Comprehensive Test Plan for DemoBlaze Website

1. Test Plan Identifier

- **Test Plan ID**: TP-DBZ-001
- **Version**: 1.1.0
- **Date**: 27- 01 - 2026
- **Prepared By**: Senior Test Engineer
- **Reviewed By**: Test Lead/ Test Manager

2. Introduction & Objective

The purpose of this Test Plan is to define the scope, strategy, resources, environment, and schedule of testing activities for the DemoBlaze e-commerce website (https://www.demoblaze.com/).

The key objective is to validate both **functional and non-functional aspects** of the site, ensuring core workflows (authentication, browsing, cart management, and checkout) deliver a **stable and positive user experience**.

Regressions will be promptly identified by using automation to handle high-priority and repetitive scenarios like checkout, cart, and login. This will be supplemented by manual exploratory testing that covers user experience and edge cases.

In the end, this strategy seeks to offer an organized and transparent testing process, ensuring that any problems are identified early, communicated effectively, and resolved prior to release. The ultimate goal is to reassure users that DemoBlaze is dependable, stable, and offers a satisfying experience.

3. Scope

In-Scope

Features to test include:

- Navigation (home, categories, product detail pages)

- User authentication (signup, login, logout)
- UI elements like modals (signup, login, about, contact)
- Browsing and filtering products by category
- Product details (images, descriptions, price)
- Cart functionality (adding items, removing items, quantity, total)
- Order / purchase end-to-end flow
- Messages / contact / feedback forms
- Responsiveness across devices

High level Test Scenarios are as mentioned below.

Test Type	Description / Coverage	Priority
User Authentication Tests	- Sign up with valid credentials. - Sign up with invalid inputs (empty fields, short password / no password, duplicate username). - Login with correct credentials. - Login with wrong credentials (invalid username/invalid password). - Logout works correctly.	**Critical**
Product Details Tests	- Images load properly. - Product description / matches price category listing. - Adding to cart from product detail adds correct item, correct price, correct quantity.	**Critical**
Navigation / Routing Tests	- All menus (Home, Categories, About Us, Contact) lead to correct pages. - Category filters work (e.g. Phones, Laptops, Monitors) show correct products. - Product detail pages show correct data when clicked from	**High**
	category list. - Carousel / image slider navigation works (Prev/Next options).	

Cart Functionality Tests	- Add item to cart - Remove item from cart. - Adjust quantity if supported - more items in cart, correct total price calculation. - Cart persists (if user logs out / in, or page refresh) (depending on Spec test).	**High**
Checkout / Purchase Flow	- If the site has a purchase or order confirmation flow: test filling payment / shipping details. - Error cases (missing data), successful completion, order confirmation. - If it is a demo only with no real backend, test what is present.	**High**
Form validations	- “Contact Us” or “Send Message” form: mandatory fields, invalid email formats, missing fields. - Signup / login forms: validation of inputs, error messaging.	**High**
Session / State Management	- After login, user sees “Logout” option, and restricted areas / user-only features available (if there are any). - After logout, no access to user-only features. - Cart state on page refresh or navigation.	**Medium**
Edge / Negative Tests	- Trying to purchase or add product when out of stock (if applicable) - Long inputs / special characters in forms - Navigating via back/forward buttons, browser refresh behavior - Behavior with slow network, or disconnecting during operations.	**Medium**
Mobile Test **Error Handling**	-Test in different port views (mobile browser, mobile app) (whatever is applicable). - If the server returns errors (if this can be simulated in test envi.) - how does the UI respond in this case - Handling 404 or missing product pages (if applicable).	**Medium**
Accessibility/ Localisation Testing	- Accessibility Testing: To ensure the app is usable for people with visual or color impairments. - Localization Testing: Ensure the app supports different languages and cultural formats.	**Medium**
Security	- Passwords hidden / not visible in login signup	**Low**

Out-of-Scope

- Third Party Payment Integrations (like Payment gateway, Card checkout etc).

4. Test Modules/ Focussed Areas

- User authentication module
- Product catalog
- Shopping cart functionality
- Checkout and order module
- Navigation menus and responsiveness

5. Test Approach / Strategy

Test environments (SIT and UAT)

Testing will be performed using both **manual** and **automated** approaches:

- **Manual Testing**: Exploratory, usability, boundary validations, and ad-hoc testing, API Testing.
- **Automated Testing**: Functional,Sanity, Regression, smoke, and end-to-end workflows automated with **Cypress** integrated into CI/CD, Run API tests through Newman.
- **Risk-based Testing**: Priority on **business-critical and high-risk flows** (sign up, login, cart, payment checkout) based on project timelines.
- **Non-functional Testing**: Compatibility (cross-browser/device testing using Saucelabs), Performance (JMeter), security (Postman for basic testing), and Accessibility Testing (JAWS, NVDA).

