UI Automation Strategy and Framework Plan: Web E2E Focus

Target Platform: Choithrams E-commerce Web Portal

Objective: To establish a stable, maintainable, and scalable End-to-End (E2E) automation framework that ensures continuous quality assurance across the web portal using modern tooling.

1. Automation Goals and Scope (Web E2E)

1.1 Key Goals

1. **Accelerate Regression:** Significantly reduce the time required for manual regression testing on the web portal.
2. **Continuous Validation:** Provide rapid, reliable validation for the most critical user flows.
3. **Cross-Browser Reliability:** Ensure core functionality works consistently across all major supported web browsers (Chromium, Firefox, WebKit).
4. **High-Risk Coverage:** Achieve maximum automated coverage for all business-critical, revenue-generating user flows.

1.2 Automation Prioritization (Web Portal)

Priority Level	**Focus Area**	**Rationale**
P1 - Smoke/Critical E2E	User Login/Logout, Homepage Loading, Site Navigation.	Validates environment stability and basic accessibility.
P2 - Critical Business Flow	**Full Checkout Funnel** (Add to Cart, Address Selection, Payment Page Load)	Directly impacts revenue generation and core business success.
P3 - Core Regression	Product Search & Filtering, Wishlist functionality, User Profile updates.	Ensures the core shopping experience remains reliable.

P4 - Feature/Acceptance	Forms (Contact Us, Newsletter Signup), Less-frequently used settings.	Only automated if time permits and the feature is stable.

1.3 Exclusions

- **Aesthetics and Usability:** Subjective visual checks or complex usability validation will remain the domain of manual QA.
- **Exploratory Testing:** Tasks requiring human intuition and deep domain knowledge to uncover unexpected defects.

2. Tool Stack and Technical Rationale

The framework will utilize a modern, JavaScript/TypeScript-based tool stack to leverage faster execution and better integration.

Platform	**Primary Tool**	**Secondary Tool/Language**	**Rationale**
Web (E2E)	**Playwright**	TypeScript (Primary Language)	Selected for its superior speed, reliable auto-wait handling, and native cross-browser support (Chromium, Firefox, WebKit).
Reporting	**Allure Reporter**	HTML/JavaScript	Generates rich, interactive HTML reports with execution history, screenshots, and visual timelines for failed tests, significantly aiding debugging.
Framework Base	**Node.js**	N/A	Provides the execution runtime environment for the entire stack.

3. Framework Architecture (The Page Object Model)

The framework will strictly adhere to the **Page Object Model (POM)** pattern to ensure high maintainability, especially critical for an e-commerce platform where UI elements are subject to frequent changes.

3.1 Framework Layers

Layer	Responsibility	Details
Test Layer (/tests)	Contains the execution flow (the *what*).	Describes the end-to-end user scenario (e.g., checkout_flow.spec.ts). **Must not contain locators.**
Page Layer (POM) (/pages)	Defines elements and methods for a specific page (the *where* and *how*).	LoginPage.ts contains locators (usernameInput) and actions (login(user, pass)). This layer acts as the single source of truth for all UI elements.
Utility Layer (/utils)	Reusable functions not tied to a specific page.	DataGenerator.ts: Generates unique test data. FixtureSetup.ts: Logic for setting up pre-conditions (e.g., managing cookies or local storage).
Configuration Layer (/config)	Stores environment-agnostic variables.	Base URLs for environments (Dev, Staging), default timeouts, and browser launch arguments. **Credentials must be handled externally.**

4. Maintenance and Quality Metrics

4.1 Framework Maintenance

- **Locator Strategy:** Prefer robust, non-volatile locators like data-testid attributes or unique IDs over unstable XPath or class names.
- **Code Standard:** All new automation code will be written in **TypeScript** to enforce type safety and improve code quality and readability across the team.

- **Flaky Test Mitigation:** Any test that fails intermittently will be immediately quarantined and assigned to the author for investigation, preventing false positives from undermining team trust in the suite.

4.2 Quality Metrics (KPIs)

The internal success of the framework's health will be measured by the following Key Performance Indicators:

Metric	Target	Focus
Test Pass Rate	> 95%	A measure of overall framework stability and test script quality.
Locator Reusability	High	A measure of how effectively the POM is reducing element duplication.
Test Execution Time	Low	Ensures the framework is fast, focusing on efficient waiting and optimal locator usage.