**Demo Web Shop – E-Commerce Website Testing**

**Test Strategy**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Author | Description |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[1. Scope 3](#_Toc534636696)

[2. Test Approach 4](#_Toc534636697)

[3. Test Environment 6](#_Toc534636698)

[4. Testing Tools 8](#_Toc534636699)

[5. Release Control 10](#_Toc534636700)

[6. Risk Analysis 12](#_Toc534636701)

[7. Review and Approvals](#_Toc534636702) 14

# Scope

This Test Strategy Document applies to the **Demo Web Shop** e-commerce web application and defines the structured approach to testing throughout the project lifecycle using the **Agile Scrum model**. It covers the testing processes, responsibilities, tools, environments, and expected deliverables to ensure a high-quality and stable product release in each sprint cycle.

The scope includes:

* Definition of **testing levels**, types of testing, and test phases.
* Identification of roles and responsibilities of QA, developers, and stakeholders.
* Test case design, execution process, defect lifecycle, and regression testing.
* Use of **manual and automated testing** integrated into CI/CD pipelines.
* Guidelines for test environment setup, test data, and reporting.
* Review and approval workflow for this document.
* Sprint-level testing activities and timeline expectations.

Document Review & Approval

|  |  |
| --- | --- |
| Activity | Responsible Role(s) |
| **Document Review** | QA Lead, Project Manager, Scrum Master |
| **Final Approval** | Test Manager, Product Owner |
| **Version Control & Updates** | QA Lead (based on retrospectives or scope change) |

**Testing Activities & Timelines**

Testing activities are carried out **iteratively per sprint** as follows:

|  |  |  |
| --- | --- | --- |
| Test Case Design | Sprint Planning / Day 1–2 | As per user stories |
| Environment Setup | Before development start | End of previous sprint |
| Test Execution (Functional) | During development | Daily during sprint |
| Defect Logging & Retesting | Mid-sprint | Ongoing |
| Regression Testing | Sprint end | Last 1–2 days of sprint |
| Test Summary Report | Sprint Review | Final day of sprint |
| Test Sign-Off | Sprint Retrospective / Review | After validation & PO approval |

# Test Approach

**1. Process of Testing**

Testing follows an **Agile sprint-based cycle** (typically 1–2 weeks). As features are developed:

* Testers begin early—right from sprint planning—participating in backlog grooming and defining acceptance criteria linked to user stories.
* Feature-level testing is conducted continuously during development, and acceptance testing (UAT) is performed at the end of each sprint before release.
* Defects are logged as they arise, retested when fixed, and regression tests run iteratively to maintain stability.

**2. Testing Levels**

The following levels are included to ensure full coverage:

* **Unit Testing** (by developers)
* **Integration Testing**
* **System Testing / Functional Testing**
* **Acceptance Testing** (user/PO validation)

**3. Roles & Responsibilities**

A cross-functional team collaborates throughout:

* **Developer**: Implements code and unit tests; resolves defects.
* **QA/Testers**: Design and execute test cases (manual and automated), report defects, and ensure coverage.
* **QA Lead / Test Manager**: Oversees the testing approach, drives triage sessions and ensures test quality.
* **Product Owner / Business Stakeholders**: Participate in defining acceptance criteria and perform UAT sign-off at sprint close.

**4. Types of Testing**

We employ a variety of test types based on feature and risk profile:

* **Functional Testing** – Validate each feature per requirement and UI behavior.
* **Regression Testing** – Ensure new changes don't break existing functionality, aided by automation.
* **Performance / Load Testing** – Assess responsiveness and stability under varying loads.
* **Security Testing** – Verify sensitive user data (e.g. login, billing) is handled securely.
* **Exploratory / Session-Based Testing** – Used for discovery, usability, and uncovering edge-case defects
* **Others (as needed)** – Including scenario testing, compliance testing, etc., based on project scope and risk.

**5. Testing Approach & Automation**

* **Parallel Testing** – QA activities run in conjunction with development (“shift-left”) to find defects early.
* **Automation Strategy** – We automate:
  + **Unit tests** (by developers)
  + **Regression suites** using Selenium WebDriver (or an equivalent framework)
  + **Performance tests** using tools like JMeter
* **Manual Testing** focuses on exploratory, UI/UX checks, and edge-case validations.
* **Continuous Integration (CI)** – Automated tests are triggered post-build via tools like Jenkins, ensuring rapid feedback.

