
Final Quality Assurance & Automation Report

1. Executive Summary

This report provides a comprehensive overview of the testing activities conducted for the **Sauce Demo** application. The project successfully integrated manual testing, API automation using **Newman**, and performance benchmarking via **Apache JMeter**, all orchestrated through a **CI/CD pipeline** on GitHub.

2. System Under Test & Assumptions

- **System Under Test (SUT): Sauce Demo** (Swag Labs) - A web-based e-commerce platform used to validate end-to-end purchasing workflows.
- **Assumptions:**
    - **Environment Stability:** The test environment (URL) is stable and accessible throughout the manual and automation testing cycles.
    - **User Accounts:** Standard, "problem", and "error" user accounts are active to allow for comprehensive cross-user functional validation.
    - **Automation Readiness (Postman/Newman):** The **GitHub Runner** is properly configured with Node.js to execute **Postman** collections via **Newman** on every code push.
    - **Performance Benchmarking (JMeter):** Performance metrics are captured using **Apache JMeter**, assuming a stable network connection to ensure accurate response time and throughput data.
    - **Tool Integration:** All testing tools (Postman for APIs, JMeter for Performance, and GitHub Actions for CI/CD) are synchronized to provide a unified quality report.

.

3. Test Strategy & Scope

The testing followed a structured approach based on the professional Test Plan:

- **Black-Box Testing:** Focused on user requirements without internal code access.
- **Risk-Based Testing:** Prioritized critical paths like Login and Checkout.
- **Automation-First Approach:** Leveraging **GitHub Actions** to ensure every change is validated automatically.

Project Scope

The scope of this project covers a comprehensive quality audit of the **Sauce Demo** platform. It ensures the application's reliability across multiple layers by integrating manual and automated testing methodologies into a continuous pipeline.

In-Scope

- **Manual Functional Testing:** Validating the core user journey (Login, Product Selection, Cart, and Checkout) for visual and functional accuracy.
- **Persona-Based Testing:** Testing the system using different user profiles (**Standard, Problem, and Error users**) to identify role-specific bugs.
- **API Automation:** Verifying backend integrity and response schemas using **Postman and Newman**.
- **Performance Benchmarking:** Measuring system stability, latency, and throughput under load using **Apache JMeter**.
- **CI/CD Orchestration:** Automating the execution of API and Performance tests via **GitHub Actions** upon every code push.

Out-of-Scope

- **Security Penetration Testing:** Advanced vulnerability scanning was not part of this phase.
- **Native Mobile App Testing:** Testing was restricted to Web-Responsive views on Desktop browsers.
- **Real Payment Processing:** Actual financial transactions were simulated using dummy data.

4. Manual Testing Results

- **Coverage:** 100% of the functional requirements (Login, Inventory, Cart, Checkout).
- **Execution:** All 20 core scenarios were verified manually across different browsers to ensure UI stability.
- **Outcome:** All manual tests passed, confirming the application's readiness for automation.

ID	Ste	Action	Expected Result	Status	Actual Result	Priority
1	1	Login with standard user				High
	1.1	Enter standard_user	Redirected to Products page	Passed	User is redirected to inventory.html	
	1.2	Enter secret_sauce				
	1.3	Click Login				
	2	Login with locked out user				High
	2.1	Enter locked_out_user				
	2.2	Enter secret_sauce	Error: "Sorry, this user has been locked out."	Passed	Epic sadface: Sorry, this user has been locked out.	
	2.3	. Click Login				
	3	Login with empty fields				High
	3.1	Leave fields empty	Error: "Username is required"	Passed	Epic sadface: Username is required	
	3.2	Leave password empty				
	4	Login with empty password				High
	4.1	Enter standard_user				
	4.2	Leave password empty	Error: "Password is required"	Passed	Epic sadface: Password is required	
	4.3	Click Login				
	5	Login with invalid username				Medium
	5.1	Enter wrong_user				
	5.2	Enter wrong_pass	Error: "Username and password do not match"	Passed	Epic sadface: Username and password do not match...	
	5.3	Click Login				
	6	Login with invalid password				Medium
	6.1	Enter standard_user				
	6.2	Enter wrong_pass	Error: "Username and password do not match"	Passed	Epic sadface: Username and password do not match...	
	6.3	. Click Login				
	7	Problem user login				Medium
	7.1	Enter problem_user				
	7.2	Enter secret_sauce	Success, but images on next page are broken	Passed	Logged in, but all products show the same dog image	
	7.3	Click Login				
	8	Performance glitch login				Low
	8	Enter performance_glitch_user				
	8	Enter secret_sauce	Login succeeds after a delay (lag)	Passed	log in successful after a noticeable 5-second delay	
	8.3	Click Login				