6. Test Types Covered

- **Functional Testing**: Verify that core features work as intended, e.g., users can log in, browse categories (Phones, Laptops, Monitors), add products to the cart, and complete

checkout successfully.

- **Regression Testing**: Process of re-running existing test cases after changes (like bug fixes or enhancements) to ensure that previously working functionality still behaves as expected and no new defects have been introduced.
- **Smoke Testing:** A quick, high-level check of the core functionalities to verify the build is stable enough for further testing.

- **Sanity Testing:** A focused check of new features or bug fixes to ensure they work correctly before deeper testing.

- **System Integration Testing:** Confirm that modules work together, e.g., product selection correctly updates the cart, and checkout reflects the right items and totals.

- **Usability Testing:** Check that navigation menus, product browsing, and cart interactions are intuitive—for example, users can easily find and purchase a laptop without confusion.

- **Performance Testing:** Measure if pages like homepage or product listings load within acceptable times; simulate multiple users adding items to the cart simultaneously.

- **Security Testing:** Done by respective team (may sit outside the organization) Confirm basic controls like password masking at login, session logout working properly, and no access to checkout without logging in.

- **Compatibility Testing: Saucelabs and BrowserStack**
  Run the DemoBlaze site on different browsers (Chrome, Firefox, Edge, Safari) and devices (mobile, tablet) to ensure consistent behavior.

- **Accessibility Testing:**
  Check if users with visual impairments can interact with DemoBlaze (e.g., screen readers can read product descriptions, buttons are keyboard accessible).

7. Entry Criteria

- Requirements reviewed and signed off.

- Test environment is configured and available.

- Test data prepared.

- Stable application build deployed in test environment.

8. Exit Criteria

- 100% execution of planned test cases with at least 80% Pass rate.
- No open **Critical/High severity (SEV1/SEV2)** defects.
- Smoke and Regression Automation suites are prepared.
- Test summary report completed.
- Approval/sign-off from stakeholders.

9. Test Deliverables

- Test Plan Document
- Test Cases and Test Data
- Automated Test Scripts, Automation Pass rate
- Defect Reports
- Daily/Weekly Test Execution Reports
- Final Test Summary Report

10. Test Environment

Hardware

- Windows, macOS.
- Mobile devices Latest Versions (Android, iOS)

Software

- Browsers: Chrome, Firefox, Safari, Edge.

- Tools: Cypress, Postman, JMeter, JAWS, NVDA.

Test Data

- Valid/ Invalid user credentials.
- Multiple product datasets for cart/checkout flows.

11. Roles & Responsibilities

- **Test Lead/ Test Manager**: Owns/Review test plan, tracks test progress, reporting.
- **QA Engineers**: Manual test execution, defect logging, Automation Script development, framework maintenance.
- **Software Engineers**: Coding, Development & Deployments.
- **DevOps Engineer**: CI/CD, environment setup.
- **Product Owner**: Requirement clarifications, UAT approval.

12. Schedule & Timelines TBC

- **Week 1-2**: Test Planning, Environment Setup
- **Week 3-5**: Test Case Design, Data Preparation, Automation script development
- **Week 6-8**: Smoke + Functional Execution
- **Week 9-10**: Retest, Regression, Automation Runs & Non functional testing
- **Week 11**: Final Execution, Reporting, Sign-off

13. RAIDS (Risks, Assumptions, Issues, Dependencies)

Risks

- **Environment instability** may delay execution.

- **Tight timelines** could reduce regression coverage.
- **Frequent UI changes** may break automation scripts.
- **Third-party dependencies** (e.g., payment gateways) cannot be tested fully.

Mitigation: Risk-based prioritisation, stable automation framework (POM/ Cypress), close DevOps collaboration.

Assumptions

- Requirements are **stable and signed off**.
- Figma Designs are finalised and locked.
- Test data is available and refreshed before execution.
- Test environment is **production-like**.
- Builds will be delivered as per sprint cycle.

Issues

- Limited **API/ Backend visibility** restricts deeper validation.
- Performance limits are not fully measurable in a demo environment.

Dependencies

- Timely deployment of **stable builds** from Dev team.
- Continuous **DevOps support** for CI/CD.
- **Product Owner, Resources** availability for clarifications and work as planned.
- **QA engineers** for automation framework setup & script development.

14. Defect Management

- Defects tracked in **Jira** with severity/priority.
- Defect template to raise defects with all the necessary details.
- Daily defect triage meetings for critical/high issues.
- Provide the Jira filter with the details Overall execution Report, Defects created, Status, Outstanding issues with Severity and Priority.

15. Test Metrics & Reporting

- % of test cases executed/passed/failed.
- Defect density by module.
- Defect leakage rate into UAT.
- Regression automation coverage %.
- Daily/weekly dashboards and final Test Summary Report.

16. Approvals & Sign-Off

Approvals required from:

- Test Lead/ Test Manager.
- Product Owner.
- Release Co-ordinator/ Business Implementation manager.
- Development Lead.

Sign-off confirms readiness for release.