Automated Testing Framework for a Web Application

Project Team

Team Member Name	ID
Omar Magdy Yousef Zain	1122143053
Amr Maged Ahmed	1122131364
Omar Sayed Amin Abdel Majeed	1122120162
Amr Ali Mustafa Gadallah	1122128338

Project Overview

The purpose of this project is to develop and implement an automated testing framework for a web application.
The application under test is a fully functional e-commerce website. The primary goal is to automate the functional testing
of the website using modern tools and methodologies while also performing manual testing to ensure comprehensive test coverage.
We leveraged the Page Object Model (POM) and TestNG framework for automation, focusing on increasing efficiency, reusability,
and scalability of the tests. The automated test cases generate reports for detailed analysis of the test runs.

Tools and Technologies

- Programming Language: Java
- Automation Framework: TestNG
- Design Pattern: Page Object Model (POM)
- IDE: IntelliJ IDEA
- Manual Testing Tools: Jira (for bug tracking and reporting)
- Test Execution: Selenium WebDriver, TestNG
- API Testing Tool: Postman

Scope of the Project

3.1 Automated Testing

- Number of Automated Test Cases: 17 out of 26 test cases
- Framework Used: TestNG with Page Object Model (POM)
- Execution: Test suites were created and executed to run the automated tests, and detailed test reports were generated.
- Features Automated:
  - User Login
  - User Registration
  - Product Search
  - Add to Cart
  - Checkout Process
  - Payment Methods
  - Error Handling for Invalid Input

3.2 Manual Testing

- Number of Test Cases: 26 manual test cases were executed to ensure the application's functionality, including areas not yet covered by automation.

3.3 API Testing

- API Test Cases: 14 APIs were manually tested using Postman, including:
  - User Authentication
  - Product Listings
  - Order Creation
  - Payment Gateway
  - Cart Management
- Bug Report: Bugs and inconsistencies identified during API testing were included in the final bug report.

Test Framework Design

We adopted the Page Object Model (POM) design pattern to enhance the maintainability and scalability of the test scripts.
Each page of the web application was modeled as a class, and UI elements were treated as object repositories. This allowed
for easy updates and reusability of test cases whenever changes were made to the UI.

The testing framework is built using TestNG, which allowed us to:
- Create test suites for organizing the tests
- Manage dependencies between tests
- Generate detailed reports of the test results
- Parameterize tests to handle different data sets

Automated Test Reports: TestNG generated detailed HTML reports after every test execution, providing insights into passed, failed, and skipped tests.

Manual and Automated Test Cases

5.1 Automated Test Cases (17 out of 26)

Test Case ID	Test Scenario	Status
TC001	LoginHappyScenario	Automated
TC002	LogoutTest	Automated
TC003	Login_NegativeScenario	Automated
TC004	AddProductsToCartTest	Automated
TC005	ContactUsTest	Automated
TC006	LoginBeforeCheckout	Automated
TC007	PlaceOrderRegisterTest	Automated
TC008	ProductsTest	Automated
TC009	TestCasesTest	Automated
TC010	RegisterBeforeCheckout	Automated
TC011	RegisterHappyScenario	Automated
TC012	RegisterNegativeScenario	Automated
TC013	RemoveProductFromCart	Automated
TC014	SearchProductTest	Automated
TC015	SubscriptionCartTest	Automated
TC016	SubscriptionTest	Automated
TC017	VerifyProductQuantityInCartTest	Automated

5.2 Manual Test Cases (26 in total)

All 26 test cases, including the 17 automated ones, were tested manually...

5.3 API Test Cases (14 in total)

The API testing covered critical endpoints like User Login, Product Search, Add to Cart, etc.

Challenges Faced

Automation Limitations, API Issues, Browser Compatibility...

Conclusion

This project was a success, with 17 test cases fully automated and extensive manual testing covering all 26 cases...

Future Enhancements

Complete automation of the remaining 9 manual test cases, Implementing CI/CD pipeline, Enhancing API testing coverage...

Appendices

- Test Case Documentation
- Source Code