Automation Strategy laboratoriodetesting.com

Automation challenge activity

Introduction

This document outlines the automation testing strategy for the e-commerce platform laboratoriodetesting.com. As the sole QA in a team of 3 developers during Sprint 6 of 18, the focus is on establishing a robust automation foundation that can scale with upcoming feature developments while addressing current quality issues.

Project Context & Business Understanding

Current State.

- Project Phase: Sprint 6 of 18 (33% complete)
- Team Structure: 1 QA + 3 Developers
- Known Issues:
  - Registration bugs (client-reported)
  - Checkout process bugs
- Upcoming Changes:
  - Payment method expansion (Sprints 7-9)
  - Homepage UX redesign (Sprints 15-18)
- Architecture: API-driven backend with frontend integration

Business Impact.

The automation strategy prioritizes:

1. **Revenue Protection:** Focus on checkout and payment flows
2. **User Acquisition:** Ensure registration and login reliability
3. **Customer Experience:** Validate core shopping journey
4. **Future-Proofing:** Build scalable framework for upcoming features

Testing Scope & Priorities

High Priority (Critical Path)

1. **User Registration & Authentication:**
   - New user registration (with known bugs)
   - User login/logout
   - Session management
2. **E-commerce Core Flow:**
   - Product browsing and selection

    - Shopping cart operations (add/remove/update)
    - Checkout process (with known bugs)
3. **Order Management:**
    - Order placement and confirmation
    - Order history viewing

Medium Priority (Supporting Features)

1. Product Search and Filtering
2. User Profile Management
3. Error Handling and Validation

Low Priority (Future Considerations)

1. Performance Testing
2. Cross-browser Compatibility
3. Mobile Responsiveness

Automation Approach

Framework Selection: Cypress.
Rationale:

- Required by the challenge
- Excellent for e-commerce applications
- Strong API testing capabilities
- Real-time debugging and testing
- Built-in waiting mechanisms

Architecture Pattern: Page Object Model (POM)
Benefits:

- Maintainable and scalable structure
- Reduces code duplication
- Better organization of locators and actions
- Easier to update when UI changes occur

Test Data Strategy
Approach:

- Use provided test users with different permission levels (TBD)
- Implement data-driven testing for comprehensive coverage
- Create test data fixtures for consistent testing
- Separate test data from test logic

Test Users :

- **User:** huge.test@gmail.com
- **Password:** Huge2025.
- **Valid credit card #** : xxxx

- **User:** huge2.test@gmail.com
- **Password:** Monday12.
- **Valid credit card #**: xxx

Critical Test Scenarios

Scenario 1: User Registration Flow

Priority: Critical
Rationale: Addresses known client-reported bugs
Coverage:

- Valid registration with all required fields
- Invalid email format validation
- Password strength requirements
- Duplicate email handling
- Success confirmation and redirect

Scenario 2: Complete Purchase Journey (HIGHEST PRIORITY - Week 1)

Priority: Critical - IMMEDIATE IMPLEMENTATION
Rationale: Revenue-generating functionality about to change in Sprint 7
Timeline: Must be completed in Week 1 before payment expansion begins
Coverage:

- Product selection and cart addition

- Cart management (quantity updates, removal)
- **Current checkout process validation** (baseline before changes)
- **Existing payment method testing** (credit card only - Sprint 6)
- Order confirmation and details
- Integration with backend APIs

Sprint-Aware Approach: This scenario tests the current state and serves as regression baseline when new payment methods are added in Sprints 7-9.

Scenario 3: User Authentication & Session Management

Priority: High
Rationale: Fundamental user experience requirement
Coverage:

- Valid login credentials
- Invalid login attempts
- Session persistence
- Logout functionality
- Access control validation

Implementation Strategy

Sprint-Aware Timeline Consideration.

Given we're in Sprint 6 of 18 with critical changes starting in Sprint 7 (payment expansion) and Sprint 15 (homepage redesign), our implementation must prioritize testing current functionality before it changes.

Phase 1: Foundation + Critical Path (Week 1 - URGENT)

Goal: Establish baseline tests for functionality that will change in Sprint 7

- **Days 1-2:** Rapid project setup and configuration
- **Days 3-7: PRIORITY 1** - Complete Purchase Journey testing
    - Current checkout process (before payment expansion)
    - Shopping cart operations
    - Existing credit card payment flow
    - Order confirmation process

Rationale: Payment method expansion starts Sprint 7. We must capture current checkout functionality as baseline before it changes.

Phase 2: Stable Functionality (Week 2)

Goal: Implement scenarios for features unlikely to change during payment expansion

1. **Days 8-10:** User Registration Flow implementation
   - Address known client-reported bugs
   - Comprehensive validation testing
2. **Days 11-14:** User Authentication & Session Management
   - Login/logout functionality
   - Session persistence testing
   - Existing credit card payment flow
   - Order confirmation process

Rationale: Payment method expansion starts Sprint 7. We must capture current checkout functionality as baseline before it changes.

