SOFTWARE TESTING QUALITY ASSURANCE FINAL TECHNICAL REPORT

SauceDemo Web Application & DummyJSON REST API

Anood Al-Naimat

February 2026

Table of Contents

1. EXECUTIVE SUMMARY

This project presents a complete Quality Assurance (QA) testing portfolio for a retail-style system, delivered as part of the Software Testing – Quality Assurance capstone assignment. The objective of the project was to plan, design, execute, automate, and report testing activities across multiple testing levels, including manual testing, API testing, API performance testing, and UI automation.

The system under test consists of two real-world practice targets: the SauceDemo web application, used to validate end-to-end e-commerce user flows, and the DummyJSON REST API, used to practice backend API validation and performance testing. The testing approach followed standard QA best practices aligned with SDLC and STLC principles.

The project outcomes include a documented test plan, executed manual test cases with evidence, identified defects, an automated Postman collection with Newman reporting, a k6 performance test with detailed analysis, and a UI automation suite. Overall, the system demonstrated functional stability, reliable API behavior, and strong performance under the tested load conditions, with identified limitations and improvement recommendations documented in this report.

2. SYSTEM UNDER TEST (SUT) AND ASSUMPTIONS

2.1 System Under Test

The project targets a simulated retail platform named Loyalty+ Store (Real Targets), represented by the following systems:

- Web UI (Frontend): SauceDemo (https://www.saucedemo.com)
- Features: Login, product browsing, sorting, cart management, checkout, and logout.
- REST API (Backend): DummyJSON (https://dummyjson.com)
- Features: Authentication, products, carts, users, and related retail-like operations.

These systems were selected according to the assignment brief and are commonly used for QA practice and automation.

2.2 Assumptions

- All testing was performed using test/demo data only.
- No real personal, payment, or sensitive data was used.
- Security testing (penetration testing, scanning, brute force) was intentionally excluded.
- DummyJSON simulates write operations (POST/PUT/DELETE), and data persistence is not guaranteed.
- The systems may change over time; therefore, tests focus on stable and core functionality.

3. TEST STRATEGY AND SCOPE

3.1 Test Strategy

A layered testing strategy was applied to ensure comprehensive coverage:

- Manual Testing: Validate core business flows and user experience.
- API Testing: Validate backend logic, data integrity, and error handling.
- API Performance Testing: Measure system behavior under concurrent load using k6.
- UI Automation: Automate critical regression paths using a maintainable framework structure.

Risk-based prioritization was applied, focusing first on critical user journeys such as authentication, cart operations, and checkout.

3.2 Test Scope

In Scope:

- Functional UI testing of SauceDemo
- Manual test case execution and defect reporting
- API functional testing with Postman
- Data-driven API testing
- Newman automated execution and reporting
- API performance testing with smoke and load profiles
- UI automation of critical flows

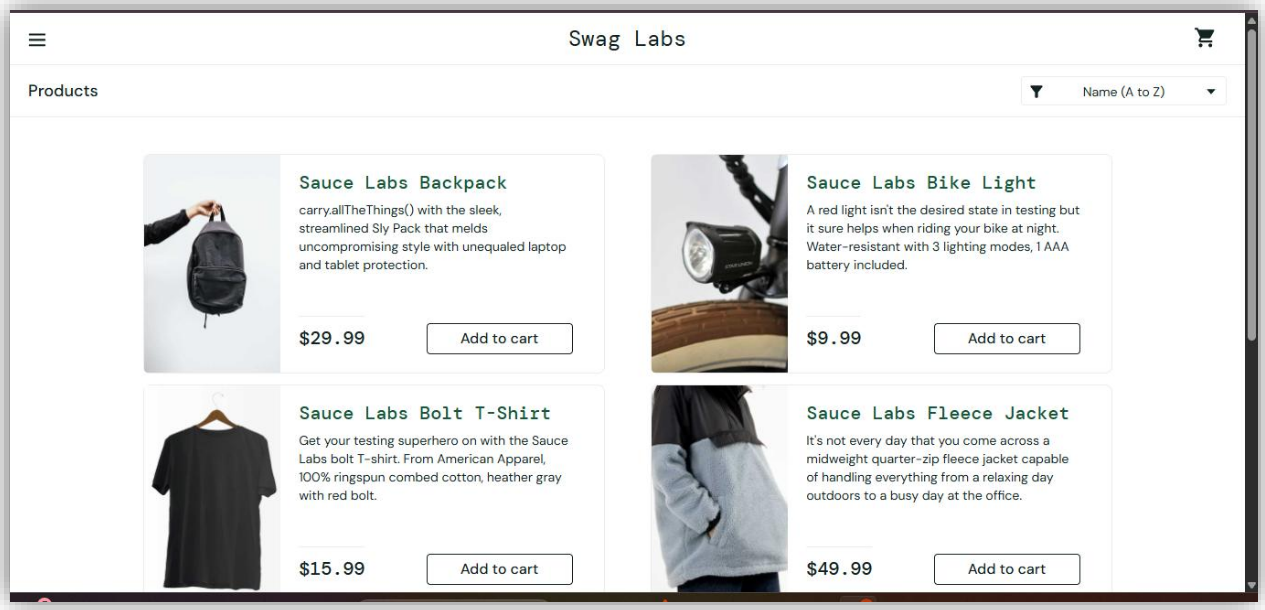
Out of Scope:

- Security testing
- Cross-browser testing
- Mobile automation (Appium)
- Production deployment testing

4. MANUAL TESTING: SCENARIOS, COVERAGE, AND EXECUTION SUMMARY

Reference to Evidence: Manual test execution screenshots are provided in Appendix A.

Manual testing covered key user scenarios including valid and invalid login, product browsing and sorting, add and remove items from the cart, checkout flow, and logout. Both positive and negative scenarios were executed to ensure adequate functional coverage.



Sauce Labs Backpack

carry.allTheThings() with the sleek, streamlined Sly Pack that melds uncompromising style with unequaled laptop and tablet protection.

