Project Title: Technical Engineering & Automation Framework for Sauce Demo

Domain: E-commerce Full-Stack Quality Assurance

Prepared by: Noor Janajrah

Technical Stack: Postman/Newman, Apache JMeter, GitHub Actions, DummyJSON API

Date: February 2026

Status: Technical Report

1.Introduction

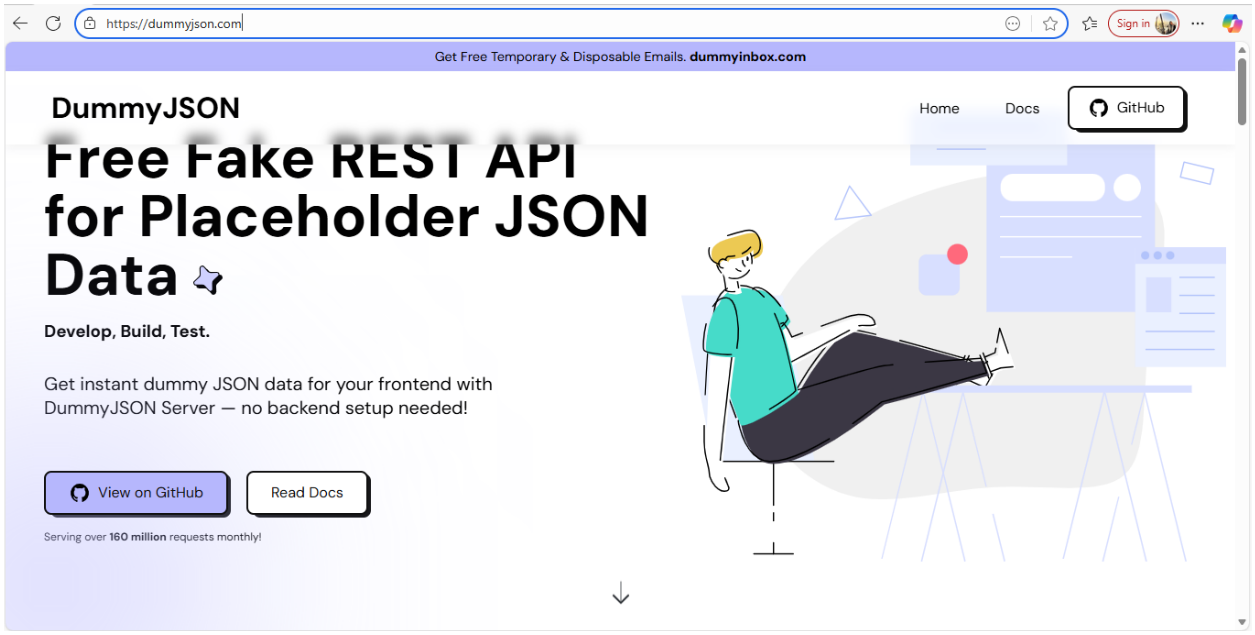
This technical report details the architectural design and implementation of a robust QA ecosystem for the Sauce Demo platform. The project demonstrates a transition from manual exploratory testing to a sophisticated Continuous Testing environment. By integrating backend API validation (via DummyJSON) and performance benchmarking into a CI/CD pipeline, this framework ensures that the application remains stable, scalable, and resilient under various stress conditions and data states.

