OpenCart Demo: Automated Testing Framework Plan

This Test Plan (TP-OC-01) outlines the comprehensive strategy for testing the OpenCart Demo – Online Shopping Platform. The primary objective is to design, develop, and implement a robust automated testing framework to validate the functionality, performance, and reliability of the e-commerce application. This framework will ensure quality across UI, API, Database, and Performance layers, following an Agile Scrum methodology.

Document Overview

- Project: OpenCart Demo
- Version: 1.0
- Prepared by: QA Team
- Date: October 2025

Testing Scope

The plan encompasses end-to-end testing, framework implementation, CI/CD setup, and detailed defect management for the initial planning phase (Sprint 1).

Project Introduction and Test Scope Definition

The purpose of this Test Plan is to formally document the scope, objectives, approach, resources, and schedule for the quality assurance phase of the OpenCart Demo Online Shopping Platform. Our focus is on establishing a resilient and scalable automated testing framework.

> Main Goal: To design, develop, and implement an automated testing framework that comprehensively validates the functionality, performance, and reliability of the target e-commerce application.

Test Plan Identifier

TP-OC-01

OpenCart Demo Automation Framework Testing Plan.

Project Context

Focusing on Sprint 1 (Planning Phase) to establish foundational documentation and strategy for subsequent implementation sprints.

Test Items: Components Under Review

The test coverage is strategically layered to ensure system integrity from the user interface down to the persistent data store.

Component	Description	Key Details
Web Application	OpenCart Demo Website	https://demo.opencart.com/
UI Features	Core customer workflows	User registration, login, product browsing, cart management, checkout.
API Layer	RESTful web services	APIs for user, product, cart, and order operations (data interactions).
Database	MySQL Data Store	Verification of stored data: users, products, and order records.
Admin Portal	Backend Administration Dashboard	Limited scope of testing, primarily focused on order validation.

Features to be Tested and Scope Exclusion

A clear demarcation of the testing scope is crucial for focused effort and timely delivery within the Agile framework. We prioritised critical user journeys and technical validations.

Included Features (Positive Scope)

Core Functionality

User Registration & Login, Product Search and Filtering, Product Details Page.

E-commerce Workflow

Add to Cart / Remove from Cart, Checkout and Order Placement.

Data and API Integrity

Order confirmation and validation in the database (CRUD), API response validation (GET/POST/PUT/DELETE).

Non-Functional Testing

Performance baseline checks using Apache JMeter for key user flows.

Excluded Features (Negative Scope)

Payment Gateways

Real payment gateway transactions are outside the scope of this demo environment testing.

Third-Party Services

Integration with external services like email or SMS providers will not be verified.

