

## CAPSTONE PROJECT

### DEMO WEB SHOP

**ABHIRAM S**

**BATCH 4**

**abhiramannur2002@gmail.com**

#### Introduction

Website to be automated: <https://demowebshop.tricentis.com/>

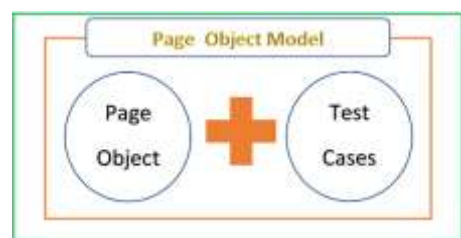
The **Demo Web Shop** ([website link](#)) is an online shopping platform that allows users to browse and purchase various electronic products. The platform provides multiple categories for users to explore, enabling them to view product details, add products to their cart, and place orders.

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 and Allure 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 and fluent 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 testng.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. Adding Multiple Products to the Cart

- **Navigating to product category:** Selects any of the categories (Computer) from the category section.
- **Adding products:** Adds products from different subcategories (Desktops, Notebooks, Accessories)

### 5. Shopping Cart Verification

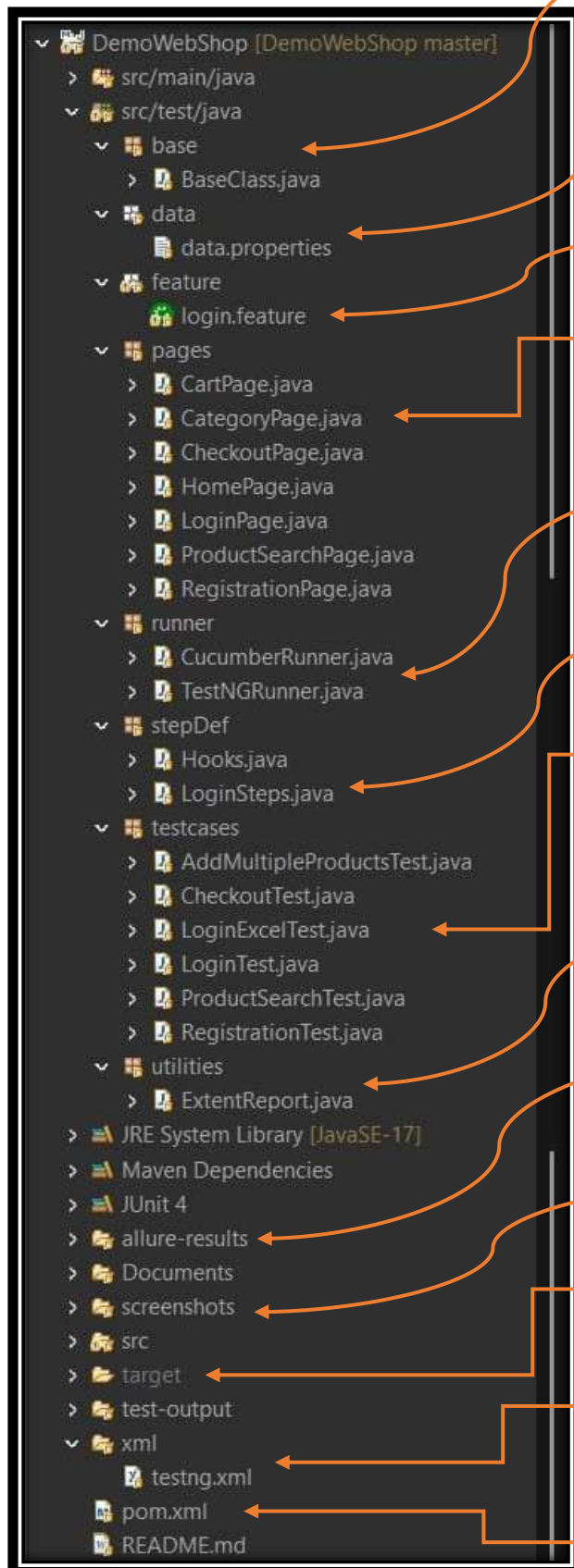
- **Navigating to Cart Page:** Ensures the cart page is accessible.
- **Checking Product Presence:** Verifies if a specific product exists in the cart.

