Front-End Test Automation: Exam Prep I

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 "Idea Center"

"Idea Center" is an **interactive platform** designed for creating and **sharing innovative ideas**. It's a space for users to engage, share, and **manage ideas** across various fields, enhancing collaboration and innovation. The platform, includes key features like user registration, idea submission, and management.

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

Access the "Idea Center" Web App through its dedicated URL:

http://softuni-qa-loadbalancer-2137572849.eu-north-1.elb.amazonaws.com:83

Application Functionality

Key Features:

- **Home Page (Unregistered/Non-Logged Users):**
  - Contains a navigation menu with "LOGIN" and "SIGN UP FOR FREE" options.
  - Features a carousel promoting community engagement and site enjoyment.
- **Sign Up Page:**
  - Allows users to register by entering a username, email, and password, with fields for repeating the password and agreeing to terms of service.
- **Sign In Page:**
  - Enables registered users to log in using their email and password.
- **Home Page (Logged Users):**
  - Provides access to additional functionalities like "My Profile," "My Ideas," and "Create Idea," along with a logout option.
- **Profile Management:**
  - Users can edit their profile details, including profile picture (via URL), first and last name, city, and a description.
- **Idea Creation:**
  - Logged users can create new ideas with a title, picture URL, and description. These ideas are then listed on the "My Ideas" page.
- **Idea Management:**
  - Includes options to view, edit, and delete ideas.
- **Search Functionality:**
  - Not implemented yet.

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 "Idea Center" 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 "Idea Center" 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 - IdeaCenter.WebApp".
- Check for the presence of the "SIGN UP FOR FREE" button using the CSS selector.
- Ensure the "LOGIN" button is also present on the Navbar.

1.2.2. Log In Test

- Using Selenium IDE, create a test that logs into the "Idea Center" application with your manually registered account.
- Input your registered email and password into the login form and submit it.
- Verify that the Navbar includes links to "My Profile," "My Ideas," "Create Idea," and a "Logout" button.
- Log out and verify that the "SIGN UP FOR FREE" button is visible on the Home page, indicating the user is logged out.

Hint: The SIGN IN button may be difficult to locate. Use the Developer Tools to inspect and identify the appropriate locator, such as a class selector or XPath. Ensure your test correctly interacts with this button by refining your selector strategy if needed.

1.2.3. Edit Profile Test

- Write a Selenium IDE 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, navigate to the "My Profile" page.
- Click on the "Edit Profile" button.
- Update the profile fields with new data:
  - **First Name:** Change to a new first name.
  - **Last Name:** Change to a new last name.
  - **City:** Update with a new city name.
  - **Description:** Add or update the description.
- Submit the changes. Verify that the updated information is correctly displayed:
  - **Names and City:** Assert that both the new name and city are displayed together.
  - **Description:** Verify that the updated description 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 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 "Idea Center" 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**

Please note that the tests will only pass if you run all of them together. Running them one by one will cause some tests to fail.

2.2.1. Create Idea With Invalid Data Test

- **Navigate to the Idea Creation page (/Ideas/Create).**
- **Use the SendKeys method to fill out the form with invalid data (empty title and description).**
  - Ensure the title and description fields are left blank to simulate invalid input.
- **Submit the form by clicking the submit button.**
  - Use CSS Selector to locate and click the submit button.
- **Verify the application's response to invalid data input.**
  - **URL Check**: Confirm that the browser remains on the Idea Creation page after submission.
  - **Error Message Check**: Verify that an appropriate error message is displayed, indicating the failure to create a new idea due to invalid input.

Hint:

The Error Message Check part might be tricky because it involves accurately identifying and selecting the specific error message element amidst potentially multiple validation messages.

1. Use a **specific CSS selector** to target the error message element.
2. The application displays **multiple error messages**, ensure your test focuses on the correct message. For instance, the **main error message** indicating that a **new idea could not be created**.

2.2.2. Create Random Idea Test

- **Navigate to the Idea Creation page (/Ideas/Create).**
- **Use the SendKeys method to fill out the form with a randomly generated title and description.**
  - Implement a helper method, such as GenerateRandomString, to generate unique titles and descriptions for each test run.
  - Use Id selectors to find form fields (title and description are mandatory; the picture URL is optional and should point to an image URL).
- **Submit the form by clicking the submit button.**
  - Use CSS Selector to locate and click the submit button.
- **Verify the idea is listed on the "My Ideas" page (/Ideas/MyIdeas) with the correct details by asserting the displayed data matches the input.**
  - **URL Check**: Confirm that the browser navigated to the correct page after submission.
  - **Idea Check**: Locate the newly created idea in the list of ideas and verify that the displayed description matches the input.

Hint:

The Idea Check part might be tricky because it involves navigating through multiple elements and ensuring you select the correct one (in this case, the most recently added idea).

1. **Use FindElements method** to retrieve a list of web elements that match the specified CSS selector.
2. The **CSS selector** should target all elements with a specific class that identifies idea cards.
3. Use **Last() method** to select the last element in the list, which represents the most recently added idea card.