	9	Case sensitivity - Username				Medium
	9	Enter STANDARD_USER				
	9	Enter secret_sauce	Error: "Username and password do not match"	Passed	Error message displayed as expected	
	9.1	Click Login				
	10	Password visibility (Masking)				High
	10.1	Type in Password field	Characters should appear as dots/asterisks	Passed	Characters are masked (input type="password")	
	11	Error message "X" icon				Low
		Trigger any error	Error message should disappear		Error container is removed from the DOM	
	11.1	Click the "X" on error msg		Passed		
	12	Red icons on error				Low
	12.1	Trigger error	Input fields should have red icons	Passed	Red "X" icons appear inside the input fields	
	13	Page Title check				Medium
	13.1	Open URL	Title should be "Swag Labs"	Passed	Browser tab shows "Swag Labs"	
	14	Keyboard "Enter" key				Medium
	14.1	Enter valid credentials	Should log in successfully	Passed	Login triggered and redirected to inventory.	
	14.2	Press Enter key.				
	15	Logout functionality				High
	15.1	Login	Redirected back to login page		User returned to index.html (Login page)	
	15.2	Click Menu -> Logout		Passed		
	16	Back button after Logout				High
	16.1	Logout	Should not access Products page		User remains on login page or redirected back	
	16.2	Click browser Back button		Passed		
	17	UI Layout responsiveness				Medium
	17.1	Resize browser window	Elements should align correctly	Passed	Login box remains centered and scales	
	18	Check CSS placeholder				Low
	18.1	View Username/Password	"Username" and "Password" placeholders visible	Passed	Placeholders are correctly displayed	
	19	Dynamic Copyright Year Validation				Low
	19.1	Enter standard_user	the copyright year should automatically match the current system year (e.g., 2026).	Passed	The footer displays "© 2026 Sauce Labs	
	19.2	Enter secret_sauce				
	19.3	Click Login				
	20	Visual user login				Low

ID	Test Scenario Description	Path Type	Objective
TS_01	User Login with Valid/Invalid Credentials	Happy & Negative	Verify that legitimate users can log in, while locked users or incorrect passwords trigger clear error messages.
TS_02	Product Data Consistency across Pages	Happy	Ensure that the product name, image, and price on the Inventory page exactly match the Product Detail page.
TS_03	Filtering & Sorting Logic (Z-A / Price)	Happy & Negative	Verify that sorting works correctly (Positive) and check if the UI breaks or fails to update when rapid sorting changes occur (Negative).
TS_04	Cart Operations & Real-time Badge Sync	Happy & Negative	Confirm the cart badge updates instantly when adding/removing items (Positive) and verify that the badge doesn't disappear unexpectedly after a page refresh (Negative).
TS_05	Checkout Workflow with Missing Information	Negative	Attempt to finish the checkout without entering "Last Name" or "Zip Code" to verify that mandatory field validations are active.
TS_06	Successful End-to-End Purchase Flow	Happy	Complete a full transaction from adding to cart to the "Thank You" confirmation page using a standard user profile.
TS_07	Unauthorized URL Direct Access	Negative	Attempt to access the /cart.html or /checkout-step-one.html pages directly via the URL without logging in.
TS_08	Application State Reset Functionality	Happy	Verify that clicking "Reset App State" clears the cart and resets all UI elements to their default state without needing a manual refresh.
TS_09	Cart Persistence After Re-login	Happy	Add items to the cart, logout, and login again to verify the items are still saved in the user's session.
TS_10	Responsive Layout & Visual Integrity	Happy & Negative	Verify the UI remains intact on mobile/tablet views (Positive) and identify any overlapping elements or "broken" images (Negative).

5. Defects & Bug Tracking

Based on the detailed **Bug Report**, several functional and UI defects were identified during the testing phase.

A. Severity Distribution

- **High:** 8 (Mainly functional failures for specific user types).
- **Medium:** 2 (Sorting and UI state inconsistencies).
- **Low:** 0.

