Test Strategy – SauceDemo UI Automation (Selenium + TestNG)

---

1. Objective

The objective of this test strategy is to define a lean, maintainable, and scalable automation approach for testing the SauceDemo e-commerce platform. The goal is to verify the core functionality (login, cart, and checkout) and ensure that critical user journeys work as expected under normal and negative scenarios using **Selenium WebDriver**, **TestNG**, and **Java**.

---

2. Automation Scope & Prioritisation

Automation is prioritised based on **critical business flows**, **risk exposure**, and **stability** of test elements.

High Priority (Automate First)

- Log in with valid credentials (standard_user)
- Locked-out user validation (locked_out_user)
- Adda single item to the cart
- Complete the checkout process with confirmation
- Price verification at checkout

Medium Priority (Automate Later)

- Sorting/filtering of products
- Cross-browser layout checks
- Multiple item checkout combinations

Low Priority (Manual/Skip)

- UI responsiveness, animations, and edge-case visuals, unless time permits.

3. Key Risks in the Application

Risk Area	Description
Authentication	Login errors, especially handling of locked users or invalid credentials.
Add to Cart	Incorrect item added, pricing mismatch, or cart not retaining state.

Checkout Flow	Calculation errors in subtotal/total, and broken navigation during checkout.
Session State	Unexpected logouts or retained sessions across logins.
UI Consistency	Misaligned buttons, missing labels, or misleading messages.

---

4. Test Architecture & Design

- **Framework**: Selenium WebDriver + TestNG
- **Design Pattern**: Page Object Model (POM)
- **Structure**:
  - pages/: All reusable page classes with locators and actions.
  - tests/: All test cases.
  - utils/: Configuration, test data handling, listeners, etc.

5. Execution Strategy

CI/CD Integration:

- Tests will be integrated with **GitHub**, **GitHub Actions**, **Jenkins**, or similar CI tools.
- The suite will:
  - Run on every **pull request** to the main branch
  - Execute in **headless mode** on Chrome for speed
  - Publish **HTML and XML reports**

Test Environments:

- **Browser**: Chrome (Headless for CI), optionally Firefox
- **OS**: Cross-platform (Windows, Mac, Linux)

6. Reporting & Maintenance

- Reports generated via **TestNG's default HTML reports**
- Future enhancement: **ExtentReports** for detailed UI logs
- Easy to plug in **Allure or ReportNG** if required
- Locators and test data are managed externally for better maintainability.