. Kick-start with MyFitnessPal – Account Creation/Login/Sign Up

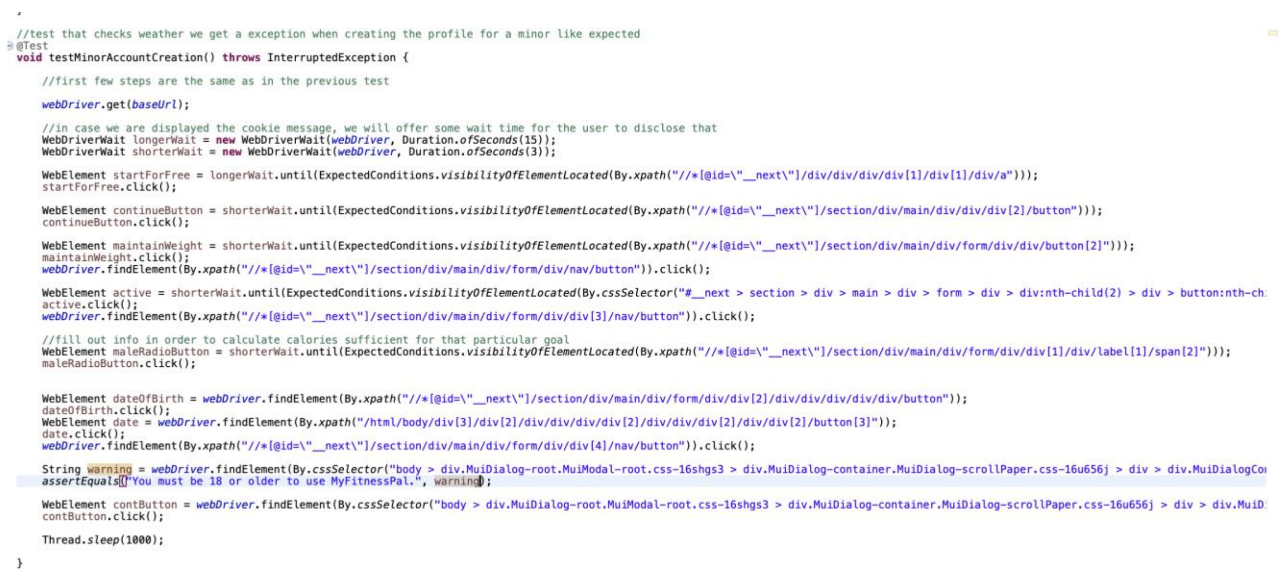
Based on the title of this test scenario, we can conclude that test cases have been written and run for the initial user encounter with the system, which commonly is triggered by one of the “start for free”/” login”/” sign up” buttons, all of which can be located on the web page.

Test Name: Test for the “How tall are you?” Title				
Description: Testing the functionalities/options found behind the “START FOR FREE” button, using one of us’ personal case in the decision-making process, regarding the direction in which the assessment will go in, alongside whether the first title on the last page we are brought to is equal to "How tall are you?"				
Pre-condition(s): Cookies have to be manually accepted in order for elements to be visible. WebDriver uses an option for starting in a maximized window.				
Test Steps: 1. Initially we would have to click on the “start for free” button 2. Furthermore we are required to fill out the fields, which are supposed to contain test data about our fitness goal. This is done by clicking on certain elements/filling them out using sendKeys(). 3. After completion of the previous step, we are brought to the last page in this test case, where we assert if the expected text value is equal to the actual content of the first title on that page.	**Test Data:** - test data in this case can be represented as the choices made as part of the decision-making process and include: -Lose Weight -Not very active -Female -13.08.2001.	**Expected Result:** The expected result is the string: “How tall are you?”.	**Actual Result:** The actual result is the string: “How tall are you?”.	**Status:** PASS
Notes: As part of the test were used waits, a longer and shorter one instead of Thread.sleep() in case certain elements were not loaded. We also have made use of the JavaScript Executor in order to scroll into the proper year. Test steps are avoided to be written in detail due to the length of the test and fact that moreover the logic can be concluded from the code and information supplied above.				



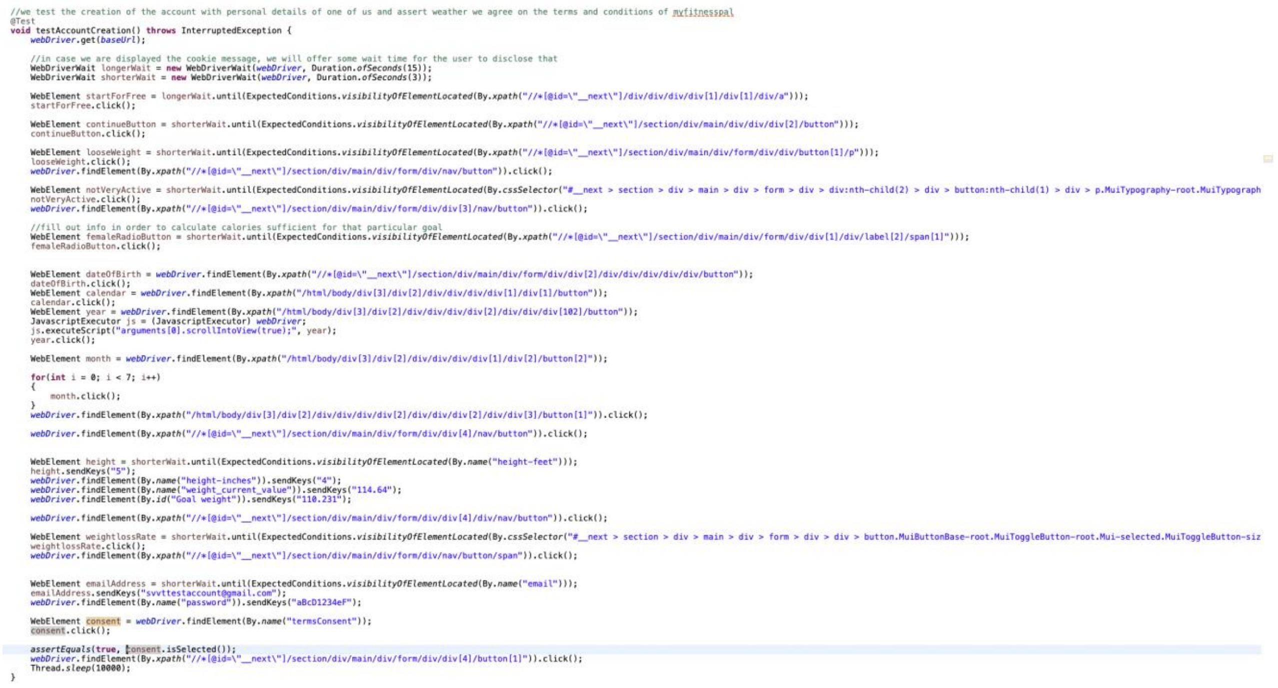
```java
/*testing the functionalities/options found behind the START FOR FREE button, using one of us's personal case in the decision making process,
 * regarding the direction in which the assessment will go in
  alongside whether the first title on the last page we are brought to is equal to "How tall are you?"
*/


@Test
void testHowTallAreYouTitleContent() throws InterruptedException {

    webDriver.get(baseUrl);

    //in case we are displayed the cookie message, we will offer some wait time for the user to disclose that
    WebDriverWait longerWait = new WebDriverWait(webDriver, Duration.ofSeconds(15));
    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(3));

    WebElement startForFree = longerWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/div/div/div/div[1]/div[1]/div/a")));
    startForFree.click();

    WebElement continueButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/section/div/main/div/div/div[2]/button")));
    continueButton.click();

    WebElement looseWeight = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div/button[1]/p")));
    looseWeight.click();
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/nav/button")).click();

    WebElement notVeryActive = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > section > div > main > div > form > div > div:nth-child(2) > div > bu
    notVeryActive.click();
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[3]/nav/button")).click();

    //fill out info in order to calculate calories sufficient for that particular goal
    WebElement femaleRadioButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[1]/div/label[2]/span[1]")))
    femaleRadioButton.click();


    WebElement dateOfBirth = webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[2]/div/div/div/div/div/button"));
    dateOfBirth.click();
    WebElement calendar = webDriver.findElement(By.xpath("/html/body/div[3]/div[2]/div/div/div/div[1]/div[1]/button"));
    calendar.click();
    WebElement year = webDriver.findElement(By.xpath("/html/body/div[3]/div[2]/div/div/div/div[2]/div/div/div[102]/button"));
    JavascriptExecutor js = (JavascriptExecutor) webDriver;
    js.executeScript("arguments[0].scrollIntoView(true);", year);
    year.click();

    WebElement month = webDriver.findElement(By.xpath("/html/body/div[3]/div[2]/div/div/div/div[1]/div[2]/button[2]"));

    for(int i = 0; i < 7; i++)
    {
        month.click();
    }
    webDriver.findElement(By.xpath("/html/body/div[3]/div[2]/div/div/div/div[2]/div/div/div[2]/div/div[3]/button[1]")).click();

    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[4]/nav/button")).click();

    Thread.sleep(1000);
    String text = webDriver.findElement(By.xpath("/html/body/div[1]/section/div/main/div/form/div/div[1]/h1")).getText();
    String compared_text_value = "How tall are you?";

    assertEquals(compared_text_value, text);

}
```

Test Name: Test Minor Account Creation				
Description: Test that checks whether we get an exception message, when creating the profile for a minor like expected.				
Pre-condition(s): We will have to go through a series of clicking and decision-making in order to get to the point in the account creation process, where we are deciding on the date of birth. Implementation of a web driver that uses an option for starting in a maximized window.				
Test Steps: 1. Initially we perform the steps from the previous test case, till we get to the date of birth field (test data is only different). 2. Furthermore we click on the elements related to this field, to display all possible options and click on any day in this year, here in particular our test data. 3. After completion of the previous step, we aim to go to the next step by clicking “next”, but what will happen instead is that due to the age limitations we will be “stuck”. 4. In addition we will compare what we expect to be shown as a message and what is actually shown, using assertEquals().	**Test Data:** - test data in this case can be represented as the choices made as part of the decision-making process and include: -Maintain weight -Active -Male -04.01.2023.	**Expected Result:** The expected result is the string: "You must be 18 or older to use MyFitnessPal."	**Actual Result:** The actual result is the string: “You must be 18 or older to use MyFitnessPal."	**Status:** PASS
Notes: Here as well a part of the test were used waits, a longer and shorter one instead of Thread.sleep() in case certain elements were not loaded.				