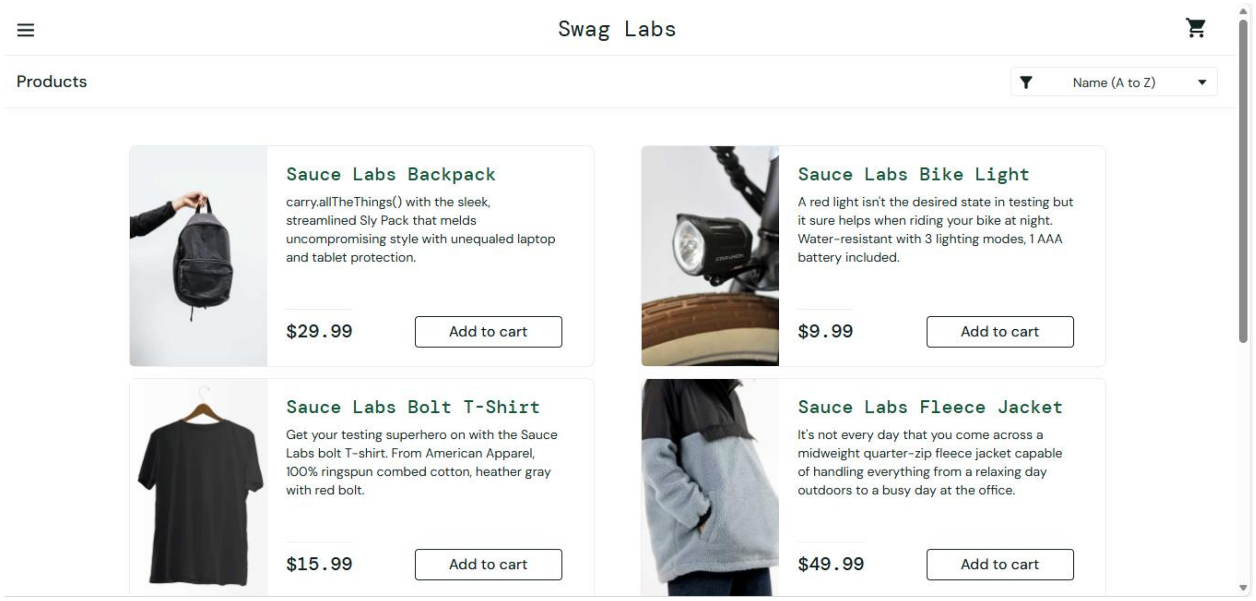
2.Project Overview and Tool Selection

2.1 Project Scope

The scope of this technical audit covers the end-to-end user journey of the Swag Labs application. A key technical highlight is the "Multi-Layered" validation approach:

- **Frontend Layer:** Manual UI/UX audit across multiple user personas.
- **Data & API Layer:** Automated backend testing using DummyJSON to simulate realistic product data and server responses.
- **Performance Layer:** Stress testing the infrastructure to identify latency and throughput limits.


https://dummyjson.com
Sign in
Get Free Temporary & Disposable Emails. dummyinbox.com
DummyJSON
Home
Docs
GitHub
Free Fake REST API for Placeholder JSON Data
Develop, Build, Test.
Get instant dummy JSON data for your frontend with DummyJSON Server — no backend setup needed!
View on GitHub
Read Docs
Serving over 160 million requests monthly!


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

2.2 Tool Selection Rationale

To build a modern testing infrastructure, the following tools were integrated:

- Postman & Newman: Selected for API automation. Newman (the CLI runner) was essential for executing tests within the GitHub cloud environment without a graphical interface.
- Selenium WebDriver (Java/TestNG): Implemented to automate the Frontend User Interface (UI). This ensures that the entire "Critical Path" from secure login and product selection to the final checkout is executed automatically, verifying functional flow and UI stability without human intervention.
- DummyJSON: Utilized as a Mock API service. It provided the necessary RESTful endpoints to test complex data structures, pagination, and product filtering without risking production data.
- Apache JMeter: Chosen for its multi-threading capabilities to simulate concurrent load, providing deep insights into system performance metrics.
- GitHub Actions: Serves as the DevOps backbone, orchestrating the automated execution of all test suites (Newman & JMeter) upon every repository push.
-

---

3. Test Data Preparation and Test Case Design

3.1 Test Data Engineering (DummyJSON & Mocking)

To validate the application's resilience, the testing suite integrated DummyJSON as a primary source for mock data. This approach was chosen to ensure that our tests are independent of the live production environment's data limitations.

- Mocking Product Data: We utilized https://dummyjson.com/products to simulate various product attributes (e.g., extremely long titles, high stock numbers, and discounted prices) to test how the Sauce Demo UI handles dynamic content.
- Persona-Driven Data: We mapped the standard users from Sauce Demo to specific data scenarios in DummyJSON to verify if the frontend correctly renders data associated with different user roles (Standard, Problem, and Error users).

3.2 Test Case Design Strategy

The testing architecture was divided into three distinct technical layers:

