Wipro Java Selenium-Batch 10, Team-06

Test Plan

Demo Web Shop

ChangeLog

Version	Change Date	By	Description
version number	Date of Change	Name of person who made changes	Description of the changes made

TABLE OF CONTENTS

1.Introduction

This document outlines the testing strategies, workflows, and methodologies adopted to verify the quality and functionality of the Demo Web Shop application. The objective is to ensure that all features perform as intended across various browsers and devices. Testing is conducted using a combination of approaches to validate that the application meets the expected functionality and quality standards.

1.1 Scope

1.1.1 In Scope

The following features will be covered in testing:

- Full customer journey:
  Registration → Login → Search → Product Details → Add to Cart → Checkout → Order Confirmation
- Search functionality, including edge cases (e.g., invalid terms, special characters)
- Shopping cart operations: **Add**, **update quantity**, **remove items**
- Payment & shipping options (e.g., credit card, purchase order)
- UI validation: Input fields, error messages, visibility of UI elements
- Cross-browser and basic mobile UI compatibility check.

1.1.2 Out of Scope

The following areas will **not** be covered during this test cycle:

- Third-party integrations, like marketing emails or external payment gateway performance testing
- Performance/load testing at scale
- Accessibility testing

1.2 Quality Objective

- Validate all core user flows from registration to order placement

- Ensure that business rules like product availability, price validation, and discounts are correctly implemented
- Verify UI compatibility across major browsers
- Identify bugs early and ensure confidence in system stability before release

1.3 Roles and Responsibilities

- **Test Lead / QA Lead**: Define test strategy and create the test plan, assign tasks and monitor overall test progress.
- **QA Engineer (Manual Tester)**: Execute test cases and report bugs, perform UI and regression testing.
- **Automation Tester**: Automate test cases using Selenium, maintain and update automation scripts.
- **Developer**: Perform unit testing before code delivery, fix defects reported by QA.
- **Product Owner / Business Analyst**: Define requirements and acceptance criteria, review and approve features during UAT (User Acceptance Testing).

2.Test Methodology

Overview

The testing methodology for the Demo Web Shop application follows the Agile development process. Testing is integrated into every phase of the software development lifecycle, beginning from sprint planning to final release. The QA team works closely with the development and product teams to understand user stories, define acceptance criteria, write and execute test cases, and provide continuous feedback. Testing is iterative and continuous, with each sprint delivering a potentially shippable product increment.

Test Levels

Test Levels specify the stages and types of testing required for verifying the application, based on project scope, time, and budget constraints. For the Demo Web Shop, we define the following core testing levels:

- **Unit Testing**

Conducted by developers, unit testing focuses on verifying the smallest components—typically a function or a module—such as product listing logic, input validation routines, or helper utility methods. This level helps catch defects early in the development cycle and supports rapid iteration.

- **Integration Testing**

Once individual components are unit-tested, integration testing verifies that these components interact correctly. For example, submitting the registration form triggers data to the backend, or adding to cart updates the appropriate session and database entries. These tests ensure modules work cohesively beyond isolated units.

- **System Testing**

System testing evaluates end-to-end workflows of the complete Demo Web Shop site in an environment that mimics production. Typical scenarios include browsing, search and filters, user registration and login, shopping cart and checkout, order confirmation, newsletter signup, and product reviews. Both functional and non-functional aspects (such as performance and usability) are validated at this level.

- **Acceptance Testing**

Acceptance Testing (or User Acceptance Testing—UAT) assesses whether the application meets business requirements and user expectations. Stakeholders (e.g. product owners) perform realistic scenarios—such as placing an order, applying coupons, checking order history—to

confirm the application is ready for production release. Acceptance tests are usually based on predefined acceptance criteria.

● Smoke (Build Verification) Testing

Often executed after each build, smoke testing performs a quick validation of critical flows—e.g. login, homepage accessibility, adding to cart. These basic checks ensure a build is stable enough for deeper testing, saving time if the build fails early.