### 6. Checkout and Logout

- **Navigating to Cart Page:** Ensures products are there and perform checkout.
- **Logout:** Verifies checkout and logout is done

## File Structure

Project Name: DemoWebShop



Initializes WebDriver, reads properties, manages browser actions.

Stores configuration details (URL, credentials, etc.)

Defines login scenarios in Gherkin syntax.

Contains Page Object Model classes for interacting with web elements.

Executes Cucumber tests using JUnit or TestNG.

Implements login steps from login.feature.

Contains the test classes that execute various test scenarios

Generates Extent Reports for test execution.

Contains raw test execution data generated by Allure Reports

Contains screenshots taken during validation, etc.

Contains the extent reports

Parameterized TestNG file for registration testing.

Manages dependencies and test execution.

### ***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.

### **CategoryPage.java**

- Handles category navigation.
- Uses PageFactory for element identification.
- Includes methods to navigate to Computers, Desktops, Notebooks, and Accessories categories.
- Uses Explicit Wait (WebDriverWait) to ensure elements are clickable before interacting.

### **CheckoutPage.java**

- Manages the checkout process from cart to order confirmation.
- Uses Explicit Wait (WebDriverWait) for stability.
- Steps included:
  1. Billing Address Selection (Checks if an existing address is available, otherwise enters a new one).
  2. Checkout Steps (Shipping, Payment, and Order Confirmation).
  3. Order Completion & Verification.
  4. View Order Details.

---

## ***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.
  - Implements explicit waits for stable element interactions.
- 

### ***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.

#### **AddMultipleProductsTest.java**

- Adds multiple products from Desktops, Notebooks, and Accessories to the cart.
- Uses Fluent Wait for handling success messages dynamically.
- Integrates with ExtentReports for logging and reporting.
- Verifies products are successfully added to the cart.

#### **CheckoutTest.java**

- Completes the full checkout process (billing, shipping, payment, confirmation).
  - Uses Allure Reporting annotations for structured test reporting.
  - Validates order success and captures a screenshot after checkout.
  - Ensures proper logout after order placement.
-

### ***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/)***

#### **testng.xml**

- TestNG XML file for running parameterized registration tests.
  - Provides input data (browser) for `RegistrationTest.java`.
  - Ensures all tests are executed in order, including Registration, Login, Product Search, Adding Multiple Products, and Checkout.
- 

### ***Target Folder (target/)***

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

### ***Allure Report Folder (allure-results/)***

- Stores Allure Report JSON files generated during test execution.
  - Contains test execution logs, steps, and screenshots for reporting.
  - Used by Allure Command Line to generate a visual test report.
- 

### ***Test-output Folder(test-output/)***

- Stores TestNG-generated reports and logs after test execution.
- Contains `index.html` for a summary of test results.
- Includes logs, screenshots, and emailable reports for debugging.
- Used for analyzing test execution status and failures.



**Login**

The screenshot shows the Selenium IDE interface. On the left, a sidebar contains icons for 'Tests', 'Log', and 'URL'. The main area is titled 'Tests' and displays a test suite named 'Login Test' with a duration of 1:33:52 and a status of 'PASS'. The test suite is expanded, showing a list of steps:

Time	Step	Description
1:33:58	GO TO URL	1:33:58 per Navigating to Login page
1:33:59	GO TO URL	1:33:59 per Performing Login
1:34:12	GO TO URL	1:34:12 per Login successful
1:34:12	GO TO URL	1:34:12 per Closing browser