Phase 3: Adaptive Evolution (Weeks 3-6)

Goal: Evolve tests to accommodate new features while maintaining regression coverage

3. **Weeks 3-4:** Monitor and adapt to payment method additions
   - Update checkout tests for new payment options
   - Maintain regression coverage for existing functionality
   - Incremental test enhancements
4. **Weeks 5-6:** Optimization and future-proofing
   - Framework optimization for upcoming homepage changes (Sprint 15)
   - Documentation and handover preparation
   - Performance baseline establishment

Risk Mitigation

Sprint Timeline Risks.

1. Payment Expansion (Sprints 7-9):

- **Mitigation:** Establish checkout baseline tests in Week 1 before changes begin
- **Strategy:** Design flexible payment testing framework for easy extension
- **Monitoring:** Weekly check-ins with development team during payment implementation

2. Homepage Redesign (Sprints 15-18):

- **Mitigation:** Use stable locator strategies (data-testid, role-based selectors)
- **Strategy:** Implement Page Object Model for easy UI change adaptation
- **Preparation:** Design homepage tests for minimal maintenance during redesign

Known Issues Management

1. Registration Bugs:

- Implement comprehensive validation testing in Week 2
- Document specific failure scenarios for development team

2. Checkout Bugs:

- **URGENT:** Test current checkout functionality in Week 1 as baseline
- Create detailed API-level validation
- Verify data consistency across systems

Change Management Strategy

1. Adaptive Testing Framework:

- Design modular test structure for easy feature addition
- Implement configuration-driven test data for different sprint phases
- Create regression test suite to validate existing functionality during changes

2. Communication Protocol:

- Weekly sync with development team during active feature development (Sprints 7-9)
- Immediate notification system for checkout-related code changes
- Documentation of test impacts for each new payment method

Technical Considerations

Change-Resilient Framework Design

- Flexible Page Objects: Design checkout components to accommodate new payment methods

- Configuration-Driven Tests: Separate test data by sprint phase (current vs. future payment options)
- Modular Architecture: Enable easy addition of new payment validation without framework changes

Locator Strategy

- Prioritize stable attributes (data-testid, role, aria-labels)
- Avoid fragile selectors (absolute CSS paths, text-based)
- Implement locator centralization in page objects
- Sprint-specific consideration: Focus on payment-related element stability during expansion

API Testing Integration

- Validate backend responses independently
- Compare frontend state with API responses
- Test API error scenarios and frontend handling
- **Priority focus**: Payment API validation before Sprint 7 changes

Sprint-Aware Test Data Management

JavaScript
```javascript
// Example: fixtures/payment-data.json
{
  "sprint6": {
    "availablePayments": ["credit-card"],
    "defaultPayment": "credit-card"
  },
  "sprint7-9": {
    "plannedPayments": ["paypal", "apple-pay", "bank-transfer"],
    "implementation": "incremental"
  }
}
```

Reporting and Monitoring

- Implement Mochawesome reporting for detailed test results
- Set up CI/CD integration for automated execution
- Create dashboard for test health monitoring

- Change tracking: Document test evolution through sprint phase

Success Metrics

Sprint-Aligned Quality Metrics

- Week 1 Success: Baseline checkout functionality captured before Sprint 7 changes
- Week 2 Success: Complete coverage of stable features (registration, authentication)
- Ongoing Success: Test suite adapts to new payment methods without breaking existing coverage
- Final Success: 90% coverage of critical user journeys with <95% pass rate on stable builds

Change Adaptation Metrics

- Regression Detection: Immediate identification when new features break existing functionality
- Feature Coverage: New payment methods tested within 1 sprint of implementation
- Framework Resilience: <20% test maintenance overhead during active development phases

Performance Metrics

- Execution Time: Complete test suite under 10 minutes
- Feedback Loop: Results available within 15 minutes of code commit
- Sprint Integration: Test results inform sprint retrospectives and planning

Dependencies & Requirements

Technical Requirements

- Cypress framework (latest stable version)
- Node.js v14+ environment
- GitHub repository setup
- Test environment access

Team Dependencies

- Development team API documentation
- Test environment provisioning
- Access to staging/test databases
- Coordination with UX team for upcoming changes

Conclusion

This automation strategy provides a sprint-aware approach to testing the laboratoriodetesting.com platform, strategically prioritizing functionality that will change in Sprint 7 while building a foundation for long-term scalability. By implementing checkout functionality tests immediately (Week 1), we establish crucial baseline coverage before payment method expansion begins.

The emphasis on change-resilient design and adaptive implementation ensures the framework can evolve seamlessly with the product throughout the remaining 12 sprints. The three critical scenarios (Purchase Journey, Registration, and Authentication) are prioritized by change probability rather than business importance alone, maximizing test value during active development phases.

This strategic approach balances immediate sprint risks with long-term quality goals, ensuring robust automation coverage while minimizing maintenance overhead during periods of rapid feature development. The framework design anticipates both the upcoming payment expansion (Sprints 7-9) and homepage redesign (Sprints 15-18), positioning the automation suite as a strategic asset rather than a maintenance burden.