$29.99

Sauce Labs Bike Light

A red light isn't the desired state in testing but it sure helps when riding your bike at night. Water-resistant with 3 lighting modes, 1 AAA battery included.

$9.99

Sauce Labs Bolt T-Shirt

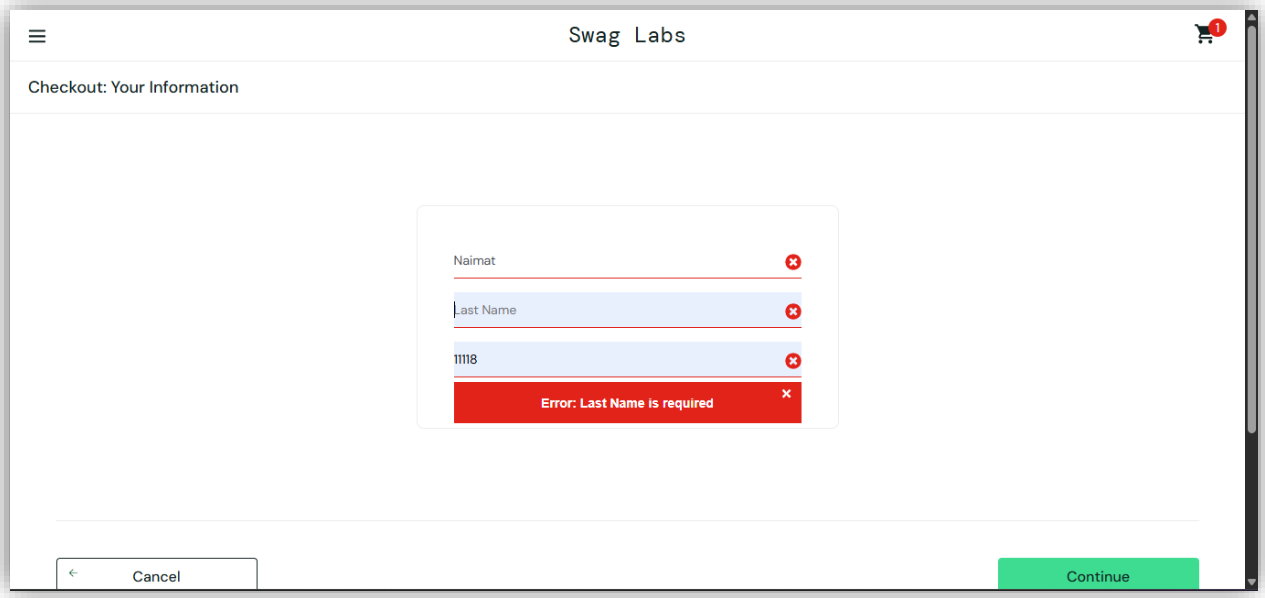
Get your testing superhero on with the Sauce Labs bolt T-shirt. From American Apparel, 100% ringspun combed cotton, heather gray with red bolt.

$15.99

Sauce Labs Fleece Jacket

It's not every day that you come across a midweight quarter-zip fleece jacket capable of handling everything from a relaxing day outdoors to a busy day at the office.

$49.99



Swag Labs

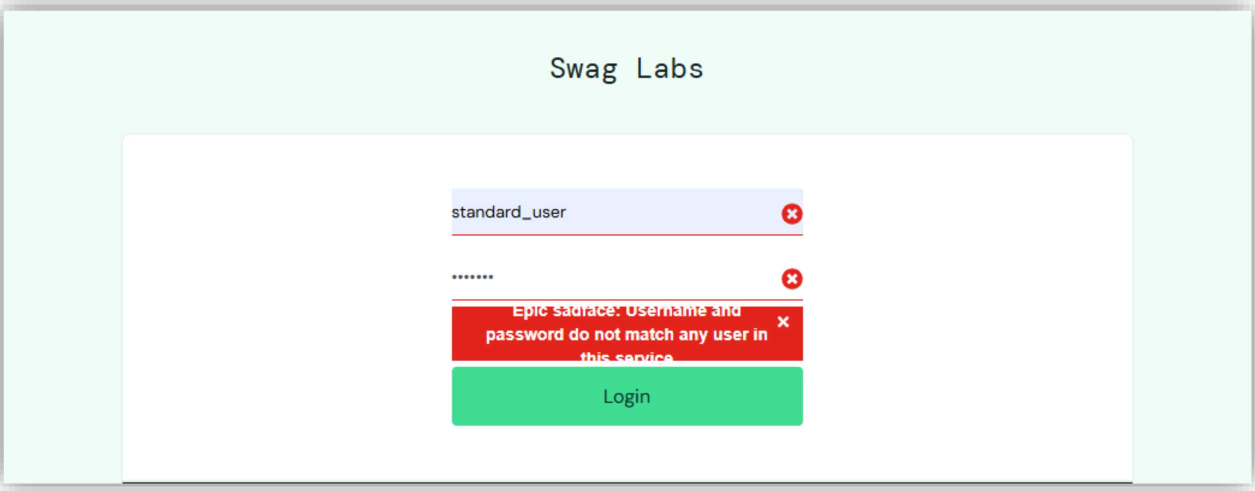
Checkout: Your Information

Naimat

Last Name

11118

Error: Last Name is required



Swag Labs

standard_user

•••••••

Epic sadface: Username and password do not match any user in this service

[Figure A1-A3: Manual Test Execution Evidence (Login, Product Browse, Add to Cart, Checkout, Logout)]

4.4 Manual Test Execution Summary Table

Table 1: Manual Test Execution Summary

Metric	Count
Total Test Cases	38
Passed	26
Failed	12
Blocked	0

5. DEFECTS: TOP ISSUES, SEVERITY DISTRIBUTION, AND EXAMPLES

Reference to Evidence: Detailed defect screenshots and reproduction evidence are provided in Appendix B.

