Wipro Java Selenium-Batch 10, Team-06

Test Strategy

Demo Web Shop

Revision History

Date	Version	Author	Description

TABLE OF CONTENTS

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

2. Test Approach

This is the comprehensive test approach for the Demo Web Shop application (https://demowebshop.tricentis.com). It covers different testing types, methods, and activities to ensure complete validation of the website's functionality, performance, security, and usability.

- **Functional Testing**

The focus will be on verifying user-centric features such as user registration, login/logout, browsing product listings, filter and search functionality, add-to-cart, checkout workflows, order placement, newsletter subscription, and product review mechanisms. Each flow will include positive, negative, boundary, and error-path scenarios.

- **Automation Strategy**

We will prioritize automation of high-risk, frequently used user journeys—login, search, add-to-cart, checkout, and order flow—within a scalable framework such as Page Object Model. Automation will be integrated into CI pipelines.

- **Regression Testing**

A comprehensive regression suite will be executed toward the end of each sprint. Key workflows are retested to ensure no features are broken due to new changes.

- **Performance Testing**

Load, stress, spike, and endurance testing will be conducted. Responsiveness of key pages will be validated, aiming for optimal performance under various traffic conditions.

- **Security Testing**

Testing will cover vulnerabilities like SQL injection, XSS, CSRF, broken authentication, and session hijacking. SSL configurations and secure cookie practices will also be validated.

- **Usability & Exploratory Testing**

Exploratory testing will be conducted to uncover usability issues. This includes layout consistency, checkout flow efficiency, and mobile responsiveness.

- **Compatibility & Accessibility**

Testing will verify site behavior across major browsers and devices. Accessibility testing will ensure keyboard navigation, screen reader support, and WCAG compliance.

- **Integration & Third-Party Services**

Testing will cover communication with email services, CAPTCHA, and newsletter systems. Mock services will be used where real integrations are unavailable.

- **Agile-specific Adaptations**

In Agile, QA is involved from story grooming to sprint closure. Test cases are created from acceptance criteria, and automated/manual tests are executed during each sprint cycle with continuous feedback.

Summary Table of Test Approach

Area	Objective
Functional Testing	Validate all user journeys and functional paths
Automation & Regression	Automate core flows; ensure regressions are caught in CI
Performance Testing	Simulate peak, spikes, endurance under load for key pages
Security Testing	Identify and resolve vulnerabilities (XSS, injection, session risks)
Usability & Exploratory Testing	Identify UI and UX issues not captured in scripts
Compatibility & Accessibility	Ensure consistent UX across browsers, devices, and accessibility support
Integration Testing	Confirm backend flows and dependencies (email, captcha, newsletter services)

3. Test Environment

A **test environment** is a setup of software, hardware, and tools used to run tests and simulate real-world conditions.

Operating Systems:
Testing will be conducted on multiple OS platforms, including Windows 10/11 and macOS, to ensure cross-platform compatibility.

Browsers:
The application will be tested on the latest versions of Chrome, Firefox, Edge, and Safari to validate performance across various browser engines.

Devices:
Tests will be run on desktop and laptop systems only. Since the web app is built for fixed screen sizes, responsive testing is not required.

Test Tools:
Automation testing will be carried out using Selenium WebDriver. Postman will be used for API testing if applicable. Test execution and reporting will utilize TestNG or JUnit frameworks. Jira will be used for bug tracking and test management.

Test Data:
Dummy user accounts, product listings, and cart/transaction data will be used during testing. All test data will be securely stored and backed up daily.

Backup & Restore Strategy:
Daily database snapshots will be created. In the event of data loss or corruption, the system will be restored using pre-test cycle backups.

Staging Environment:
Testing will be performed in a dedicated staging server that mirrors the production setup to avoid disrupting live systems.

Internet Speed
Testing will be conducted on connections ranging from 2 Mbps to 100 Mbps to simulate different user network conditions.