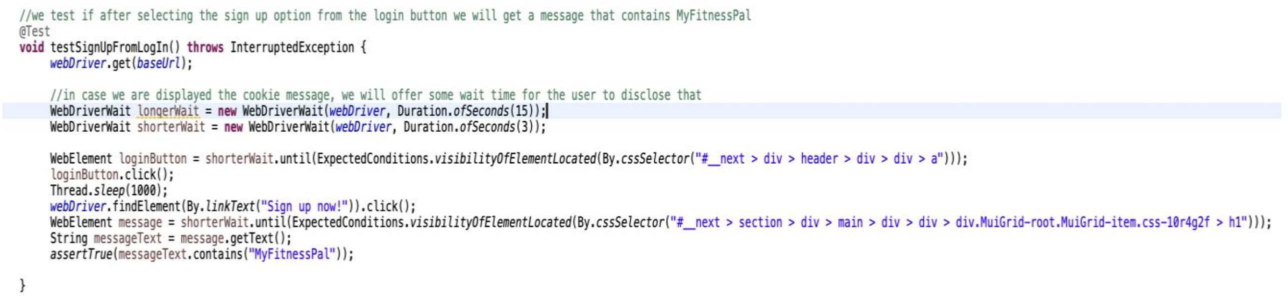
```java
,
//test that checks weather we get a exception when creating the profile for a minor like expected
@Test
void testMinorAccountCreation() throws InterruptedException {

    //first few steps are the same as in the previous test

    webDriver.get(baseUrl);

    //in case we are displayed the cookie message, we will offer some wait time for the user to disclose that
    WebDriverWait longerWait = new WebDriverWait(webDriver, Duration.ofSeconds(15));
    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(3));

    WebElement startForFree = longerWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/div/div/div/div[1]/div[1]/div/a")));
    startForFree.click();

    WebElement continueButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/section/div/main/div/div/div[2]/button")));
    continueButton.click();

    WebElement maintainWeight = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div/button[2]")));
    maintainWeight.click();
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/nav/button")).click();

    WebElement active = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > section > div > main > div > form > div > div:nth-child(2) > div > button:nth-ch
    active.click();
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[3]/nav/button")).click();

    //fill out info in order to calculate calories sufficient for that particular goal
    WebElement maleRadioButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[1]/div/label[1]/span[2]")));
    maleRadioButton.click();


    WebElement dateOfBirth = webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[2]/div/div/div/div/div/button"));
    dateOfBirth.click();
    WebElement date = webDriver.findElement(By.xpath("/html/body/div[3]/div[2]/div/div/div/div[2]/div/div/div[2]/div/div[2]/button[3]"));
    date.click();
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[4]/nav/button")).click();

    String warning = webDriver.findElement(By.cssSelector("body > div.MuiDialog-root.MuiModal-root.css-16shgs3 > div.MuiDialog-container.MuiDialog-scrollPaper.css-16u656j > div > div.MuiDialogCo
    assertEquals("You must be 18 or older to use MyFitnessPal.", warning);

    WebElement contButton = webDriver.findElement(By.cssSelector("body > div.MuiDialog-root.MuiModal-root.css-16shgs3 > div.MuiDialog-container.MuiDialog-scrollPaper.css-16u656j > div > div.MuiD
    contButton.click();

    Thread.sleep(1000);

}
```

Test Name: Test if Account Agreed on Terms and Conditions				
Description: We test the creation of the account with personal details of one of us and assert whether we agree on the terms and conditions of MyFitnessPal.				
Pre-condition(s): Implementation of a web driver that uses an option for starting in a maximized window. Performing steps for the creation of an account on MyFitnessPal, which we have encountered as part of the previous test(s).				
Test Steps: 1. As per usual we perform the steps from the previous test case, till we get to the fields that have not yet been covered, then fill them too with test data. 2. The aforementioned is done with multiple occurrences of click(), sendKeys(), with which we will make decisions regarding our goal, activity level, gender, date of birth, height, weight, goal weight, email and password. 3. Completing the previous we are only left to agree to the terms and conditions by clicking on the radio button. 4. Finally we check if that radio button has been clicked using isClicked() and assertions.	**Test Data:** - test data in this case can be represented as the choices made as part of the decision-making process and include: -Lose Weight -Not very active -Female -13.08.2001. -5.4ft -114.64 -110.231 svvttestaccount@gmail.com -aBcD1234eF	**Expected Result:** The expected result is the Boolean: True.	**Actual Result:** The actual result is the Boolean: True	**Status:** PASS
Notes: Longer and short waits, to control the presence of elements at a given point in time.				



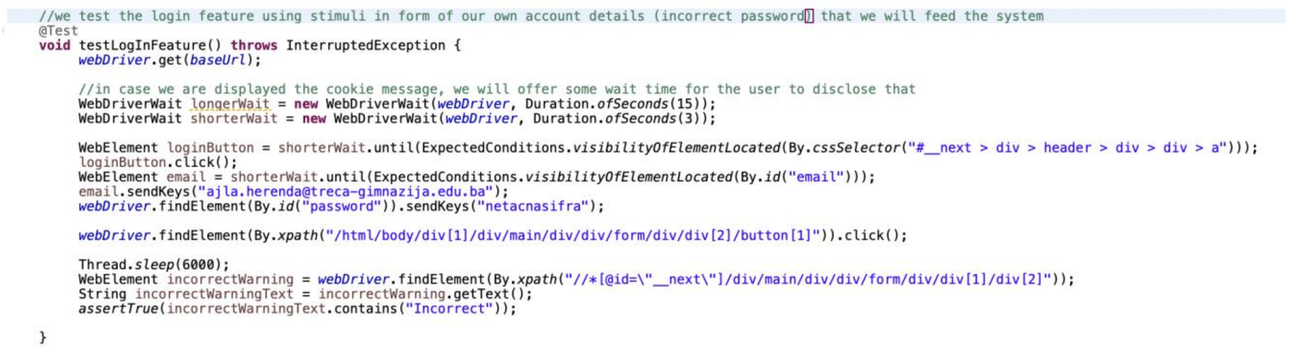
```java
//we test the creation of the account with personal details of one of us and assert weather we agree on the terms and conditions of myfitnesspal
@Test
void testAccountCreation() throws InterruptedException {
    webDriver.get(baseUrl);

    //in case we are displayed the cookie message, we will offer some wait time for the user to disclose that
    WebDriverWait longerWait = new WebDriverWait(webDriver, Duration.ofSeconds(15));
    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(3));

    WebElement startForFree = longerWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/div/div/div/div[1]/div[1]/div/a")));
    startForFree.click();

    WebElement continueButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/section/div/main/div/div/div[2]/button")));
    continueButton.click();

    WebElement looseWeight = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div/button[1]/p")));
    looseWeight.click();
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/nav/button")).click();

    WebElement notVeryActive = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > section > div > main > div > form > div > div:nth-child(2) > div > button:nth-child(1) > div > p.MuiTypography-root.MuiTypograph
    notVeryActive.click();
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[3]/nav/button")).click();

    //fill out info in order to calculate calories sufficient for that particular goal
    WebElement femaleRadioButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[1]/div/label[2]/span[1]")));
    femaleRadioButton.click();


    WebElement dateOfBirth = webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[2]/div/div/div/div/div/button"));
    dateOfBirth.click();
    WebElement calendar = webDriver.findElement(By.xpath("/html/body/div[3]/div[2]/div/div/div/div[1]/div[1]/button"));
    calendar.click();
    WebElement year = webDriver.findElement(By.xpath("/html/body/div[3]/div[2]/div/div/div/div[2]/div/div/div[102]/button"));
    JavascriptExecutor js = (JavascriptExecutor) webDriver;
    js.executeScript("arguments[0].scrollIntoView(true);", year);
    year.click();

    WebElement month = webDriver.findElement(By.xpath("/html/body/div[3]/div[2]/div/div/div/div[1]/div[2]/button[2]"));

    for(int i = 0; i < 7; i++)
    {
        month.click();
    }
    webDriver.findElement(By.xpath("/html/body/div[3]/div[2]/div/div/div/div[2]/div/div/div[2]/div/div[3]/button[1]")).click();

    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[4]/nav/button")).click();


    WebElement height = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.name("height-feet")));
    height.sendKeys("5");
    webDriver.findElement(By.name("height-inches")).sendKeys("4");
    webDriver.findElement(By.name("weight_current_value")).sendKeys("114.64");
    webDriver.findElement(By.id("Goal weight")).sendKeys("110.231");

    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[4]/div/nav/button")).click();

    WebElement weightlossRate = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > section > div > main > div > form > div > div > button.MuiButtonBase-root.MuiToggleButton-root.Mui-selected.MuiToggleButton-siz
    weightlossRate.click();
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/nav/button/span")).click();


    WebElement emailAddress = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.name("email")));
    emailAddress.sendKeys("svvttestaccount@gmail.com");
    webDriver.findElement(By.name("password")).sendKeys("aBcD1234eF");

    WebElement consent = webDriver.findElement(By.name("termsConsent"));
    consent.click();

    assertEquals(true, consent.isSelected());
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/section/div/main/div/form/div/div[4]/button[1]")).click();
    Thread.sleep(10000);
}
```

Test Name: Test Sign Up from Login				
Description: We test if after selecting the sign-up option from the login button we will get a message that contains "MyFitnessPal".				
Pre-condition(s): Implementation of a web driver that uses an option for starting in a maximized window.				
Test Steps: 1. We locate and click on the "Log in" button. 2. After waiting, using the shorter wait, which we have previously implemented, we will follow the link used for signing up. 3. When the link and the contents of the page are loaded, we assert if the title message on the screen contains "MyFitnessPal".	**Test Data:** - There is no test data that we are supplying to this test.	**Expected Result:** The expected result is the Boolean: True.	**Actual Result:** The actual result is the Boolean: True	**Status:** PASS
Notes: Shorter wait, to control the presence of elements at a given point in time.				



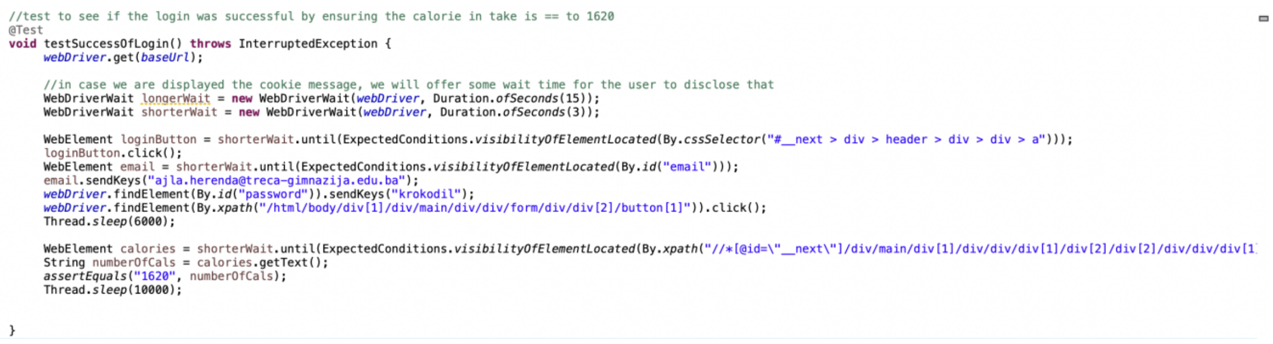
```java
//we test if after selecting the sign up option from the login button we will get a message that contains MyFitnessPal
@Test
void testSignUpFromLogIn() throws InterruptedException {
    webDriver.get(baseUrl);

    //in case we are displayed the cookie message, we will offer some wait time for the user to disclose that
    WebDriverWait longerWait = new WebDriverWait(webDriver, Duration.ofSeconds(15));
    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(3));

    WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
    loginButton.click();
    Thread.sleep(1000);
    webDriver.findElement(By.linkText("Sign up now!")).click();
    WebElement message = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > section > div > main > div > div > div.MuiGrid-root.MuiGrid-item.css-10r4g2f > h1")));
    String messageText = message.getText();
    assertTrue(messageText.contains("MyFitnessPal"));

}
```

Test Name: Test Log in with Incorrect Stimuli and Assert Error Message				
Description: We test the login feature using stimuli in form of our own account details that we will feed the system, which do not satisfy any existing account, meaning that it will result in an error message, whose value we will assert.				
Pre-condition(s): Implementation of a web driver that uses an option for starting in a maximized window.				
Test Steps:	**Test Data:**	**Expected Result:**	**Actual Result:**	**Status:**
1. We once again locate and click on the “Log in” button. 2. After waiting, using thread.sleep(), we will be using sendKeys() to fill out the email and password field with test data. 3. Afterwards we click on the “Next” button, and assert that the error message contains the word “Incorrect”.	- We are using test data for the email and password of our “test account”, which are: ajla.herenda@treca-gimnazija.edu.b -”netacnasifra”	The expected result is the Boolean: True.	The actual result is the Boolean: True	PASS
Notes: Thread.sleep, to control the presence of elements at a given point in time.				



```
//we test the login feature using stimuli in form of our own account details (incorrect password) that we will feed the system
@Test
void testLogInFeature() throws InterruptedException {
      webDriver.get(baseUrl);

      //in case we are displayed the cookie message, we will offer some wait time for the user to disclose that
      WebDriverWait longerWait = new WebDriverWait(webDriver, Duration.ofSeconds(15));
      WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(3));

      WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
      loginButton.click();
      WebElement email = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("email")));
      email.sendKeys("ajla.herenda@treca-gimnazija.edu.ba");
      webDriver.findElement(By.id("password")).sendKeys("netacnasifra");

      webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div/div/form/div/div[2]/button[1]")).click();

      Thread.sleep(6000);
      WebElement incorrectWarning = webDriver.findElement(By.xpath("//*[@id=\"__next\"]/div/main/div/div/form/div/div[1]/div[2]"));
      String incorrectWarningText = incorrectWarning.getText();
      assertTrue(incorrectWarningText.contains("Incorrect"));

}
```

Test Name: Test Success of Log In				
Description: We test to see if the login was successful by ensuring the calorie intake is == to 1620.				
Pre-condition(s): Implementation of a web driver that uses an option for starting in a maximized window. Alongside this standard prerequisite, we are here needing the account details of an actually existing account.				
Test Steps: 1. We locate and click on the “Log in” button. 2. After waiting, using the thread.sleep() for waiting, we will be using sendKeys() to fill out the email and password field with test data. 3. Afterwards we click on the “Next” button, and are forwarded to a new page-account's dashboard. 4. There we can find information about our daily calorie intake, which is 1620 and we assert it to be so.	**Test Data:** - We are using real test data for the email and password of our “test account”, which are: - ajla.herenda@treca-gimnazija.edu.b -”krokodil”	**Expected Result:** The expected result of that field is 1620.	**Actual Result:** The actual result of the field is indeed 1620, causing the assertion to be true.	**Status:** PASS
Notes: Thread.sleep, to control the presence of elements at a given point in time.				