● Regression Testing

Regression testing ensures that new changes or enhancements haven't adversely impacted existing features. For the Demo Web Shop, regression includes core user flows like search, cart, checkout, login, and product reviews. Regression cycles are automated as much as possible to support frequent sprint releases.

Test Level	Objective for Demo Web Shop
Unit Testing	Verify individual components/modules before integration
Integration Testing	Validate interactions between modules, e.g. UI and backend flows
System Testing	Validate end-to-end functionality and non-functional requirements
Acceptance Testing (UAT)	Confirm application meets business/user requirements based on acceptance criteria
Smoke Testing	Quick sanity checks on critical paths post-build
Regression Testing	Ensure new features don't break existing functionality across releases

Agile-Specific Considerations

● **Unit and Integration Testing**: Executed continuously by developers as features are developed—ideally before QA picks up testing in a sprint.

● **Smoke Testing**: Performed immediately after build deployment to the QA environment to confirm build stability.

● **System, Regression & Acceptance Testing**: Managed by QA in each sprint; key scenarios are automated for efficiency.

● **Regression Suite Updates**: Updated regularly to cover newly added features (e.g. newsletter signup, product reviews), ensuring automation keeps pace with development.

Bug Triage

The purpose of bug triage is to systematically evaluate each defect to determine its appropriate resolution route and to prioritize bug fixes effectively. This process ensures that high-impact issues are addressed promptly while lower-impact items are scheduled appropriately, helping maintain sprint momentum and quality release cycles.

Key Triage Outcomes

1.**Resolution Path Definition**
During each triage session, the QA lead and development team classify each bug into one of several resolution types:

○ **Fix in Current Sprint**: Critical or blocking bugs that must be resolved immediately (e.g., checkout failure, login outage).

○ **Fix in Future Sprint**: Medium- to low-priority bugs with minimal impact on current functionality.

○ **Will Not Fix**: Issues classified as non-reproducible, cosmetic, or outside the current release scope.

2.**Defect Prioritization & Scheduling**
Each bug is assessed by **severity** (Critical, Major, Minor) and **priority** (P1, P2, P3). Bugs are then scheduled based on their impact and urgency:

○ **P1/Critical**: Blockers that disrupt essential workflows—assigned immediately.

○ **P2/Major**: Important bugs that may affect user experience or business logic—planned in the current sprint if feasible.

○ **P3/Minor**: Cosmetic issues or edge-case scenarios—tracked for resolution in later sprints or backlog refinement.

Triage Process Workflow

1.**Bug Identification**: QA logs defects with clear steps to reproduce, expected vs actual behavior, screenshots, and assigned priority/severity in the defect tracking tool (e.g., JIRA).

2.**Triage Meeting**: Conducted daily or at least mid-sprint. The QA Lead, Dev Lead, and Product Owner review newly logged bugs and update their status.

3.**Assignment & Scheduling**:

○ Critical bugs are assigned to developers for immediate resolution.

○ Others are assigned based on sprint capacity and business value.

4.**Tracking & Review**: Each bug's status is updated and monitored through to closure, with retesting by QA once fixes are delivered.

5.**Post-Triage Communication**: Summaries are shared with stakeholders to maintain transparency and ensure alignment.

Benefits of Effective Triage

● Prevents backlog overflow and confusion by filtering out invalid or low-priority bugs promptly.

● Ensures critical defects are resolved before release, protecting key user flows (e.g., adding to cart, placing orders, user login).

● Facilitates sprint planning and clear allocation of development and QA effort.

● Encourages shared ownership and accountability between QA, Development, and Product teams.

Sample Triage Schedule & Roles

Frequency	Participants	Main Activities
Daily (or Mid-Sprint)	QA Lead, Dev Lead, Product Owner	Review new bugs, assign severity/priority, and assign fixes
Post-Triage Sync	QA Lead, Dev Lead	Confirm scheduled fixes and review retesting plan
Retest Sessions	QA Tester + Dev	Verify bug fixes and update bug status

Suspension Criteria and Resumption Requirements

A. Suspension Criteria

Testing activities (whole or in part) may be temporarily halted if one or more of the following critical conditions arise:

● **Critical Environment Failures**: Major issues in the QA environment (e.g. database down, corrupted test data, or failed configurations) that prevent meaningful test execution.