A. Manual Test Design (User Experience Layer):

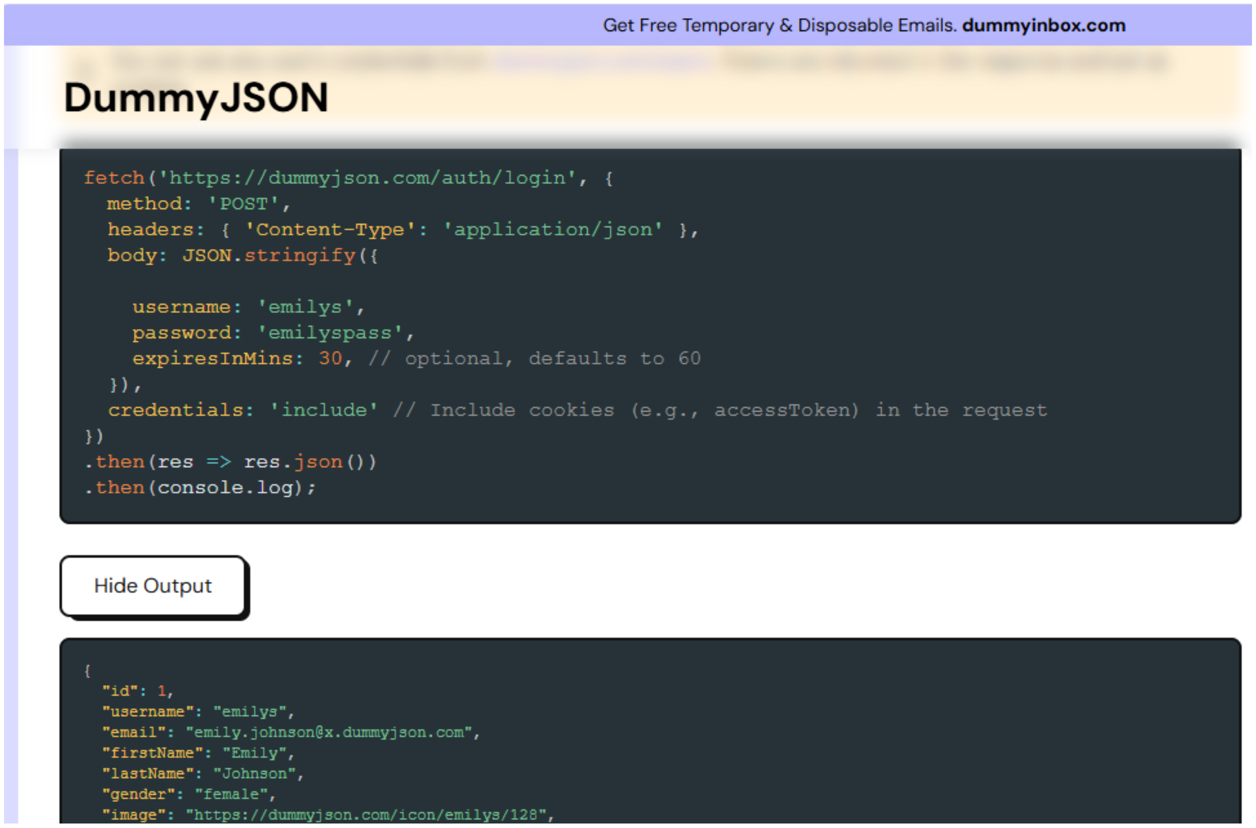
- Designed 20 manual test cases focusing on the "Critical Path" (Login → Cart → Checkout).
- UI Automation (Selenium WebDriver): "Developed robust automation scripts using Java and TestNG to validate the core web pages, including Login, Product Inventory, and the Shopping Cart. The primary focus was to ensure functional correctness and verify the visibility and integrity of UI elements across different stages of the user journey."
- Technique: Used Equivalence Partitioning to test input fields (e.g., verifying that the zip code field in the checkout page accepts only specific formats).

B. API Automation Scripts (Backend Integrity Layer):

- Tool: Postman/Newman.
- **Validation Logic: JavaScript-based scripts were embedded within the Postman collection to perform Automated Assertions:**
    - **pm.response.to.have.status(200): To ensure endpoint availability.**
    - **pm.expect(pm.response.json().products).to.be.an('array'): To verify data structure integrity.**
    - Latency Check: Ensuring that the DummyJSON response time is consistently below 500ms to maintain a smooth user experience.