Advanced Security

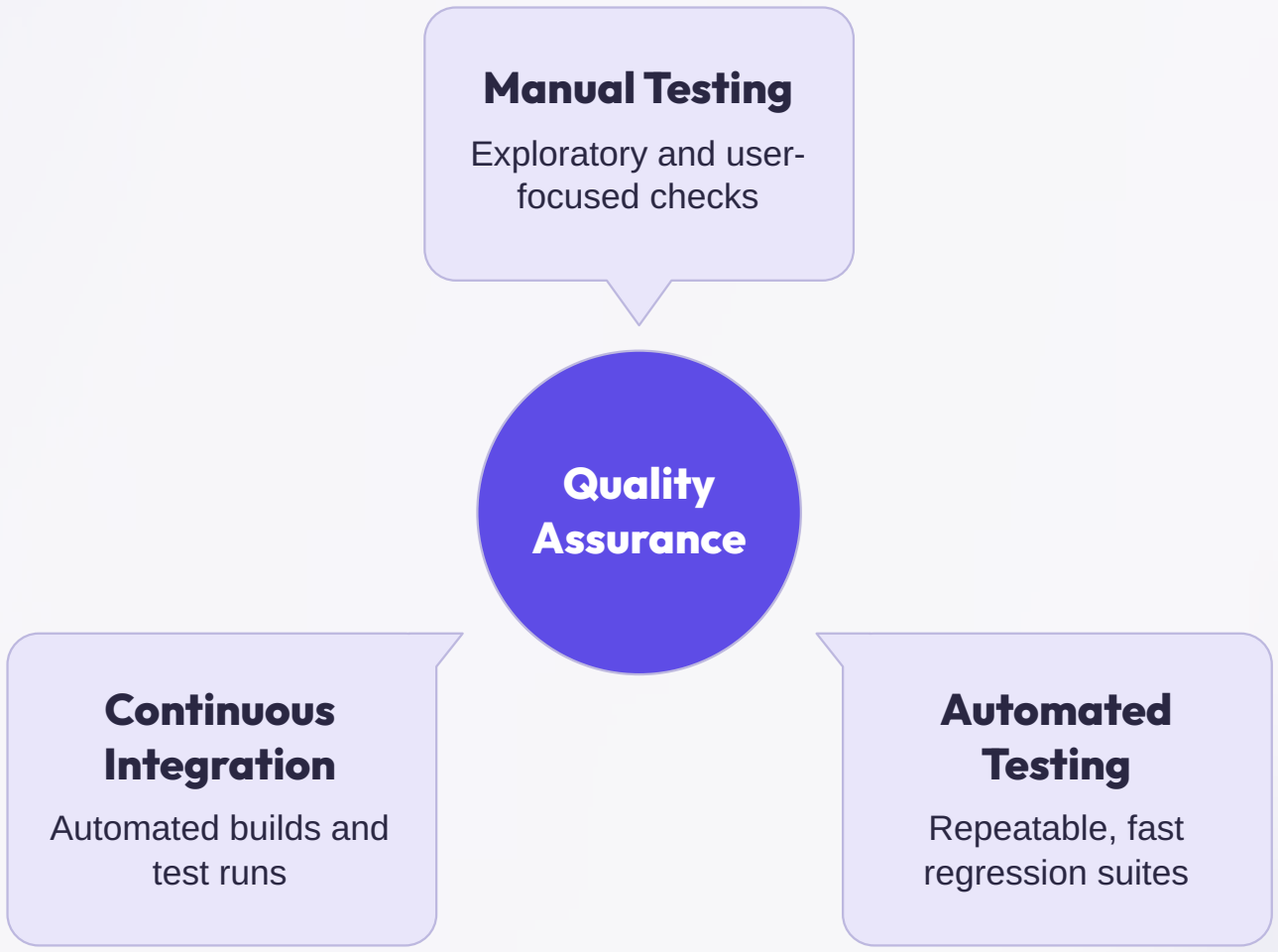
Advanced security testing, penetration testing, and vulnerability scans are excluded.

Client Platforms

Testing of dedicated mobile applications or native clients is not included.

Test Strategy: A Hybrid Approach

Our testing methodology adopts a hybrid strategy, combining exploratory manual testing with robust automated techniques. This ensures comprehensive end-to-end coverage, reliability, and speed.



Layered Testing Coverage

The automation framework is designed to address multiple layers of the application stack, providing detailed feedback on system health at every level.

UI Testing

Verifying front-end interaction and visual fidelity using Selenium.

API Testing

Validating request/response cycles, status codes, and payloads using Postman/Newman.

Database Testing

Ensuring data integrity and ACID properties are maintained for orders and user records.

Performance Testing

Establishing performance baselines for critical user paths using JMeter.

The foundation of our automation is the Page Object Model (POM), which enhances script maintainability and scalability for the UI layer.

Test Tools and Environment Configuration

A fully defined and stable test environment is a crucial prerequisite for successful automation runs. We have specified the technology stack and the necessary tools for each testing activity.

Automation Stack

Selenium for UI interaction, Cucumber (BDD) for feature definition, and POM for framework structure. Language: Java/JavaScript.

API Validation

Postman for manual API exploration and Newman for command-line execution within the automated CI/CD pipeline.

Load Testing

Apache JMeter will be used to conduct basic load testing to determine performance baselines of user workflows.

Continuous Testing

Jenkins will host the CI/CD test runs, triggered by changes in the GitHub repository, ensuring continuous quality checks.

Test Environment Specifications

Application Under Test	https://demo.opencart.com/
Operating System	Windows 11 / Ubuntu (for diverse environment testing)
Browsers	Chrome (latest), Firefox (latest)
Database	MySQL 8.x (Validated via MySQL Workbench)
Test Data	Dummy users, sample products, mock orders (prepared in Sprint 1/2)
Source Control	Git, GitHub

Roles, Responsibilities, and Team Structure

The QA team is comprised of four dedicated members, each assigned specific roles to maximise expertise and efficiency across the testing layers. Clear ownership is vital for agile delivery.

QA Lead (Member 1)

Approves the Test Plan, tracks overall progress, and oversees execution across all sprints. Serves as the primary stakeholder liaison.

Manual Tester (Member 2)

Executes critical manual test cases, performs exploratory testing, and diligently logs defects using Jira.

Automation Engineer (Member 3)

Responsible for building the robust Page Object Model framework and developing automated UI and API test scripts.

API/DB Tester (Member 4)

Focuses on writing comprehensive API and database test scripts, ensuring data integrity and validation at the backend layer.