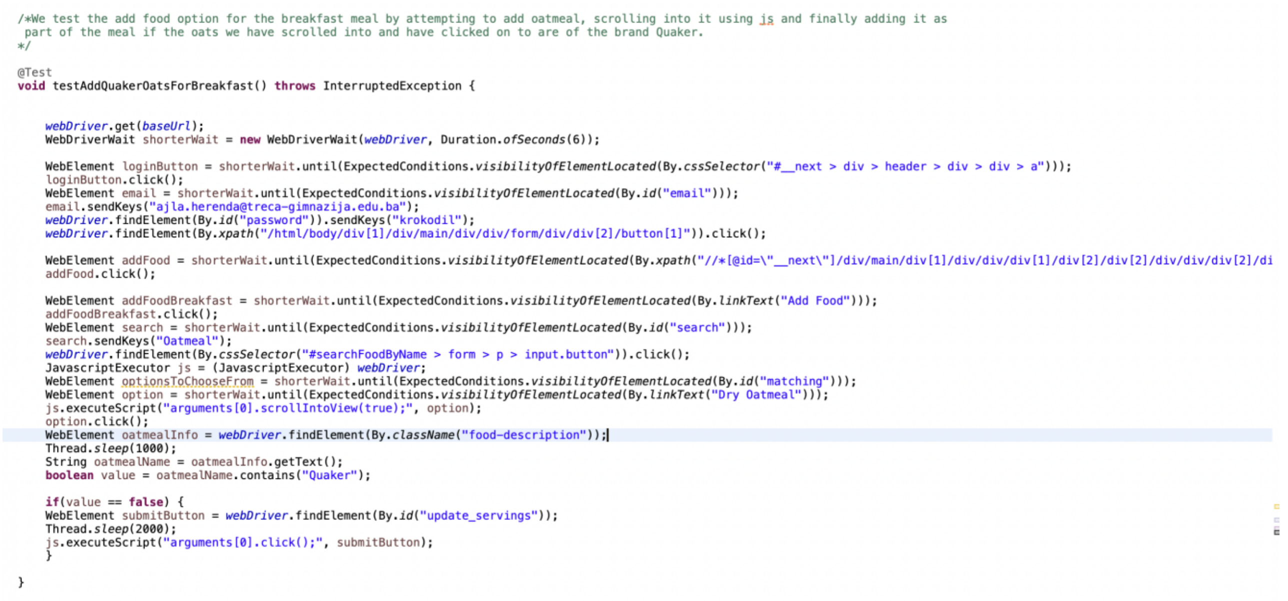


```
//test to see if the login was successful by ensuring the calorie in take is == to 1620
@Test
void testSuccessOfLogin() throws InterruptedException {
     webDriver.get(baseUrl);

     //in case we are displayed the cookie message, we will offer some wait time for the user to disclose that
     WebDriverWait longerWait = new WebDriverWait(webDriver, Duration.ofSeconds(15));
     WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(3));

     WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
     loginButton.click();
     WebElement email = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("email")));
     email.sendKeys("ajla.herenda@treca-gimnazija.edu.ba");
     webDriver.findElement(By.id("password")).sendKeys("krokodil");
     webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div/div/form/div/div[2]/button[1]")).click();
     Thread.sleep(6000);

     WebElement calories = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/div/main/div[1]/div/div/div[1]/div[2]/div[2]/div/div/div[1
     String numberOfCals = calories.getText();
     assertEquals("1620", numberOfCals);
     Thread.sleep(10000);


}
```

3.2. Using Features Related to the Daily Calorie Intake

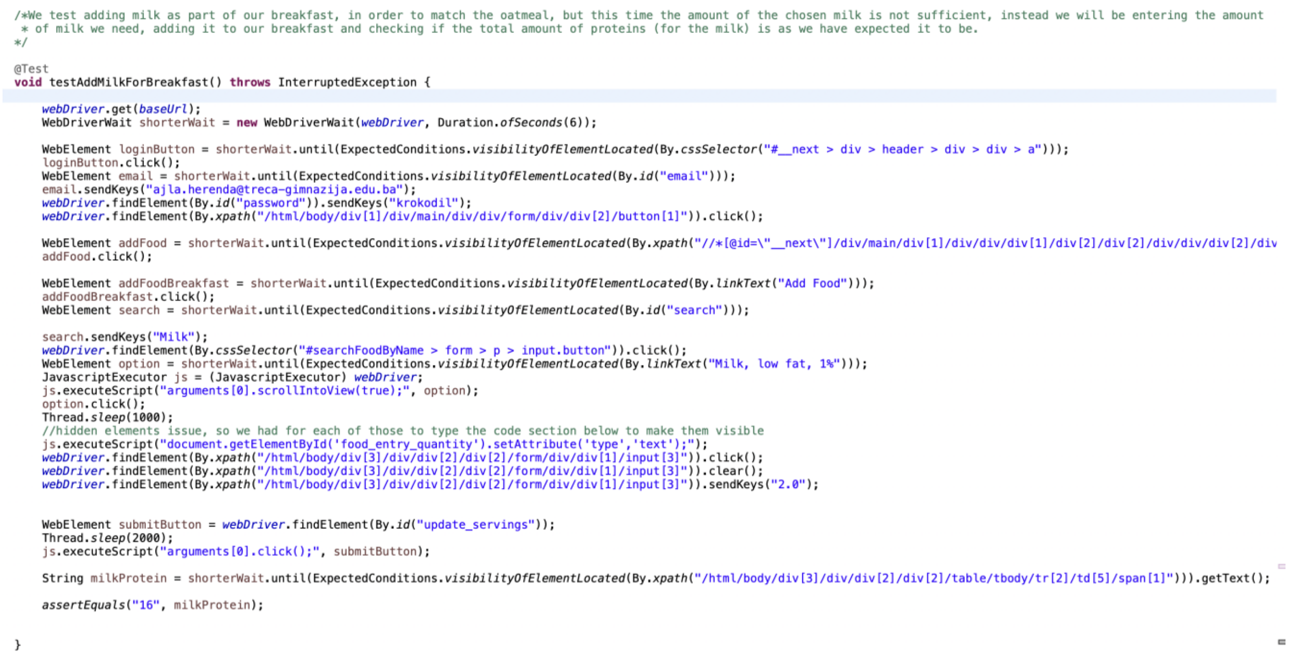
As a user of this website, we are supposed to be able to enter, alter and remove foods from our daily calorie intake tracker, all of which can be a part of any meal in the day, those being breakfast, lunch, dinner and snack.

Test Name: Test Adding Quaker Oats for Breakfast				
Description: We test the add food option for the breakfast meal by attempting to add oatmeal, scrolling into it using js and finally adding it as part of the meal if the oats we have scrolled into and have clicked on to are of the brand Quaker.				
Pre-condition(s): Being logged into an existing account, so we are able to use and test the aforementioned features.				
Test Steps:	**Test Data:**	**Expected Result:**	**Actual Result:**	**Status:**
1. We locate and click on the “Log in” button. 2. After waiting, using the thread.sleep() for waiting, we will be using sendKeys() to fill out the email and password field with test data. 3. Afterwards we click on the “Next” button, and are forwarded to a new page-account's dashboard. 4. There we can find a “Add Food” button, that we will click, causing us to be shifted to a new page, where we will in the breakfast segment click on another “Add Food” button, which will forward us to a new page. 5. On the new page we will in the input field add our test data, click on the search button to find it. 6. Scrolling into the food option is done using js and afterwards we check if those are the right oats, depending on which they will/will not be added.	- We are using real test data for the email and password of our “test account”, which are: ajla.herenda@treca-gimnazija.edu.b - ”krokodil” - ”Oatmeal” - ”Quaker”	The expected result is “Quaker” being contained in the name of our scrolled into oatmeal option.	The actual result indeed contains “Quaker” in its name, so the oats will be added to our breakfast.	PASS
Notes: Shorter wait (explicit), to handle the wait time needed for certain elements to occur.				



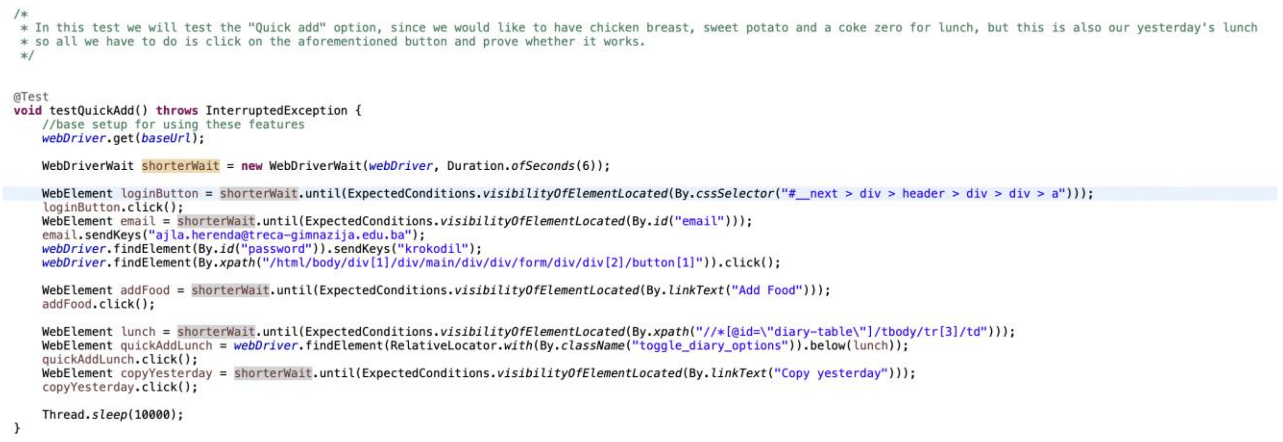
```java
/*We test the add food option for the breakfast meal by attempting to add oatmeal, scrolling into it using js and finally adding it as
 part of the meal if the oats we have scrolled into and have clicked on to are of the brand Quaker.
*/

@Test
void testAddQuakerOatsForBreakfast() throws InterruptedException {


    webDriver.get(baseUrl);
    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));

    WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
    loginButton.click();
    WebElement email = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("email")));
    email.sendKeys("ajla.herenda@treca-gimnazija.edu.ba");
    webDriver.findElement(By.id("password")).sendKeys("krokodil");
    webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div/div/form/div/div[2]/button[1]")).click();

    WebElement addFood = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/div/main/div[1]/div/div/div[1]/div[2]/div[2]/div/div/div[2]/di
    addFood.click();

    WebElement addFoodBreakfast = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Add Food")));
    addFoodBreakfast.click();
    WebElement search = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("search")));
    search.sendKeys("Oatmeal");
    webDriver.findElement(By.cssSelector("#searchFoodByName > form > p > input.button")).click();
    JavascriptExecutor js = (JavascriptExecutor) webDriver;
    WebElement optionsToChooseFrom = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("matching")));
    WebElement option = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Dry Oatmeal")));
    js.executeScript("arguments[0].scrollIntoView(true);", option);
    option.click();
    WebElement oatmealInfo = webDriver.findElement(By.className("food-description"));
    Thread.sleep(1000);
    String oatmealName = oatmealInfo.getText();
    boolean value = oatmealName.contains("Quaker");

    if(value == false) {
    WebElement submitButton = webDriver.findElement(By.id("update_servings"));
    Thread.sleep(2000);
    js.executeScript("arguments[0].click();", submitButton);
    }

}
```