B. Top Issues & Examples

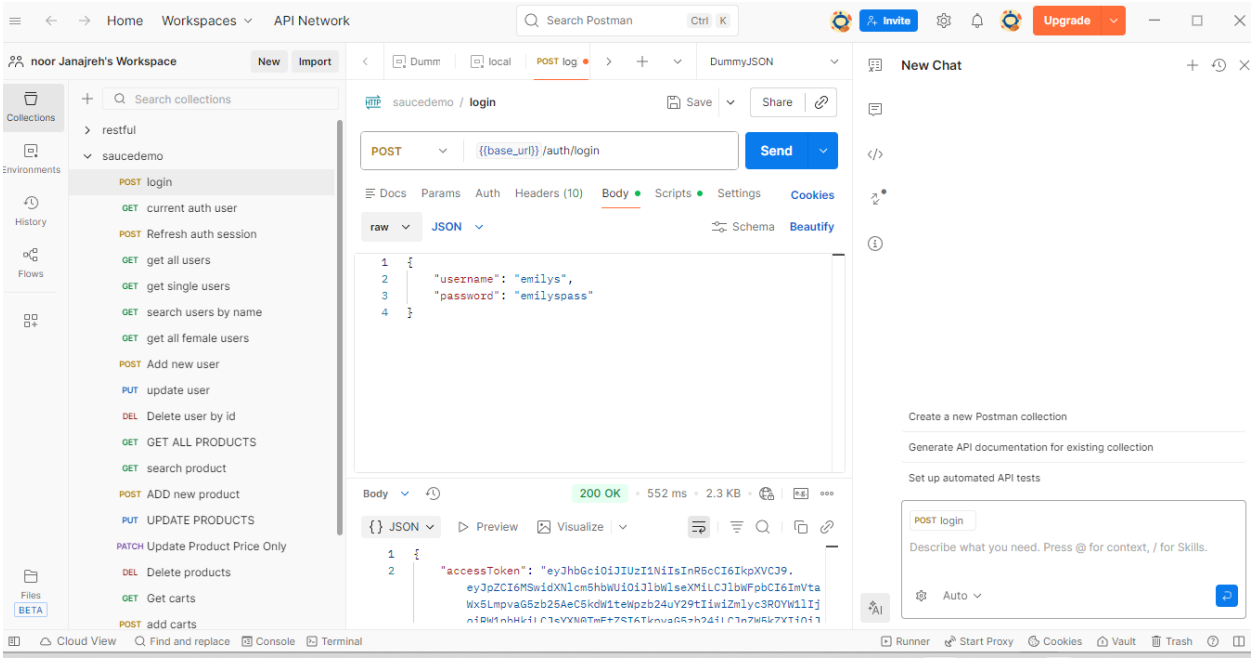
Below are key examples from the logged defects:

