# Testing Manual

Helix

August 12, 2024

## Contents

# 1 Introduction

This manual outlines the testing strategy and processes for the Smart Inventory Web App. It serves as a guide for quality assurance team members, developers, and stakeholders involved in the testing process.

# 2 Test Strategy

Our test strategy focuses on a comprehensive approach, combining various testing types to ensure the reliability, usability, and performance of the Smart Inventory Web App. We employ a shift-left testing approach, integrating testing throughout the development life-cycle.

# 3 Types of Tests

1. Unit Tests: Verify individual functions, methods, and small code units in isolation.
2. Component Tests: Validate the functionality of isolated UI components.
3. Integration Tests: Ensure different parts of the application work together correctly.
4. End-to-End (E2E) Tests: Simulate real user scenarios, validating the entire application workflow.
5. API Tests: Verify the functionality and reliability of AWS backend services.
6. Performance Tests: Evaluate the system's responsiveness and stability under various load conditions.
7. Security Tests: Identify vulnerabilities and ensure data protection.
8. Accessibility Tests: Ensure the application is usable by people with disabilities.

Test Coverage Goal: Minimum 80% for unit and integration tests.

# 4 Test Tools

Cypress

- Primary tool for unit, component, integration, and E2E testing
- Provides a browser-based testing environment
- Supports automated test script creation and execution

Postman

- Used for API testing
- Simulates HTTP requests to API endpoints
- Manages test environments and collections
- Supports automated API test scripts

AWS CloudWatch

- Monitors and logs AWS service performance
- Helps in identifying and debugging issues in AWS resources

# 5 Quality Assurance Requirements

- Reliability: Ensured through comprehensive test coverage across all test types.
- Usability: Primarily addressed by E2E tests and usability testing sessions.
- Maintainability: Enhanced by well-documented, modular test suites.
- Scalability: Verified through load testing and AWS service configuration tests.
- Security: Ensured through regular security audits and penetration testing.

# 6 Testing Processes

## 6.1 Component Tests

- Use Cypress Component Testing
- Mount individual Angular components in isolation
- Interact with components using Cypress commands
- Assert expected behavior and DOM changes

## 6.2 Integration Tests

- Test interactions between different modules or services
- Use Cypress for frontend integration testing
- Implement API mocking when necessary

## 6.3 End-to-End (E2E) Tests

Write tests simulating complete user journeys Use Cypress for E2E testing Interact with UI elements using Cypress commands Assert expected application state and API responses Run tests with npx cypress run

## 6.4 API Tests

Use Postman to test AWS API Gateway endpoints Create collections for each API resource Write tests for different HTTP methods (GET, POST, PUT, DELETE) Test both success and error scenarios Use environment variables for different stages (dev, staging, prod) Run tests manually or integrate with CI/CD pipeline

# 7 Test Data Management

Maintain a separate test database with anonymized data Use data factories to generate test data programmatically Implement data cleanup procedures after test execution Version control test data alongside the codebase

# 8 Reporting and Documentation

Generate automated test reports after each test run Maintain up-to-date test case documentation Create and update user guides and API documentation Document known issues and workarounds

## 9 Continuous Integration and Deployment

Integrate all tests into the CI/CD pipeline Run unit and integration tests on every commit Execute E2E tests before deployment to staging Perform smoke tests after each production deployment

## 10 Issue Tracking and Resolution

Use a dedicated issue tracking system (e.g., Jira, GitHub Issues) Categorize and prioritize issues based on severity and impact Implement a clear workflow for issue resolution and retesting Conduct regular triage meetings to review and update issue status

## 11 Test Review and Sign-off

Conduct peer reviews of test cases and scripts Hold regular test results review meetings with stakeholders Establish clear criteria for test pass/fail and release readiness Obtain formal sign-off from relevant stakeholders before major releases