Test Name: Test Add Milk for Breakfast				
Description: We test adding milk as part of our breakfast, in order to match the oatmeal, but this time the amount of the chosen milk is not sufficient, instead we will be entering the amount of milk we need, adding it to our breakfast and checking if the total amount of proteins (for the milk) is as we have expected it to be.				
Pre-condition(s): Being logged into an existing account, so we are able to use and test the aforementioned features.				
Test Steps: 1. We click on the “log in” button. 2. Enter test data, that is needed in order for the login to be successful, after clicking on “log in” again. 3. Furthermore, we click on the “add food” button on the dashboard and again in the breakfast area. 4. Look for milk (enter milk, click “Search” button). 5. Choose/scroll into using the JavascriptExecutor the right milk kind. 6. Adjust the amount of milk. 7. Add the food by clicking on the “add food to diary” button. 8. We assert if the total number of proteins for that amount of milk is as we have expected it to be.	**Test Data:** Similarly, to before, we are using real test data for the email and password of our “test account”, which are: ajla.herenda@treca-gimnazija.edu.b - ”krokodil” In addition, we have: -Milk	**Expected Result:** The total number of proteins found in that particular amount of milk to be 16.	**Actual Result:** The assertion returns true, since the expected and actual result match.	**Status:** PASS
Notes: Explicit wait/Thread.sleep used for managing the visibility of particular elements. We had also to undergo the process of “unhiding” hidden elements using the JavascriptExecutor and clicking “input buttons” using the aforementioned, due to the standard submit ()/ click () not working.				



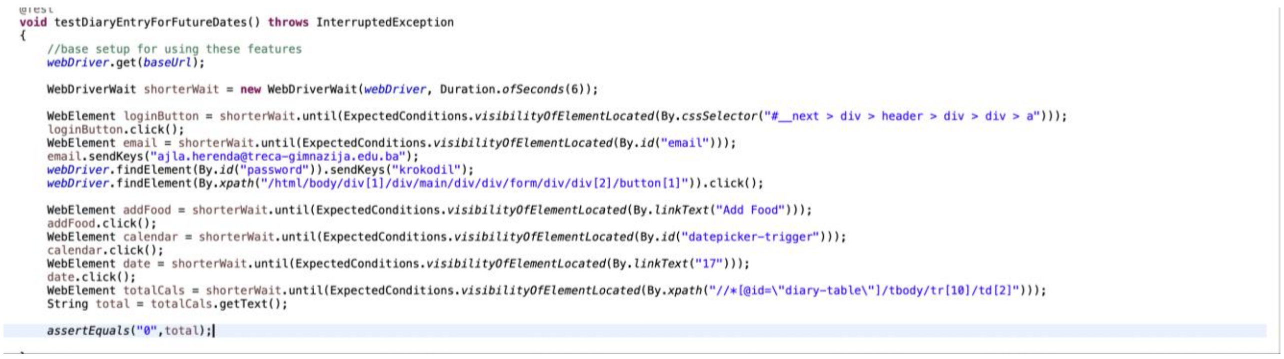
```java
/*We test adding milk as part of our breakfast, in order to match the oatmeal, but this time the amount of the chosen milk is not sufficient, instead we will be entering the amount
 * of milk we need, adding it to our breakfast and checking if the total amount of proteins (for the milk) is as we have expected it to be.
*/

@Test
void testAddMilkForBreakfast() throws InterruptedException {

    webDriver.get(baseUrl);
    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));

    WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
    loginButton.click();
    WebElement email = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("email")));
    email.sendKeys("ajla.herenda@treca-gimnazija.edu.ba");
    webDriver.findElement(By.id("password")).sendKeys("krokodil");
    webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div/div/form/div/div[2]/button[1]")).click();

    WebElement addFood = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/div/main/div[1]/div/div/div[1]/div[2]/div[2]/div/div/div[2]/div
    addFood.click();

    WebElement addFoodBreakfast = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Add Food")));
    addFoodBreakfast.click();
    WebElement search = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("search")));

    search.sendKeys("Milk");
    webDriver.findElement(By.cssSelector("#searchFoodByName > form > p > input.button")).click();
    WebElement option = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Milk, low fat, 1%")));
    JavascriptExecutor js = (JavascriptExecutor) webDriver;
    js.executeScript("arguments[0].scrollIntoView(true);", option);
    option.click();
    Thread.sleep(1000);
    //hidden elements issue, so we had for each of those to type the code section below to make them visible
    js.executeScript("document.getElementById('food_entry_quantity').setAttribute('type','text');");
    webDriver.findElement(By.xpath("/html/body/div[3]/div/div[2]/div[2]/form/div/div[1]/input[3]")).click();
    webDriver.findElement(By.xpath("/html/body/div[3]/div/div[2]/div[2]/form/div/div[1]/input[3]")).clear();
    webDriver.findElement(By.xpath("/html/body/div[3]/div/div[2]/div[2]/form/div/div[1]/input[3]")).sendKeys("2.0");


    WebElement submitButton = webDriver.findElement(By.id("update_servings"));
    Thread.sleep(2000);
    js.executeScript("arguments[0].click();", submitButton);

    String milkProtein = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[3]/div/div[2]/div[2]/table/tbody/tr[2]/td[5]/span[1]"))).getText();

    assertEquals("16", milkProtein);


}
```

Test Name: Test Quick Tools				
Description: In this test we will test the "Quick Tools" option, since we would like to have chicken breast, sweet potato and a coke zero for lunch, but this is also our yesterday's lunch so all we have to do is click on the aforementioned button and prove whether it works.				
Pre-condition(s): Being logged into an existing account, so we are able to use and test the aforementioned features (includes having a logged lunch meal).				
Test Steps: 1. We click on the “Log in” button. 2. Enter test data, that is needed in order for the login to be successful, after clicking on “Log in” again. 3. Furthermore, we click on the “Add Food” button that can be seen on the dashboard. 4. Under the lunch area, next we will click on the “Quick Tools” button. 5. Consequently we are shown a range of links. 6. We click on the element “Copy yesterday”. 7. Finally we copied our yesterday’s lunch for today.	**Test Data:** Like before, we are using real test data for the email and password of our “test account”, which are: ajla.herenda@treca-gimnazija.edu.b - ”krokodil” No additional test data is required.	**Expected Result:** Our yesterday’s lunch is copied for today and a notification is displayed.	**Actual Result:** The actual result matches the expected one, in addition the notification states:” Meal copied.”	**Status:** PASS
Notes: Explicit wait/Thread.sleep used for managing the visibility of particular elements/being able to see what is going on.				



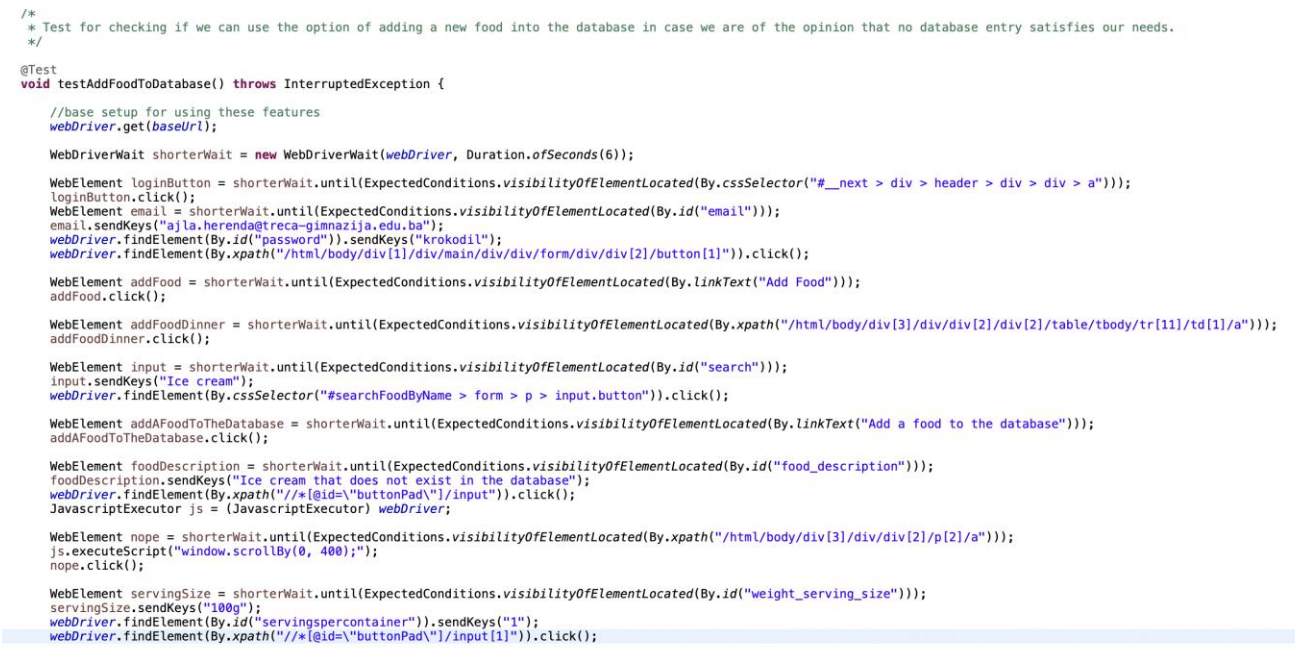
```java
/*
 * In this test we will test the "Quick add" option, since we would like to have chicken breast, sweet potato and a coke zero for lunch, but this is also our yesterday's lunch
 * so all we have to do is click on the aforementioned button and prove whether it works.
 */


@Test
void testQuickAdd() throws InterruptedException {
    //base setup for using these features
    webDriver.get(baseUrl);

    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));

    WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
    loginButton.click();
    WebElement email = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("email")));
    email.sendKeys("ajla.herenda@treca-gimnazija.edu.ba");
    webDriver.findElement(By.id("password")).sendKeys("krokodil");
    webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div/div/form/div/div[2]/button[1]")).click();

    WebElement addFood = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Add Food")));
    addFood.click();

    WebElement lunch = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"diary-table\"]/tbody/tr[3]/td")));
    WebElement quickAddLunch = webDriver.findElement(RelativeLocator.with(By.className("toggle_diary_options")).below(lunch));
    quickAddLunch.click();
    WebElement copyYesterday = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Copy yesterday")));
    copyYesterday.click();

    Thread.sleep(10000);
}
```