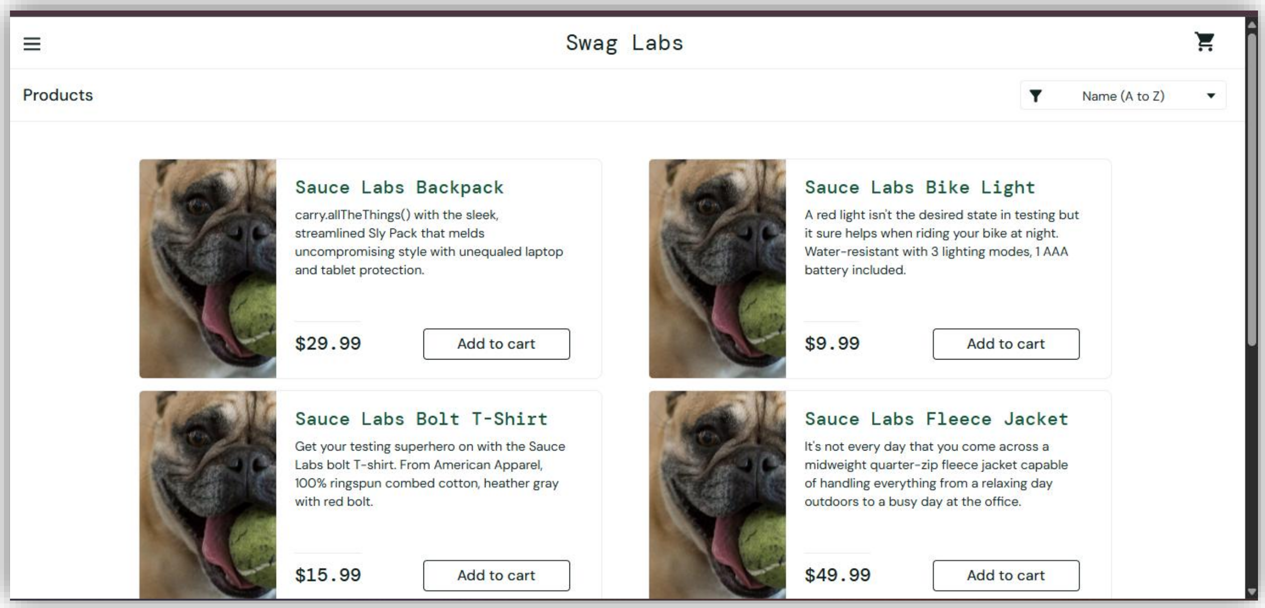
5.1 Defect Overview

A defect log was created to capture identified issues during manual testing. Defects were documented with clear steps to reproduce, expected results, actual results, severity, and supporting evidence.

5.2 Severity Distribution

Table 2: Defect Severity Distribution

Severity	Number of Defects
High	4
Medium	7
Low	1
Total	12



Swag Labs

Products

Name (A to Z)

Sauce Labs Backpack

carry.allTheThings() with the sleek, streamlined Sly Pack that melds uncompromising style with unequaled laptop and tablet protection.

$29.99

Add to cart

Sauce Labs Bike Light

A red light isn't the desired state in testing but it sure helps when riding your bike at night. Water-resistant with 3 lighting modes, 1 AAA battery included.

$9.99

Add to cart

Sauce Labs Bolt T-Shirt

Get your testing superhero on with the Sauce Labs bolt T-shirt. From American Apparel, 100% ringspun combed cotton, heather gray with red bolt.

$15.99

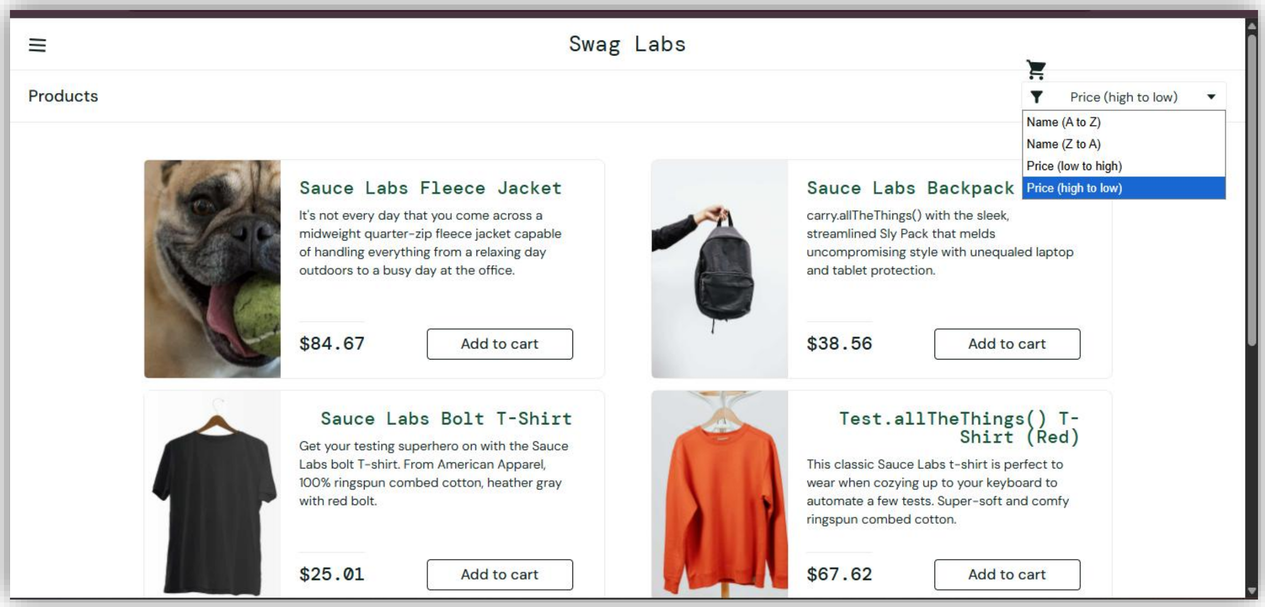
Add to cart

Sauce Labs Fleece Jacket

It's not every day that you come across a midweight quarter-zip fleece jacket capable of handling everything from a relaxing day outdoors to a busy day at the office.

