Purpose of the Document

Use the Test Plan document to describe the testing approach and overall framework that will drive the testing of the project.

Template Instructions

Note that the information in italics is guidelines for documenting testing efforts and activities. To adopt this template, delete all italicized instructions and modify as appropriate

Table of Contents

1 Introduction

1.1 PURPOSE

This document provides a summary of the strategy, approach, execution plan and test management framework for testing the AutomationExercise website. It describes what will be tested, how testing will be conducted, what data and environments will be used, and how defects will be managed.

1.2 PROJECT OVERVIEW

The project is to test the public-facing e-commerce-style demo site AutomationExerciseThe site includes user-signup/login flows, product browsing/search, shopping cart/checkout, categories/brands navigation, subscription & contact forms. The objective is to ensure the entire system (UI + APIs + database) functions reliably, supports automated regression, and preserves data integrity as expected.

2 Scope

2.1 IN-SCOPE

The following features and workflows will be tested:

- UI layer: user registration/signup, login/logout, product listing & detail, search, categories/brands, cart operations, checkout, subscription, contact forms.
- API/Service layer: REST endpoints supporting the UI flows (e.g., signup API, login API, product list API, cart API, order submission API).
- Database/back-end layer: verifying that data (users, orders, cart items, subscriptions) is correctly stored/updated in the database, schema validations, data integrity.
- Automation testing: building automated scripts covering key flows (UI + API) for regression.
- Regression across all layers (UI, API, DB) after enhancements or changes.

2.2 OUT-OF-SCOPE

The following items are out of scope for this test cycle:

- Performance testing under heavy load (unless scheduled separately)
- Integration with third-party systems not visible on the site (e.g., external payment gateways beyond test simulation)
- Administrative/back-office interfaces (because the practice site focuses on front-end)
- Mobile app (if any) — only the web interface is being tested
- Highly specialized accessibility testing (unless separately scheduled)

3 Testing Strategy

3.1 TEST OBJECTIVES

- Verify that UI flows (signup/login, browsing, search, cart, checkout) operate correctly for positive and negative scenarios.
- Validate API endpoints: correct status codes, response payloads, expected business logic for valid and invalid inputs.
- Ensure database integrity: records created/updated/deleted as expected, data constraints enforced, correct relationships maintained.
- Develop and maintain automated tests to reduce manual regression effort and provide quick feedback.
- Enable regression coverage across UI, API and DB layers to catch unintended side-effects after changes.
- Confirm cross-layer consistency: e.g., a UI action triggers correct API calls and correct DB entries.

3.2 TEST ASSUMPTIONS

- The test environment (UI, APIs, database) will be stable and accessible during the testing window.
- API documentation or discovery mechanisms exist (to identify endpoints to test).
- Testers have access to necessary test credentials, test database or read-only view of DB where needed.
- Automation framework and tools are available and configured (for UI and API).
- There are no major UI or API changes during the primary functional test cycle unless communicated in advance.
- Database resets or clean-ups will be done between test cycles or test data will be isolated.

3.3 DATA APPROACH

- UI test data: unique email addresses for signup, valid/invalid login credentials, product names for search, cart operations.
- API test data: datasets for endpoint testing (valid inputs, boundary/edge inputs, invalid inputs) and expected responses.
- DB test data: known baseline data; ability to query database to verify states before/after tests; cleanup scripts/seeding.
- Automation data: parameterised data sets for automation scripts (UI + API); environment variables for endpoints, credentials, DB connection.
- Test data lifecycle: creation → use → cleanup/reset to baseline or isolated environment.

3.4 LEVEL OF TESTING

List the types of testing to be performed.

Test Type	Description
Unit Testing	(Handled by developers) basic component/logic level tests
API/Service Testing	Testing REST endpoints for correctness, status codes, payloads, error handling
Database/Back-end Testing	Validate correct storage and retrieval of data, schema integrity, constraints
Functional/UI Testing	Verification of end-to-end user flows through the UI
Automation Testing	Automated execution of UI + API + DB tests for regression and faster feedback
User Acceptance Testing	Business stakeholder validation of the system from UI perspective
Regression Testing	Re-execution of previously passed tests across UI/API/DB layers

API/SERVICE TESTING

Specify what features are to be tested.

Participants:

Tester's Name	Department/ Area	Role
Marian Ghaly	Login API valid/invalid credentials → status codes & messages.	Tester
Ahmed Salah Qatar	User Account API Create, Delete, Update, Get user account.	Tester
Mohammed Abedel Hamed	Product listing API GET all Products, POST all Products List.	Tester
Belal Mohammed Taher	Brand Listing API GET all Brands, PUT all Brand List	Tester
Mohammed abo El Yazeed	Search Product API Search product & search without parameter	Tester

DATABASE/SERVICE TESTING

Specify what features are to be tested.

