## DemoQA.com

## Automation Cucumber BDD Selenium Framework

This project is a **Cucumber BDD Selenium Automation Framework** designed to automate test scenarios on demoqa.com. It follows the **Page Object Model (POM)** for better maintainability and readability.

Step1: Setup & Configuration.

1. JDK
2. Eclipse
3. Cucumber Plugin
4. Set Environmental Variables for Java home and Maven Home

Step2: Create Maven Project in eclipse and update pom.xml with the following dependencies.

1. cucumber-core
2. cucumber-html
3. cucumber-java
4. selenium-java
5. cucumber-junit
6. cucumber-jvm-deps
7. cucumber-reporting
8. hamcrest
9. gherkin
10. junit
11. maven-cucumber-reporting
12. extentreports-cucumber7-adapter
13. testng
14. cucumber-testng
15. tools (local system JAR)
16. log4j-api
17. log4j-core

Step3: Create Project Structure.

com.demoQA

── src/test/java  
 ── pageObjects(package)  
 ── stepDefinitions(package)  
 ── testRunner(package)  
 ── utilities(package)

── src/test/resources  
 ── config.properties(file)  
 ── extent.properties(file)  
 ── log4j2.xml(file)  
 ── pdf-config.yaml(file)  
 ── spark-config.xml(file)

── JRE System Library [JavaSE-1.8]  
 ── Maven Dependencies  
 ── Downloads(folder)  
 ── Drivers(folder)  
 ── Features(folder)  
 ── logs(folder)  
 ── src  
 ── target(folder)  
 ── test-output(folder)  
 ── pom.xml(file)  
 ── Toolsqa.jpg(file)

Step4 : Copy drivers into drivers folder.

Step5: Automating Test Cases

1. Create feature files in the Features folder.
2. Create pageobject classes under pageObjects package.
3. Create steps.java classes under the stepDefinitions package.
4. Create TestRunner.java class under the testRunner package
5. Create a class to take screenshots on failure under the utilities package.
6. Configure browser in config properties and add configuration file needed to generate extent reports and logs under src/test/resources folder.
7. Create reports and screenshots folder under target folder to store the report and screenshots.

## Reports & Logs

* **Extent Report**: Automatically generated under /test-output/ folder after test execution.
* **Screenshots**: Captured for failures and stored under /screenshots/.
* **Logs**: Execution logs maintained using log4j2.xml.

## Project Structure & Approach

This project automates the below test scenarios on DemoQA using the Cucumber BDD framework, Selenium WebDriver, Java, and Maven. It follows the **Page Object Model (POM)** design pattern and includes reporting, logging, and reusable utility support.

Form interaction with data input and validation  
File upload/download functionality  
Calendar/date picker interaction  
Tabular data interaction (e.g., sorting, pagination, row selection)  
Multi-page navigation with data sharing  
Validation messages or tooltips  
Handling of dynamic elements or synchronization techniques

**pageObjects :** Contains classes where each class represents a specific page or component on the webpage. These classes include element locators and reusable methods to interact with them.

**CommonConstructor\_POM.java:** Inherits the WebDriver setup for reuse in all page classes.  
**ElementsCard\_POM.java:** Handles Tabular data interaction (e.g., sorting, pagination, row selection), Form interaction with data input and validation, File upload/download functionality, Handling of dynamic elements or synchronization techniques  
**FormsCard\_POM.java**: Form interaction with data input and validation  
**HomePage\_POM**:Handles different elements in the Homepage  
**StudentRegistration\_POM**:Handles Form interaction with data input,Calendar/date picker interaction  
**Widgets\_POM.**java: Handles widget-based interactions like tooltips

**stepDefinitions**

Includes all **Cucumber step definition** files that link the Gherkin steps in feature files to their respective Java method implementations.

Each class corresponds to a test scenario category (e.g., DynamicColorChangeValidation.java, FormDataValidation.java).  
**Hooks.java**: Contains @Before and @After hooks for setup and teardown like launching or quitting the browser.

**testRunner**

Contains the **Cucumber Test Runner** class to trigger feature execution using @CucumberOptions.  
TestRunner.java: Central point for test execution, specifying glue paths and reporting plugins.  
Extends AbstractTestNGCucumberTests to access TestNG

**utilities**

Utility/helper classes to support common tasks such as screenshots

TakeScreenshot.java: Captures screenshots on failure or as needed for evidence.