**6. Defect Lifecycle & Test Sign-off**

Defect handling follows a clear process:

* **Defect Logging** – Reported to tracking tool (e.g., Jira) with severity, steps, and environment.
* **Triage Meetings** – Held mid-sprint to prioritize and assign defect fixes.
* **Re-testing** – Defects are validated post-fix.
* **Regression Testing** – Before each sprint end, key regression scenarios are re-executed to prevent regressions.
* **Test Sign-off** – Occurs when:
  + All acceptance criteria are met,
  + No critical/high defects remain open,
  + Regression results are stable,
  + Test summary report is reviewed and approved by QA Lead and Product Owner.

# Test Environment

This section outlines the infrastructure and requirements for the test environments used in validating the Demo Web Shop application.

**Environment Setup Requirements**

|  |  |  |
| --- | --- | --- |
| Environment | Purpose | Required Setup |
| **Development (Dev)** | Developer unit testing | Local machines, Git repository, basic DB and web server setup |
| **QA/Test** | Manual & automation test runs | Stable build, latest database, test data, automation hooks, Selenium/Jenkins setup |
| **Staging (Pre-Prod)** | UAT & Final verification | Near-production environment with complete data and integrated services |

**Number of Requirements for Each Environment**

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Dev | QA/Test | Staging |
| Web Server | 1 | 1 | 1 |
| Application Server | Local | 1 | 1 |
| Database Instance | Local/Shared | 1 | 1 |
| Browsers | Chrome, Firefox | Chrome, Firefox, Edge | Chrome, Firefox, Edge |
| Test Users (Dummy data) | Optional | 15–20 | 20–30 |

**Test Data Backup & Restore Strategy**

To ensure consistency and avoid data loss across test cycles:

* **Backup Strategy:**
  + Database snapshots are taken **before each test cycle or major release build**.
  + Test data is exported using SQL dump or DB export tools.
  + Data files are stored securely in versioned folders on the QA shared drive.
* **Restore Strategy:**
  + QA team restores the environment to a clean baseline using latest backup before:
    - Regression testing
    - Major sprint or release test execution
  + Restoration is automated using SQL scripts or Docker volumes where applicable.
  + Restore validations are done to confirm data consistency.
* **Responsibility:**  
  QA Lead coordinates backups; DevOps/Support team executes the restore process as per schedule.

# Testing Tools

**Automation and Test Management Tools**

|  |  |  |
| --- | --- | --- |
| Category | Tool | Purpose |
| Test Management | **Jira + Zephyr / TestRail** | Managing test cases, linking to user stories |
| Test Automation | **Selenium WebDriver (Java)** | Functional UI automation |
| CI/CD | **Jenkins** | Triggering automated tests after each build |
| Defect Tracking | **Jira** | Logging, tracking, and triaging defects |
| Performance Testing | **JMeter (Open Source)** | Load & stress testing |
| Collaboration | **Confluence / Slack** | Test planning, documentation, team discussions |

**Open-Source Tools**

|  |  |  |
| --- | --- | --- |
| Tool | Max Users Supported | Remarks |
| Selenium WebDriver | Unlimited | Lightweight and scalable test automation |
| JMeter | Unlimited | Used for performance/load testing |
| Jenkins | Unlimited (self-hosted) | Requires DevOps to manage agents |

**Commercial Tools (Optional / Scalable)**

|  |  |  |
| --- | --- | --- |
| Tool | Max Users (Standard Plan) | Remarks |
| TestRail | 5–20 users (scalable) | Web-based test management for manual + automated TC |
| Zephyr for Jira | 10–50 users (based on plan) | Integrated inside Jira; useful for Agile tracking |
| BrowserStack | Limited by plan | Cross-browser testing and mobile emulation |

**Tool Planning Strategy**

* Based on team size (5–10 testers + 2 developers), open-source tools are preferred to avoid licensing costs in early stages.
* If scaling to larger teams or CI/CD maturity increases, tools like **TestRail** and **BrowserStack** may be evaluated.
* Each QA engineer will have access to Selenium, Jira, and Slack.
* **Jenkins** and **JMeter** will be maintained by a shared DevOps admin to support the full team.
* **Tool onboarding & documentation** will be maintained in Confluence.