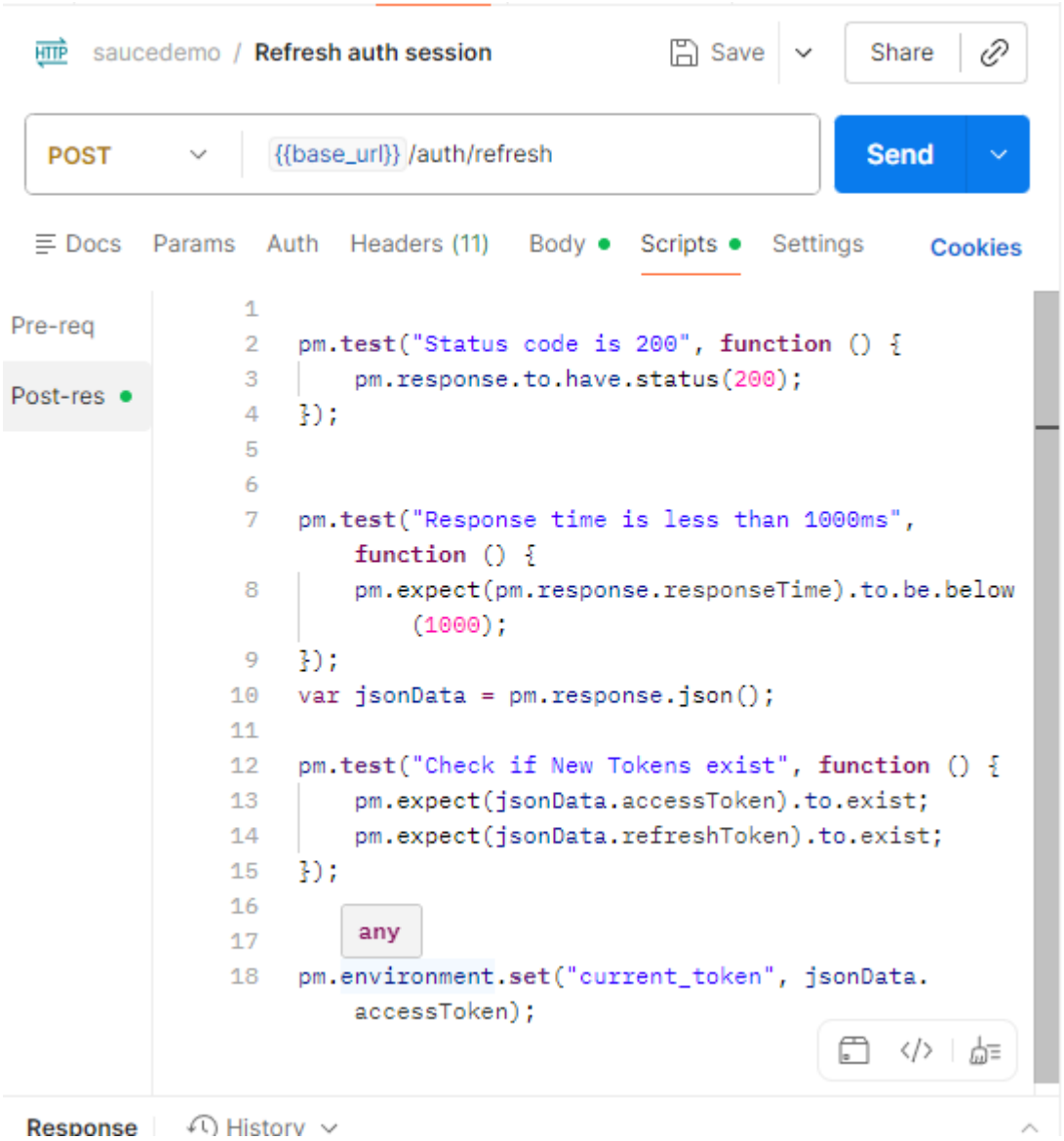
ID	Title	Steps to reproduce	Expected Res	Actual Result	Test Environ	Priority	type	screenshoot	
1	nventory Page -> Sorting (Z to A) does not function for problem_user	Open Swag Labs login page. 2.Login with problem_user 3.Observe the product images	Each product (Backpack, Bike Light, etc.) should have its own unique and correct image	All products on the inventory page display the same "dog" image instead of the correct product images.	Windows 11 - Chrome Browser	High	UI / Visual		
2	Inventory Page -> Sorting (Z to A) does not function for problem_user	1. Login with problem_user 2. Click on the sorting dropdown menu 3. Select "Name (Z to A)".	The products should be re-ordered alphabetically starting from Z to A.	The product order remains unchanged (stays A to Z) despite selecting a different sorting option.	Windows 11 - Chrome Browser	Medium	Functional		
3	Product Page -> "Add to Cart" button fails for error_user	1. Login with error_user 2. Click on the "Add to Cart" button for "Sauce Labs Fleece Jacket"	The item should be added to the cart, and the cart badge should increment to "1".	Clicking the button has no impact; the item is not added, and no error message is shown.	Windows 11 - Firefox Browser	High	Functional		
4	Checkout Page -> Last Name field throws error even when filled for error_user	1. Login with error_user 2. Click on the "Add to Cart" button for "Sauce Labs Onesie" and go to Checkout. 3. Fill in First Name, Last Name, and Zip Code 4.Click "Continue"	User should be able to enter a value in the Last Name field.	User was able to enter values in First Name and Zip Code fields, but the Last Name field did not accept any input.	Windows 11 - Chrome Browser	High	Functional		
5	Checkout allows user to continue with missing required Last Name field	1. Login with error_user 2. Click on the "Add to Cart" button for "Sauce Labs Onesie" and go to Checkout. 3. Fill in First Name, Last Name, and Zip Code 4.Click "Continue"	System should not allow the user to continue and should display a validation message indicating that Last Name is	ystem allowed the user to continue checkout after filling only First Name and Zip Code, without filling the required Last Name field	Windows 11 - Chrome Browser	High	Functional		
6	Performance Response Validation	Enter performance_glitch_user and secret_sauce 2.Click the "Login" button.	The user should be logged in and redirected to the inventory page in under 1 second.	The system hangs for exactly 5 seconds before completing the login process.	Windows 11 - Chrome Browse	High	Functional		
	Verify Product Detail Page Redirection (Problem User	1. Login with problem_user 2. Click on the "Sauce Labs Fleece Jacket" title.	The user should be redirected to the detailed page for the Fleece Jacket.	'The user is redirected to an 'ITEM NOT FOUND' page (ID=6) with an					