C. Performance Load Profiles (Stability Layer):

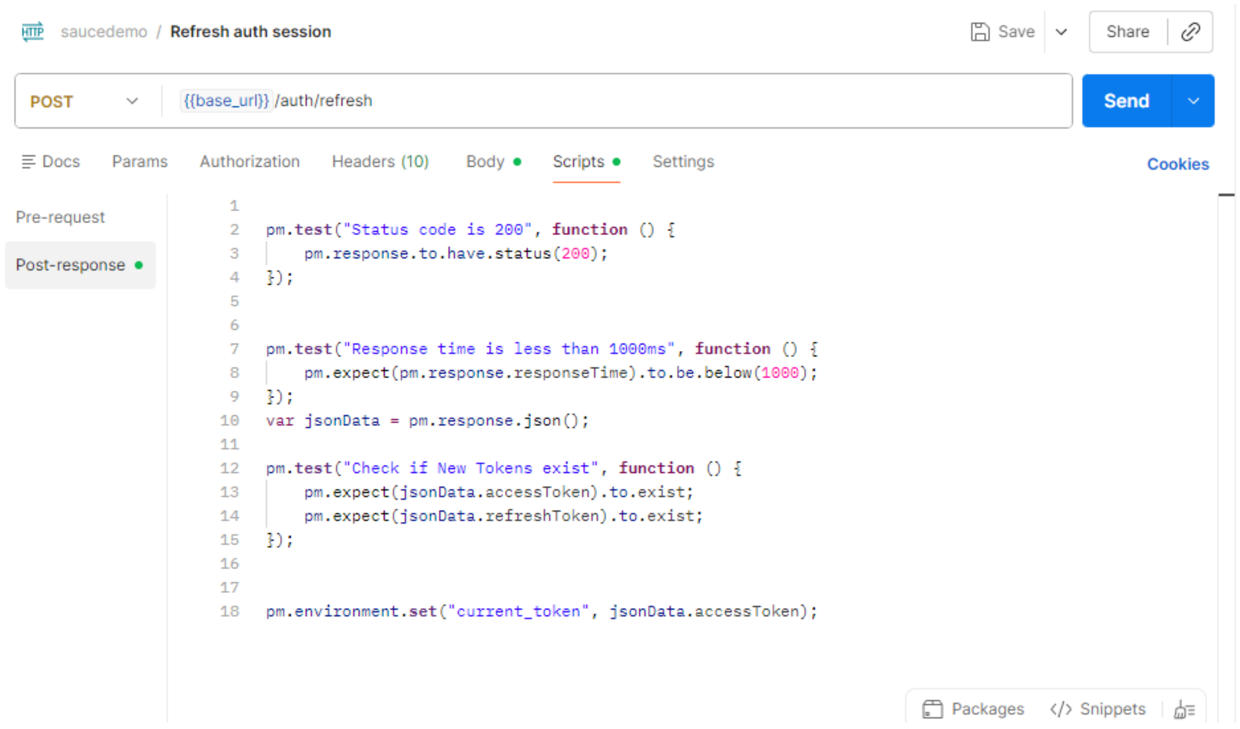
- Tool: Apache JMeter.
- Scenario: Designed a Thread Group targeting DummyJSON endpoints to simulate concurrent API calls. This validates how the system's "Data Layer" behaves when multiple users are fetching products at the same time.


