**CAPSTONE PROJECT**

**DEMO WEB SHOP**

**ABHIRAM S BATCH 4**

[**abhiramannur2002@gmail.com**](mailto:abhiramannur2002@gmail.com)

**Introduction**

Website to be automated**:** <https://demowebshop.tricentis.com/>  
The **Demo Web Shop** ([website link](https://demowebshop.tricentis.com/)) is an online shopping platform that allows users to browse and purchase various electronic products. The platform provides multiple categories for users to explore, enabling them to view product details, add products to their cart, and place orders.

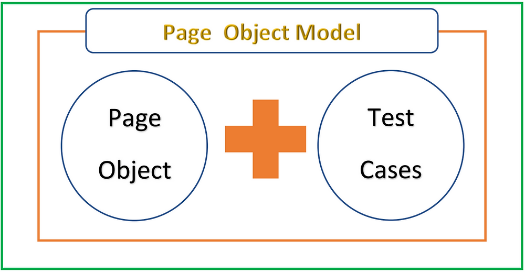
This project was undertaken as part of an automation testing initiative to apply the knowledge and skills gained during automation testing training. It incorporates industry-standard tools and frameworks such as Selenium, TestNG, Cucumber, a hybrid framework, and advanced reporting tools like Allure/Extent.

**Objective**

The main objective of this project is to automate various workflows of the Demo Web Shop platform using Selenium with TestNG, BDD Cucumber, the Page Object Model (POM). The data required for the tests is retrieved from external sources such as data sheets (Excel) and properties files. By implementing automation, the testing process becomes faster, more accurate, and more efficient, reducing manual effort and ensuring consistency across multiple test executions.

The project follows a hybrid automation framework, integrating different testing techniques and tools to provide a robust and scalable solution. This document provides a detailed breakdown of the concepts used, the structure of the project, the purpose of each file, and the functionalities implemented.

**Testing Concepts, Methodologies, and Features Used**

**Page Object Model (POM)**

* Organized test automation framework by separating elements and actions into individual page classes.
* Created separate Page Classes (e.g., LoginPage, RegistrationPage, etc).
* Used Page Factory for better element management.

**Selenium WebDriver**

* Browser automation using WebDriver for Chrome, Firefox, and Edge.
* Finding elements using By locators (id, xpath).
* Interacting with web elements (click(), sendKeys(), etc).
* Taking screenshots using TakesScreenshot.

**TestNG**.

* Annotations (@Test, @BeforeMethod, @AfterMethod, etc).
* Assertions (Assert.assertTrue()).
* Data-driven testing using @DataProvider.
* Running tests with an XML file.

**(BDD) Cucumber**

* Defined test scenarios in Gherkin syntax (Given, When, Then) and implemented step definitions for login function. Also made use of Hooks and Tags.
* Cucumber with TestNG: Your TestNGRunner ensures execution with TestNG.
* Cucumber Standalone: CucumberRunner allows execution in pure Cucumber.

**Properties File Usage**

* Loaded data.properties to fetch test data dynamically (email, password, browser).

**Data-Driven Testing**

* Executed tests with multiple inputs using TestNG XML parameters and Cucumber’s Scenario Outline.
* LoginExcelTest fetches login data from Excel file using Apache POI.
* @DataProvider used to pass multiple test cases.

**Extent Reports**

* Generated structured reports with detailed logs, test status, and screenshots for execution tracking.

**Screenshot Capture**

* Captured screenshots for debugging and validation during test execution.

**Synchronization**

* Used implicit wait, explicit wait.

**TestNG XML for Parameterization**

* Passed dynamic test data (e.g., registration inputs) through testng.xml.

**Jenkins, GitHub**

* Cloned the project into github.
* Tested through Jenkins by giving local path
* Tested through Jenkins-Github integration.