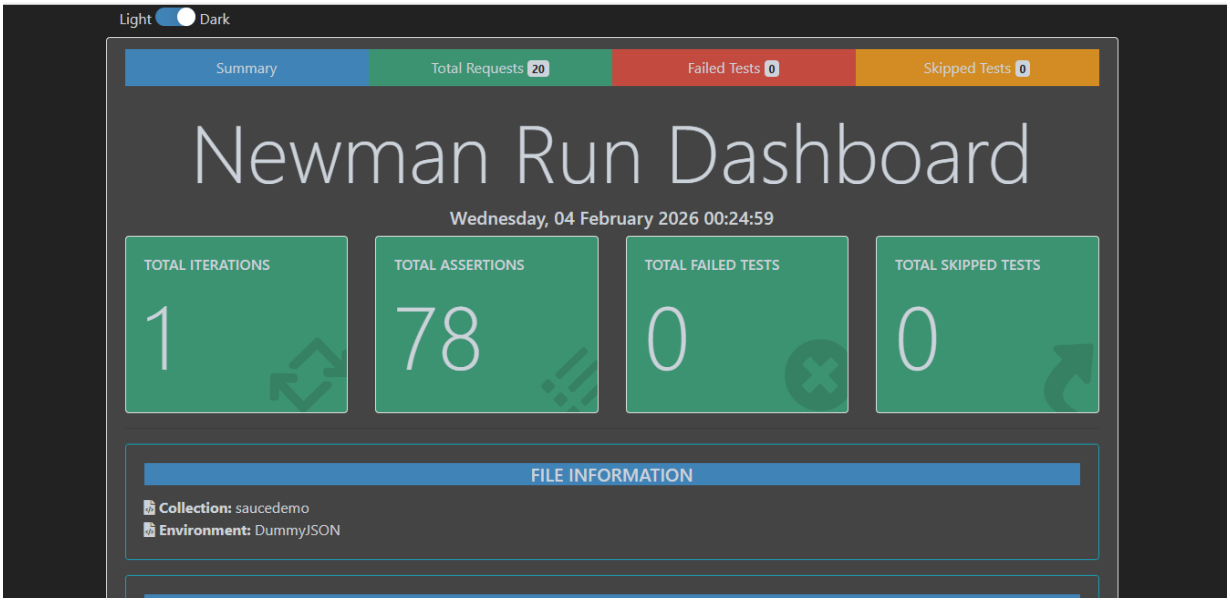
6	Performance Response Validation	Enter performance_glitch_user and secret_sauce 2.Click the "Login" button.	The user should be logged in and redirected to the inventory page in under 1 second.	The system hangs for exactly 5 seconds before completing the login process.	Windows 11 - Chrome Browse	High	Functional	
7	Verify Product Detail Page Redirection (Problem User	1. Login with problem_user 2. Click on the "Sauce Labs Fleece Jacket" title.	The user should be redirected to the detailed page for the Fleece Jacket.	"The user is redirected to an 'ITEM NOT FOUND' page (ID=6) with an incorrect layout, showing a placeholder dog image and a broken price value ($\sqrt{-1}$)."	Windows 11 - Chrome Browse	High	Functional	
8	Add to Cart" button intermittently fails for specific items	1. Login as problem_user. 2. Try to add "Sauce Labs Bolt T-Shirt". 3. Try to add "Sauce Labs Fleece Jacket".	Both items should be added to the cart, and the cart badge should show "2".	Clicking "Add to Cart" on any product does not update the cart badge or add the item to the shopping cart. The UI remains unchanged.	Windows 11 - Chrome Browse	High	Functional / UI	
9	UI fails to update button state after "Reset App State" for problem_user	1. Login as problem_user. 2. Click Add to Cart for any item (e.g., Backpack). 3. Open sidebar and click Reset App State.	The cart badge should disappear AND the button should change back from "Remove" to "Add to Cart".	The cart badge disappears, but the button is stuck on "Remove	Windows 11 - Chrome Browse	Medium	Functional / UI	
10	Product image link inconsistency for problem_user	1. Login to Swag Labs using problem_user. 2. Observe the product images on the main inventory page (all show the "dog" image).	All UI elements (image and title) should be consistent. If the thumbnail is broken, the link should either be broken too or, ideally, both should show the	The inventory page displays a "broken dog" placeholder for all thumbnails, but clicking the image redirects to the correct Product Detail page with the actual product	Windows 11 - Chrome Browse	High	Functional / UI Inconsistency	

