**DATE: 04/08/202**

**ABSTRACT**

A test plan is a comprehensive document that outlines the approach, scope, resources, schedule, and activities required for testing a software application or system for E-Commerce web application.

**TEST PLAN**

**For E-Commerce Web Application**

**Company Name: CGI**

**Version:**1.0

**Created:** 04/08/2025

**Last Updated:**04/08/2025

**Status:** DRAFT (The status would change to finalized post the BA, PM and dev team review and sign off)

**Table of Contents**

1. Test Plan Identifier
2. Introduction

   2.1 Overview of the Software Application

   2.2 Purpose of the Document

1. Test Scope

   3.1 In Scope

   3.2 Out of Scope

1. Extent of Testing
2. Features to Test
3. Features Not to Test
4. Test Objectives

   7.1 Testing Goals

   7.2 Testing Guidelines

1. Assumptions and Dependencies
2. Risk Analysis
3. Testing Strategy

   10.1 Manual Testing Approach

   10.2 Automation Testing Approach

   10.3 Execution and Efficiency

   10.4 Reporting and Framework Validation

1. Features to be Tested
2. Roles and Responsibilities

   12.1 QA Lead / Test Manager

   12.2 Manual Tester

   12.3 Automation Tester

   12.4 Project Coordinator / Team Member

   12.5 Reporting & Documentation Owner

1. Test Schedule
2. Entry Criteria
3. Exit Criteria
4. Suspension Criteria
5. Resumption Criteria
6. Test Environment
7. Communication Plan
8. Testing Tools to be Used
9. Document Control

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **E-Commerce Web Application** | | | |  |
|  |
| **Sl. No.** | **Team Members** | **Designation** | **Remark** |  |
| **1** | **Priyanka V** | **SE** |  |  |
| **2** | **Kaviyanjali M** | **SE** |  |  |
| **3** | **Asha R** | **SE** |  |  |
| **4** | **Keerthana V S** | **SE** |  |  |

**1. Test Plan Identifier**

|  |  |
| --- | --- |
| **Field** | **Details** |
| **Project Name** | **Hybrid Testing Strategy for E-Commerce Web Application** |
| **Version** | **version\_1.0** |
| **Unique ID** | **A6\_EcomApp\_01** |
| **Date** | **04-Aug-2025** |

**2. Introduction**

This test plan outlines the testing approach for an e-commerce web application that includes both manual and automation testing using Selenium and Java. The document is intended for QA team members, developers, and project stakeholders to understand the testing coverage, process, tools, and deliverables involved in the project.

**3. Overview of the Software Application**

The application is a mock e-commerce website (e.g., Automation Exercise / OpenCart Demo) where users can:

* Sign up / Log in
* Search for products
* Add items to cart
* Proceed to checkout and payment

It is a web-based application accessed through Chrome/Firefox. Users include regular customers, guest users, and admin (optional for test reference).

**4. Purpose of the Document**

The purpose of this document is to define the scope, strategy, and approach for testing the e-commerce application. It serves as a reference for the QA team to plan, execute, and report both manual and automated tests consistently and effectively. This ensures key functionalities are validated and project quality goals are met.

**5. Test Scope**

**In Scope:**

* User Signup / Login
* Product Search
* Add to Cart
* Checkout Process
* Payment Flow
* Smoke Testing
* Functional Testing
* Regression Testing
* Automation using Selenium + Java + TestNG

**Out of Scope:**

* Load/Performance Testing
* Security Testing
* API Testing
* Admin Dashboard features (unless mentioned)

**6. Extent of Testing**

* Both manual and automated testing will be performed.
* Manual testing will cover functional, smoke, UI, and basic regression tests.
* Automation will cover major regression test cases using Java + Selenium with TestNG.
* Testing will be conducted on Chrome and Firefox browsers.
* Backend or API-level testing is not included.

**7. Features to Test**

* User Signup/Login – Test user registration, login, and logout functionality.
* Product Search – Verify users can search for products using keywords.
* Add to Cart – Ensure products can be added or removed from the cart.
* Checkout Process – Check billing, shipping, and mock payment flow.
* Order History – Test viewing past orders after a purchase.
* Basic UI Testing – Ensure elements are properly aligned and functional.
* Cross-Browser Testing – Run tests on Chrome and Firefox.