**Scenarios Covered**

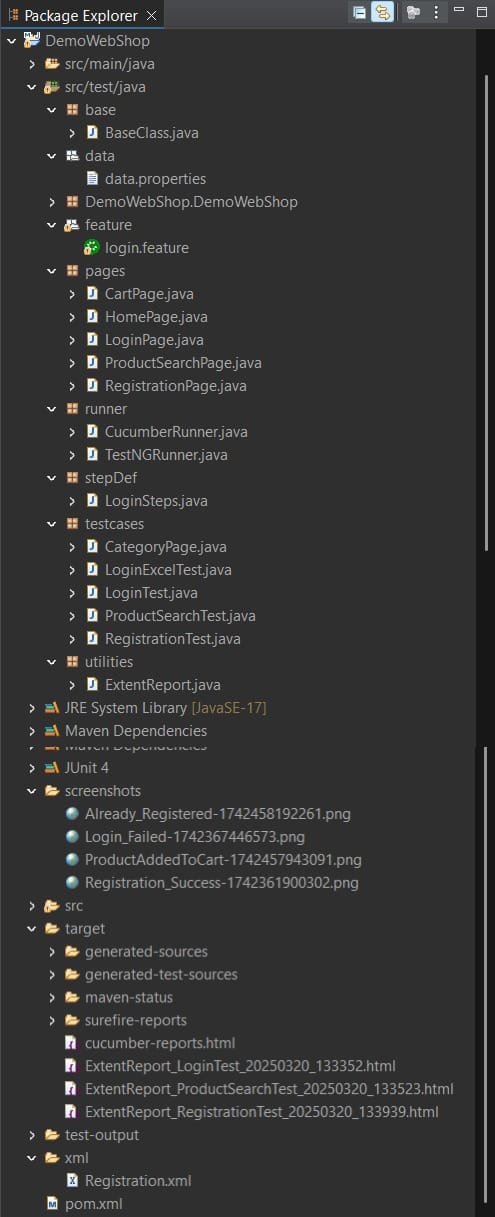
1. **User Login (Multiple Approaches)**
   * **TestNG-based Login Test:** Validates login with page object model model, assertions.
   * **BDD Cucumber Login Test:** Uses feature file and step definitions for login validation.
   * **Data-Driven Login Test:** Reads credentials from an Excel file for multiple test cases.
2. **User Registration**
   * **Registration Test:** Automates form filling and validation.
   * **Parameterization:** Tested using parameterized values from Registration.xml..
   * **Executed via Maven:** The test is triggered by running pom.xml as Maven Test.
3. **Product Search and Add to Cart**
   * **Searching for a Product:** Searches for "Simple Computer" and verifies results.
   * **Selecting a Product:** Clicks on a product from search results.
   * **Adding Product to Cart:** Adds the selected product to the shopping cart.
   * **Verifying Product in Cart:** Confirms the product is successfully added.
4. **Shopping Cart Verification**
   * **Navigating to Cart Page:** Ensures the cart page is accessible.
   * **Checking Product Presence:** Verifies if a specific product exists in the cart.
5. **Extent Reports & Screenshots**
   * **Test Execution Logs:** Generates detailed logs for each test.
   * **Screenshots for Verification:** Captures screenshots for success and failure cases.

**File Structure:**

**Initializes WebDriver, reads properties, manages browser actions.**

**Project name: DemoWebShop**

**Stores configuration details (URL, credentials, etc.)**

****

**Contains the extent reports**

**Parameterized TestNG file for registration testing.**

**Contains screenshots taken during validation, etc.**

**Executes Cucumber tests using JUnit or TestNG.**

**Contains Page Object Model classes for interacting with web elements.**

**Defines login scenarios in Gherkin syntax.**

**Contains Page Object Model classes for interacting with web elements.**

**Manages dependencies and test execution.**

**Generates Extent Reports for test execution.**

**Implements login steps from login.feature.**

**Base Package (base/)**

**BaseClass.java**

* Initializes WebDriver and manages browser setup (invokeBrowser()).
* Contains common utility methods like closeBrowser() and screenshot().
* Uses data.properties for configuration settings.

**Data Package (data/)**

**data.properties**

* Stores configurable values like URLs, credentials, and browser type.
* Loaded in BaseClass.java for test execution.

**Feature Package (feature/)**

**login.feature**

* Defines BDD scenarios for login testing using Cucumber.
* Uses parameterization for valid and invalid credentials.

**Pages Package (pages/)**

**HomePage.java**

* Implements navigation to Register, Login, and Computers categories.
* Uses locators and BaseClass.driver to interact with elements.

**LoginPage.java**

* Provides login functionality.
* Uses POM approach to locate and interact with input fields.

**CartPage.java**

* Manages shopping cart interactions.
* Checks if a specific product is in the cart.

