# Front-End Test Automation: Retake 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 "SimpleNotes" Web App

**"SimpleNotes"** is an **interactive platform** for taking notes.

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

**Access the "SimpleNotes" Web App through its dedicated URL:**

[**https://d5wfqm7y6yb3q.cloudfront.net/**](https://d5wfqm7y6yb3q.cloudfront.net/)

## Application Functionality

**Key Features:**

* **Landing page**
* Contains a call-to-action (CTA) button, labeled "JOIN US NOW".
* Upon clicking the "JOIN US NOW" button the user is redirected to the **registration page**.
* **Registration Page**
* Allows users to register by entering First Name, Middle Name (optional), Last Name, Username, Email, Password, and Repeat Password.
* Includes an option for existing users to swap to login page.
* **Login Page**
* Registered users can log in using their **email** (not with their username, it's a bug) and password.
* **Main Page (Logged Users)**
  + Features a sidebar menu with options to add new notes, view completed notes, and log out.
  + Includes a user profile icon in the top-right corner.
  + Users can view their profile upon clicking the user profile icon and choosing "My profile".
* **Profile Management**
  + Users can only vew their profiles. Edit functionality is not yet implemented.
* **Note Creation**
* Logged users can create new notes with a title, description, and status. Start and end dates are optional.
* **Note Management**
  + Users can change the status of the notes from "New" to "Done"
* Options to edit and delete notes are also available.

**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 "Movie Catalog" 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 "SimpleNotes" application**. **Use this account for tasks involving logged-in user actions.**

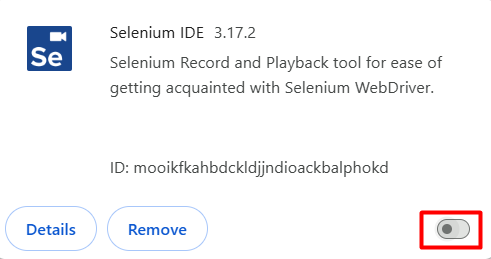
### Tests

**Selenium IDE & Google Chrome - Important Notice**

Selenium IDE is no longer officially supported in Google Chrome. The browser may disable it automatically due to security updates, and you will have to manually enable it each time from the extensions settings.

✅ **Recommended:** Use Mozilla Firefox, where Selenium IDE works reliably without restrictions.

✅ **If you must use it in Chrome:** Go to chrome://extensions/. Find Selenium IDE. Enable it manually (toggle switch).



⚠️ However, Chrome might disable it again after a restart! To avoid issues, we **strongly recommend using Firefox** instead. Also you might experience problems with BFCache. So, again, we strongly recommend using Firefox.

### Tests

#### 1.2.1. Landing Page Navigation Test (Non-Logged User)

**Use Selenium IDE to record a test that navigates to the Landing page as an unregistered or non-logged user.**

* Open the Landing Page
* Set Window Size
* Verify Page Title, e.g "Home Page - SimpleNotes.WebApp"
* Verify CTA Button Text, e.g. "JOIN US NOW!"
* Close the Browser

#### 1.2.2. Log In Test

**Using Selenium IDE, create a test that logs into the "SimpleNotes" application with your manually registered account.**

* Open the Landing Page
* Set Window Size
* Click the CTA "JOIN US NOW!" Button
* Click the Login Tab (**hint**: id=tab-login)
* Click on the Username Field
* Enter Username
* Click on the Password Field
* Enter Password
* Click the Login Button (**hint:** xpath=//button[@type='submit'])
* Verify Successful Login, e.g. title of the page
* Click Logout (**hint:** linkText=Logout)
* Verify CTA Button is Visible Again
* Close the Browser

#### 1.2.3. View 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.**

* Open the Landing Page
* Set Window Size
* Click the CTA "JOIN US NOW!" Button
* Click the Login Tab (**hint**: id=tab-login)
* Click on the Username Field
* Enter Username
* Click on the Password Field
* Enter Password
* Click the Login Button (**hint:** xpath=//button[@type='submit'])
* Open Profile Menu (**hint:** css=#navbarDropdownMenuLink > .rounded-circle)
* Click "My Profile" Option (**hint**: linkText=My Profile)
* Assert username and email
* Click Logout
* Verify Title of the page, e.g. " Home Page - SimpleNotes.WebApp"
* Close the Browser

### 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.

## Selenium WebDriver

In this section, you will set up and execute automated UI tests for the "SimpleNotes" 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] method**

* Configure Chrome options and set preferences.
* Initialize the WebDriver.
* Maximize the browser window and set timeouts.
* Navigate to the Application.
* Locate and Click the "Join Us" Button.
* Scroll to the "Join Us" Button.
* Click "Join Us" Button
* Perform Login
  + Enter Login Credentials
  + Click the Login Button