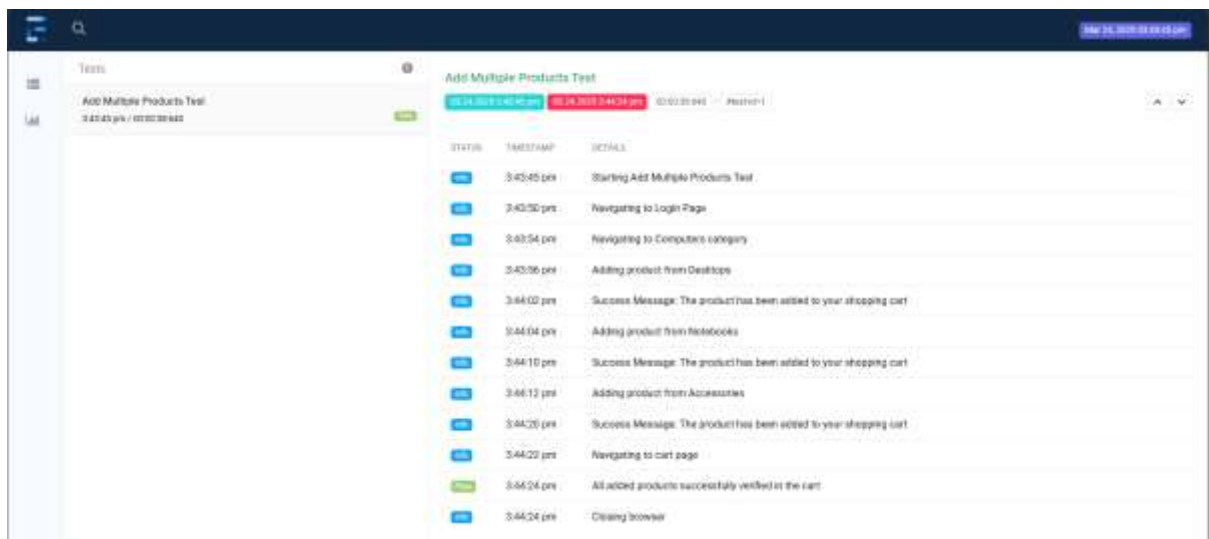
The screenshot displays the Selenium IDE interface. On the left, a sidebar shows the 'Test Suite' structure. The main area shows the 'Registration Test' suite, which is currently running. The test suite is represented by a green bar with a progress indicator at 100%. The test suite is titled 'Registration Test' and has a duration of 1:38:28. The test suite is currently running, with a progress bar at 100%.

Step	Command	Value	Status	Time	Description
1	start		Pass	1:38:28	Starting Registration Test
2	goto	https://demo.guru99.com/otp/	Pass	1:38:44	Navigated to URL: https://demo.guru99.com/otp/
3	goto	https://demo.guru99.com/otp/	Pass	1:39:04	Navigating to Registration Page
4	fill		Pass	1:39:05	Filling Registration Form
5	assert		Fail	1:39:52	User is already registered with the provided email
6	screenshot		Pass	1:39:52	Screenshot captured and saved in 'screenshots' folder
7	close		Pass	1:39:52	Closing browser

The screenshot shows the Cypress Test Runner interface. At the top, there's a dark blue header with the Cypress logo and a search icon. Below the header, the test suite 'Product Search Test' is listed with a green bar indicating it passed. The test suite details show it ran for 1.00:29 (pass) on 08/08/19 at 08:02. The test steps are listed in a table with columns for status, timestamp, and details. The steps are: Starting Product Search Test, Navigating to Login Page, Searching for product: Simple Computer, Product list displayed successfully, Selecting product: Simple Computer, Adding product to cart, Navigating to cart page, Product successfully added to cart, Screenshot captured and saved in 'screenshots' folder, and Closing browser. The first 7 steps have a blue status icon, and the last two have a green status icon.