# Release Control

**Release Management Plan**

|  |  |
| --- | --- |
| Activity | Details |
| **Release Identification** | Every release will follow a versioning format: vMajor.Minor.Sprint (e.g., v1.3.5) |
| **Release Artifacts** | Each build must include: Release notes, change log, build ID, deployment date |
| **Change Impact Analysis** | QA will analyze modified modules/features and update regression scope accordingly |
| **Code Freeze** | Initiated 1–2 days before sprint end; no code commits allowed post freeze unless critical |
| **Build Tagging** | Jenkins/CI will tag tested builds with version number for traceability |
| **Test Case Mapping** | All changes are mapped to updated test cases via Jira/TestRail |
| **Regression Planning** | Based on impact and module complexity; regression is mandatory for major releases |
| **UAT Readiness** | Build must pass QA regression before being deployed for UAT testing |
| **Production Release Sign-Off** | Final approval from QA Lead and Product Owner after UAT clearance |

**Version History & Audit Trail**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Sprint | Release Date | Summary of Changes | QA Status |
| v1.0.0 | Sprint 1 | 12-Aug-2025 | Initial release with core modules | Passed |
| v1.1.0 | Sprint 2 | 26-Aug-2025 | Added Wishlist & Sorting filters | Passed |
| v1.2.0 | Sprint 3 | 10-Sep-2025 | Introduced Payment Gateway & Coupons | Passed |
| v1.3.0 | Sprint 4 | 24-Sep-2025 | UI enhancements, product review feature | In Progress |

**Integration with Test Execution**

* Every release triggers:
  + **Smoke Testing** on deployment
  + **Functional Testing** of newly added features
  + **Regression Testing** of impacted modules
* QA team links updated test cases with release versions for full traceability.

# Risk Management

**Identified Risks and Mitigation Plan**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk ID | Risk Description | Impact Level | Mitigation Strategy | Contingency Plan |
| R1 | Unclear or changing requirements mid-sprint | High | Involve QA early in backlog grooming and sprint planning | Log as change request, adjust test scope, reallocate effort |
| R2 | Delay in build delivery or unstable builds | High | Set build freeze deadlines; maintain CI/CD pipelines | Use last stable build for partial testing, escalate via project manager |
| R3 | Limited test data or outdated test data | Medium | Maintain reusable data sets; schedule test data refreshes | Use backup data or request DevOps to restore previous DB snapshot |
| R4 | Lack of environment availability or downtime | Medium | Plan test cycles based on environment calendar | Switch to alternate test environment if available; communicate impact early |
| R5 | High defect density during UAT/regression | High | Include smoke tests & early regression cycles | Extend sprint testing window, re-prioritize defect triage based on severity |
| R6 | Unavailability of key QA resources | Medium | Cross-train team members, rotate features | Shift test ownership temporarily or extend test cycle based on velocity |
| R7 | Test automation failures post-integration | Medium | Review automation stability with every sprint | Re-run scripts manually or isolate flaky tests and log bugs for fix |
| R8 | Security or performance issues missed due to tight deadlines | Medium | Add baseline load/security tests in each sprint | Plan focused non-functional testing every alternate sprint |
| R9 | Last-minute changes by business owner | High | Define scope lock after sprint planning | Raise risk alert, re-baseline test cases and get formal approval |

**Risk Management Summary**

* All risks will be monitored by the **QA Lead**.
* Weekly risk reviews during sprint retrospectives.
* Risks logged and tracked using **Jira Risk Register** (or documented in Confluence).
* Contingency efforts are factored into sprint buffers when needed.

# Review and Approvals

**Review and Sign-off Responsibilities**

|  |  |
| --- | --- |
| Role | Responsibility |
| **QA Lead** | Ensure completeness and alignment with testing scope and methodology |
| **Project Manager** | Review for project timelines, test coverage, and resource alignment |
| **Product Owner** | Approve business requirement alignment and user acceptance criteria |
| **Development Lead** | Confirm technical feasibility, environments, and automation scope |
| **Business Team** | Final review and sign-off from business perspective |

**Document Version Control and Review Log**

A summary of changes made during the review process will be logged at the beginning of this document. It will include:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Reviewed By | Comments / Summary of Changes | Approved By |
| v1.0 | 05-Aug-2025 | Krishna Babu (QA) | Initial draft created | Naveen Ravalavalasa |
| v1.1 | 07-Aug-2025 | Project Manager | Added risk matrix and test data restore strategy | Project Manager |
| v1.2 | 08-Aug-2025 | Product Owner | Included release versioning and sign-off flow | Product Owner |

**Approval Statement**

Upon review, all responsible stakeholders acknowledge and approve this Test Strategy Document, enabling execution of the defined testing activities for the Demo Web Shop project.