$49.99

Add to cart


Swag Labs

Products

Price (high to low)

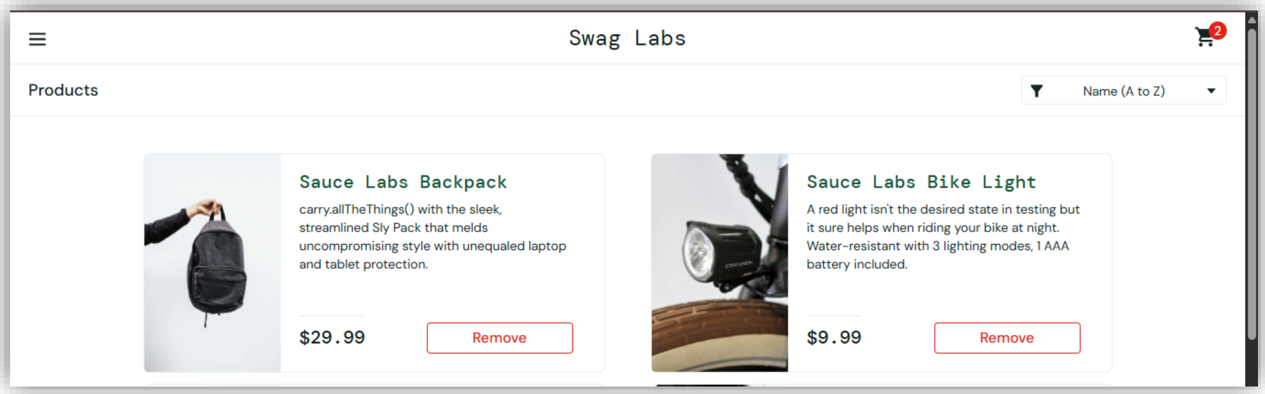
Name (A to Z)
Name (Z to A)
Price (low to high)
Price (high to low)

Sauce Labs Fleece Jacket
It's not every day that you come across a midweight quarter-zip fleece jacket capable of handling everything from a relaxing day outdoors to a busy day at the office.
$84.67 Add to cart

Sauce Labs Backpack
carry.allTheThings() with the sleek, streamlined Sly Pack that melds uncompromising style with unequaled laptop and tablet protection.
$38.56 Add to cart

Sauce Labs Bolt T-Shirt
Get your testing superhero on with the Sauce Labs bolt T-shirt. From American Apparel, 100% ringspun combed cotton, heather gray with red bolt.
$25.01 Add to cart

Test.allTheThings() T-Shirt (Red)
This classic Sauce Labs t-shirt is perfect to wear when cozying up to your keyboard to automate a few tests. Super-soft and comfy ringspun combed cotton.
$67.62 Add to cart


Swag Labs

2

Products

Name (A to Z)

Sauce Labs Backpack
carry.allTheThings() with the sleek, streamlined Sly Pack that melds uncompromising style with unequaled laptop and tablet protection.
$29.99 Remove

Sauce Labs Bike Light
A red light isn't the desired state in testing but it sure helps when riding your bike at night. Water-resistant with 3 lighting modes, 1 AAA battery included.
$9.99 Remove

[Figure B1-B3: Defect Evidence (Image Issues, Sorting Problems, Remove Button)]

5.3 Example Defect

- Defect Title: Checkout swaps First Name and Last Name and shows incorrect validation error
- Severity: Medium
- Steps: Proceed to checkout without filling all mandatory fields
- Expected Result: Validation message displayed
- Actual Result: Checkout proceeds without blocking

6. API TESTING (POSTMAN): COLLECTION DESIGN, ASSERTIONS, AND NEWMAN SUMMARY

Reference to Evidence: Postman collection structure, assertions, and Newman execution reports are provided in Appendix C and Appendix D.

6.1 Collection Design

A structured Postman collection was created with over 20 API requests, organized into logical folders:

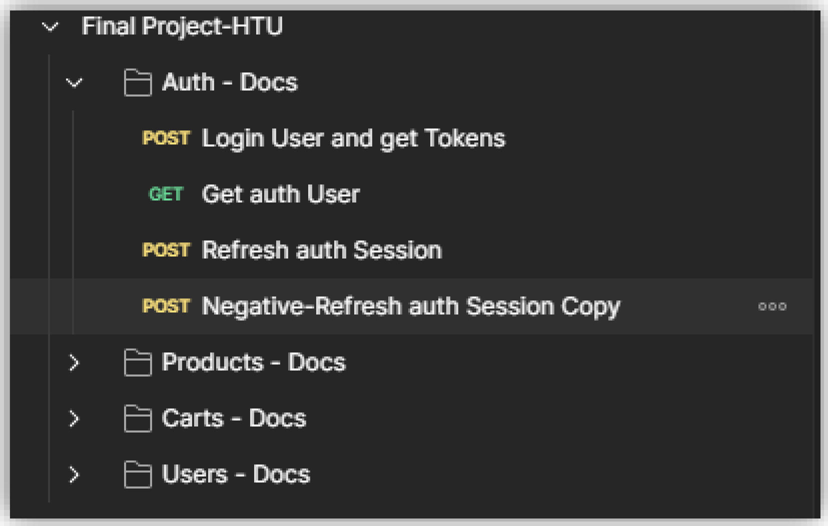
- Authentication
- Products
- Carts
- Users

Environment variables were used for baseURL, accessToken, and dynamic IDs to ensure reusability and maintainability.