Get Free Temporary & Disposable Emails. **dummyinbox.com**

DummyJSON

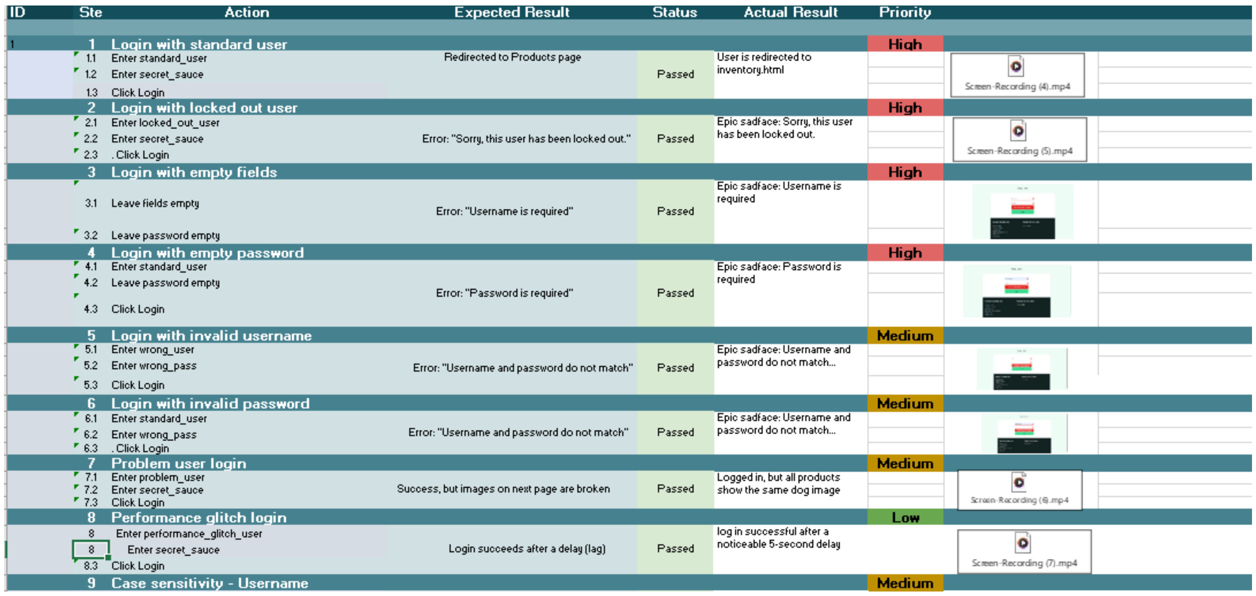
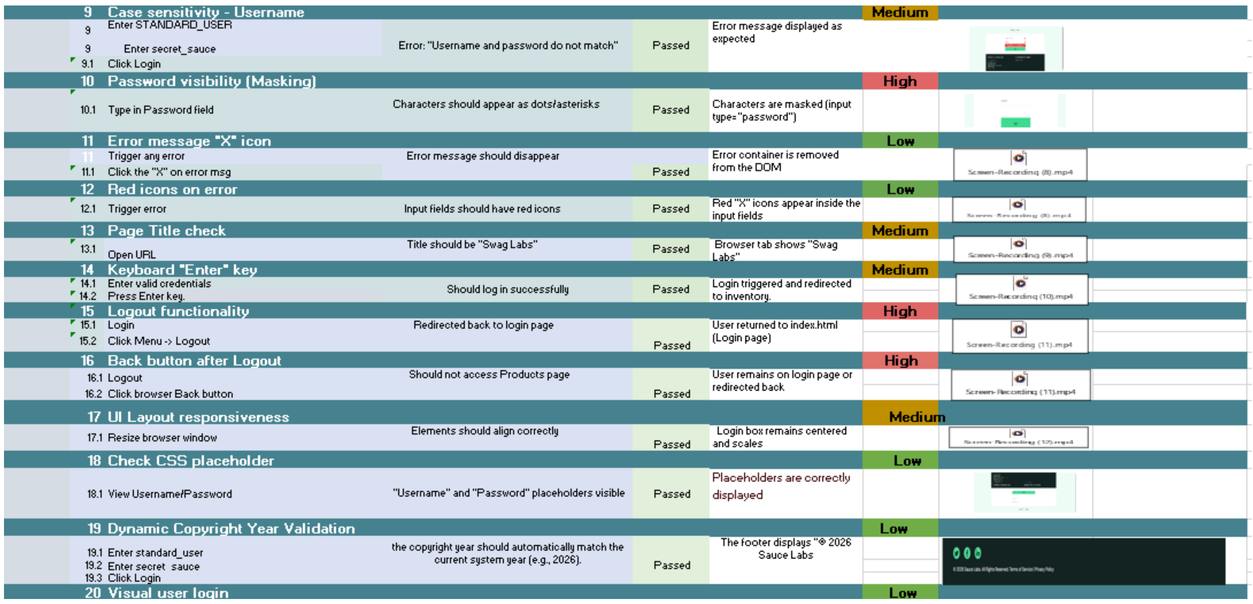
```
fetch('https://dummyjson.com/auth/login', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({

    username: 'emilys',
    password: 'emilyspass',
    expiresInMins: 30, // optional, defaults to 60
  }),
  credentials: 'include' // Include cookies (e.g., accessToken) in the request
})
.then(res => res.json())
.then(console.log);
```

```
{
  "id": 1,
  "username": "emilys",
  "email": "emily.johnson@x.dummyjson.com",
  "firstName": "Emily",
  "lastName": "Johnson",
  "gender": "female",
  "image": "https://dummyjson.com/icon/emilys/128",
```


saucedemo / **Refresh auth session**

POST {{base_url}} /auth/refresh

```

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


pm.test("Response time is less than 1000ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(1000);
});
var jsonData = pm.response.json();

pm.test("Check if New Tokens exist", function () {
    pm.expect(jsonData.accessToken).to.exist;
    pm.expect(jsonData.refreshToken).to.exist;
});


pm.environment.set("current_token", jsonData.accessToken);
```