Participants:

Tester's Name	Department/ Area	Role
Mohammed abo El Yazeed	Products & Categories (e.g., oc_product, oc_product_description, oc_product_to_category, oc_category, oc_category_description, oc_category_path)	Tester
Ahmed Salah Qatar	Settings& Extensions (e.g., oc_setting, oc_extension, oc_module,)	Tester
Mohammed Abedel Hamed	Orders & Order-details (e.g., oc_order, oc_order_product, oc_order_total, oc_order_history)	Tester
Belal Mohammed Taher	Customer & Address & Admin User data (e.g., oc_customer, oc_address, oc_customer_login, oc_user, oc_user_group).	Tester
Marian Ghaly	Functional & relationship testing of core tables across above domains & Wishlist & cart & checkout	Tester

3.4 FUNCTIONAL TESTING

All user flows and UI behavior as outlined in Scope Section 2.1.

Participants:

Tester's Name	Department/ Area	Role
Mohammed Abedel Hamed	Signup/Login page	Tester
Belal Mohammed Taher	Home page	Tester
Mohammed abo El Yazeed	Products page	Tester
Marian Ghaly	Shopping cart page	Tester
Ahmed Salah Qatar	Contact Us page	Tester

3.5 AUTOMATION TESTING

Specify what features are to be tested.

Participants:

Tester's Name	Scenarios	Role
Belal Mohammed Taher	All about registration / accounts. 1. Register User 5. Register with existing email 14. Place Order: Register while Checkout	Tester
Mohammed abo El Yazeed	All related to authentication and session management. 2. Login with correct email/password 3. Login with incorrect email/password 4. Logout User	Tester
Ahmed Salah Qatar	Browsing, product discovery & navigation 8. Verify All Products & product detail page 9. Search Product 18. View Category Products 19. View & Cart Brand Products	Tester
Mohammed Abedel Hamed	Cart behavior + subscription features. 10. Verify Subscription on Home Page 11. Verify Subscription on Cart Page 12. Add Products to Cart 13. Verify Product quantity in Cart 17. Remove Products from Cart	Tester
Marian Ghaly	End-to-end flows, UI/UX, reviews, checkout, invoices, scrolling. 6. Contact Us Form 7. Verify Test Cases Page 15. Place Order: Register before Checkout 16. Place Order: Login before Checkout 20. Search Products & Verify Cart After Login 21. Add review on product 22. Add to cart from Recommended items 23. Verify address details in checkout page 24. Download Invoice after purchase order 25. Verify Scroll Up (Arrow) & Scroll Down 26. Verify Scroll Up (without Arrow) & Scroll Down	Tester

3.6 ESTIMATED TIME

Specify what features are to be tested.

Participants:

Focus Area	Simulation time
Exploratory Testing & test plan	1 weeks
Dividing tasks on Jira &Manual testing & BDD	2 weeks
Database testing & Api testing	1 weeks
Automation testing & performance testing	2 weeks
Execute all test cases and make full execution report & Documentation & test completion report	1 week

4 Execution Strategy

4.1 ENTRY CRITERIA

- Test environment (test server) is available and reachable.
- Test data required for initial tests is created (e.g., test user, test product items).
- Testers have access to application, API endpoints, database (or read access).
- Test plan, test cases and data sets are reviewed and approved.
- Test build (application version) is deployed and stable enough for testing (no major show-stopping bugs).

4.2 EXIT CRITERIA

- All planned test cases executed (or deferred with justification) and results recorded.
- All critical/high severity defects resolved or work-around accepted.
- No outstanding critical defects, and only acceptable number of medium/low defects remain (as per agreed defect threshold).
- Test summary report issued with coverage, defect status, quality metrics.
- Regression pack executed and passed.

4.3 VALIDATION AND DEFECT MANAGEMENT

- Test cases for all layers will be reviewed and approved.
- Defects will be tracked with details including layer (UI/API/DB), severity, steps to reproduce, logs/screenshots, and status.
- Defects found during the Testing should be categorized as below:

Severity	Impact
1 (Critical)	▪ *Functionality is blocked and no testing can proceed* ▪ *Blocker: major function unavailable (UI/API/DB)*
2 (High)	▪ *Functionality is not usable and there is no workaround but testing can proceed*
3 (Medium)	▪ *Functionality issues but there is workaround for achieving the desired functionality*
4 (Low)	▪ *Unclear error message or cosmetic error which has minimum impact on product use.*

5 Environment Requirements

5.1 TEST ENVIRONMENTS

- **UI environment:** Web application accessible via browser on test domain.
- **API environment:** Service endpoints accessible (staging or dedicated test).
- **Database environment:** Test database instance mirroring production schema.
- **Automation infrastructure:** Machines/VMs for running automated scripts (e.g., in CI/CD pipeline).
- **Browsers/devices:** Latest versions of Chrome, Firefox, Edge; optionally mobile devices/responsive view.
- **Security/access:** Testers will have necessary credentials/permissions for API calls and DB queries (read or read/write as needed).
- **Data reset:** Mechanism to reset DB or clear test data after each cycle or as needed to maintain clean baseline.

6 Dependencies

- API documentation and endpoint readiness must be provided before API layer testing.
- Database schema must be stable before DB testing begins.
- Automation framework and test scripts must be created, reviewed and approved.
- Availability of test credentials, unique data sets (emails, product IDs) for UI + API test flows.
- No major release changes (UI/API/Schema) during the testing window without coordination.