4. Testing Tools

Automation and Test management tools needed for test execution are identified based on the application architecture, testing scope, and team size. Both **open-source** and **commercial** tools are considered to achieve high coverage and efficiency. The number of supported users for each tool is evaluated to plan licenses and team access accordingly.

4.1 Test Management Tools

Tool	Type	Purpose	User Support
Jira + Xray/TestRail	Commercial	Test case design, execution, defect tracking, and traceability	Up to 10 QA and dev team members via project access

4.2 Automation Tools

Tool	Type	Purpose	User support
Selenium WebDriver + TestNG	Open-source	UI automation for functional and regression testing	Used by 3–5 automation engineers
Cypress	Open-source	Frontend e2e automation testing for dynamic web elements	Used by 2 QA testers for rapid UI feedback
REST Assured	Open-source	API-level automated validation	Integrated with Selenium; used by 2–3 testers
Jenkins	Open-source	CI/CD pipeline for automation execution	Integrated for automated nightly runs by all QA

4.3 Performance Testing Tools

Tool	Type	Purpose	User support
Apache JMeter	Open-source	Load and performance testing on endpoints like Checkout and Cart	Used by 2 performance engineers
Gatling (Optional)	Open-source	Scripting-based performance testing	Optional use based on test complexity

4.4 Cross-Browser & Responsive Testing Tools

Tool	Type	Purpose	User support
BrowserStack	Commercial	Real device and browser testing	License for 2 concurrent users planned
Chrome DevTools – Responsive Mode	Open-source	Ad-hoc responsive testing	Used by all frontend testers (no license required)

4.5 API Testing Tools

Tool	Type	Purpose	User support
Postman	Freemium/Open-source	Manual API testing and collection sharing	Used by 3 testers
Swagger UI (if exposed)	Open-source	Visualize and test REST APIs	Public access or team access if available

4.6 Security & Vulnerability Testing

Tool	Type	Purpose	User support
OWASP ZAP	Open-source	Vulnerability scanning (XSS, SQLi, CSRF)	Used by 1 security tester for baseline scans
Burp Suite Community	Open-source	Manual security testing	Used by 1 tester as needed

4.7 Test Data Management

Tool	Type	Purpose	User support
Mockaroo	Open-source	Fake test data generation (names, emails, addresses)	Used by 2 testers
Excel/CSV + Selenium DataProvider	Open-source	Data-driven testing for login, checkout	Used by all automation testers

Tool Utilization Plan

- Open-source tools will be used wherever possible to reduce cost while maintaining testing coverage.
- Commercial tools like Jira (with Xray/TestRail) and BrowserStack will be licensed for key users (QA leads, automation testers).
- Total user capacity is balanced across tools to support ~6–8 QA engineers and 2–3 dev leads.
- Tools will be integrated with Jenkins for continuous automation runs and report generation

5. Release Control

5.1 Release & Version Structure

Maintain structured versioning using test-management tool:

- **Releases**: Represent major delivery milestones with defined start/end dates, scope and release notes.
- **Builds**: Subsets or incremental development checkpoints within a Release.

5.1.1 Timeline

Release v1.0 (Core Webshop – July 1 to July 28)

├ Build 1.0.1 (Login & Registration)

├ Build 1.0.2 (Product Browse & Cart)

├ Build 1.0.3 (Wishlist)

Release v1.1 (Payment & Checkout – July 29 to August 6)

├ Build 1.1.0 (Payment Integration)

├ Build 1.1.1 (Order confirmation)

5.2. Feature Requirements

- **Capture Requirements**: Map each requirement, bug fix, or enhancement to a specific Release and Build within Test.
- **Version History**: For each version or build, maintain a changelog:
- New features
- Fixed bugs/issues
- Test impact (modified, added, skipped, deprecated)

5.3. Test Planning & Execution Hierarchy

Test Execution module, structure tests as follows:

1. **Release Container**
2. **Test Cycles**: e.g. “Cycle – Build 1.1.1 Regression”
3. **Test Suites**: Group by functionality or feature (e.g. “Checkout Flow Suite”)
4. **Test Runs**: Individual test cases executed by testers or automation

This allows tracking execution status per version/build and linking test runs to defects or requirements.