Test Name: Test Food Diary Calendar				
Description: Test for writing into our food diary, for a date that has not happened yet, by exercising the calendar option. In order to prove that we have been moved to the 17th of January, we will assert the total number of calories on that day.				
Pre-condition(s): Being logged into an existing account, so we are able to use and test the aforementioned features.				
Test Steps: 1. Once again we repeat the few staring steps by clicking on the “Log in” button. 2. Entering test data, that is needed in order for the login to be successful, afterwards clicking on “Log in” again. 3. Furthermore, we click on the “Add Food” button that can be seen on the dashboard. 4. We locate the calendar icon and click on it. 5. Consequently we are shown a mini calendar 6. We click on the element 17th of January. 7. We are redirected to the diary for the 17th of January, where we assert the total number of calories.	**Test Data:** Like before, we are using real test data for the email and password of our “test account”, which are: ajla.herenda@treca-gimnazija.edu.b - ”krokodil” -As date we choose the 17th of January.	**Expected Result:** The total number of calories for the 17th of January is equal to 0.	**Actual Result:** The actual result matches the expected, since there were no entries in the food diary for a day that did not occur yet.	**Status:** PASS
Notes: Explicit wait (shorterWait) used for managing the visibility of particular elements.				



```
@Test
void testDiaryEntryForFutureDates() throws InterruptedException
{
    //base setup for using these features
    webDriver.get(baseUrl);

    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));

    WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
    loginButton.click();
    WebElement email = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("email")));
    email.sendKeys("ajla.herenda@treca-gimnazija.edu.ba");
    webDriver.findElement(By.id("password")).sendKeys("krokodil");
    webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div/div/form/div/div[2]/button[1]")).click();

    WebElement addFood = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Add Food")));
    addFood.click();
    WebElement calendar = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("datepicker-trigger")));
    calendar.click();
    WebElement date = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("17")));
    date.click();
    WebElement totalCals = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"diary-table\"]/tbody/tr[10]/td[2]")));
    String total = totalCals.getText();

    assertEquals("0",total);
```

Test Name: Test Add Food to the Database				
Description: Test for checking if we can use the option of adding a new food into the database in case, we are of the opinion that no database entry satisfies our needs.				
Pre-condition(s): Being logged into an existing account/logging in as part of the test steps, which we will avoid to be addressed in detail in the test steps in order to at least avoid some repetitiveness, so we are able to use and test the aforementioned feature.				
Test Steps: 1. We kick start the test by clicking on the "Log in" button. 2. Entering test data, that is needed in order for the login to be successful, afterwards clicking on "Log in" again. 3. Furthermore, we click on the "Add Food" button that can be seen on the dashboard. 4. We locate the "Add Food" button in the dinner segment. 5. Afterwards we are redirected to a page, where we are entering our test data and searching for it. 7. Since the results of the search do not satisfy our needs, we click on the "Add Food to the Database" button, which redirects us to a page where we are once again assured to that there are multiple options. 8. Next, we click on the "Nope" button, which forwards us to a page where we enter the remaining test data and click on the "Save" button.	**Test Data:** Like before, we are using real test data for the email and password of our "test account", which are: ajla.herenda@treca-gimnazija.edu.b - "krokodil" For additional test data we have: -" Ice cream" -" Ice cream that does not exist in the database" -"100g" -"1"	**Expected Result:** The new food item is added to the database and to our dinner.	**Actual Result:** The actual result matches the expected, meaning that we are brought back to the food diary page, where we can see that our food has been entered as a dinner food.	**Status:** PASS
Notes: Explicit wait instance for assuring visibility of elements.				

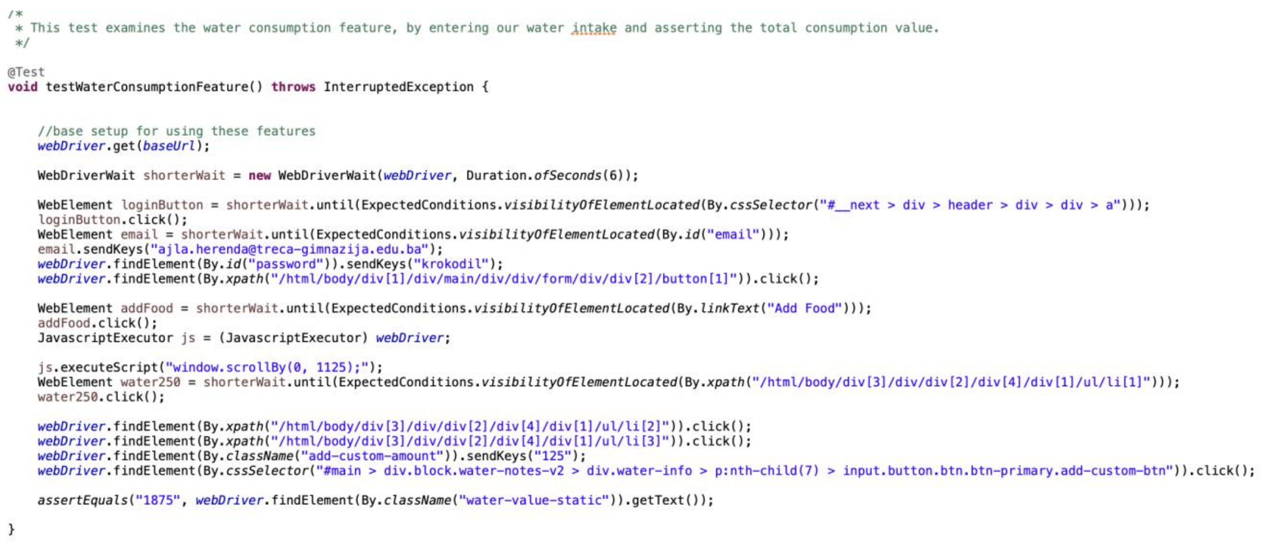


```java
/*
 * Test for checking if we can use the option of adding a new food into the database in case we are of the opinion that no database entry satisfies our needs.
 */

@Test
void testAddFoodToDatabase() throws InterruptedException {

    //base setup for using these features
    webDriver.get(baseUrl);

    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));

    WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
    loginButton.click();
    WebElement email = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("email")));
    email.sendKeys("ajla.herenda@treca-gimnazija.edu.ba");
    webDriver.findElement(By.id("password")).sendKeys("krokodil");
    webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div/div/form/div/div[2]/button[1]")).click();

    WebElement addFood = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Add Food")));
    addFood.click();

    WebElement addFoodDinner = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[3]/div/div[2]/div[2]/table/tbody/tr[11]/td[1]/a")));
    addFoodDinner.click();

    WebElement input = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("search")));
    input.sendKeys("Ice cream");
    webDriver.findElement(By.cssSelector("#searchFoodByName > form > p > input.button")).click();

    WebElement addAFoodToTheDatabase = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Add a food to the database")));
    addAFoodToTheDatabase.click();

    WebElement foodDescription = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("food_description")));
    foodDescription.sendKeys("Ice cream that does not exist in the database");
    webDriver.findElement(By.xpath("//*[@id=\"buttonPad\"]/input")).click();
    JavascriptExecutor js = (JavascriptExecutor) webDriver;

    WebElement nope = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[3]/div/div[2]/p[2]/a")));
    js.executeScript("window.scrollBy(0, 400);");
    nope.click();

    WebElement servingSize = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("weight_serving_size")));
    servingSize.sendKeys("100g");
    webDriver.findElement(By.id("servingspercontainer")).sendKeys("1");
    webDriver.findElement(By.xpath("//*[@id=\"buttonPad\"]/input[1]")).click();
```

3.3. Using Features Related to the Daily Water Intake

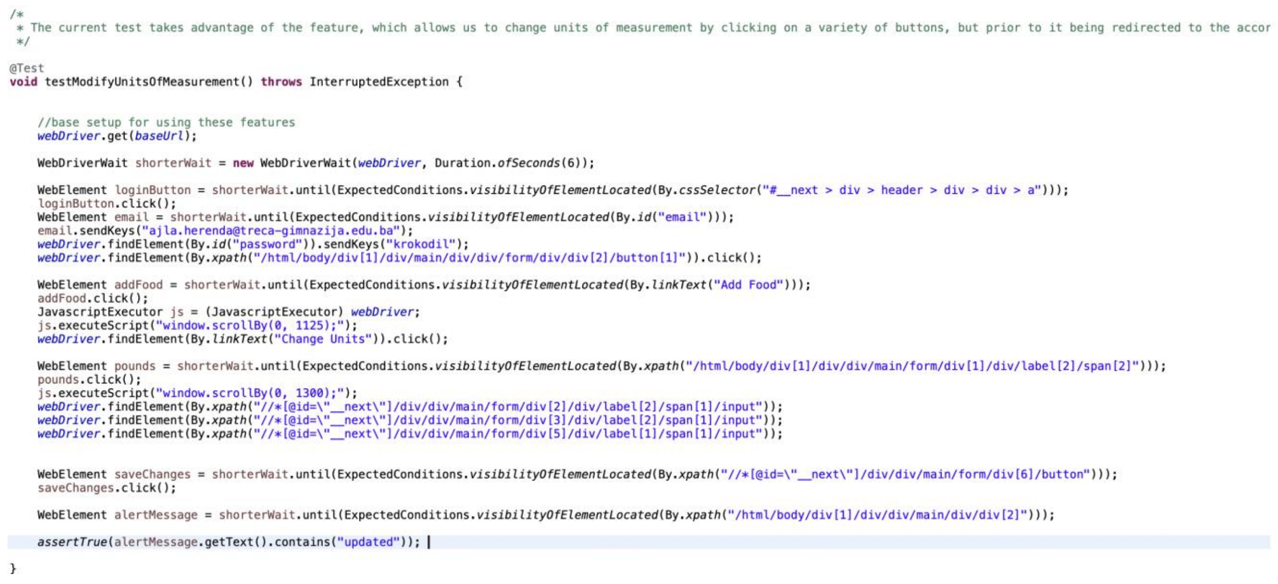
Users are given the option to add, remove and track their daily water intake. Water can be added by clicking on previously generated amounts or by creating custom amounts and adding them. Alongside the aforementioned, this segment of page can also redirect us to a new page where an alteration of all units can be made.

Test Name: Test Water Consumption Feature				
Description: This test examines the water consumption feature, by entering our water intake and asserting the total consumption value.				
Pre-condition(s): Being logged into an existing account/as we do it here => logging in as part of the test, so we are able to use and test the aforementioned feature.				
Test Steps: 1. We kick start the test by clicking on the "Log in" button. 2. Entering test data, that is needed in order for the login to be successful, afterwards clicking on "Log in" again. 3. Furthermore, we click on the "Add Food" button that can be seen on the dashboard. 4. We scroll down using the JSExecutor to the water intake segment. 5. Next, we click on the links for adding 250, 500 and 1000ml of water. 7. Additionally, we create a custom amount of water (test data) intake we log, by clicking on the "Add" button. 8. Finally, we assert the final water intake value.	**Test Data:** Like before, we are using real test data for the email and password of our "test account", which are: ajla.herenda@treca-gimnazija.edu.b - "krokodil" For additional test data we have: - "125" ml for adding custom amount of water.	**Expected Result:** The total water intake is equal to 1875 ml.	**Actual Result:** The actual result matches the expected water intake value.	**Status:** PASS
Notes: Explicit wait, which is in the task named shorterWait.				



```java
/*
 * This test examines the water consumption feature, by entering our water intake and asserting the total consumption value.
 */

@Test
void testWaterConsumptionFeature() throws InterruptedException {


    //base setup for using these features
    webDriver.get(baseUrl);

    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));

    WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
    loginButton.click();
    WebElement email = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("email")));
    email.sendKeys("ajla.herenda@treca-gimnazija.edu.ba");
    webDriver.findElement(By.id("password")).sendKeys("krokodil");
    webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div/div/form/div/div[2]/button[1]")).click();

    WebElement addFood = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Add Food")));
    addFood.click();
    JavascriptExecutor js = (JavascriptExecutor) webDriver;

    js.executeScript("window.scrollBy(0, 1125);");
    WebElement water250 = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[3]/div/div[2]/div[4]/div[1]/ul/li[1]")));
    water250.click();

    webDriver.findElement(By.xpath("/html/body/div[3]/div/div[2]/div[4]/div[1]/ul/li[2]")).click();
    webDriver.findElement(By.xpath("/html/body/div[3]/div/div[2]/div[4]/div[1]/ul/li[3]")).click();
    webDriver.findElement(By.className("add-custom-amount")).sendKeys("125");
    webDriver.findElement(By.cssSelector("#main > div.block.water-notes-v2 > div.water-info > p:nth-child(7) > input.button.btn.btn-primary.add-custom-btn")).click();

    assertEquals("1875", webDriver.findElement(By.className("water-value-static")).getText());

}
```

Test Name: Test Modify Units of Measurement				
Description: The current test takes advantage of the feature, which allows us to change units of measurement by clicking on a variety of buttons, but prior to it being redirected to the according page.				
Pre-condition(s): Being logged into an existing account/logging in as part of the test, so we are able to use and test the aforementioned feature.				
Test Steps: 1. We kick start the test by clicking on the "Log in" button. 2. Entering test data, that is needed in order for the login to be successful, afterwards clicking on "Log in" again. 3. Furthermore, we click on the "Add Food" button that can be seen on the dashboard. 4. We scroll down using the JSExecutor to the water intake segment. 5. Here, we click on the "Change units of measurement" link. 7. It redirects us to a page where we are offered a number of radio buttons for each data type, where we individually select particular units of measurement (test data). 8. Afterwards, the "Save changes" button is made visible, so we click on it. 9. The alert message is what we assert in the last step.	**Test Data:** Like before, we are using real test data for the email and password of our "test account", which are: ajla.herenda@treca-gimnazija.edu.b - "krokodil" For additional test data we have: - "Pounds" for weight - "Inches/Feet" for height - "Miles" for distance - "Cups" for liquids	**Expected Result:** The alert message shown on the screen contains the word "update".	**Actual Result:** The actual result matches the expected, since the message states: "Your unit preferences have been updated.".	**Status:** PASS
Notes: Explicit wait, which is in the task named shorterWait.				

