Front-End Test Automation: Regular Exam

Exam assignment for the "**QA Front-End Automation" Course @ SoftUn**i.

Submit your work as a **single zip / rar / 7z archive** holding your solutions for each problem at SoftUni Website.

Please refer to the **end of this document** for instructions on **how to submit your work.**

The "Foody" Web App

"Foody" is an **interactive platform** for food enthusiasts to **share, describe, and manage their food experiences**. The application provides a space for users to **register, create profile, and post about their food adventures**.

Your task is to **conduct automated UI tests using Selenium IDE** and **Selenium WebDriver**, ensuring the "Foody" application's functionalities perform as expected.

Access the "Foody" Web App through its dedicated URL:

http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:85

Application Functionality

Key Features:

- **Home Page (Unregistered/Non-Logged Users)**
  - Contains a navigation menu with "LOG IN" and "SIGN UP" options.
  - Features sections like "Share your food," "Describe Your Dish," and "Sign Up Now," which encourage user engagement.
- **Sign Up Page**
  - Allows users to register by entering Username, Email, First Name, Middle Name (optional), Last Name, Password, and Repeat Password.
  - Includes an option for existing users to log in.
- **Log In Page**
  - Registered users can log in using their Username and Password.
- **Home Page (Logged Users)**
  - Features a welcome message, user profile link, and options to add food, and log out.
  - Includes a search functionality for filtering foods by title.
- **Profile Management**
  - Users can edit profile details including profile picture URL, personal information, and a description of their food passion.
- **Food Creation**
  - Logged users can create new food posts with a title, description, and picture URL.
- **Food Management**
  - Options to edit (not fully implemented) and delete food posts are available on the Home page for logged users.

- **Search Functionality**
  - A search function allows users to find specific food posts by title.

Locators or selectors are intentionally tricky to find in this exam. In real practice, QAs often have to deal with similar problems. It's important to insist that developers add clear IDs and other locators to make automated testing easier and more reliable.

1. Selenium IDE

In this section, you will use Selenium IDE to automate and validate core user flows within the "Foody" application. This involves recording and running tests to ensure key functionalities are accessible and perform as expected for both unregistered and registered users.

1.1. Preparation

Before beginning the automated tests, **manually register an account on the "Foody" application**. **Use this account for tasks involving logged-in user actions.**

1.2. Tests

1.2.1. Home Page Navigation Test (Non-Logged User)

- Use Selenium IDE to record a test that navigates to the Home page as an unregistered or non-logged user.
- Verify that the page title is "Home Page - Foody.WebApp".
- Check for the presence of the "SIGN UP" and "LOG IN" in the navbar.
- Ensure the "LEARN MORE" button is also present in the page.

1.2.2. Login In Test

- Using Selenium IDE, create a test that logs into the "Foody" application with your manually registered account.
- Input your registered username and password into the login form and submit it.
- Verify that the Navbar includes links to "👤" "ADD FOOD", and a "Logout" button.
- Assert that the app welcomes you with the message "**Welcome, your_user_name!"**
- Log out and verify that the welcome message on the home page for non-logged users is "**Share your homemade or store-bought food with us!**".

1.2.3. Edit Profile Test

- Using Selenium IDE create test that logs in with the user's credentials from the manually registered account and updates the profile information.
- Use the credentials to log in to the application.
- After logging in, click on "👤" to navigate to the profile page.
- Click the "Edit" button.
- Update the profile fields with new data:
  - **First Name:** Change to a new first name.
  - **Middle Name:** Change to a new middle name.
  - **Last Name:** Change to a new last name.
  - **About:** Add something about you.
- Submit the changes. Verify that the updated information is correctly displayed:
  - **Full name:** Assert that first, middle and last name are displayed correctly under the full name section
  - **About me:** Verify that the updated about me section is correctly shown.
- Logout and verify redirection to home page.

1.3. Tests Organization

- Ensure all recorded tests are organized into a test suite.
- Give the test suite a clear and descriptive name that indicates its purpose.
- Name each test appropriately to reflect the specific task or functionality being tested.
- Arrange the tests in a logical sequence within the suite.
- Each test should start in a new browser window with a non-logged user to ensure a fresh state.
- Maximize the browser window at the start of each test to ensure all elements are visible and to maintain consistency across different screen resolutions.

2. Selenium WebDriver

In this section, you will set up and execute automated UI tests for the "Foody" application using Selenium WebDriver. The goal is to validate various functionalities of the application, ensuring it behaves as expected under different scenarios.

2.1. Setup and Preparation

Create a New NUnit Project

Install Necessary Packages

- Selenium.WebDriver
- Selenium.Support
- ChromeDriver or FirefoxDriver

Include the relevant namespaces in your test classes by adding using directives

- `using OpenQA.Selenium;`
  Instantiating and managing WebDriver instances, interacting with web elements, and controlling browser actions.
- `using OpenQA.Selenium.Chrome;`
  Launching Chrome, setting Chrome-specific options, and managing ChromeDriver.
- `using OpenQA.Selenium.Interactions;`
  Performing complex user interactions that go beyond simple clicks or text entry.
- `using OpenQA.Selenium.Support.UI;`
  Implementing explicit waits, which are essential for handling dynamic content and synchronizing tests.