2.2.3. View Last Created Idea Test

- **Navigate to the "My Ideas" page (/Ideas/MyIdeas)**
- **Locate the last created idea**
  - Use the FindElements method to get a list of all idea cards on the page.
  - The process might involve navigating through multiple elements to ensure the correct one (the most recently added idea) is selected.
- **Click the "View" button**
  - Within the last idea card, locate the "View" button using a CSS selector that targets the hyperlink with a specific pattern.
  - Use the Actions class to move to the element and click it. This can help avoid issues where elements are not directly clickable due to overlapping elements or other layout issues.
  - After clicking the "View" button, the application should navigate to a page displaying the details of the idea.

- **Locate the element containing the title of the idea (e.g., using a CSS selector).**
- **Extract the text of the title and compare it with the expected value using an assertion.**

Hint: At this point you should think about declaring static class-level variables to maintain their value across the test class.

2.2.4. Edit Last Created Idea Title Test

- **Navigate to the "My Ideas" page (/Ideas/MyIdeas)**
- **Locate the last created idea**
- **Click the "Edit" button:**
  - Within the last idea card, locate the "Edit" button using a CSS selector that targets the hyperlink with a specific pattern.
  - Use the Actions class to move to the element and click it.
  - Click the "Edit" button to navigate to the edit page for the selected idea.
- **Modify the Idea Title:**
  - Locate the input field for the idea title using its ID.
  - Clear the existing text and enter a new title, prefixing it with "Changed Title: " for easy identification.
- **Submit the Changes:**
  - Click the submit button to save the changes. This action should update the idea and return the user to the "My Ideas" page.
- **Verify the Changes:**
  - Locate the last idea card "My Ideas" page again.
  - Click the "View" button to open the details of the edited idea.
  - Locate the element displaying the idea title and verify that the title matches the newly set value.

2.2.5. Edit Idea Description Test

- **Navigate to the "My Ideas" page (/Ideas/MyIdeas)**
- **Locate the last created idea**
- **Click the "Edit" button:**
  - Within the last idea card, locate the "Edit" button using a CSS selector that targets the hyperlink with a specific pattern.
  - Use the Actions class to move to the "Edit" button and click it, ensuring the element is interactable.
- **Modify the Idea Description:**
  - On the edit page, locate the description input field using its ID.
  - Clear the existing text and enter a new description, prefixing it with "Changed Description: " for easy identification.
- **Submit the Changes:**
  - Click the submit button to save the changes. This action should update the idea and return the user to the "My Ideas" page.
- **Verify the Changes:**
  - Locate the last idea card again, find the element containing the description, and verify that the description matches the newly set value.

2.2.6. Delete Last Idea Test

- **Navigate to the "My Ideas" page (/Ideas/MyIdeas):**
- **Locate the last created idea:**
- **Click the "Delete" button:**
  - Within the last idea card, locate the "Delete" button using a CSS selector that targets the hyperlink with a specific pattern.
  - Use the Actions class to move to the "Delete" button and click it, ensuring the element is interactable.
- **Verify Deletion:**
  - After the deletion, check the list of ideas again to ensure that the idea has been removed.

- **Hint:** Verify that none of the remaining idea cards contains the description of the deleted idea.

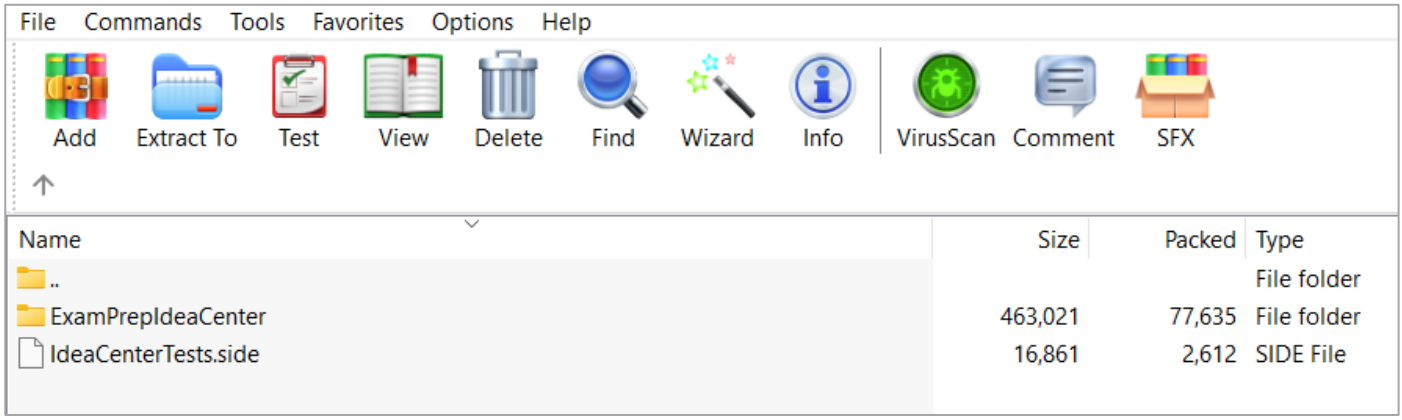
3. How to submit your exam

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.**