● **Unstable or Broken Builds**: If the nightly build is repeatedly failing or unstable—such as frequent crashes or missing critical functionality—testing should be paused until a stable build is available.

● **Blocking or Critical Defects**: Presence of high-severity defects that completely obstruct test progress in key workflows (e.g. checkout, login, cart operations).

● **Major Infrastructure Outage**: Network, server, or CI/CD pipeline outages affecting environment accessibility or deployment capabilities.

● **Test Corruption**: If the test suite itself is broken—e.g., automated scripts fail due to framework errors, broken locators, or outdated configurations—manual or scripted tests must be paused for timely maintenance.

During suspension, no new test execution should be commenced, and all ongoing test work should be carefully documented (including logs, errors, and failure contexts).

B. Resumption Requirements

Testing may safely resume only when clear prerequisites are met to ensure stability and effectiveness:

● **Stable QA Environment Restored**: All environment components (frontend, backend, database) are functional and verified, with successful smoke test results confirming readiness.

● **Deployment of a Valid Build**: A new, stable build has been deployed that addresses previous failures and passes sanity checks.

● **Critical Bugs Resolved and Verified**: All blocking/high-severity defects have been fixed, deployed, and validated by QA.

● **Automated Test Suite Ready**: Regression or smoke automation scripts updated and executed successfully—any broken tests must be fixed prior to continuation.

● **Infrastructure Functionality Restored**: CI/CD, network, server, and service configurations are operational and tested.

● **Approval from Stakeholders**: QA lead and Dev lead confirm the status post-suspension and formally agree on restarting test activities.

This clear and comprehensive definition of suspension and resumption criteria helps keep the Agile testing cycle on track—ensuring that test interruptions are deliberate, controlled, and the team acts only when conditions are safe and reliable for effective testing.

Test Completeness

Test completeness defines the specific criteria that must be met to consider testing activities for a sprint or release cycle as fully complete. These criteria ensure that the software has been thoroughly validated, all planned test objectives have been achieved, and the product is ready for release or progression to the next phase.

In the context of Agile development for the Demo Web Shop, the following key metrics and checkpoints will be used to determine test completeness:

Test Completeness Criteria

1. **100% Test Case Execution**
All manual and automated test cases planned for the sprint (including functional, regression, integration, and UI tests) must be executed at least once. Any deviations should be formally reviewed and justified.

2.**Critical & High-Priority Bugs Resolved**
All defects marked as Critical or High severity that impact key features such as user login, product purchase, checkout flow, or payment integration must be resolved and retested. No unresolved high-impact bugs should remain open.

3.**Test Coverage Goals Met**

- **Requirement Coverage**: All user stories and acceptance criteria are fully covered by test cases.

- **Code Coverage (for automated tests)**: A minimum threshold (e.g., 80%) is achieved where applicable.

4.**All Regression Tests Passed**
The existing functionalities (especially core workflows like cart operations, payment gateway, order history, and user profile updates) must pass all relevant regression test cases after each code change.

5.**No Critical Defects in UAT/Pre-Production**
Any bugs found in the UAT or staging environment should be reviewed. All showstopper or user-facing issues must be closed before Go-Live approval.

6.**Test Summary Report Approved**
A final test summary must be prepared and reviewed by QA Lead and Product Owner. This includes test coverage metrics, defect analysis, pass/fail statistics, and sign-off documentation.

7.**Agile Definition of Done (DoD) Satisfied**
All sprint-level "Done" criteria including unit testing, peer code reviews, deployment validation, and QA sign-off are completed per Agile DoD standards.

8.**Non-Functional Requirements Verified (if applicable)**
Tests related to performance (e.g., page load time), responsiveness (e.g., mobile view rendering), security (e.g., login validation, session timeout), and browser compatibility are executed and passed.