Create [OneTimeSetup] or [Setup] method

- Configure Chrome options and set preferences.
- Initialize the WebDriver.
- Maximize the browser window and set implicit wait times.
- Perform an initial login to the application, ensuring the session is ready for the tests.

Add [OneTimeTearDown] or [TearDown] method

- Close the browser and end the WebDriver session.

2.2. Automated Tests

Tests should follow a logical sequence - Use the `[Order]` attribute

Each test should start with a consistent state

Name your tests descriptively

2.2.1. Add Food with Invalid Data Test

- **Navigate to the Food Creation page:** Go to the URL for adding food.
- **Fill out the form with invalid data:** Leave title and description blank to simulate invalid input.
- **Submit the form:** Click the submit button.
- **Verify the application's response:**
  - Ensure the page does not navigate away and that an appropriate error message is displayed.
  - The application displays multiple error messages, ensure your test focuses on the correct message. For instance, the main error message indicating "Unable to add this food revue!"

2.2.2. Add Random Food Test

- **Navigate to the Food Creation page:** Go to the specified URL where food can be added.
- **Fill out the form with randomly generated data:**
  - Use a helper method to generate unique titles and descriptions.
- **Submit the form:** Locate and click the submit button.
- **Verify the newly added food appears on the Home Page with the correct details.**
  - **URL Check**: Confirm that the browser navigated to the home page after submission.
  - **Food Check**: Locate the newly added food in the list of foods and verify that the displayed title matches the input.

2.2.3. Edit Last Added Food Test

- **Navigate to the Home Page:** Ensure you start from a consistent state.
- **Locate the last added food:** Retrieve all food elements and select the last one.
- **Click the Edit button:** Scroll and click the Edit button.
- **Modify the food title:** Update the title to a new value.
- **Submit the changes:** Click the add button.
- **Tricky Part:** Asserting the **title change won't be possible due to incomplete functionality**. Verify that the title remains unchanged and print an explanatory message on the console.

2.2.4. Search For Food Title Test

- **Search for the last added food:** Use the search functionality to look for the recently added food title.
- **Assert the search results:** Ensure only one food item is returned and that it matches the searched title.
  - Asserting the search results involves ensuring the search returns exactly one food item and that the title matches.

2.2.5. Delete Last Added Food Test

- **Navigate to the Home Page:** Start from a consistent state by navigating to the base URL.
- **Locate and Count Food Containers:** Retrieve and count all foods on the page to establish an initial count.
- **Scroll to the Last Food:** Scroll to the last food container to ensure it is visible and interactable.
- **Click the Delete Button:** Locate and click the Delete button within the last food.
- **Verify the Deletion:** After deletion, recount the foods to ensure the count has decreased by one and verify that the last food title in the list of foods is not the same as the last added food.

2.2.6. Search for Deleted Food Test

- **Navigate to the Home Page:** Start from a consistent state by navigating to the base URL.
- **Search for the Last Added Food by Title:** Use the search functionality to search for the title of the last added food, which was previously deleted.
- **Assert No Foods Message**: Verify that the application displays the message "There are no foods :(".
- **Assert Add Food Button**: Ensure that the "Add food" button is displayed, providing the user with an option to add new food.

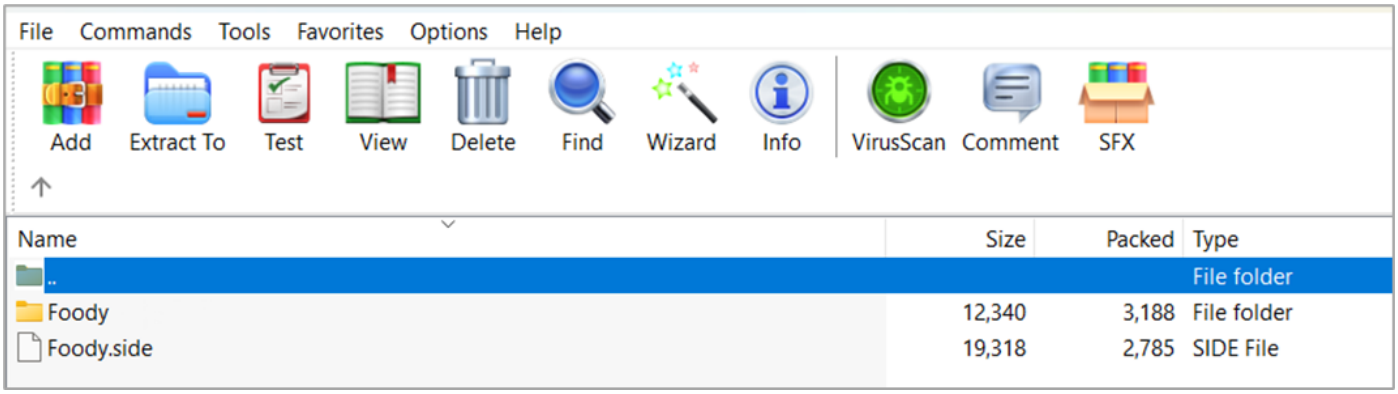
3. How to submit your exam

You should attach a single zip / rar / 7z archive containing all of your tasks.

Upload your archive at SoftUni website, into **Regular Exam section**.

- The Selenium IDE task exported in a single **.side** file.
- Your **Selenium WebDriver** project should be **in a folder**.

At the end, the content of your archive should look similar:



Before archiving, please make sure that you **deleted all bin and obj folders from your VS Test project.**