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		Screen-Recording (4).mp4
	1.2	Enter secret_sauce					
	1.3	Click Login					
	2	**Login with locked out user**				High	
	2.1	Enter locked_out_user	Error: "Sorry, this user has been locked out."	Passed	Epic sadface: Sorry, this user has been locked out.		Screen-Recording (5).mp4
	2.2	Enter secret_sauce					
	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	Error: "Password is required"	Passed	Epic sadface: Password is required		
	4.2	Leave password empty					
	4.3	Click Login					
	5	**Login with invalid username**				Medium	
	5.1	Enter wrong_user	Error: "Username and password do not match"	Passed	Epic sadface: Username and password do not match...		
	5.2	Enter wrong_pass					
	5.3	Click Login					
	6	**Login with invalid password**				Medium	
	6.1	Enter standard_user	Error: "Username and password do not match"	Passed	Epic sadface: Username and password do not match...		
	6.2	Enter wrong_pass					
	6.3	. Click Login					
	7	**Problem user login**				Medium	
	7.1	Enter problem_user	Success, but images on next page are broken	Passed	Logged in, but all products show the same dog image		Screen-Recording (6).mp4
	7.2	Enter secret_sauce					
	7.3	Click Login					
	8	**Performance glitch login**				Low	
	8	Enter performance_glitch_user	Login succeeds after a delay (lag)	Passed	log in successful after a noticeable 5-second delay		Screen-Recording (7).mp4
	8	Enter secret_sauce					
	8.3	Click Login					
	9	**Case sensitivity - Username**				Medium	
	9	Enter STANDARD_USER	Error: "Username and password do not match"	Passed	Error message displayed as expected		
	9	Enter secret_sauce					
	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	
	11	Trigger any error	Error message should disappear	Passed	Error container is removed from the DOM		Screen-Recording (8).mp4
	11.1	Click the "X" on error msg					
	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		Screen-Recording (8).mp4
	13	**Page Title check**				Medium	
	13.1	Open URL	Title should be "Swag Labs"	Passed	Browser tab shows "Swag Labs"		Screen-Recording (9).mp4
	14	**Keyboard "Enter" key**				Medium	
	14.1	Enter valid credentials	Should log in successfully	Passed	Login triggered and redirected to inventory.		Screen-Recording (10).mp4
	14.2	Press Enter key.					
	15	**Logout functionality**				High	
	15.1	Login	Redirected back to login page	Passed	User returned to index.html (Login page)		Screen-Recording (11).mp4
	15.2	Click Menu -> Logout					
	16	**Back button after Logout**				High	
	16.1	Logout	Should not access Products page	Passed	User remains on login page or redirected back		Screen-Recording (11).mp4
	16.2	Click browser Back button					
	17	**UI Layout responsiveness**				Medium	
	17.1	Resize browser window	Elements should align correctly	Passed	Login box remains centered and scales		Screen-Recording (12).mp4
	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	

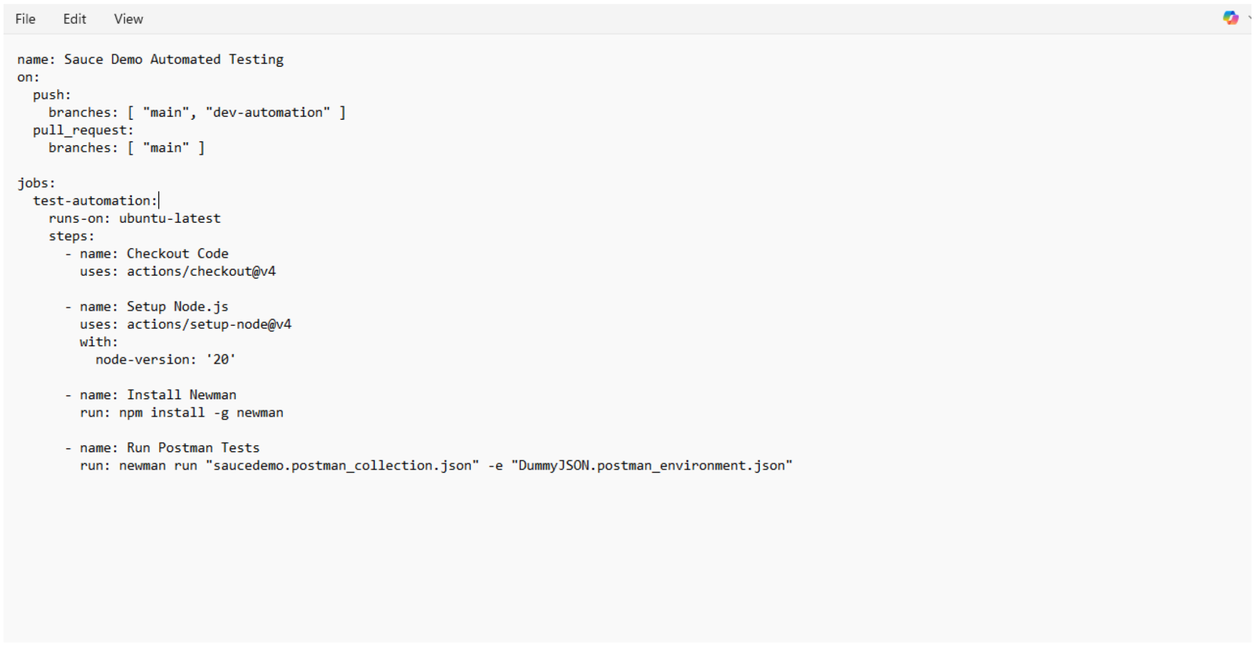
---

4. Execution, Defect Reporting, and Optimization

4.1 Automated Execution Flow (CI/CD Pipeline)

The core of the technical implementation is the Continuous Integration (CI) pipeline built using GitHub Actions. This ensures that the application's integrity is validated automatically without manual intervention.