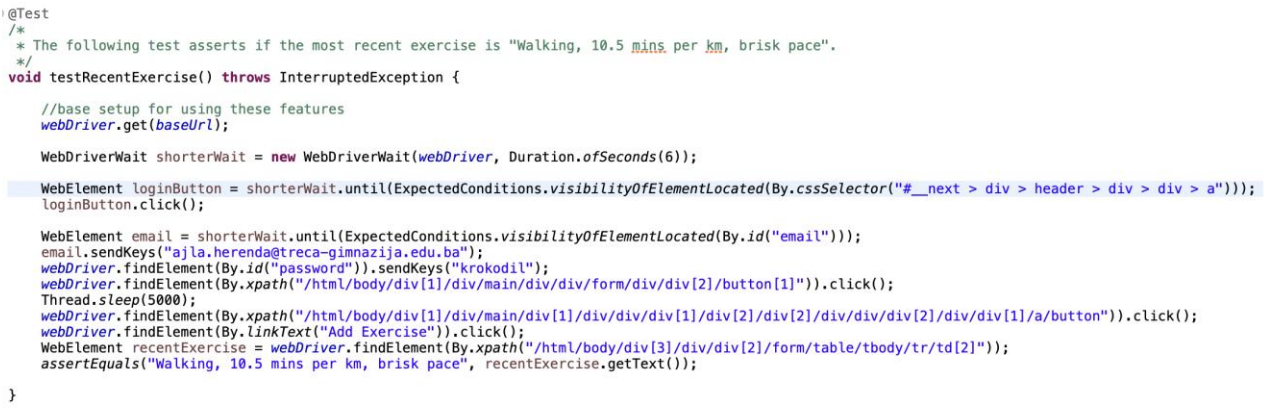


```java
/*
 * The current test takes advantage of the feature, which allows us to change units of measurement by clicking on a variety of buttons, but prior to it being redirected to the accor
 */

@Test
void testModifyUnitsOfMeasurement() throws InterruptedException {


    //base setup for using these features
    webDriver.get(baseUrl);

    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));

    WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
    loginButton.click();
    WebElement email = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("email")));
    email.sendKeys("ajla.herenda@treca-gimnazija.edu.ba");
    webDriver.findElement(By.id("password")).sendKeys("krokodil");
    webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div/div/form/div/div[2]/button[1]")).click();

    WebElement addFood = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.linkText("Add Food")));
    addFood.click();
    JavascriptExecutor js = (JavascriptExecutor) webDriver;
    js.executeScript("window.scrollBy(0, 1125);");
    webDriver.findElement(By.linkText("Change Units")).click();

    WebElement pounds = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[1]/div/div/main/form/div[1]/div/label[2]/span[2]")));
    pounds.click();
    js.executeScript("window.scrollBy(0, 1300);");
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/div/div/main/form/div[2]/div/label[2]/span[1]/input"));
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/div/div/main/form/div[3]/div/label[2]/span[1]/input"));
    webDriver.findElement(By.xpath("//*[@id=\"__next\"]/div/div/main/form/div[5]/div/label[1]/span[1]/input"));


    WebElement saveChanges = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/div/div/main/form/div[6]/button")));
    saveChanges.click();

    WebElement alertMessage = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[1]/div/div/main/div/div[2]")));

    assertTrue(alertMessage.getText().contains("updated"));

}
```

3.4 Using Features Related to Exercise

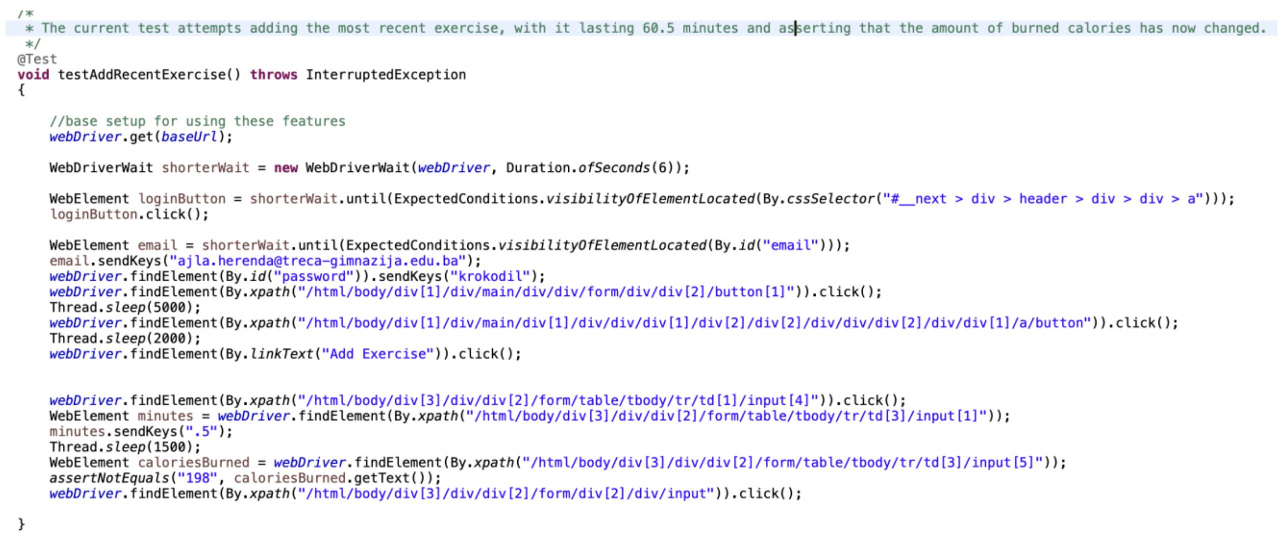
MyFitnessPal offers alongside features used for tracking the daily food and water intake, also the exercise level. As part of this segment, we will be testing the aforementioned and providing additional information for the use cases if needed.

Test Name: Test Recent Exercise				
Description: The following test asserts if the most recent exercise is "Walking, 10.5 mins per km, brisk pace".				
Pre-condition(s): Being logged into an existing account.				
Test Steps: 1. First things first, we open our baseUrl. 2. The few next steps are already known and relate to the login process, with the provided test data. 3. Being forwarded to the new page, we will click on the "Add Exercise" button. 4. On the next page we will click on the "Add Exercise" link and get channeled to a new page. 5. Finally we assert that the field, that contains the most recent exercises, on the new page states the expected.	**Test Data:** As it is always the case when we are testing features exclusive to being logged into an account, we have to be using real test data for the email and password of our "test account", which are: ajla.herenda@treca-gimnazija.edu.b - "krokodil"	**Expected Result:** The observed field, says: "Walking, 10.5 mins per km, brisk pace".	**Actual Result:** The result is such that it matches the expected one.	**Status:** PASS
Notes: Thread.sleep and shorterWait used for managing the visibility of particular elements/being able to see what is going on.				



```
@Test
/*
 * The following test asserts if the most recent exercise is "Walking, 10.5 mins per km, brisk pace".
 */
void testRecentExercise() throws InterruptedException {

    //base setup for using these features
    webDriver.get(baseUrl);

    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));

    WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
    loginButton.click();

    WebElement email = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("email")));
    email.sendKeys("ajla.herenda@treca-gimnazija.edu.ba");
    webDriver.findElement(By.id("password")).sendKeys("krokodil");
    webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div/div/form/div/div[2]/button[1]")).click();
    Thread.sleep(5000);
    webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div[1]/div/div/div[1]/div[2]/div[2]/div/div/div[2]/div/div[1]/a/button")).click();
    webDriver.findElement(By.linkText("Add Exercise")).click();
    WebElement recentExercise = webDriver.findElement(By.xpath("/html/body/div[3]/div/div[2]/form/table/tbody/tr/td[2]"));
    assertEquals("Walking, 10.5 mins per km, brisk pace", recentExercise.getText());

}
```

Test Name: Test Add Recent Exercise				
Description: The current test attempts adding the most recent exercise, with it lasting 60.5 minutes and asserting that the amount of burned calories has now changed/is not 198.				
Pre-condition(s): Being logged into an existing account.				
Test Steps: 1. First things first, we open our baseUrl. 2. The few next steps are already known and relate to the login process, with the provided test data. 3. Being forwarded to the new page, we will click on the "Add Exercise" button. 4. On the next page we will click on the "Add Exercise" link and get channeled to a new page. 5. Here we can find our most recent form of exercise, which we will click on and whose longevity we will change, this way impacting the total number of calories burned for that exercise and test data period. 6. Lastly, we will assert that the number of calories burned has changed and differs from the expected value, using assertNotEquals.	**Test Data:** Being familiar with the fact that these are features exclusive to being logged into an account, we have to be using real test data for the email and password of our "test account", which are: ajla.herenda@treca-gimnazija.edu.b - "krokodil" Additionally, we are entering "0.5" to an input field to change its value.	**Expected Result:** The calories field does not state what it used to (198).	**Actual Result:** Test execution shows the expected result as the actual result.	**Status:** PASS
Notes: Thread.sleep and shorterWait used for managing the visibility of particular elements/being able to see what is going on.				