6.2 Key Assertions

Automated tests were added to validate:

- HTTP status codes
- Response structure and mandatory fields
- Token generation and reuse
- Error handling for negative scenarios
- Response time thresholds


Final Project-HTU

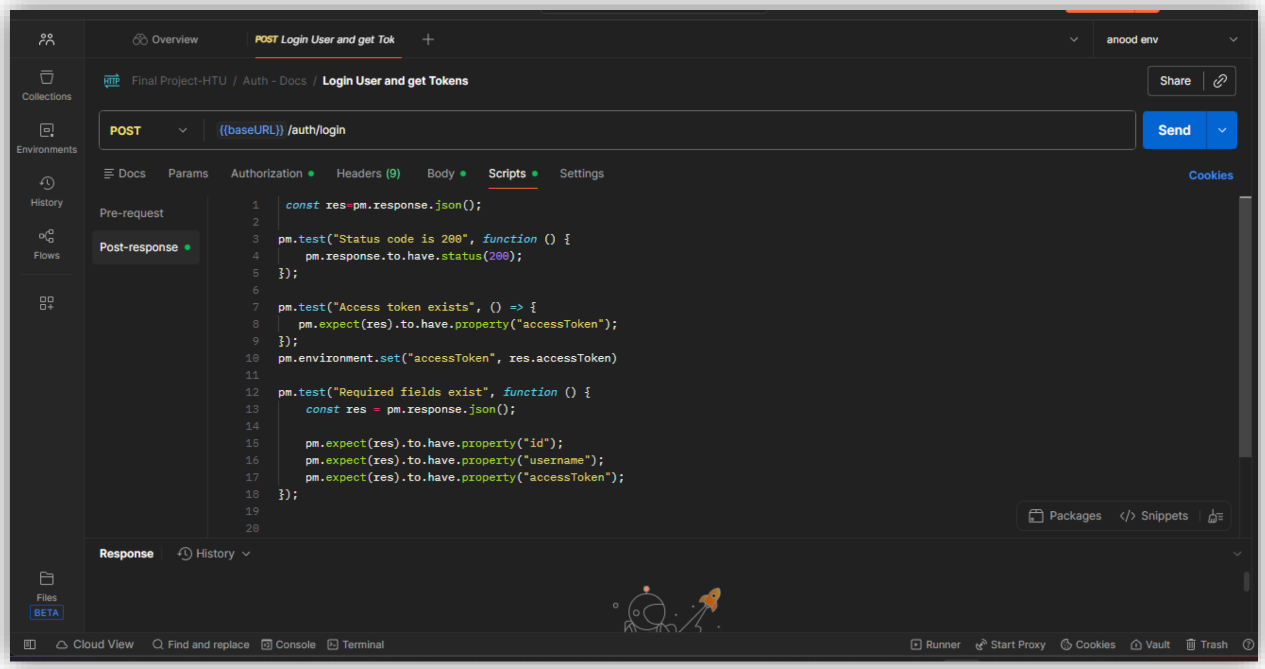
Auth - Docs

- POST Login User and get Tokens
- GET Get auth User
- POST Refresh auth Session
- POST Negative-Refresh auth Session Copy

Products - Docs

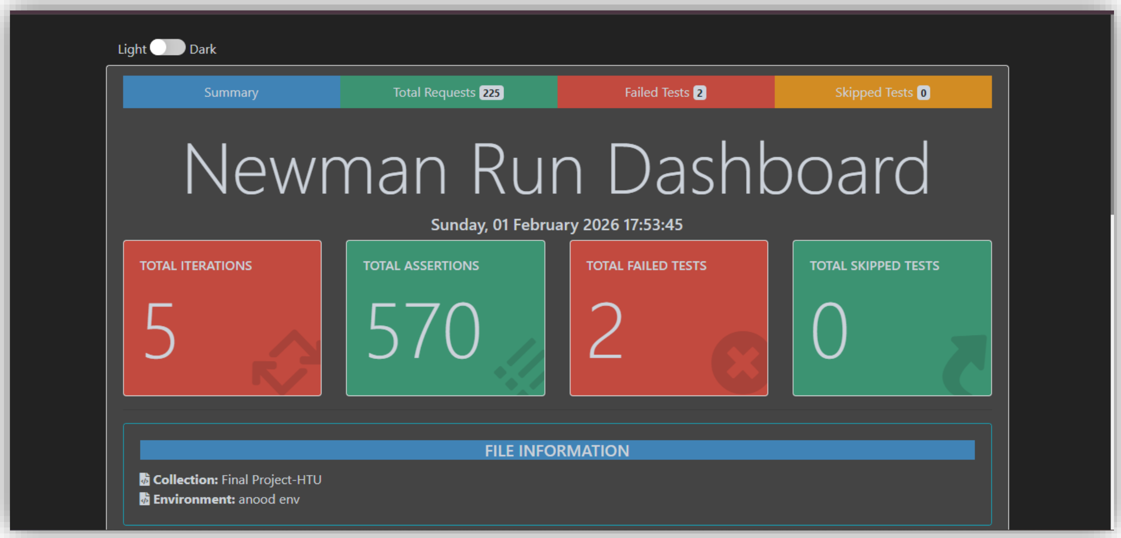
Carts - Docs

Users - Docs



POST {{baseURL}} /auth/login

```
 const res=pm.response.json();

pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Access token exists", () => {
   pm.expect(res).to.have.property("accessToken");
});
pm.environment.set("accessToken", res.accessToken)

pm.test("Required fields exist", function () {
    const res = pm.response.json();

    pm.expect(res).to.have.property("id");
    pm.expect(res).to.have.property("username");
    pm.expect(res).to.have.property("accessToken");
});
```


Light Dark

Summary | Total Requests 225 | Failed Tests 2 | Skipped Tests 0

Newman Run Dashboard

Sunday, 01 February 2026 17:53:45

TOTAL ITERATIONS	TOTAL ASSERTIONS	TOTAL FAILED TESTS	TOTAL SKIPPED TESTS
5	570	2	0

FILE INFORMATION

Collection: Final Project-HTU
Environment: anood env

[Figure C1-C3, D1: Postman Collection Structure and Newman Execution Reports]

6.3 Newman Execution Summary

Table 3: Postman and Newman Execution Summary

Metric	Result
Total Requests	225
Total Assertions	570
Failed Assertions	2
Average Response Time	164 ms

7. PERFORMANCE TESTING (K6): TEST PLAN, LOAD PROFILES, RESULTS, AND ANALYSIS

Reference to Evidence: k6 execution outputs, KPI summaries, and threshold results are provided in Appendix E.