5.4 Release Version & History

Release	Build	Date	Description of Changes	Testing Focus	Status
v1.0.	1.0.1	2025-07-01	Initial baseline deployment	Full regression test	Complete
v1.0	1.0.2	2025-07-20	New feature: wishlist functionality	Functional + regression	In progress
v1.0	1.0.3	2025-08-28	Small bug fix: login error	Targeted smoke testing	Pending
v1.1	1.1.0	2025-08-01	Feature enhancements: search filters	Regression + UI testing	Planned
v1.1	1.1.1	2025-08-06	Patch: checkout flow fix	End-to-end scenario test	Planned

6. Risk Analysis

- List all risks that you can estimate
    - Environment Instability
        - Testing environment may not always mirror production or could experience downtime, resulting in failed or invalid tests.
    - Regression Failures After Updates
        - Updates or new feature deployments may unintentionally break existing functionalities (e.g., cart, checkout).
    - Integration Issues
        - Failures or issues when integrating with third-party services (e.g., payment gateways, email notifications).
    - Limited Test Data
        - Lack of or outdated test data may impede proper validation, especially for rare or edge cases.
    - Time Constraints in Sprints/Releases
        - Compressed timelines could lead to reduced coverage, rushed executions, or missed defects.
    - Resource Availability
        - Key QA or development team members may be unavailable, impacting planning and execution.
    - Browser and Device Compatibility
        - The site may behave inconsistently across different browsers or devices if not adequately tested.

- Give a clear plan to mitigate the risks also a contingency plan

    - Continuous Requirement Review: Regular backlog grooming and close collaboration with the Product Owner or business user to clarify requirements and changes.
    - Stable & Mirrored Environments: Automate environment setup and refresh to closely match production, and monitor uptime.
    - Automated Regression Suite: Maintain and regularly run automated tests for key workflows (e.g., login, add-to-cart, checkout) to quickly detect breakages.
    - Rich Test Data Management: Use tools/scripts to create and refresh real-world-like test data.
    - Prioritized Coverage: Focus first on critical user journeys and high-risk features; expand coverage as time allows.
    - Resource Planning: Ensure backup testers and cross-training; stagger vacations and plan ahead for key project phases.
    - Compatibility Testing: Maintain a browser/device matrix and use tools like BrowserStack for cross-platform checks.

- Security and Performance: Include dedicated security checks (manual and automated) and run performance tests before major releases.

7. Review and Approvals

Review and Approval:

All activities related to this release such as requirement analysis, test case creation, test execution, and defect resolution have been reviewed and approved by the respective stakeholders including the Business Team, Project Management, Development Team, and QA Team.

Review Log:

Review Date	Name	Role	Comments	Approved
[YY/MM/DD]	[Business Lead Name]	Business Team Lead	Validated product filters and UI alignment	Yes
[YY/MM/DD]	[Project Lead Name]	Project Manager Lead	Confirmed build scope and checkout enhancements.	Yes
[YY/MM/DD]	[Dev Lead Name]	Development Lead	Reviewed UI adjustments and backend fixes.	Yes
[YY/MM/DD]	[QA Lead Name]	QA Lead	Approved test case updates and regression scope.	Yes

Approval:

Final Sign-Off:

Stakeholder Team	Name	Role	Signature / Initials	Approval Date
Business Team	[Business Lead Name]	Business Analyst	—----------	YY/MM/DD
Project Management	[Project Lead Name]	Project Manager	—----------	YY/MM/DD

Development Team	[Dev Lead Name]	QA Lead	—----------	YY/MM/DD
QA Team	[QA Lead Name]	Dev Lead	—----------	YY/MM/DD