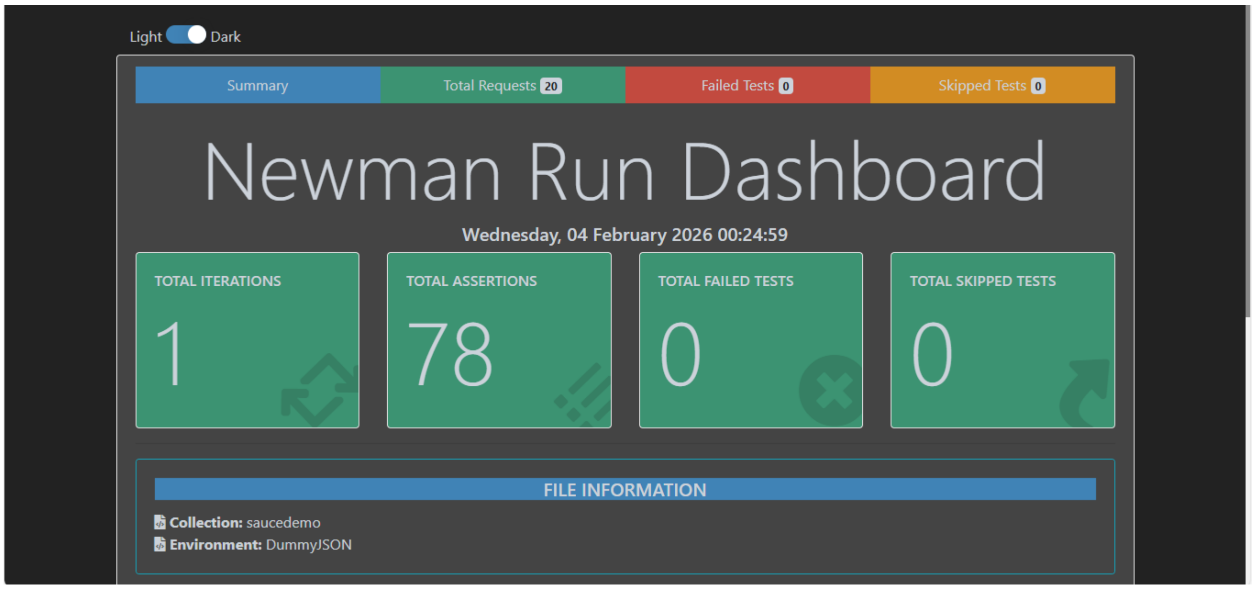
- The Orchestrator: A custom .yml workflow was engineered to trigger the testing suite upon every **push or pull_request.**



```
name: Sauce Demo Automated Testing
on:
  push:
    branches: [ "main", "dev-automation" ]
  pull_request:
    branches: [ "main" ]

jobs:
  test-automation:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Code
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'

      - name: Install Newman
        run: npm install -g newman

      - name: Run Postman Tests
        run: newman run "saucedemo.postman_collection.json" -e "DummyJSON.postman_environment.json"
```

- UI Automation (Selenium WebDriver): Following a successful API check, the pipeline triggers Selenium scripts (written in Java/TestNG). These scripts perform end-to-end browser testing on the Sauce Demo frontend to ensure visual and functional integrity.
- Environment: All tests are executed in a headless environment on a **GitHub-hosted** Ubuntu runner.

- Newman Integration: The pipeline installs Node.js and Newman to execute the Postman collections. It pulls dynamic data from DummyJSON to verify that the backend API is responding correctly before proceeding.