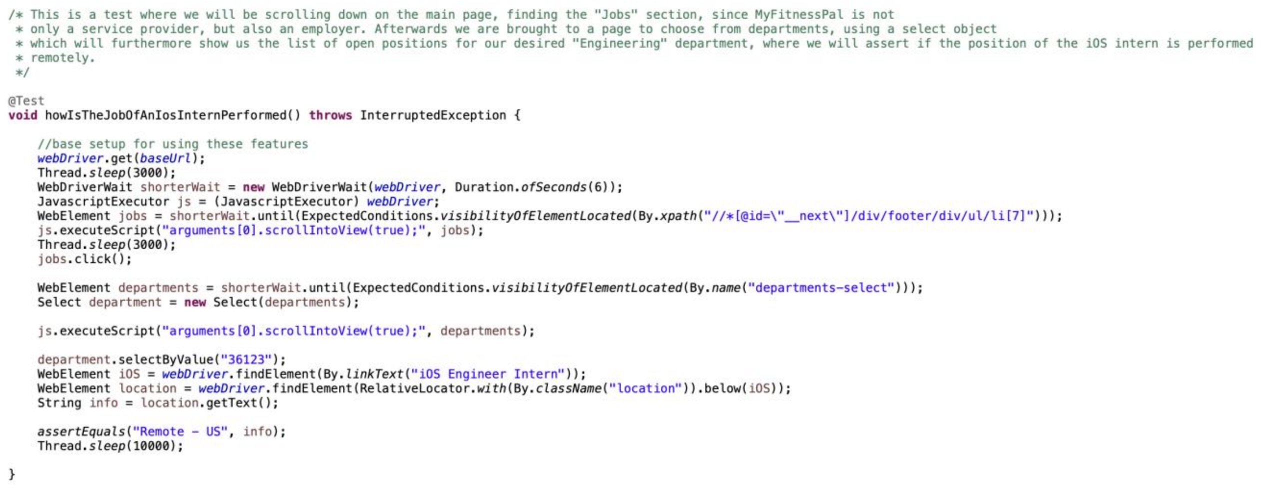


```java
/*
 * The current test attempts adding the most recent exercise, with it lasting 60.5 minutes and asserting that the amount of burned calories has now changed.
 */
@Test
void testAddRecentExercise() throws InterruptedException
{

    //base setup for using these features
    webDriver.get(baseUrl);

    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));

    WebElement loginButton = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.cssSelector("#__next > div > header > div > div > a")));
    loginButton.click();

    WebElement email = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.id("email")));
    email.sendKeys("ajla.herenda@treca-gimnazija.edu.ba");
    webDriver.findElement(By.id("password")).sendKeys("krokodil");
    webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div/div/form/div/div[2]/button[1]")).click();
    Thread.sleep(5000);
    webDriver.findElement(By.xpath("/html/body/div[1]/div/main/div[1]/div/div/div[1]/div[2]/div[2]/div/div/div[2]/div/div[1]/a/button")).click();
    Thread.sleep(2000);
    webDriver.findElement(By.linkText("Add Exercise")).click();


    webDriver.findElement(By.xpath("/html/body/div[3]/div/div[2]/form/table/tbody/tr/td[1]/input[4]")).click();
    WebElement minutes = webDriver.findElement(By.xpath("/html/body/div[3]/div/div[2]/form/table/tbody/tr/td[3]/input[1]"));
    minutes.sendKeys(".5");
    Thread.sleep(1500);
    WebElement caloriesBurned = webDriver.findElement(By.xpath("/html/body/div[3]/div/div[2]/form/table/tbody/tr/td[3]/input[5]"));
    assertNotEquals("198", caloriesBurned.getText());
    webDriver.findElement(By.xpath("/html/body/div[3]/div/div[2]/form/div[2]/div/input")).click();

}
```

3.5. Using Features for Becoming Not Only a User, but Also an Employee

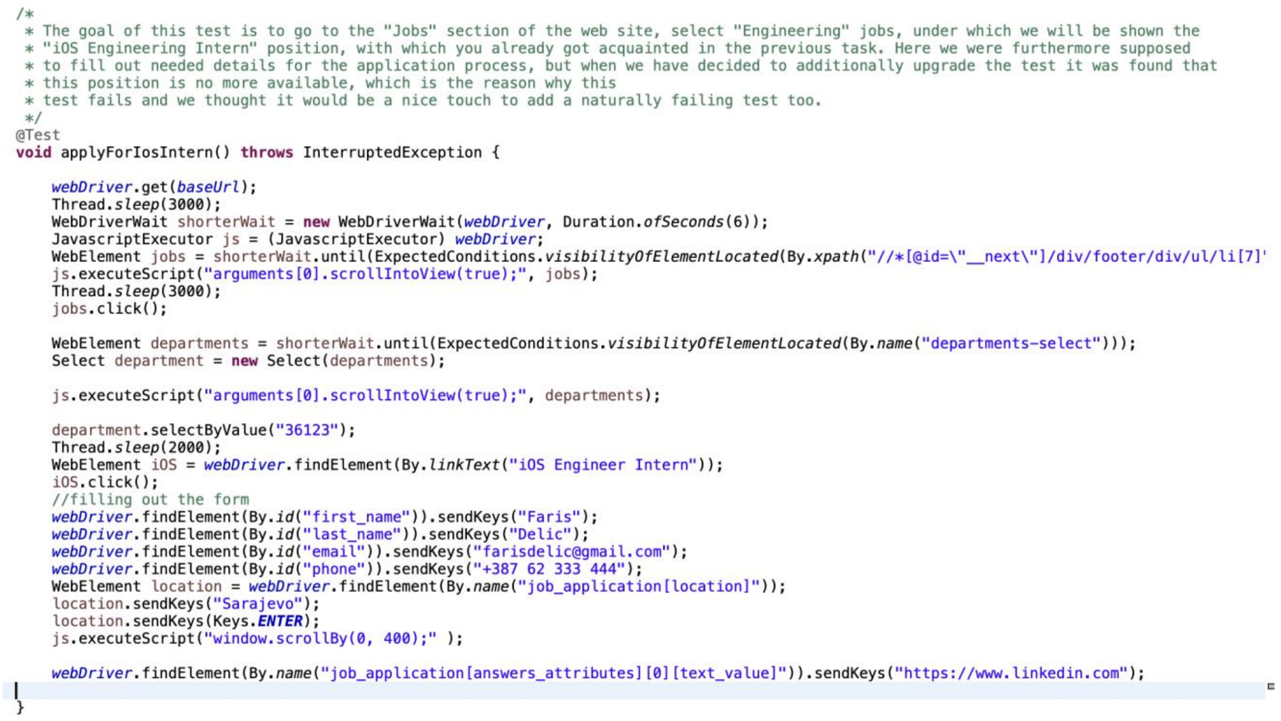
MyFitnessPal is not only a service provider, but also a job provider, meaning that as a company they offer job and partnership opportunities. Additional information in case of interest for those segments of the website can be found under appropriately named sections, which we will be testing in the section below.

Test Name: How is the Job of an iOS Intern Performed				
Description: This is a test where we will be scrolling down on the main page, finding the "Jobs" section, since MyFitnessPal is not only a service provider, but also an employer. Afterwards we are brought to a page to choose from departments, using a select object, which will furthermore show us the list of open positions for our desired "Engineering" department, where we will assert if the position of the iOS intern is performed remotely.				
Pre-condition(s): None				
Test Steps: 1. Initially, we open the MyFitnessPal website and scroll down the whole page, where we find the “Jobs” segment. 2. We click on “Jobs”. 3. After being forwarded to the “Jobs” page, here we access a dropdown of departments, using a select object. 4. We select the “Engineering” department. 5. Consequently, we are shown several job positions. 6. We assert what the text below the “iOS Engineer Intern” position says.	**Test Data:** Test data here conforms to us selecting the “Enegineering” department and “iOS Engineer Intern” as our objects of observation.	**Expected Result:** The text below the observed test data says “Remote - US”.	**Actual Result:** Indeed, the actual result matches the expected one.	**Status:** PASS
Notes: Explicit wait/Thread.sleep used for managing the visibility of particular elements/being able to see what is going on.				



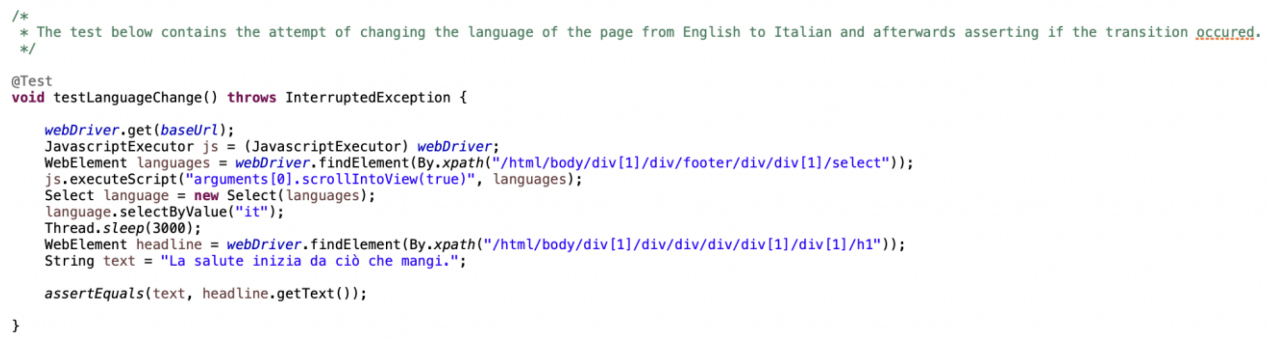
```java
/* This is a test where we will be scrolling down on the main page, finding the "Jobs" section, since MyFitnessPal is not
 * only a service provider, but also an employer. Afterwards we are brought to a page to choose from departments, using a select object
 * which will furthermore show us the list of open positions for our desired "Engineering" department, where we will assert if the position of the iOS intern is performed
 * remotely.
 */

@Test
void howIsTheJobOfAnIosInternPerformed() throws InterruptedException {

    //base setup for using these features
    webDriver.get(baseUrl);
    Thread.sleep(3000);
    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));
    JavascriptExecutor js = (JavascriptExecutor) webDriver;
    WebElement jobs = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/div/footer/div/ul/li[7]")));
    js.executeScript("arguments[0].scrollIntoView(true);", jobs);
    Thread.sleep(3000);
    jobs.click();

    WebElement departments = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.name("departments-select")));
    Select department = new Select(departments);

    js.executeScript("arguments[0].scrollIntoView(true);", departments);

    department.selectByValue("36123");
    WebElement iOS = webDriver.findElement(By.linkText("iOS Engineer Intern"));
    WebElement location = webDriver.findElement(RelativeLocator.with(By.className("location")).below(iOS));
    String info = location.getText();

    assertEquals("Remote - US", info);
    Thread.sleep(10000);

}
```

Test Name: Apply for the Job of an iOS Intern				
Description: The goal of this test is to go to the "Jobs" section of the website, select "Engineering" jobs, under which we will be shown the "iOS Engineering Intern" position, with which you already got acquainted in the previous task. Here we were furthermore supposed to fill out needed details for the application process, but when we have decided to additionally upgrade the test it was found that this position is no more available, which is the reason why this test fails and we thought it would be a nice touch to add a naturally failing test too.				
Pre-condition(s): None				
Test Steps: 1. Initially, we open the MyFitnessPal website and scroll down the whole page, where we find the "Jobs" segment. 2. We click on "Jobs". 3. After being forwarded to the "Jobs" page, here we access a dropdown of departments, using a select object. 4. We select the "Engineering" department. 5. Consequently, we are shown several job positions. 6. We attempt to click on the "iOS Engineering Intern" position. 7. On the next page we click on the "Apply" button, which leads us to a series of field, we attempt to populate using our test data.	**Test Data:** Test data here conforms to us selecting the "Enegineering" department and "iOS Engineer Intern" as our objects of observation.	**Expected Result:** The field we have intended to populate using the test data, being filled.	**Actual Result:** The test fails at the attempt of matching the actual result with the expected result, due to this position not being anymore available.	**Status:** PASS
Notes: Explicit wait/Thread.sleep used for managing the visibility of particular elements/being able to see what is going on.				



```java
/*
 * The goal of this test is to go to the "Jobs" section of the web site, select "Engineering" jobs, under which we will be shown the
 * "iOS Engineering Intern" position, with which you already got acquainted in the previous task. Here we were furthermore supposed
 * to fill out needed details for the application process, but when we have decided to additionally upgrade the test it was found that
 * this position is no more available, which is the reason why this
 * test fails and we thought it would be a nice touch to add a naturally failing test too.
 */
@Test
void applyForIosIntern() throws InterruptedException {

    webDriver.get(baseUrl);
    Thread.sleep(3000);
    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));
    JavascriptExecutor js = (JavascriptExecutor) webDriver;
    WebElement jobs = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//*[@id=\"__next\"]/div/footer/div/ul/li[7]'
    js.executeScript("arguments[0].scrollIntoView(true);", jobs);
    Thread.sleep(3000);
    jobs.click();

    WebElement departments = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.name("departments-select")));
    Select department = new Select(departments);

    js.executeScript("arguments[0].scrollIntoView(true);", departments);

    department.selectByValue("36123");
    Thread.sleep(2000);
    WebElement iOS = webDriver.findElement(By.linkText("iOS Engineer Intern"));
    iOS.click();
    //filling out the form
    webDriver.findElement(By.id("first_name")).sendKeys("Faris");
    webDriver.findElement(By.id("last_name")).sendKeys("Delic");
    webDriver.findElement(By.id("email")).sendKeys("farisdelic@gmail.com");
    webDriver.findElement(By.id("phone")).sendKeys("+387 62 333 444");
    WebElement location = webDriver.findElement(By.name("job_application[location]"));
    location.sendKeys("Sarajevo");
    location.sendKeys(Keys.ENTER);
    js.executeScript("window.scrollBy(0, 400);" );

    webDriver.findElement(By.name("job_application[answers_attributes][0][text_value]")).sendKeys("https://www.linkedin.com");

}
```