**8. Features Not to Test**

* Real Payment Gateway– No actual payment integration testing.
* Security Features– No deep testing of password security or encryption.
* Performance Testing – Load and speed tests are not included.
* Mobile Testing– Only desktop browser testing will be done.
* Admin Panel– Backend features for admin (like adding products) are out of scope.
* Multi-language/Currency Support – Only default language and currency tested.

**9. Test Objective**

The objective of this project is to ensure the reliability, functionality, and usability of an e-commerce web application by applying both manual and automated testing techniques.

* To verify that critical user flows like Login, Search, Add to Cart, and Checkout work as expected.
* To identify defects early using manual test cases.
* To reduce repetitive effort through Selenium automation using Java.
* To improve test coverage and efficiency using a hybrid testing strategy.

**10. Provide the guidelines of Testing.**

* Follow the test plan and stick to the defined scope.
* Write clear test cases with expected and actual results.
* Prioritize testing of critical features like login, cart, and checkout.
* Perform smoke, functional, and regression testing.
* Use clean and reusable test data.
* Report bugs clearly with proper details.
* Use Page Object Model (POM) in automation.
* Validate with assertions and generate reports.
* Test on both Chrome and Firefox.
* Re-test after bug fixes.

**11. Assumptions and Dependencies**

* The application under test is stable and deployed in a test environment
* All functional requirements are clearly defined and accessible.
* Manual and automated test cases will be created based on the available feature set.
* Required tools such as Java, Selenium, Maven, TestNG, ExtentReports, Eclipse, and browsers (Chrome/Firefox) are installed and configured.
* The team has basic knowledge of Java programming and automation frameworks.
* Internet connection is available for testing hosted demo sites like Automation Exercise or OpenCart.
* All third-party services (e.g., payment gateways in the mock site) are simulated or mocked.
* Test data will be assumed or generated unless real credentials are provided.

**12. Risk Analysis**

|  |  |  |
| --- | --- | --- |
| **Risk** | **Impact** | **Mitigation Strategy** |
| Unstable application during testing | High | Coordinate with developers to freeze builds before testing starts |
| Delay in requirement availability | Medium | Use exploratory testing to cover features while waiting |
| Test environment not ready | High | Prepare fallback with local setups or mock environments |
| Lack of automation skills | Medium | Allocate time for learning or use recorded tools for basic tasks |
| Incomplete test data | Medium | Generate test data or simulate scenarios |
| Browser version incompatibility | Low | Test on multiple versions of Chrome/Firefox |
| Test script maintenance due to UI changes | Medium | Use POM design pattern and reusable functions |

## **13. Testing Strategy**

To achieve this goal, a hybrid testing strategy is employed that integrates both manual testing and Selenium-based automation testing using Java. The strategy ensures full lifecycle testing—from requirement validation to regression testing—by aligning the right method with the right test type:

### 1. Manual Testing Approach

* Focuses on exploratory testing, UI/UX validation, and new/complex workflows where human judgment is valuable.
* Used during initial development phases to validate the look, feel, and behavior of new features.
* Helps in identifying usability issues, visual inconsistencies, and unexpected behavior that automation might miss.

### 2. Automation Testing Approach

* Focuses on regression testing, data-driven testing, and repeated workflows like login, cart operations, and checkout.
* Implements a robust automation framework using Selenium WebDriver, Java, TestNG, and Maven.
* Applies the Page Object Model (POM) design pattern to maximize test reusability, modularity, and maintainability.
* Validates application workflows with assertions and generates professional test execution reports.

3. Execution and Efficiency

* Both manual and automated tests are executed in parallel to compare execution time, coverage, and effort.
* Automation scripts are designed to cover high-impact, repeatable features to reduce test cycle time.
* Manual testing complements automation by focusing on areas where human insight and exploratory testing are more effective.

4. Reporting and Framework Validation

* Utilizes tools like ExtentReports for detailed reporting of test results.
* Logs bugs and tracks their lifecycle using JIRA or Excel logs.
* Demonstrates how frameworks improve maintainability and test reusability in real-world QA setups.