STATUS	TIMESTAMP	DETAILS
pass	1:05:22 pm	Starting Product Search Test
pass	1:05:26 pm	Navigating to Login Page
pass	1:05:32 pm	Searching for product: Simple Computer
pass	1:05:34 pm	Product list displayed successfully
pass	1:05:34 pm	Selecting product: Simple Computer
pass	1:05:36 pm	Adding product to cart
pass	1:05:42 pm	Navigating to cart page
pass	1:05:43 pm	Product successfully added to cart
pass	1:05:45 pm	Screenshot captured and saved in 'screenshots' folder
pass	1:05:45 pm	Closing browser

## Adding Multiple Products

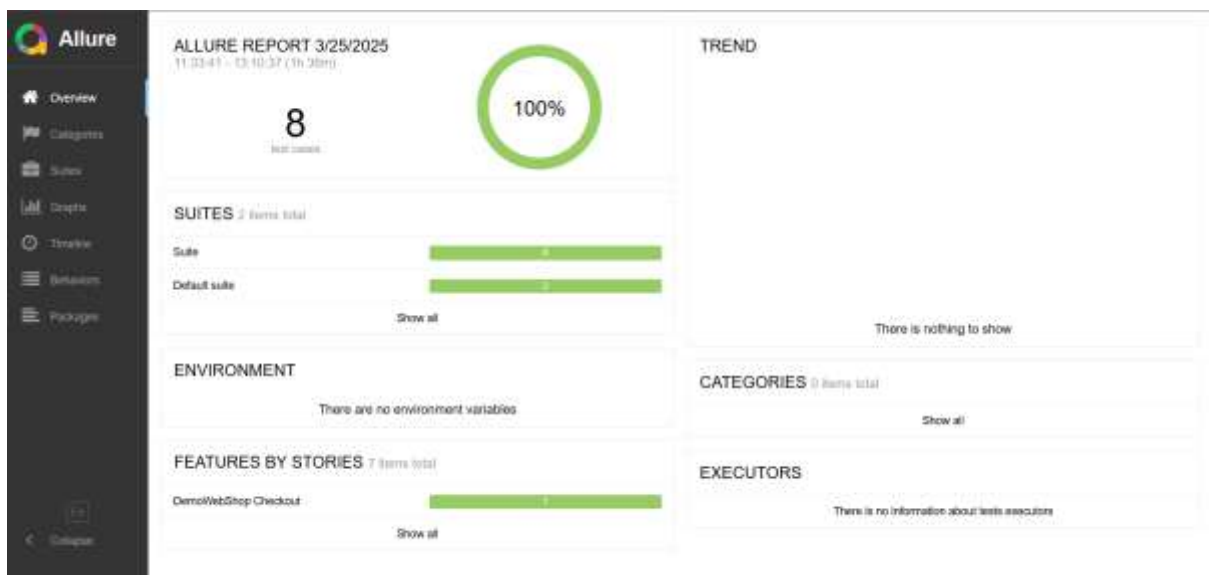


The screenshot displays the Allure Test Runner interface. On the left, a sidebar shows a list of tests, with 'Add Multiple Products Test' selected. The main area shows the details of this test run. At the top, the test name 'Add Multiple Products Test' is displayed along with its duration (00:00:00.000000000) and a status of 'Passed'. Below this, a table lists the steps of the test run, including navigation to the login page, adding products from Desktops, Notebooks, and Accessories, and navigating to the cart page. The final status is 'All added products successfully verified in the cart'.

STATUS	TIMESTAMP	DETAILS
passed	3:43:40 pm	Starting Add Multiple Products Test
passed	3:43:50 pm	Navigating to Login Page
passed	3:43:54 pm	Navigating to Computers category
passed	3:43:56 pm	Adding product from Desktops
passed	3:44:02 pm	Success Message: The product has been added to your shopping cart
passed	3:44:04 pm	Adding product from Notebooks
passed	3:44:10 pm	Success Message: The product has been added to your shopping cart
passed	3:44:12 pm	Adding product from Accessories
passed	3:44:20 pm	Success Message: The product has been added to your shopping cart
passed	3:44:22 pm	Navigating to cart page
passed	3:44:24 pm	All added products successfully verified in the cart
passed	3:44:24 pm	Closing browser

## *Allure Report (allure-results/)*

Ran this command from projects's cmd: "allure serve allure-results"





## Cucumber HTML Report