Defect Management Protocol

Defects are managed and tracked rigorously through Jira, ensuring transparent communication and swift resolution. Severity levels dictate the priority of fix implementation.

Severity	Description
1 – Critical	Application crash or complete system failure (e.g., checkout process broken).
2 – High	Major functionality failure where a significant user workflow is blocked.
3 – Medium	Minor functionality issue where a workaround is available, impacting non-critical path.
4 – Low	UI/UX issue, cosmetic defects, or spelling mistakes.

> **Defect Lifecycle:** New → Assigned → In Progress → Fixed → Retest → Closed.

Test Schedule and Agile Sprint Breakdown

The testing engagement is structured across four sprints, spanning seven weeks, following the initial planning and approval phase. This timeline ensures adequate time for framework development and rigorous execution.

Sprint 1: Planning and Setup (1 Week)

Objective: Define scope, resources, and schedule.
Deliverables: Test Plan Document, Project Roadmap.

Sprint 2: Analysis & Design (2 Weeks)

Objective: Detailed design of test scenarios.
Deliverables: Manual + Automated Test Cases, API + DB Scenarios, BDD Feature Files.

Sprint 3: Implementation (2 Weeks)

Objective: Code the automation framework.
Deliverables: Framework setup (POM), Automation, API, and DB scripts completed.

Sprint 4: Execution & Reporting (2 Weeks)

Objective: Full test execution and reporting.
Deliverables: CI/CD Integration, Final Execution Reports, and Documentation.

All timelines are subject to change based on development dependencies and defect fix cycles. Regular communication through daily standups will ensure schedule adherence.

Entry and Exit Criteria for Testing Stages

Defining clear criteria ensures that testing activities begin and conclude under optimal conditions, providing measurable gates for project progression and quality sign-off.

1

Entry Criteria (Commencement)

- Test environment is fully ready and accessible.
- All required tools (Selenium, Jenkins, etc.) are installed and configured.
- Initial test data (users, products) has been prepared and loaded.
- The Test Plan document is formally reviewed and approved by the QA Lead.

2

Exit Criteria (Completion)

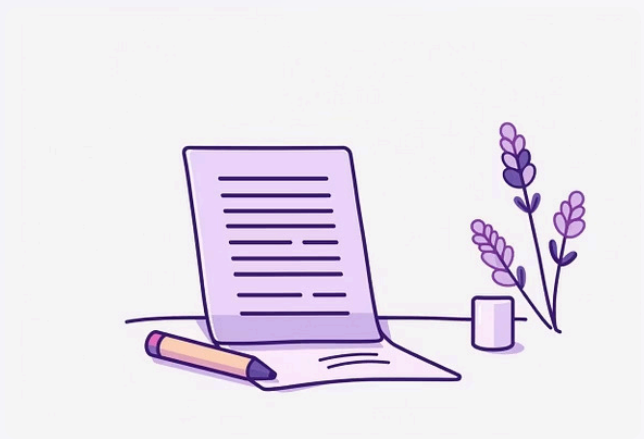
- All critical 'smoke tests' are executed successfully, validating system stability.
- Zero critical defects (Severity 1) or high defects (Severity 2) remain open.
- All defined deliverables for Sprint 1 (planning phase) are completed and reviewed.
- The Sprint 1 retrospective has been conducted to capture lessons learned.

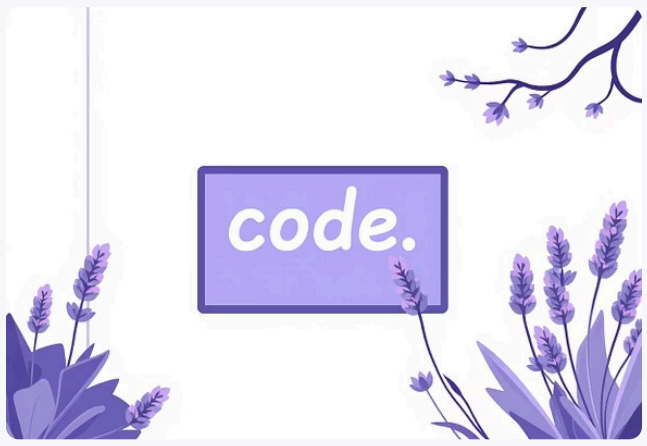
These criteria prevent premature execution and ensure that the final sign-off represents a high-quality product state.