**ProductSearchPage.java**

* Implements product search and selection logic.
* Uses @FindBy annotations (Page Factory) for element identification.
* Scrolls the page using JavaScriptExecutor and handles delays with Thread.sleep().

**RegistrationPage.java**

* Automates user registration.
* Uses TestNG parameterization from Registration.xml.

**Runner Package (runner/)**

**TestNGRunner.java**

* Runs Cucumber tests using TestNG.

**CucumberRunner.java**

* Runs tests in JUnit format for Cucumber execution.
* Generates HTML reports (target/cucumber-reports.html).

**Step Definitions Package (stepDef/)**

**Hooks.java**

* Creates WebDriver before the scenario.
* Quits WebDriver after the scenario

**LoginSteps.java**

* Implements step definitions for login using Selenium WebDriver.
* Use the shared WebDriver instance from Hooks.java.
* Uses @Given, @When, and @Then Cucumber annotations.
* Handles login validation and browser quit in @Then step.

**Test Cases Package (testcases/)**

**LoginTest.java**

* Executes login tests using TestNG.
* Uses assertions to verify successful and failed logins.
* Integrates with ExtentReport for logging and reporting.

**ProductSearchTest.java**

* Tests product search and add-to-cart functionality.
* Integrates with ExtentReport for logging and reporting.

**RegistrationTest.java**

* Validates user registration process.
* Uses TestNG XML parameterization (Registration.xml).
* Integrates with ExtentReport for logging and reporting.

**LoginExcelTest.java**

* Reads login credentials from an Excel file.
* Implements Data-Driven Testing using Apache POI.

**Utilities Package (utilities/)**

**ExtentReport.java**

* Generates test execution reports with logs.

**Screenshots Folder (screenshots/)**

* Stores failure and success screenshots.
* Captured using BaseClass.screenshot().

**XML Configuration (xml/)**

**Registration.xml**

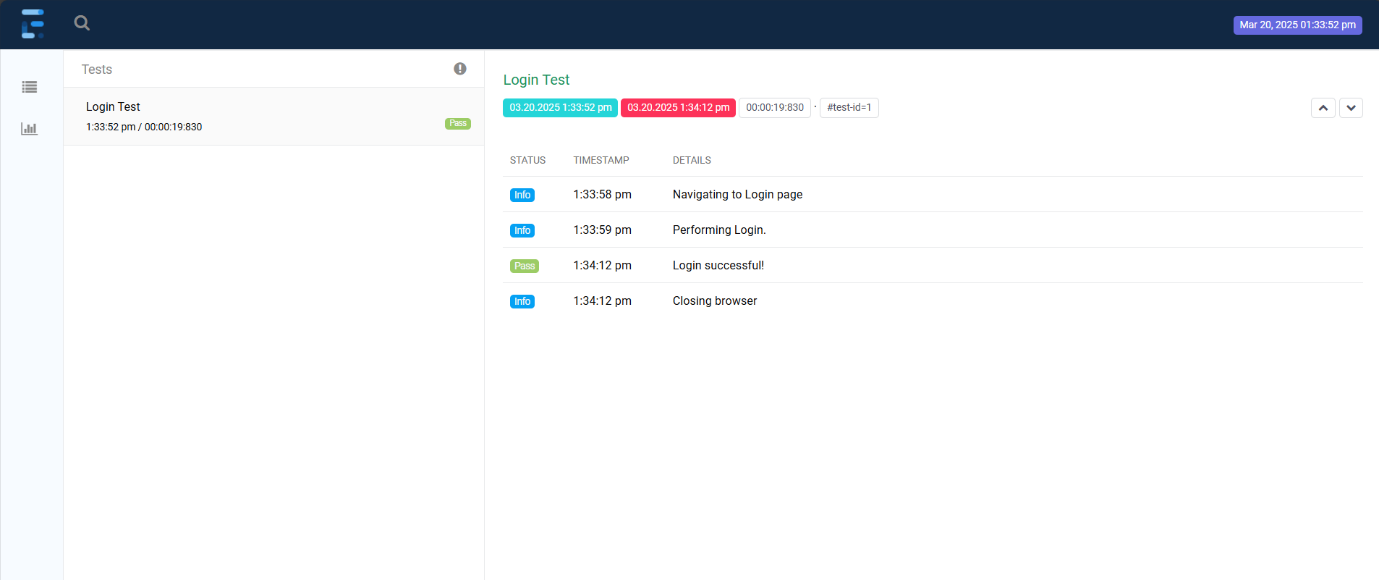
* TestNG XML file for running parameterized registration tests.
* Provides input data (browser) for RegistrationTest.java.