## **14. Features to be Tested**

To validate the strategy, the following functional and non-functional features are selected from a mock e-commerce web application (e.g., Automation Practice or OpenCart Demo):

### 1. User Authentication

* Signup with valid and invalid data
* Login with correct/incorrect credentials
* Logout and session handling

### 2. Product Search & Filtering

* Search for products using keywords and categories
* Apply filters (price, size, color)
* Navigate product listings and view product details

### 3. Shopping Cart Management

* Add items to the cart from listings and product pages
* Update or remove items from the cart
* Verify correct price calculations (subtotal, tax, total)

### 4. Checkout & Payment

* Proceed to checkout with or without login
* Fill out address and shipping details
* Select payment method (dummy COD or demo card)
* Confirm and place order, validate success page

### 5. User Account Management

* Edit profile and personal information
* View order history
* Change password and update address

### 6. Functional Validations

* Form validations (email format, required fields)
* Proper alerts and success/error messages
* Page redirection and workflow transitions

### 7. Cross-Browser Compatibility

* Execute tests on Chrome, Firefox, and optionally Edge to ensure UI and functions work uniformly

**15. Roles and Responsibilities**

### 1. QA Lead / Test Manager

* Define overall test strategy and test plan.
* Allocate tasks and manage timelines for manual and automation testers.
* Monitor test progress and defect resolution.
* Ensure alignment with project objectives and reporting requirements.
* Review final reports, dashboards, and metrics before submission.

2. Manual Tester

* Analyze requirements and write detailed test scenarios and test cases.
* Execute **smoke**, **functional**, **UI**, and **regression** tests manually.
* Log and track defects found during manual testing.
* Perform exploratory testing and document edge case behaviors.
* Support retesting and regression cycles after bug fixes.

### 3. Automation Tester

* Set up and maintain the **Selenium + Java + TestNG + Maven** automation framework.
* Design reusable components using the **Page Object Model (POM)** design pattern.
* Write and execute automated test scripts for major workflows (e.g., login, add to cart).
* Validate outputs using assertions and generate automated reports.
* Integrate test data from Excel/CSV using Apache POI or OpenCSV.
* Debug scripts and maintain test code as the application evolves.

4. Project Coordinator / Team Member

* Help maintain documentation including test cases, bug logs, and summary reports.
* Assist with test data preparation and browser compatibility testing.
* Participate in regular reviews and team discussions.
* Support manual or automation efforts wherever required.

5. Reporting & Documentation Owner

* Consolidate results from both manual and automation executions.
* Create the Test Summary Report, including metrics like test coverage, bug status, pass/fail ratio, and effort comparison.
* Maintain screenshots and logs from test runs.
* Document lessons learned and improvements for future testing cycles.

**16. Test schedule**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sl. NO** | **PHASE** | **ACTIVITY** | **START DATE** | **END DATE** | **DURATION** |
| 1 | Requirement Analysis | - Understand E-Commerce app workflows  - Identify testable features | 4/8/25 | 5/8/25 | 2 days   |  | | --- | |  | |
| 2 | Test Planning | - Define test objectives and scope  - Identify test tools and resources  - Draft test plan document | 6/8/25 | 7/8/25 | |  | | --- | |  |   2 days |
| 3 | Test Case Design | - Write manual test cases for key modules  - Identify scenarios for automation | 11/8/25 | 13/8/25 | 3 days |
| 4 | Environment Setup | - Configure Selenium + Java + Maven + TestNG  - Prepare browser drivers and folder structure | 14/8/25 | 15/8/25 | 2 days |
| 5 | Manual Test Execution | - Execute functional, UI, smoke, and regression tests manually | 18/8/25 | 21/8/25 | 4 days |
| 6 | Automation Scripting | - Develop POM-based Selenium scripts  - Use TestNG for assertions and grouping  - Include data-driven tests | 22/8/25 | 29/08/25 | 6 days |
| 7 | Automation Execution | - Run automated scripts across modules  - Generate test reports (Extent/Allure/TestNG XML) | 1/9/25 | 2/9/25 | 2 days |
| 8 | Defect Reporting And Retesting | - Log bugs in bug tracker (JIRA/Excel)  - Retest fixed bugs  - Execute regression tests | 3/9/25 | 4/9/25 | 2 days |
| 9 | Reporting And Documentation | - Summarize manual vs automation test results  - Prepare test summary report  - Finalize documentation | 5/9/25 | 6/9/25 | 2 days |
| 10 | Final Review | - Team walkthrough  - Submit project artifacts  - Final QA review and presentation | 7/9/25 | 8/9/25 | 2 days |

## **17. Entry Criteria**

Entry criteria define the prerequisites that must be met before beginning the test execution phase in both manual and automation testing.

