
📄 Project Introduction

ShopperStack Automation Testing Project

---

1. Project Overview

This project focuses on **Automation Testing of the ShopperStack E-Commerce Web Application** using **Java, Selenium WebDriver, TestNG, Maven, and Page Object Model (POM)**.
The objective of this project is to automate critical end-to-end user workflows and ensure functional correctness of the ShopperStack application.

The automation framework is designed in a **modular and reusable manner** using utility classes, page object classes, and TestNG XML files to support scalable and maintainable automation testing.

---

2. Application Description

ShopperStack is a **web-based e-commerce application** that allows users to:

- Login into the application
- Search and view products
- Add products to cart and wishlist
- Manage user profile and addresses
- Change account password
- Submit product reviews

The application simulates real-time online shopping behavior and is suitable for functional and regression automation testing.

---

3. Objective of Automation

The main objectives of this automation project are:

- To validate critical business functionalities of the ShopperStack application
- To reduce manual testing effort for repetitive test cases
- To ensure faster and reliable regression testing
- To implement industry-standard automation framework using POM
- To improve test execution accuracy and consistency

---

4. Scope of Automation Testing

In-Scope Functionalities (Based on Your Project)

The following modules are automated in this project:

- User Login and Logout
- Home Page Validation
- Product Search
- Product Details Page Validation
- Add Product to Cart
- Wishlist Functionality
- Address Management
    - Add Address
    - Edit Address
    - Delete Address
- Profile Management
- Change Password and Re-login
- Product Review Submission

Out-of-Scope

- Payment Gateway Automation
- Order Placement
- Performance Testing
- Security Testing
- OTP / CAPTCHA related flows

---

5. Automation Framework Design

The automation framework follows **Page Object Model (POM)** design pattern to maintain separation between test logic and UI elements.

Framework Components (As Implemented)

Generic Utility Package

- `Base_Test.java` – Browser initialization and teardown
- `Webdriver_Utility.java` – Waits, window handling, browser utilities
- `File_Utility.java` – Configuration file handling
- `Java_Utility.java` – Random test data generation
- `FrameWorkConstants.java` – Framework constants
- `Listeners_Utility.java` – Screenshot capture on failure

Page Object Model Package

- `Login_page.java`
- `Home_page.java`
- `SearchResult_Page.java`
- `ProductDetails_Page.java`
- `Wishlist_Page.java`
- `MyProfile_Page.java`
- `Myaddresses_Page.java`
- `AddressForm_page.java`
- `ChangePassword_Page.java`
- `ProductReview_Page.java`
- `Welcome_page.java`

---

6. Test Execution Strategy

- Test cases are executed using **TestNG XML files**
- Separate XML files are created for each functional module:
  - Add Address
  - Edit Address
  - Delete Address
  - Product Search & Add to Cart
  - Wishlist Validation
  - Password Change & Re-Login
  - Product Review Submission
- Test execution reports are generated after each run
- Screenshots are captured automatically for failed test cases

---

7. Tools and Technologies Used

- **Programming Language:** Java
- **Automation Tool:** Selenium WebDriver
- **Test Framework:** TestNG
- **Build Tool:** Maven
- **Framework Design:** Page Object Model (POM)
- **IDE:** Eclipse
- **Version Control:** GitHub

---

8. Challenges Faced

- Handling dynamic web elements
- Synchronization using waits
- Managing test data for multiple scenarios
- Designing reusable utility classes
- Maintaining test stability during execution

---

9. Conclusion

This ShopperStack Automation Testing project provides hands-on experience in automating real-world e-commerce workflows.
The project demonstrates practical knowledge of automation framework design, test execution using TestNG, and effective use of Selenium WebDriver with Java.