7.1 Test Plan

A k6 test script was created targeting three API endpoints:

- Search Users
- Filter Users
- Paginated Users

7.2 Load Profiles

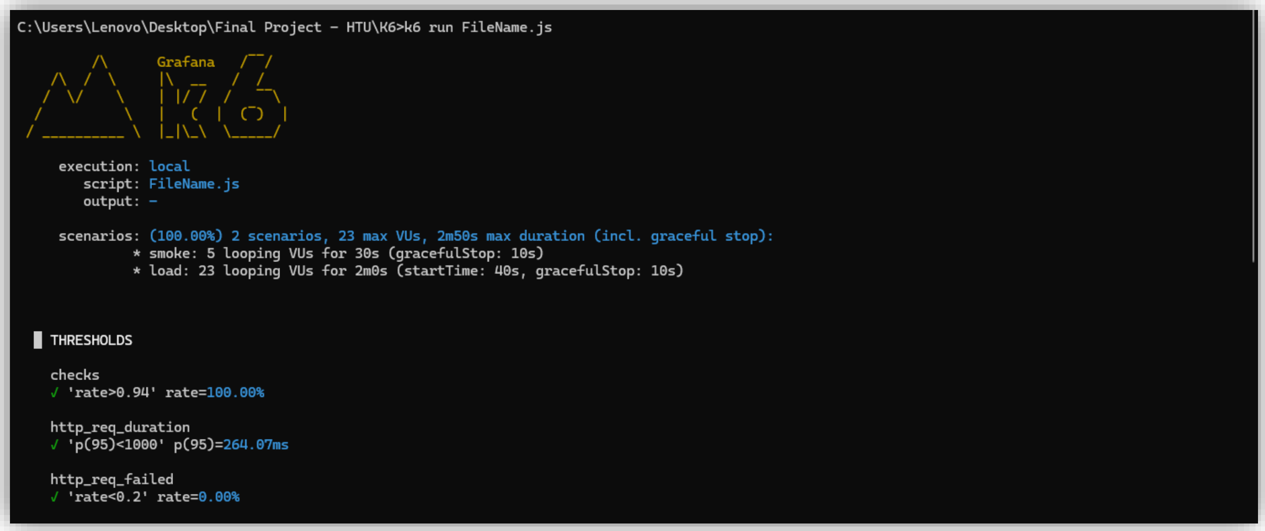
Table 4: k6 Load Profile Configuration

Scenario	**Virtual Users**	**Duration**
Smoke	5	30 sec
Load	23	2 min

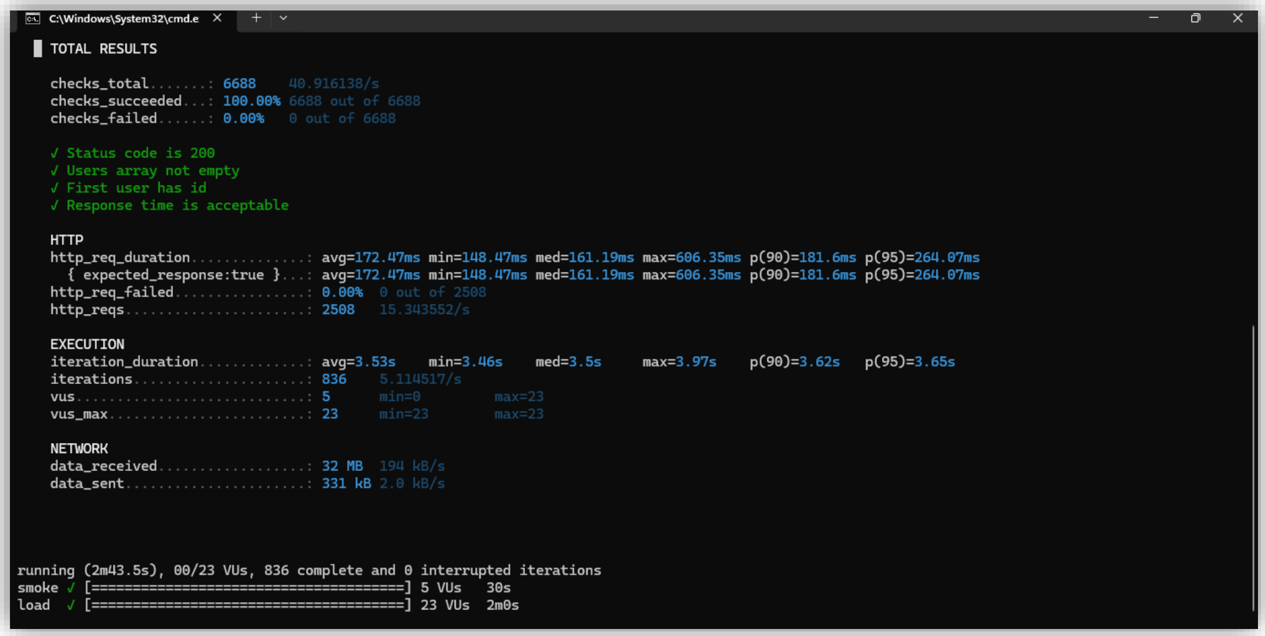
7.3 Key KPIs

Table 5: k6 Performance Key Metrics

Metric	**Value**
Average Response Time	172 ms
p95 Response Time	264 ms
Error Rate	0%
Max VUs	23



```
C:\Users\Lenovo\Desktop\Final Project - HTU\K6>k6 run FileName.js

         /\      Grafana   /‾‾/
    /\  /  \     |\  __   /  /
   /  \/    \    | |/ /  /   ‾‾\
  /          \   |   (  |  (‾)  |
 / __________ \  |_|\_\  \_____/

     execution: local
        script: FileName.js
        output: -

     scenarios: (100.00%) 2 scenarios, 23 max VUs, 2m50s max duration (incl. graceful stop):
              * smoke: 5 looping VUs for 30s (gracefulStop: 10s)
              * load: 23 looping VUs for 2m0s (startTime: 40s, gracefulStop: 10s)



  █ THRESHOLDS

    checks
    ✓ 'rate>0.94' rate=100.00%

    http_req_duration
    ✓ 'p(95)<1000' p(95)=264.07ms

    http_req_failed
    ✓ 'rate<0.2' rate=0.00%
```



```
C:\Windows\System32\cmd.e

  █ TOTAL RESULTS

    checks_total.......: 6688    40.916138/s
    checks_succeeded...: 100.00% 6688 out of 6688
    checks_failed......: 0.00%   0 out of 6688

    ✓ Status code is 200
    ✓ Users array not empty
    ✓ First user has id
    ✓ Response time is acceptable

    HTTP
    http_req_duration..............: avg=172.47ms min=148.47ms med=161.19ms max=606.35ms p(90)=181.6ms p(95)=264.07ms
