# Front-End Test Automation: Exam Prep II



**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 "Revue Crafters" Web App

"**Revue Crafters**" is a web application designed to facilitate the **creation and management of reviews** (referred to as "revues") across various domains. This interactive platform allows users to register, log in, edit their profiles, and manage their revues efficiently. The primary objective of this exam is to ensure the application's functionalities perform as expected through automated UI tests using Selenium IDE and Selenium WebDriver.

**Access the "Revue Crafters" Web App through its dedicated URL:**

[**https://d3s5nxhwblsjbi.cloudfront.net/**](https://d3s5nxhwblsjbi.cloudfront.net/)

## Application Functionality

**Key Features:**

* **Home Page (Unregistered/Non-Logged Users)**
* Contains a navigation menu with "HOME," "ABOUT," "SERVICES," "REGISTER," and "LOGIN" options.
* Features sections such as "Explore the World of Reviews," "There is no limitations!!!," and "Join the RevueCrafter Community Today."
* **Sign Up**
* Allows users to register by entering a username, email, password, and repeat password.
* **Log In**
* Enables registered users to log in using their email and password.
* **Home Page (Logged Users)**
* Provides access to additional functionalities like "MY REVUES," "CREATE REVUE," "PROFILE," and "LOGOUT."
* **Profile Management**
* Users cannot edit their profile details, since this functionality is not yet implemented.
* **Revue Creation:**
* Logged users can create new revues with a title, picture URL (optional), and description. These revues are then listed on the "My Revues" page.
* **Revue Management:**
* Includes options to edit and delete revues. View functionality is not yet implemented.
  + **Search Functionality**
* A search function allows users to find specific revues 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.

## Selenium IDE

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

### Preparation

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

### 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 "RevueCrafters".
* Verify the presence of the About, Sevices, Register and Login links in the Navigation bar.

#### 1.2.2. Log In Test

* Using Selenium IDE, create a test that logs into the "RevueCrafters" 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 Revues" "Create Revue", "Profile" and a "Logout" button.
* Also verify the welcome message reads "Dear, your-username"
* Log out and verify the text "Explore the World of Reviews" is present on the Home page, indicating the user is logged out.

#### 1.2.3. Verify Profile Test

* Write a Selenium IDE test that logs in with the user's credentials from the manually registered account and checks the username in the profile section is correctly displayed.
* Use the credentials to log in to the application.
* After logging in, navigate to the "Profile".
* Verify that your username is correctly displayed.
* Logout and verify redirection to home page with the presence of the Register link.

### 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.

## Selenium WebDriver

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

### 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.

### Automated Tests

* **Tests should follow a logical sequence -** Use the [Order] attribute
* **Each test should start with a consistent state**
* **Name your tests descriptively**

#### Create Revue with Invalid Data Test

* **Navigate to the Revue Creation Page**
* **Use the Actions class to scroll to the form element. Ensure the form is visible and interactable.**
* **Fill Out the Form with Invalid Data:**
  + Leave the title and description fields blank to simulate invalid input.
  + Use SendKeys to input the invalid data into the form fields identified by their respective IDs.
* **Submit the Form:** Locate and click the submit button identified by the CSS selector.
* **Verify the Application's Response:**
  + Confirm that the browser remains on the creation page after submission.
  + Locate the error message element using the CSS selector.
  + Verify that the main error message is displayed and matches the expected text "Unable to create new Revue!".

#### Create Random Revue Test

* **Create a random title and description for the revue.**
* **Navigate to the Revue Creation Page.**
* **Scroll to the Create Form:** Use the Actions class to scroll to the form element. This ensures the form is visible and interactable.
* **Fill Out the Form with Random Data**:
  + Use SendKeys to input the random title into the form field identified by ID.
  + Use SendKeys to input the random description into the form field identified by ID.
* **Submit the Form:** Locate and click the submit button identified by the CSS selector.
* **Verify the Application's Response:**
  + Confirm that the browser navigates to the "My Revues" page after submission.
* **Locate all created revues on the "My Revues" page using the CSS selector.**
* **Identify the last created revue by selecting the last element from the list.**
* **Within the last created revue, locate the title element using the CSS selector.**
* **Verify that the title of the last created revue matches the generated title by asserting the text of the title element is equal to the random title.**