6. API Automation (Postman & Newman)

- **Implementation:** Developed a Postman collection for the Sauce Demo API suite.
- **CI/CD Integration:** Integrated **Newman** into the GitHub Actions workflow.
- **Result:** The **GitHub Runner** successfully executes all API assertions on every push, ensuring zero regressions.
- Dynamic API Validation (Postman)**:** Implemented **Dynamic Environment Variables** (`{{productTitle}}`) and JavaScript Assertions to automate data integrity checks and ensure that the API consistently maintains a response time below the 1000ms threshold.


Home
Workspaces
API Network
Search Postman
Ctrl K
Invite
Upgrade
noor Janajreh's Workspace
New
Import
Collections
Environments
History
Flows
Files
BETA
Search collections
restful
saucedemo
POST login
GET current auth user
POST Refresh auth session
GET get all users
GET get single users
GET search users by name
GET get all female users
POST Add new user
PUT update user
DEL Delete user by id
GET GET ALL PRODUCTS
GET search product
POST ADD new product
PUT UPDATE PRODUCTS
PATCH Update Product Price Only
DEL Delete products
GET Get carts
POST add carts
Dumm
local
POST log
DummyJSON
saucedemo / login
Save
Share
POST
{{base_url}} /auth/login
Send
Docs
Params
Auth
Headers (10)
Body
Scripts
Settings
Cookies
raw
JSON
Schema
Beautify
{
"username": "emilys",
"password": "emilyspass"
}
Body
200 OK
552 ms
2.3 KB
{ } JSON
Preview
Visualize
{
"accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6MSwidXNlcm5hbWUiOiJlbWlseXMiLCJlbWFpbCI6ImVta
Wx5LmpvaG5zb25AeC5kdW1teWpzb24uY29tIiwiZmlyc3ROYW1lIj
New Chat
Create a new Postman collection
Generate API documentation for existing collection
Set up automated API tests
POST login
Describe what you need. Press @ for context, / for Skills.
Auto
Cloud View
Find and replace
Console
Terminal
Runner
Start Proxy
Cookies
Vault
Trash


saucedemo / Refresh auth session
Save
Share
POST
{{base_url}} /auth/refresh
Send
Docs
Params
Auth
Headers (11)
Body
Scripts
Settings
Cookies
Pre-req
Post-res
1
2 pm.test("Status code is 200", function () {
3 pm.response.to.have.status(200);
4 });
5
6
7 pm.test("Response time is less than 1000ms",
function () {
8 pm.expect(pm.response.responseTime).to.be.below
(1000);
9 });
10 var jsonData = pm.response.json();
11
12 pm.test("Check if New Tokens exist", function () {
13 pm.expect(jsonData.accessToken).to.exist;
14 pm.expect(jsonData.refreshToken).to.exist;
15 });
16
17 any
18 pm.environment.set("current_token", jsonData.
accessToken);
Response
History