### 1. Requirements Finalized

* Functional and non-functional requirements of the e-commerce web application must be documented and reviewed.
* Testable use cases and workflows (e.g., login, cart, checkout) must be clearly defined.

### 2. Test Plan Approved

* The test plan, including strategy, scope, resources, tools, and timelines, must be completed and signed off.

### 3. Test Cases Prepared

* Manual test cases for all major modules (e.g., user login, product search, add to cart, checkout) must be written, reviewed, and approved.
* Automation scenarios must be identified and mapped.

### 4. Test Environment Ready

* Browsers (Chrome, Firefox, etc.) and required test data are available.
* Selenium WebDriver and Java environment (Eclipse/IntelliJ, Maven, TestNG) must be set up and working.

### 5. Automation Framework Setup

* The basic Page Object Model (POM)-based framework must be implemented.
* Folder structure for test scripts, resources, test data, and reports should be organized.

### 6. Access to Tools

* Access to required tools such as:  
  + Test management (Excel/TestLink/JIRA)
  + Defect tracking (JIRA/Bugzilla)
  + Reporting tools (Extent Reports/TestNG reports)

### 7. Roles Assigned

* Team members must have clearly defined responsibilities such as test designer, automation developer, manual tester, etc.

### 8. Smoke Test Completed

* A basic smoke test run (manual or automated) must be completed to confirm application stability.

## **18. Exit Criteria**

Exit criteria define the conditions that must be satisfied before testing can be formally concluded. These ensure the project goals have been met and deliverables are complete.

### 1. Test Case Execution Completed

* All planned manual and automated test cases must be executed.
* Critical and high-priority test cases must have passed successfully.

### 2. Defect Resolution

* All critical and major defects must be fixed, re-tested, and closed.
* Medium and low-priority bugs should either be resolved or documented for future releases with acceptable business impact.

### 3. Test Coverage Achieved

* 100% coverage for critical features like:  
  + Signup/Login
  + Product Search
  + Add to Cart
  + Checkout and Payment
* Minimum of 90–95% test case coverage achieved overall.

### 4. Regression Testing Completed

* Final regression test cycle must be completed to ensure no new defects were introduced by recent changes or fixes.

### 5. Automation Execution

* All automated test scripts must run successfully without failure.
* Results and logs must be generated and validated using reports (e.g., TestNG or ExtentReports).

### 6. Test Summary Report Submitted

* A final test summary report must be prepared, including:
* Total test cases
* Pass/fail counts
* Defect status
* Automation coverage
* Lessons learned and recommendations

**19. Suspension Criteria:**

* Build is not stable for testing.
* Critical functionality is broken.
* The environment is down or inaccessible.
* High number of blocker defects found.
* Automation scripts are failing continuously due to application changes.

**20. Resumption Criteria:**

* Stable build deployed.
* Blocker defects resolved and verified.
* Test environment restored
* Updated scripts deployed if needed.
* QA team receives go-ahead to resume.

**21. Test Environment:**

* OS: Windows 10
* Browsers: Chrome, Firefox, Edge (latest versions)
* Automation: Java, Selenium WebDriver, TestNG
* Tools:Git, Excel
* Database: MySQL (QA instance)

**22. Communication Plan:**

* Daily stand-up for status updates
* Defect triage every alternate day.
* Progress reporting via email and Slack.
* Use JIRA for defect logging and tracking.
* Final test summary shared with stakeholders.

**23.Testing Tools to be Used:**

* Manual: Excel for test case management.
* Automation: Selenium WebDriver, Java, TestNG.
* Reporting: Extent Reports for automation results.
* Version Control: Git & GitHub.
* CI/CD Integration: Jenkins for nightly runs.

**24. Document Control:**

* Test Plan Version 1.0 created on August 04, 2025.
* Author: Team A6
* Reviewers: Project Manager.
* Approval to be documented in change log
* Document stored in shared project repository