Example – Functional Areas Covered in Test Completion for Demo Web Shop:

- User Registration & Login
- Product Browsing and Filtering
- Shopping Cart & Wishlist Functionality
- Checkout & Payment Processing
- Order Confirmation and History
- Email Notifications
- Profile Management
- Admin Panel (if applicable)

Establishing these well-defined completion criteria ensures that the Demo Web Shop platform is thoroughly validated before it reaches customers, supports Agile sprint goals, and reduces production-level risks significantly.

3.Test Deliverables

This section outlines the test deliverables associated with the testing lifecycle of the Demo Web Shop application. These deliverables ensure effective tracking, validation, and reporting of the testing process throughout each phase of the software development lifecycle (SDLC).

Deliverable	**Description**	**Relevance to demowebshop.tricentis.com**
Test Plan	Defines scope, approach, resources, schedule, test environment details	Clarifies testing objectives for product catalog, cart flows, checkout, and user account features
Test Cases	Step-by-step scenarios with inputs/expected results	Covers key workflows like browsing products, adding to cart, order checkout, managing product reviews
Requirement Traceability Matrix (RTM)	Maps each requirement to associated test cases	Ensures every feature (search, checkout, payment, categories) is validated against functional specifications
Bug Reports	Logs issues with severity, reproduction steps, screenshots, and status	Captures defects such as payment failures, missing UI elements, browser-specific rendering issues
Test Strategy	High-level approach describing tools, levels of testing, and test environment	Includes functional testing via Selenium, API validation (if APIs exist), performance checks, and compatibility

		testing
Test Metrics	Quantitative measurements such as test coverage, defect density, pass/fail rate	Enables tracking of test progress, overall stability, and readiness for release based on data
Test Summary Report	Summary of test results for major application blocks	Provides clear pass/fail summary for features like Login, Payment, Cart, Search, and My Orders
Customer Sign-Off	Formal approval post successful UAT and closure of open defects	Confirms that the webshop—product search, user login, cart checkout, reviews—is tested and meets expectations

4.Resource & Environment Needs

4.1 Testing Tools

For efficient execution and tracking of the testing process, the following tools will be `` utilized:

1. **Requirements Tracking Tool** – To document, manage, and track requirements throughout the project lifecycle.
   Example: JIRA, IBM Rational DOORS, Confluence.

2. **Bug Tracking Tool** – To log, prioritize, and track defects until resolution.
   Example: JIRA, Bugzilla, MantisBT.

3. **Automation Tools** – To execute regression and repetitive tests automatically, saving time and improving accuracy.
   Example: Selenium, UFT (Unified Functional Testing), Cypress, Playwright.

4. **Performance Testing Tools** – To measure application responsiveness, scalability, and stability under load.
   Example: JMeter, LoadRunner, NeoLoad.

4.2 Test Environment

The test environment setup specifies the minimum hardware and software configuration required to test the application effectively.

Hardware Requirements:

- Processor: Intel Core i5 (or equivalent) and above
- RAM: Minimum 8 GB
- Storage: Minimum 256 GB SSD or 500 GB HDD

- Display: 1366×768 resolution or higher

Software Requirements:

- Operating System: Windows 8 or above
- Microsoft Office: Version 2013 or above
- Email Client: MS Exchange or equivalent
- Web Browsers: Latest versions of Chrome, Firefox, Edge (client-specified)
- Database: MySQL, Oracle, or SQL Server (based on project requirement)

Network Requirements:

- Minimum Internet Speed: 10 Mbps for stable connectivity during testing, tool access, and bug tracking system updates.
- Recommended Speed: 25 Mbps or higher for smooth execution of cloud-based or remote testing tools.

5.Terms/Acronyms

Make a mention of any terms or acronyms used in the project

TERM/ACRONYM	DEFINITION
API	Application Program Interface
AUT	Application Under Test
QC	Quality Control
SIT	System Integration Testing
STLC	Software Testing Life Cycle
SDLC	Software Development Life Cycle
TC	Test Case
TS	Test Suit
Selenium	Popular open-source automation tool for web UI testing
JIRA	Test management and defect tracking tool
Postman	Tool for manual/API testing
API	Application Programming Interface