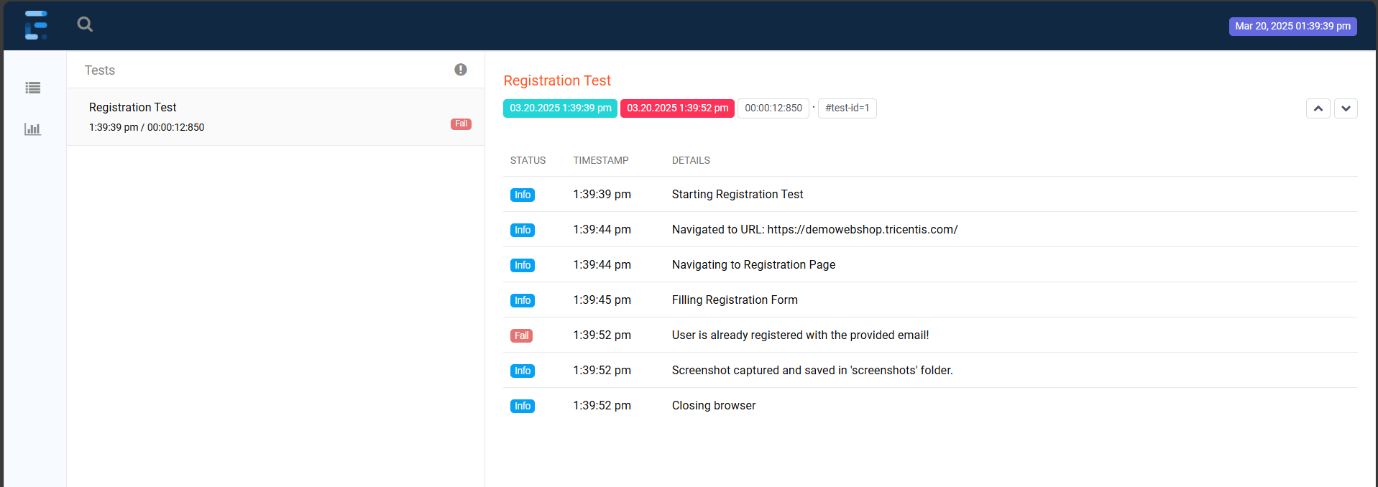
**Target Folder (target/)**

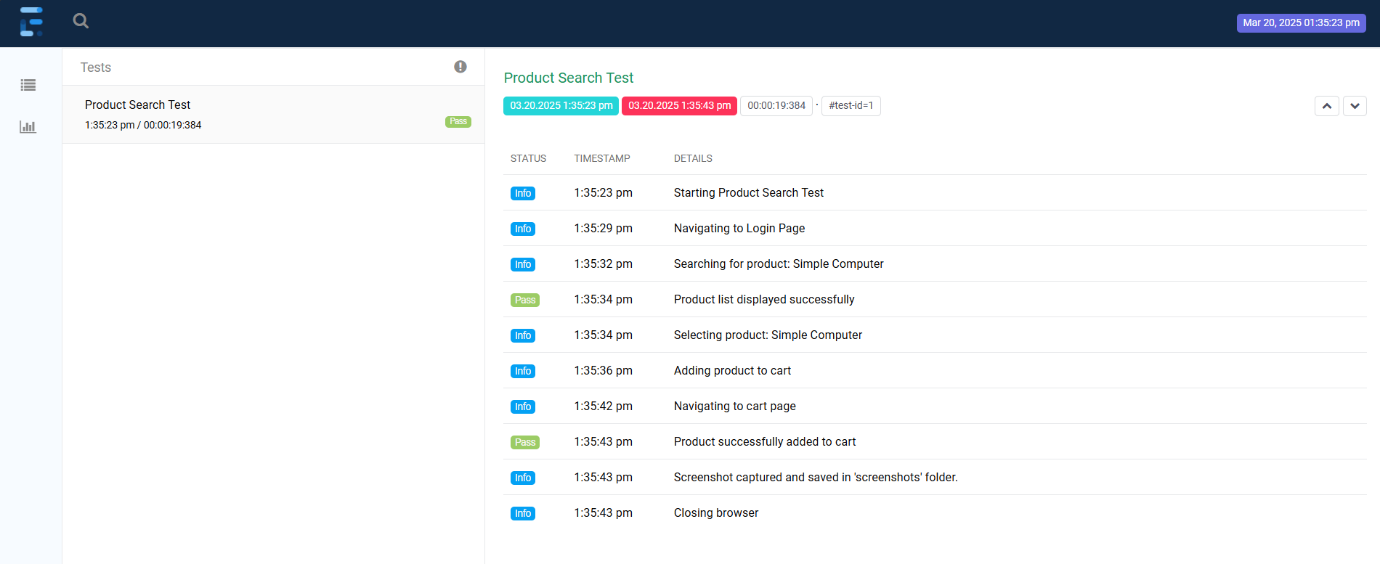
* Stores generated test reports (ExtentReport and Cucumber HTML).
* Contains compiled test outputs (surefire-reports).

