TEST PLAN FOR E-COMMERCE WEBSITE (Playwright + TypeScript)

1. Introduction

This test plan describes the QA strategy, scope, tools, environments, test types, execution cycles, and reporting structure for automating an e-commerce website using Playwright with TypeScript.

2. Objectives

- Validate core e-commerce functionality (browse, search, cart, checkout, payments, order tracking).
- Ensure stable coverage across major browsers.
- Implement scalable & maintainable Playwright automation framework using TypeScript.
- Integrate automated tests into CI/CD.
- Ensure performance, accessibility, security, and usability standards.

3. Scope

3.1 In-Scope

- Web UI automation using Playwright (Chromium, Firefox, WebKit).
- Functional testing (search, product pages, cart, orders).
- Smoke, regression, sanity test suites.
- API testing via Playwright request context.
- Visual regression (optional).
- Cross-browser validation.
- Data-driven testing.

3.2 Out-of-Scope

- Mobile native apps.
- Load testing beyond basic page benchmarks.
- Manual UX validation.

4. Test Approach

4.1 Automation Framework

Architecture:

- Playwright + TypeScript
- Page Object Model (POM)
- Fixture-based test design
- Reusable helper utilities
- CI integration

5. Features to be Tested

- Landing pages & navigation
- Authentication workflows
- Product listing & detail pages
- Search functionality
- Shopping cart
- Checkout
- Payment module
- Order management
- User profile
- (Optional) Admin module

6. Test Types

- Functional tests (UI + API)
- Integration tests
- Accessibility tests
- Performance basics
- Security checks

7. Test Deliverables

- Test plan
- Automated scripts
- Test data
- Reports (HTML/Allure)
- CI pipeline configuration

8. Tools & Technology

- Playwright, TypeScript, Node.js
- Allure or HTML Reporter
- GitHub Actions/Jenkins
- Docker (optional)
- axe-core

9. Test Execution Plan

Environments:

- QA
- Staging
- Production (smoke only)

Schedule:

- Environment Prep
- Framework Setup
- Test Case Design
- Script Development
- Test Execution
- Reporting

10. Entry & Exit Criteria

Entry:

- Stable build deployed

- Automation framework ready

Exit:

- No high-severity defects

- 95% test completion

11. Risks & Mitigation

- Environment instability → Mock/stub data

- UI changes → POM structure

- Third-party failures → API stubs

12. Reporting

- Daily execution summary

- Defect tracking

- Weekly progress reports

13. Sample Folder Structure

tests/

pages/

fixtures/

utils/

config/

reports/

14. Exit Deliverables

- Automated test suite

- Documentation

- CI integration

- Coverage report