**Add [OneTimeTearDown] method**

* Dispose 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.**

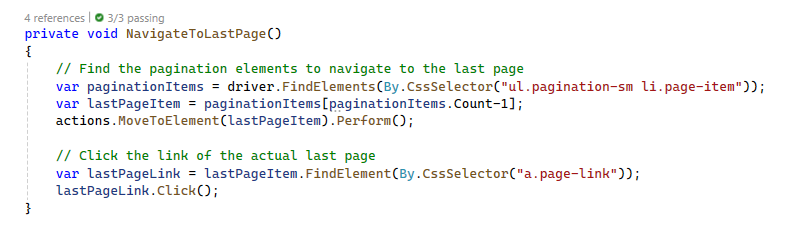
**Name your tests descriptively.**

**IMPORTANT!** When testing the Simple Ntes application, it is very important to understand the following points regarding **pagination** and **Note listings**:

* The application displays up to six notes per page.
* Notes are arranged in two rows of three per page.
* If you add more than six notes, a new page will be created, and the additional notes will appear there.
* What this means for your tests:
  + If your test setup includes only six or fewer notes, your tests will pass without issues.
  + If you add a seventh note or more, it will appear on a new page.
  + If your test does not handle pagination, it may fail when trying to locate a note that is on another page.

**What You Should Do:**

* **Option 1: Keep It Simple (Recommended)**
  + To avoid pagination issues, ensure that your test setup includes only six notes at a time.
  + This approach is valid and reliable, as it allows your tests to focus on note creation, editing, and deletion without pagination complexity.
* **Option 2 (Bonus Points! 🎯):**
  + Try implementing pagination handling in your tests to verify notes across multiple pages.
* **Hint:** Here is a method to help you handle pagination.



#### Add Note with Invalid Data Test

This Selenium WebDriver test verifies that attempting to create a new note with invalid data (empty title and description) results in an error message and prevents submission.

* Navigate to <https://d5wfqm7y6yb3q.cloudfront.net/Note/New>
* Click "Create Note".
* Enter Invalid Note Title.
* Enter Invalid Note Description.
* Click the "Create" Button.
* Verify Page Remains on Note Creation Page.
* Verify Error Message is Displayed.

#### Add Random Note Test

This test verifies that a user can successfully create a new note with randomly generated data and that the note creation success message is displayed correctly.

* Navigate to <https://d5wfqm7y6yb3q.cloudfront.net/Note/New>
  + Click "Create Note".
* Fill Out the Form with Randomly Generated Data.
  + Title
  + Description
* Select status "New" from the Dropdown.
* Click the "Create" Button.
* Verify Success Message After Submission – Note created successfully! (**hint:** "div[@class='toast-message']").

#### 2.2.3. Edit Last Added Note Test

This test verifies that a user can successfully edit the **last added note**, update its title, and confirm that the changes are saved correctly.

* Navigate to <https://d5wfqm7y6yb3q.cloudfront.net/Note/New>
* Locate the Last Added Note

When testing note editing, it's important to understand how notes are structured on the page:

✅ Each page contains up to six notes, arranged in two separate containers.

✅ Each container holds exactly three notes.

**To find the edit button of the last note**, your test needs to:

* Locate the **last container** on the page. (**hint:** "section.p-4.d-flex.justify-content-center.text-center.w-100").
* Then find the Edit button of the **last note inside that container**. (**hint:** "a[href\*='/Note/Edit']").
* Click the Edit button.
* Modify the Note Title.
* Click the Edit button to save the changes.
* Save the Updated Note.
* Verify the Updated Note Appears with the New Title
  + You should find again the container and the title of the note (hint: "div.card-body h5.card-title")

#### 2.2.4. Move Edited Note to Done Test

This test verifies that a user can successfully **move the last edited note to "Done" status** and that a success message is displayed after the status change.

* For this test you should follow the steps from the previous one up until editing the title.
* This time don't edit the title but change the status of the from New to Done (**hint:** "a[href\*='ChangeToDone']")
* Then verify success message - Note status changed successfully! (**hint:** //div[@class='toast-message']")

#### Delete Edited Note Test

This test verifies that a user can successfully **delete a previously edited note** after it has been moved to the "Done" section.

* Navigate to <https://d5wfqm7y6yb3q.cloudfront.net/Note/Done>.
* Locate the Last Note in the "Done" List. It is similar to tests 2.2.3 and 2.2.4.
* Click the Delete Button.
* Verify That the Delete Confirmation Page is Displayed.
* Confirm the Deletion clicking the "YES" button. (hint: "form[action\*='/Note/Delete'] button[type='submit']")

#### Logout Test

This test verifies that a user can successfully **log out**, is redirected to the home page, and cannot access restricted pages after logging out.

* Locate and Click the Logout Button
* Verify That the User is Redirected to the Landing Page
* Attempt to Access a Restricted Page <https://d5wfqm7y6yb3q.cloudfront.net/Note/New>
* Verify "Access Denied" Message is Displayed (**hint:** FindElement(By.TagName("pre")).Text;)

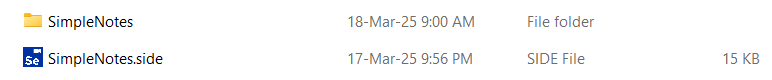
# How to submit your exam

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

Upload your archive at SoftUni website, into **Retake 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.**