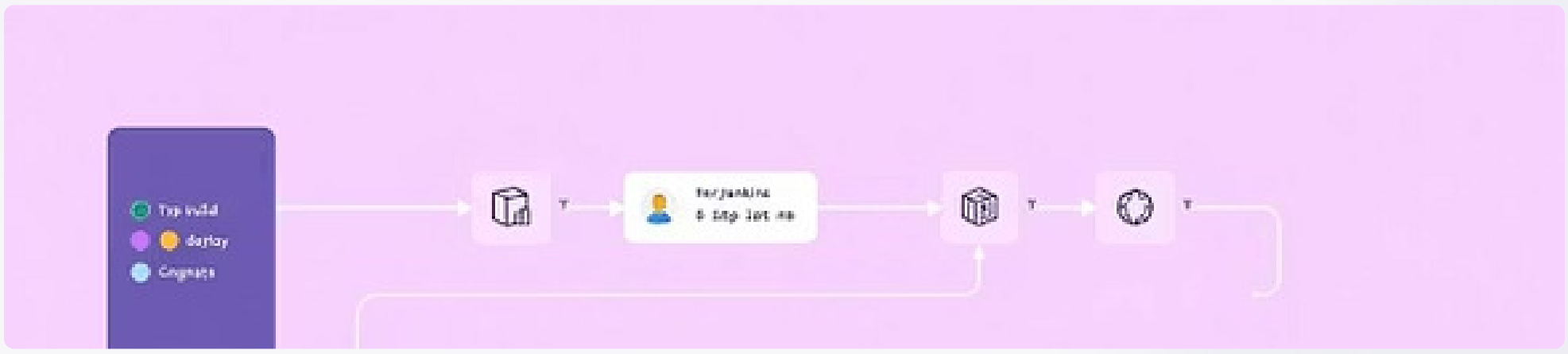
Test Deliverables and Risk Mitigation Strategy

The QA team is committed to producing comprehensive outputs that not only demonstrate the quality of the application but also contribute to the long-term maintainability of the testing process.

Key Deliverables







Risk Analysis and Contingency Planning

Proactive identification of risks allows the team to implement mitigation strategies to minimise impact on the project schedule and quality.

Tight Timeline

Daily standups & Scrum master monitoring.

Limited Test Data

Prepare mock user/product data in Sprint 1/2.

Tool Configuration Issues

Validate environment setup before Sprint 2.

Demo Site Downtime

Use backup environment or focus on API/DB testing with mock data.

Team Distribution Plan – OpenCart Demo Project

In the OpenCart Demo Project, every team member participates in all testing types to ensure comprehensive coverage and shared understanding. However, each member has a unique primary focus and leadership role, allowing for specialized expertise and efficient task allocation across the different layers of the application.

Member	Primary Focus	Key Responsibilities	Extra Responsibilities
Amr Mohamed – QA Lead	Project Coordination & Reporting	• Review/validate manual test cases • Create smoke automation (login, cart) • Review API collections & reports • Validate DB after checkout tests	Sprint planning, progress tracking, and defect triage
Ziad Mahmoud – QA Engineer	Functional / UI Testing	• Design & execute manual cases (registration, search, checkout) • Automate UI flows (login, search, add to cart) • Validate API responses (user, product modules) • Check DB records (user creation, orders)	Maintain Jira test cases & report bugs
Ali Haitham – Automation Engineer	Framework & CI/CD Setup	• Support manual testing (complex scenarios) • Build reusable automation framework • Automate API tests (Newman, Jenkins) • Validate DB queries via automation	Setup Jenkins/GitHub CI pipeline
Omar Hisham – API/DB Tester	Backend & Data Validation	• Assist manual checkout/backend scenarios • Automate regression suite (order management) • Build/maintain API collections in Postman • Write SQL scripts for CRUD & data verification	Generate DB + API execution reports

This structured distribution ensures that while individual strengths are leveraged, there is also cross-functional support and a clear path for each testing discipline throughout the project lifecycle.

Sprint-wise Team Assignment Details

Sprint-wise Assignment

The following table outlines the specific assignments for each team member across the different sprints, ensuring a focused approach to achieve sprint goals.

Sprint	Main Focus	Amr Mohamed	Ali Haitham	Ziad Mahmoud	Omar Hesham
Sprint 1 (1 week)	Planning	Create Test Plan & Roadmap	Define test schedule & criteria	Setup initial repo structure	Prepare API/DB checklist
Sprint 2 (2 weeks)	Analysis & Design	Review test cases	Write manual UI test cases	Create POM classes	Write SQL & API test cases
Sprint 3 (2 weeks)	Implementation	Oversee framework setup	Automate UI smoke cases	Integrate Jenkins + Newman	Validate DB results, update API scripts
Sprint 4 (2 weeks)	Execution & Reporting	Final review + report	Execute manual regression	Execute automation suite	Run DB/API verification + reporting

This detailed sprint breakdown facilitates clear accountability and provides a roadmap for each team member's contributions, crucial for the agile development process.