      { expected_response:true }...: avg=172.47ms min=148.47ms med=161.19ms max=606.35ms p(90)=181.6ms p(95)=264.07ms
    http_req_failed................: 0.00%  0 out of 2508
    http_reqs......................: 2508   15.343552/s

    EXECUTION
    iteration_duration.............: avg=3.53s    min=3.46s    med=3.5s     max=3.97s    p(90)=3.62s   p(95)=3.65s
    iterations.....................: 836    5.114517/s
    vus............................: 5      min=0         max=23
    vus_max........................: 23     min=23        max=23

    NETWORK
    data_received..................: 32 MB  194 kB/s
    data_sent......................: 331 kB 2.0 kB/s




running (2m43.5s), 00/23 VUs, 836 complete and 0 interrupted iterations
smoke ✓ [======================================] 5 VUs   30s
load  ✓ [======================================] 23 VUs  2m0s
```

[Figure E1-E2: k6 Test Execution Output, Load Profiles, and Performance Metrics]

7.4 Analysis

All thresholds were met successfully. The API handled concurrent requests efficiently with no failed requests and consistent response times. No bottlenecks were observed under the tested load.

8. UI AUTOMATION: APPROACH, STRUCTURE, EXECUTION, AND RESULTS

Reference to Evidence: UI automation project structure and execution results are provided in Appendix F.

8.1 Automation Approach

UI automation was implemented using Selenium WebDriver (Java) following best practices.

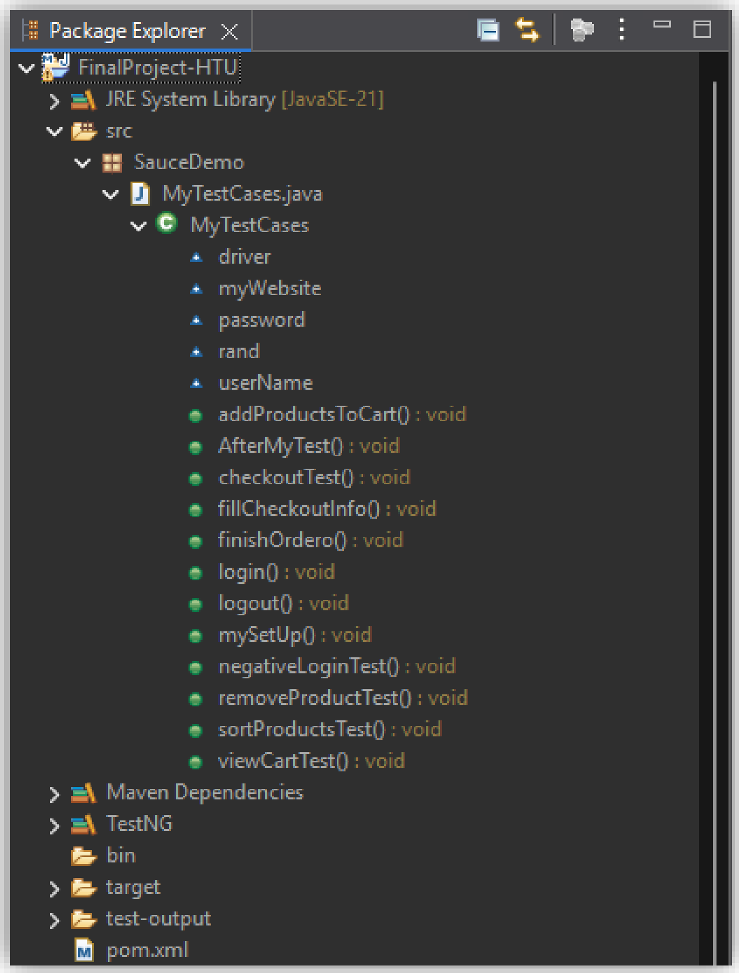
8.2 Project Structure

- Page Object Model (POM) design
- Separation of test logic and page locators
- Reusable utilities and configuration files