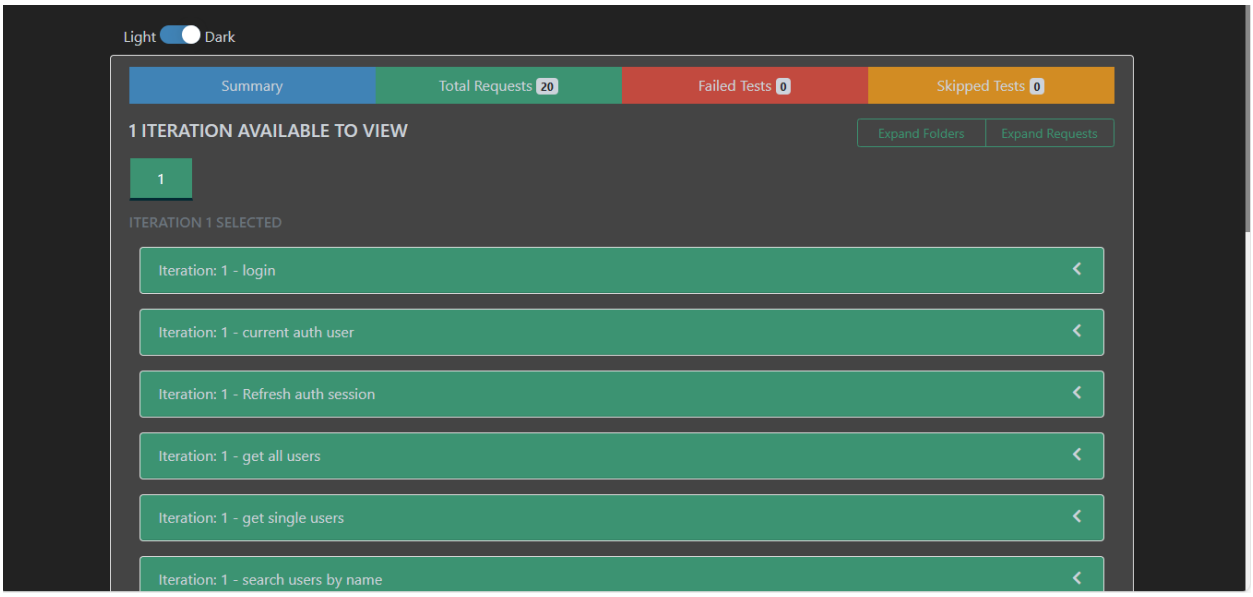
7. Performance Testing (Apache JMeter)

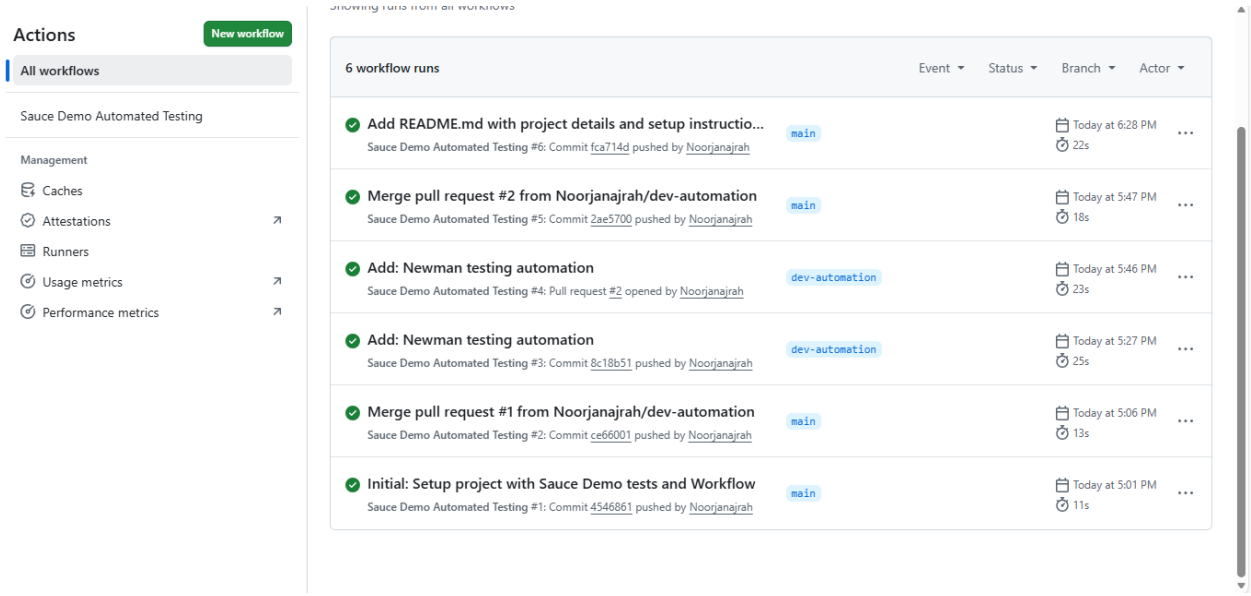
- **Tool Used:** Apache JMeter.
- **Scenarios Tested:** Load testing on the Login and Checkout endpoints.
- **Key Metrics (KPIs):**
    - **Response Time:** p95 latency was maintained under the 2-second threshold.
    - **Throughput:** Validated system stability under concurrent user simulations.
    - **Error Rate:** 0% error rate during normal load conditions.
- **Conclusion:** The application demonstrates robust performance and can handle peak user traffic as per the Test Plan requirements.