#### 2.2.3. Search for Revue Title Test

* **Navigate to the "My Revues" Page.**
* **Use the Actions class to scroll to the search field element identified by the CSS selector.**
* **Perform the Search:**
  + Locate the search input field using the By.Name selector.
  + Input the title of the last created revue.
  + Locate and click the search button identified by the.
* **Verify the Search Results:**
  + Locate the title of the searched revue in the search results using the CSS.
* **Assert that the title of the revue in the search results matches the title of the last created revue by checking the text content of the title element.**

#### Edit Last Created Revue Title Test

* **Navigate to the "My Revues" Page**
* **Locate the Last Created Revue:**
  + Retrieve all revue elements using the CSS selector.
  + Assert that there is at least one revue element on the page.
  + Select the last revue element in the list.
  + Use the Actions class to scroll to the last revue element to ensure it is visible and interactable.
* **Within the last revue element, locate the "Edit" button using the CSS.**
* **Click the "Edit" button to navigate to the edit page.**
* **Edit the Revue Title:**
* **Locate the edit form using the CSS selector and scroll to it using the Actions class.**
* **Find the title input field by its ID.**
* **Clear the existing title and enter a new title.**
* **Save the Changes:**
* **Locate and click the save changes button identified by the CSS selector button.**
* **Assert that the current URL matches the expected URL for the "My Revues" page.**
* **Verify the Updated Title:**
  + Retrieve all revue elements again using the CSS selector.
  + Select the last revue element in the list.
  + Locate the title element within the last revue using the CSS selector.
  + Assert that the title of the last revue matches the newly updated title.

#### Delete Last Created Revue Title Test

* **Navigate to the "My Revues" Page.**
* **Locate the Last Created Revue:**
  + Retrieve all revue elements using the CSS selector.
  + Assert that there is at least one revue element on the page.
  + Select the last revue element in the list.
* **Use the Actions class to scroll to the last revue element to ensure it is visible and interactable.**
* **Click the Delete Button:**
* Within the last revue element, locate the "Delete" button using the CSS selector.
* Click the "Delete" button to delete the revue.
* **Verify the Page Redirection:**
  + Assert that the current URL matches the expected URL for the "My Revues" page.
* **Verify the Deletion:**
  + Retrieve all revue elements again using the CSS selector.
  + Assert that the number of revues has decreased.
  + Select the last revue element in the list.
  + Locate the title element within the last revue using the CSS selector.
  + Assert that the title of the last revue is not the same as the last created revue title.

#### Search for Deleted or Non-Existent RevueTest

* **Navigate to the "My Revues" Page.**
* **Locate the search field container using the CSS selector.**
  + Use the Actions class to scroll to the search field to ensure it is visible and interactable.
* **Locate the search input field by its name attribute keyword and enter the title of the last created revue   
  or a non-existig one.**
* **Locate and click the search button using its ID search-button.**
* **Assert the No Revues Message:**
* Locate the message indicating no revues using the CSS selector.
* Assert that the message text matches "No Revues yet!" to confirm that the revue is not listed.

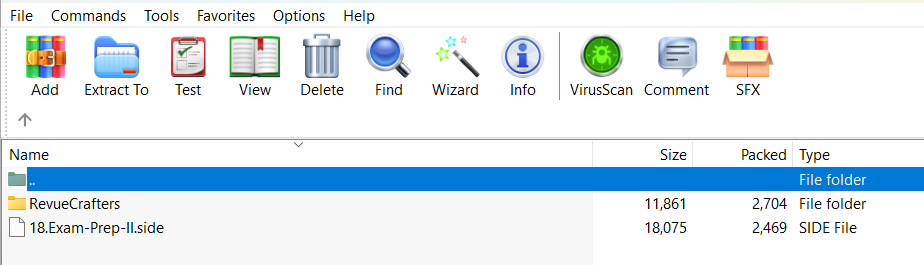
# How to submit your exam

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.**