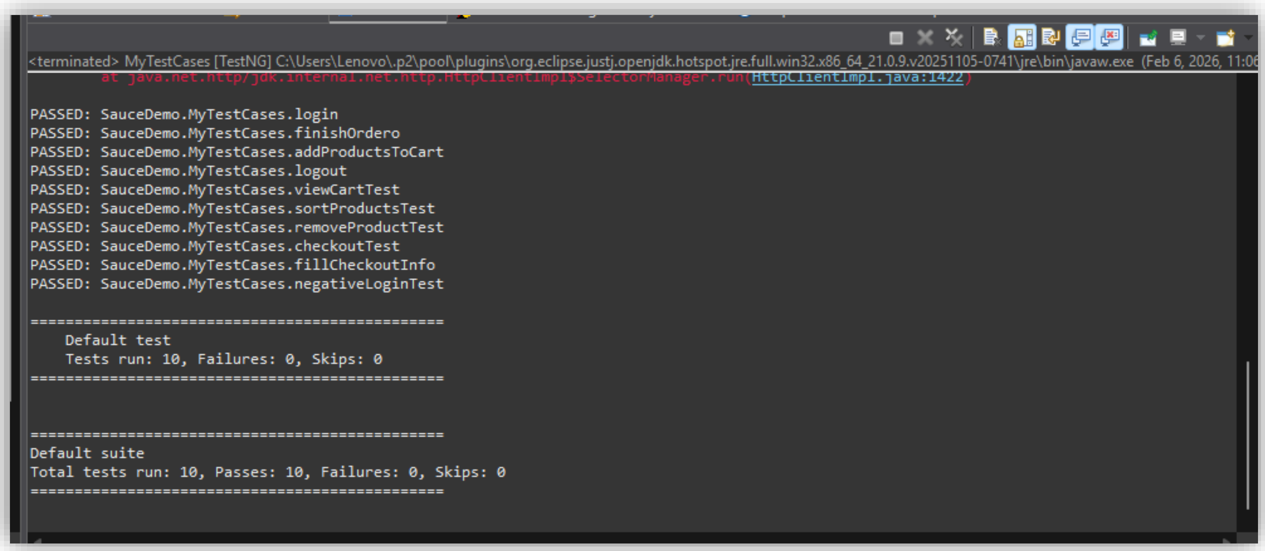
8.3 Automated Scenarios

Table 6: Automated UI Test Coverage

Scenario	Automated
Login	Yes
Product Browse	Yes
Add to Cart	Yes
Checkout	Yes
Logout	Yes



```
Package Explorer
v FinalProject-HTU
  > JRE System Library [JavaSE-21]
  v src
    v SauceDemo
      v MyTestCases.java
        v MyTestCases
            driver
            myWebsite
            password
            rand
            userName
            addProductsToCart() : void
            AfterMyTest() : void
            checkoutTest() : void
            fillCheckoutInfo() : void
            finishOrdero() : void
            login() : void
            logout() : void
            mySetUp() : void
            negativeLoginTest() : void
            removeProductTest() : void
            sortProductsTest() : void
            viewCartTest() : void
  > Maven Dependencies
  > TestNG
    bin
  > target
  > test-output
    pom.xml
```



```
<terminated> MyTestCases [TestNG] C:\Users\Lenovo\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.9.v20251105-0741\jre\bin\javaw.exe (Feb 6, 2026, 11:06
	at java.net.http/jdk.internal.net.http.HttpClientImpl$SelectorManager.run(HttpClientImpl.java:1422)

PASSED: SauceDemo.MyTestCases.login
PASSED: SauceDemo.MyTestCases.finishOrdero
PASSED: SauceDemo.MyTestCases.addProductsToCart
PASSED: SauceDemo.MyTestCases.logout
PASSED: SauceDemo.MyTestCases.viewCartTest
PASSED: SauceDemo.MyTestCases.sortProductsTest
PASSED: SauceDemo.MyTestCases.removeProductTest
PASSED: SauceDemo.MyTestCases.checkoutTest
PASSED: SauceDemo.MyTestCases.fillCheckoutInfo
PASSED: SauceDemo.MyTestCases.negativeLoginTest

===============================================
    Default test
    Tests run: 10, Failures: 0, Skips: 0
===============================================


===============================================
Default suite
Total tests run: 10, Passes: 10, Failures: 0, Skips: 0
===============================================
```

[Figure F1-F2: UI Automation Project Structure, Page Objects, Tests, and Results]

8.4 Execution and Results

Tests can be executed locally via the provided configuration. The UI automation suite can be run by executing the test runner after setting up the required environment and dependencies as described in the project README. Sample execution results show successful completion of all automated scenarios, demonstrating regression stability.

9. RISKS, LIMITATIONS, AND RECOMMENDATIONS

9.1 Risks and Limitations

- Demo environments may change without notice
- API write operations are simulated and not persistent
- Limited load levels due to environment constraints

9.2 Recommendations

- Add CI integration (GitHub Actions) for automated runs
- Expand negative and boundary test coverage
- Include cross-browser and mobile testing in future iterations
- Implement comprehensive test data management framework
- Add performance baselines for continuous monitoring

10. APPENDICES

Appendix A – Manual Testing Execution Evidence

Figures A1-A3: Manual test execution screenshots showing:

- Figure A1: Passed test case execution (Login – valid credentials)
- Figure A2: Failed test case execution (Checkout validation)
- Figure A3: Negative test scenario execution (Invalid login)

Appendix B – Defect Evidence Screenshots

Figures B1-B3: Sample defect evidence. The complete defect log containing all 12 reported issues is provided in the defect tracking sheet.

- Figure B1: BUG-002 – Product images are incorrect for problem user
- Figure B2: BUG-009 – Products are not sorted correctly for visual user
- Figure B3: BUG-006 – Remove button does not work directly on Products page for error user

Appendix C – Postman Collection Evidence

Figures C1-C3: Postman collection structure and configuration:

- Figure C1: Postman collection folder structure (Auth, Products, Carts, Users)
- Figure C2: Sample API request with automated test assertions

Appendix D – Newman Execution Report

Figures D1: Newman execution results:

- Figure D1: Newman HTML report summary (total requests, assertions, failures)

Appendix E – k6 Performance Testing Evidence

Figures E1-E2: k6 performance test execution results:

- Figure E1: k6 test execution output (smoke and load scenarios)
- Figure E2: Performance KPI summary and threshold evaluation

Appendix F – UI Automation Evidence

Figures F1-F2: UI automation project structure and results:

- Figure F1: UI automation project structure inspired by the Page Object Model (POM)
- Figure F2: UI automation execution results (passed test scenarios)

All screenshots, logs, and reports referenced above are included in the submission folders according to the required assignment structure.