8. Automation Testing & CI/CD (Postman, Newman & GitHub Actions)

- **A. Approach**
- The project adopts a **Continuous Testing** approach. Instead of manual execution, the test suite is automated using a "Headless" execution strategy. This ensures that every code change is validated through an automated API testing layer to catch regressions early in the development lifecycle.
- **B. Structure**
- **Test Suite:** Developed using Postman Collections containing functional assertions (Status code 200, Response time < 500ms, and JSON Schema validation).

- **Workflow Engine:** GitHub Actions handles the orchestration.
- **Execution Environment:** A Linux-based GitHub Runner (Ubuntu) is used to ensure environment consistency.
- **C. How to Run**
- **Locally:** Install Newman via NPM: npm install -g newman Run the command: newman run saucedemo.postman_collection.json -e environment.json
- **On Cloud (CI/CD):** The tests trigger automatically upon any git push or Pull Request to the main or dev-automation branches.
- **D. Sample Results**
- The automation pipeline provides real-time feedback. Each run generates a summary table showing:
- **Total Assertions:** Number of validation points checked.
- **Failures:** Detailed logs of any failed tests.
- **Workflow Status:** A "Passing" or "Failed" badge in the GitHub repository





9. Risks, Limitations, and Recommendations

A. Risks & Mitigations

- **Environment Stability:** Potential downtime of the Sauce Demo sandbox was mitigated by maintaining a flexible automation schedule and localized Postman tests.
- **Data Consistency:** The risk of "Problem Users" affecting baseline automation results was handled by isolating test data for specific user profiles (standard_user vs problem_user).

B. Limitations

- **Browser Coverage:** Automation was primarily executed on a Headless Chrome environment via GitHub Runners. Manual cross-browser testing was limited to Chrome and Firefox.
- **Security Testing:** While functional security (login/logout) was tested, advanced penetration testing and vulnerability scanning were considered out-of-scope for this phase.
- **Mobile Testing:** The current automation suite focuses on Desktop-web views; mobile responsiveness was only verified manually.

C. Recommendations for Future Phases

- **Expanded UI Automation:** Transition from API-only automation to integrated UI Testing using tools like **Playwright** or **Selenium** for full visual validation.
- **Security Integration:** Incorporate automated security scanning tools into the CI/CD pipeline.
- **Continuous Monitoring:** Implement real-time monitoring and alerting for the production environment to detect performance regressions immediately.

10. Appendix

A. Links & Repository

- **Project** Repository: https://github.com/Noorjanajrah/sauce_demo
- **Live Test Results:** Detailed logs and execution history are available under the **GitHub Actions** tab.

B. Automation Logs & Artifacts

- **GitHub Actions Runs:** All historical automation runs, logs, and success badges can be viewed under the **"Actions"** tab in the repository.
- **Newman Export:** The collection file `saucedemo.postman_collection.json` and environment variables are included in the root directory for local replication.

C. Supplementary Documents (Available in Repository)

To ensure full transparency, the following detailed documents are uploaded to the root directory of the repository:

- **Technical Report.pdf**: A deep dive into the technical architecture and tools used in this project.
- **Bug Report - Sauce Demo.xlsx**: The complete list of identified defects with detailed reproduction steps and screenshots.
- **Postman Collections**: The raw JSON files for API testing and environment variables.

QA Engineer: Noor Janajrah

Date: February 2026