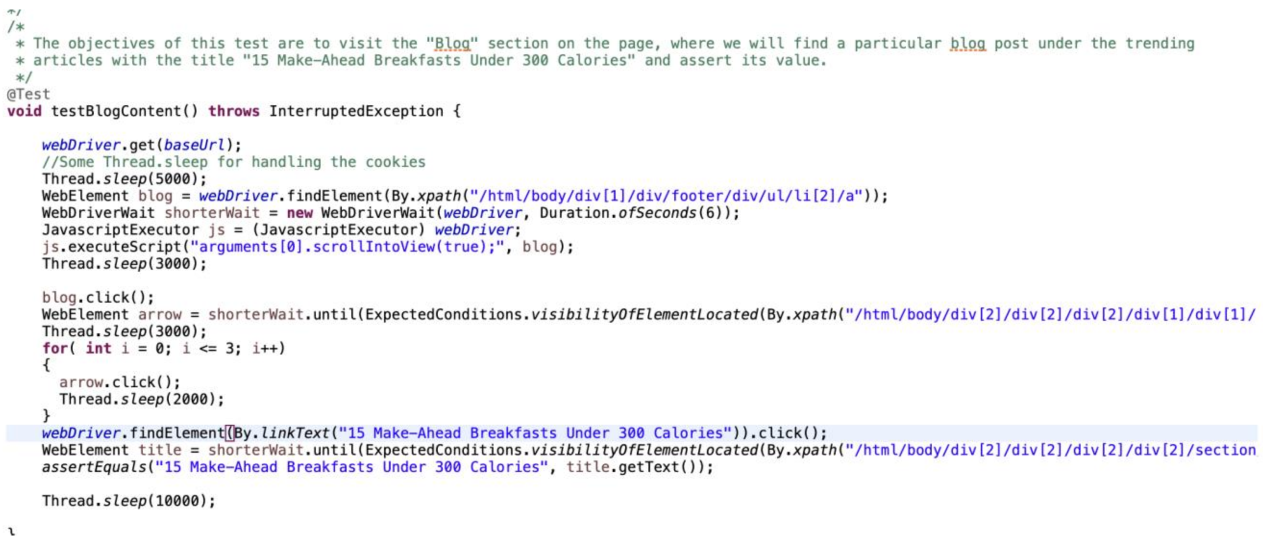
Test Name: Test the Language Change				
Description: The test below contains the attempt of changing the language of the page from English to Italian and afterwards asserting if the transition has occurred.				
Pre-condition(s): None				
Test Steps: 1. First things first, we open our baseUrl. 2. The next step is to scroll down the page to our drop down, which contains a variety of languages. 3. We access the desired test data option by using a select object. 4. Afterwards we assert if the change of language of the page has occurred.	**Test Data:** The selected language, “Italian”, can be seen as a form of test data.	**Expected Result:** The observed title, says now: “La salute inizia da ciò che mangi.”;	**Actual Result:** The result is such that it matches the expected one.	**Status:** PASS
Notes: Thread.sleep used for managing the visibility of particular elements/being able to see what is going on.				



```
/*
 * The test below contains the attempt of changing the language of the page from English to Italian and afterwards asserting if the transition occured.
 */

@Test
void testLanguageChange() throws InterruptedException {

    webDriver.get(baseUrl);
    JavascriptExecutor js = (JavascriptExecutor) webDriver;
    WebElement languages = webDriver.findElement(By.xpath("/html/body/div[1]/div/footer/div/div[1]/select"));
    js.executeScript("arguments[0].scrollIntoView(true)", languages);
    Select language = new Select(languages);
    language.selectByValue("it");
    Thread.sleep(3000);
    WebElement headline = webDriver.findElement(By.xpath("/html/body/div[1]/div/div/div/div[1]/div[1]/h1"));
    String text = "La salute inizia da ciò che mangi.";

    assertEquals(text, headline.getText());

}
```

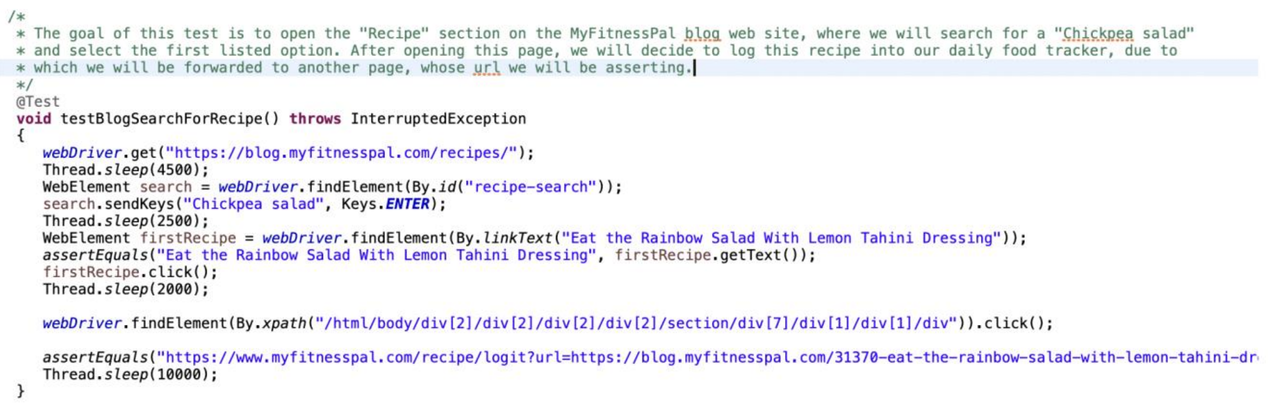
3.6. Using Features Related to the Blog and Its Contents

Test Name: Test Blog Content				
Description: The objectives of this test are to visit the "Blog" section on the page, where we will find a particular blog post under the trending articles with the title "15 Make-Ahead Breakfasts Under 300 Calories" and assert its value.				
Pre-condition(s): None				
Test Steps: 1. First things first, we open our baseUrl. 2. The next step is to scroll down the page to our drop down, which contains the "Blog" section. 3. We click on it and get forwarded to the MyFitnessPal blog. 4. Here we will search for our test data post, by clicking on arrow objects and after successfully navigating to it, we will open it. 5. Lastly, we assert positively the title of the blog post.	**Test Data:** In this case the test data can be seen as the blog post of our choice: "15 Make-Ahead Breakfasts Under 300 Calories".	**Expected Result:** The blog post title is: "15 Make-Ahead Breakfasts Under 300 Calories".	**Actual Result:** The test behaves like expected.	**Status:** PASS
Notes: Thread.sleep and shorterWait used for managing the visibility of particular elements/being able to see what is going on.				



```java
*/
/*
 * The objectives of this test are to visit the "Blog" section on the page, where we will find a particular blog post under the trending
 * articles with the title "15 Make-Ahead Breakfasts Under 300 Calories" and assert its value.
 */
@Test
void testBlogContent() throws InterruptedException {

    webDriver.get(baseUrl);
    //Some Thread.sleep for handling the cookies
    Thread.sleep(5000);
    WebElement blog = webDriver.findElement(By.xpath("/html/body/div[1]/div/footer/div/ul/li[2]/a"));
    WebDriverWait shorterWait = new WebDriverWait(webDriver, Duration.ofSeconds(6));
    JavascriptExecutor js = (JavascriptExecutor) webDriver;
    js.executeScript("arguments[0].scrollIntoView(true);", blog);
    Thread.sleep(3000);

    blog.click();
    WebElement arrow = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/div[2]/div[2]/div[1]/div[1]/
    Thread.sleep(3000);
    for( int i = 0; i <= 3; i++)
    {
      arrow.click();
      Thread.sleep(2000);
    }
    webDriver.findElement(By.linkText("15 Make-Ahead Breakfasts Under 300 Calories")).click();
    WebElement title = shorterWait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/div[2]/div[2]/div[2]/div[2]/section
    assertEquals("15 Make-Ahead Breakfasts Under 300 Calories", title.getText());

    Thread.sleep(10000);

}
```

Test Name: Test Blog Search for Recipe				
Description: The goal of this test is to open the "Recipe" section on the MyFitnessPal blog website, where we will search for a "Chickpea salad" and select the first listed option. After opening this page, we will decide to log this recipe into our daily food tracker, due to which we will be forwarded to another page, whose URL we will be asserting.				
Pre-condition(s): None				
Test Steps: 1. This time we are directly opening the “Recipe” section on the blog website. 2. Here we are location the “search” input field, where we will be entering and searching for our test data. 3. We assert that the first option title matches the expected title and further select it. 4. After opening the recipe, we decide to log it by clicking on a button, which opens a new tab. 5. We assert the value of the new tab’s URL.	**Test Data:** As part of this task as test data can observed the input to “search”: “Chickpea salad”, the title of the recipe used for assertion: "Eat the Rainbow Salad With Lemon Tahini Dressing" and the URL of the new tab (too long to be attached here), which is as well a part of ongoing assertions.	**Expected Result:** We expect the URL fetched from the new tab, to match our assertion.	**Actual Result:** The test does not behave like expected, since the Chrome Driver opens a separate window, whose URL does not conform to the one in the regular Chrome.	**Status:** PASS
Notes: Thread.sleep and shorterWait used for managing the visibility of particular elements/being able to see what is going on.				



```
/*
 * The goal of this test is to open the "Recipe" section on the MyFitnessPal blog web site, where we will search for a "Chickpea salad"
 * and select the first listed option. After opening this page, we will decide to log this recipe into our daily food tracker, due to
 * which we will be forwarded to another page, whose url we will be asserting.
 */
 @Test
 void testBlogSearchForRecipe() throws InterruptedException
 {
    webDriver.get("https://blog.myfitnesspal.com/recipes/");
    Thread.sleep(4500);
    WebElement search = webDriver.findElement(By.id("recipe-search"));
    search.sendKeys("Chickpea salad", Keys.ENTER);
    Thread.sleep(2500);
    WebElement firstRecipe = webDriver.findElement(By.linkText("Eat the Rainbow Salad With Lemon Tahini Dressing"));
    assertEquals("Eat the Rainbow Salad With Lemon Tahini Dressing", firstRecipe.getText());
    firstRecipe.click();
    Thread.sleep(2000);

    webDriver.findElement(By.xpath("/html/body/div[2]/div[2]/div[2]/div[2]/section/div[7]/div[1]/div[1]/div")).click();

    assertEquals("https://www.myfitnesspal.com/recipe/logit?url=https://blog.myfitnesspal.com/31370-eat-the-rainbow-salad-with-lemon-tahini-dr
    Thread.sleep(10000);
 }
```

4. Conclusion

4.1. Testing Summary

Provide a summary of all your executed tests. Something like this would be alright:

Testing Tool	**Total Tests**	**Passed Tests**	**Failed Tests**
Selenium implemented as a dependency in Eclipse	20	18	2

Failed tests:
Apply for The Job of an iOS Intern (3.5.)
Test Blog Search for Recipe (3.6.)

4.2. Final Thoughts

Considering the fact that MyFitnessPal is a widely used application and website, there were no flaws to be found (at least by us - "professionals"), but this meant at the same time that certain components of the website would regularly become updated, causing eventually a particular test case of ours to fail. The scale at which the website was developed also gave us trouble in terms, that certain elements' type was set as "hidden", meaning that we had to find first a solution how to tackle that obstacle and then access the element in question.
Coming to an end we would like to state that we are satisfied with the project process and its outcomes and hope that you will be too.

P.S. We apologize for the repetitiveness of certain steps (login steps) and step descriptions, but those could not be avoided by executing in the manner "test after test", since in that case the Chrome Driver was not able to recognize the elements properly, alongside the longevity of the execution which would be unbearable for us.