**Extent Reports (target/)**

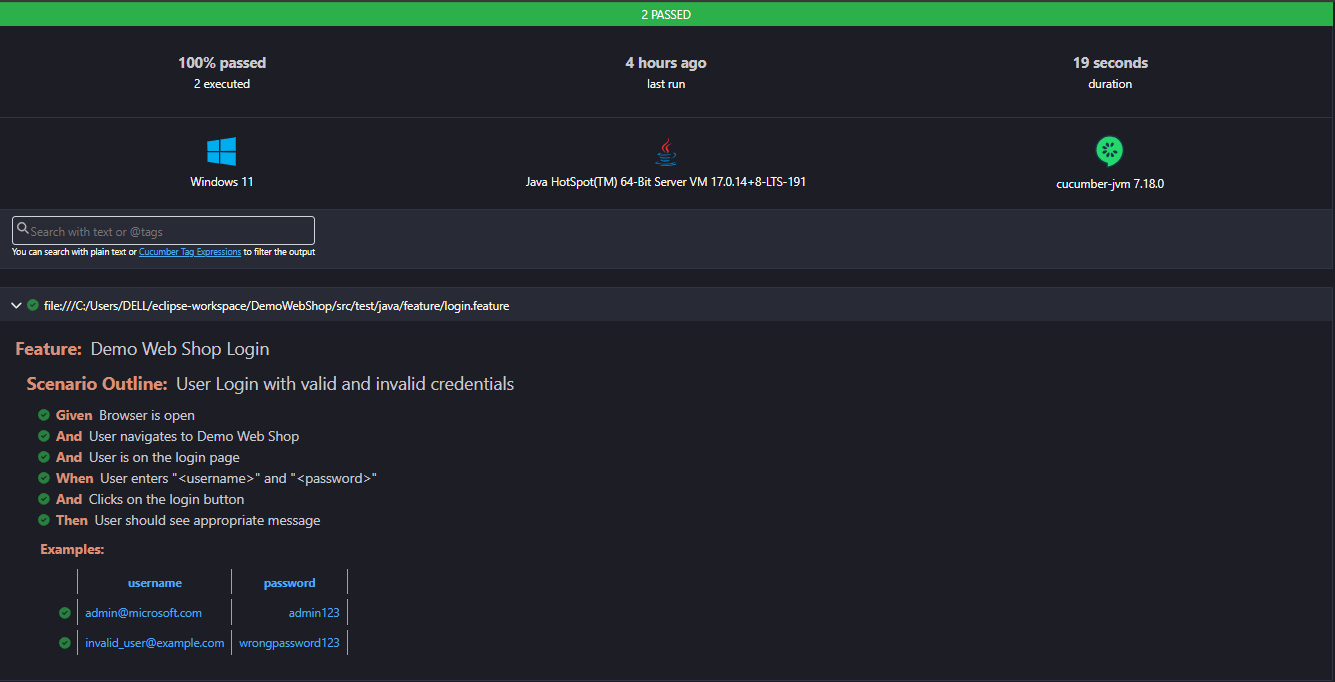
**Login**

****

**Registration  
**

**ProductSearch  
**

**Cucumber HTML Report**

****