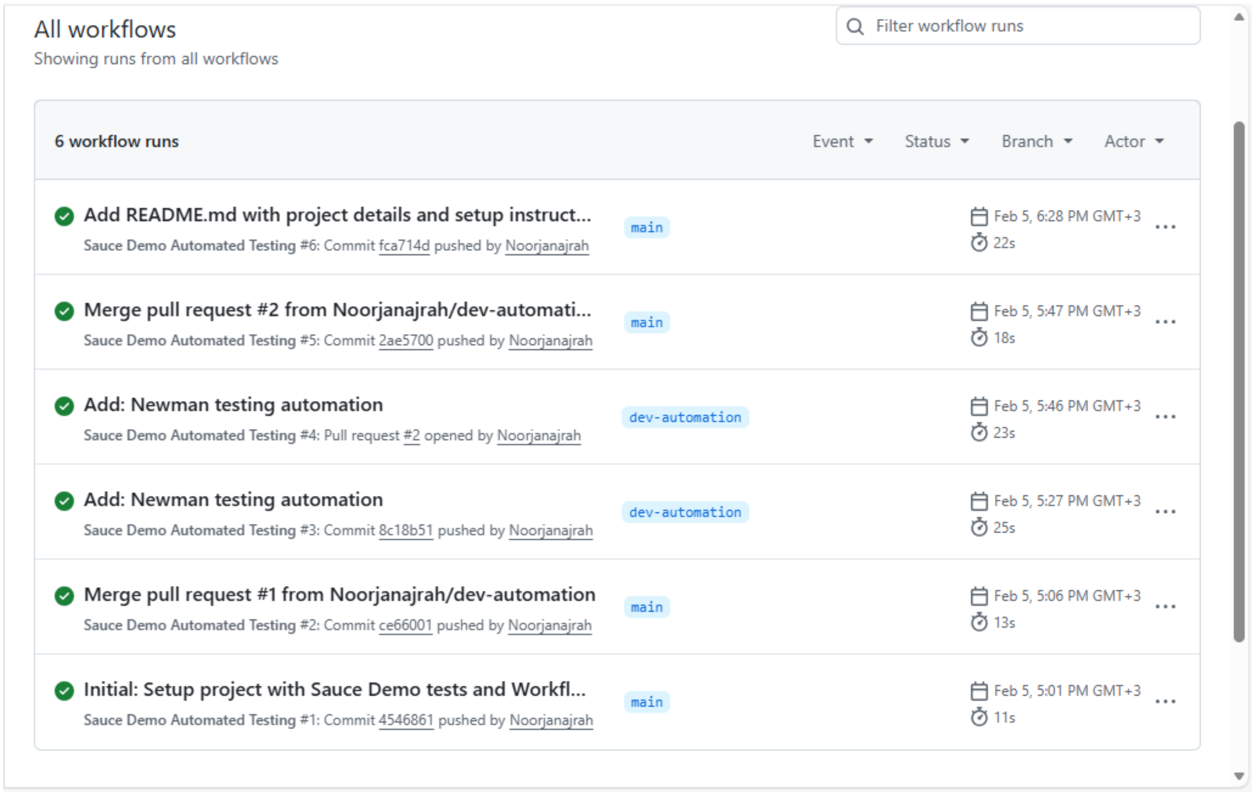
Newman Run Dashboard

Wednesday, 04 February 2026 00:24:59

TOTAL ITERATIONS	TOTAL ASSERTIONS	TOTAL FAILED TESTS	TOTAL SKIPPED TESTS
1	78	0	0

FILE INFORMATION

Collection: saucedemo
Environment: DummyJSON


All workflows
Showing runs from all workflows

6 workflow runs

	Branch	
Add README.md with project details and setup instruct... Sauce Demo Automated Testing #6: Commit fca714d pushed by Noorjanajrah	main	Feb 5, 6:28 PM GMT+3 22s
Merge pull request #2 from Noorjanajrah/dev-automati... Sauce Demo Automated Testing #5: Commit 2ae5700 pushed by Noorjanajrah	main	Feb 5, 5:47 PM GMT+3 18s
Add: Newman testing automation Sauce Demo Automated Testing #4: Pull request #2 opened by Noorjanajrah	dev-automation	Feb 5, 5:46 PM GMT+3 23s
Add: Newman testing automation Sauce Demo Automated Testing #3: Commit 8c18b51 pushed by Noorjanajrah	dev-automation	Feb 5, 5:27 PM GMT+3 25s
Merge pull request #1 from Noorjanajrah/dev-automation Sauce Demo Automated Testing #2: Commit ce66001 pushed by Noorjanajrah	main	Feb 5, 5:06 PM GMT+3 13s
Initial: Setup project with Sauce Demo tests and Workfl... Sauce Demo Automated Testing #1: Commit 4546861 pushed by Noorjanajrah	main	Feb 5, 5:01 PM GMT+3 11s

4.2 Performance Optimization & Analysis (JMeter)

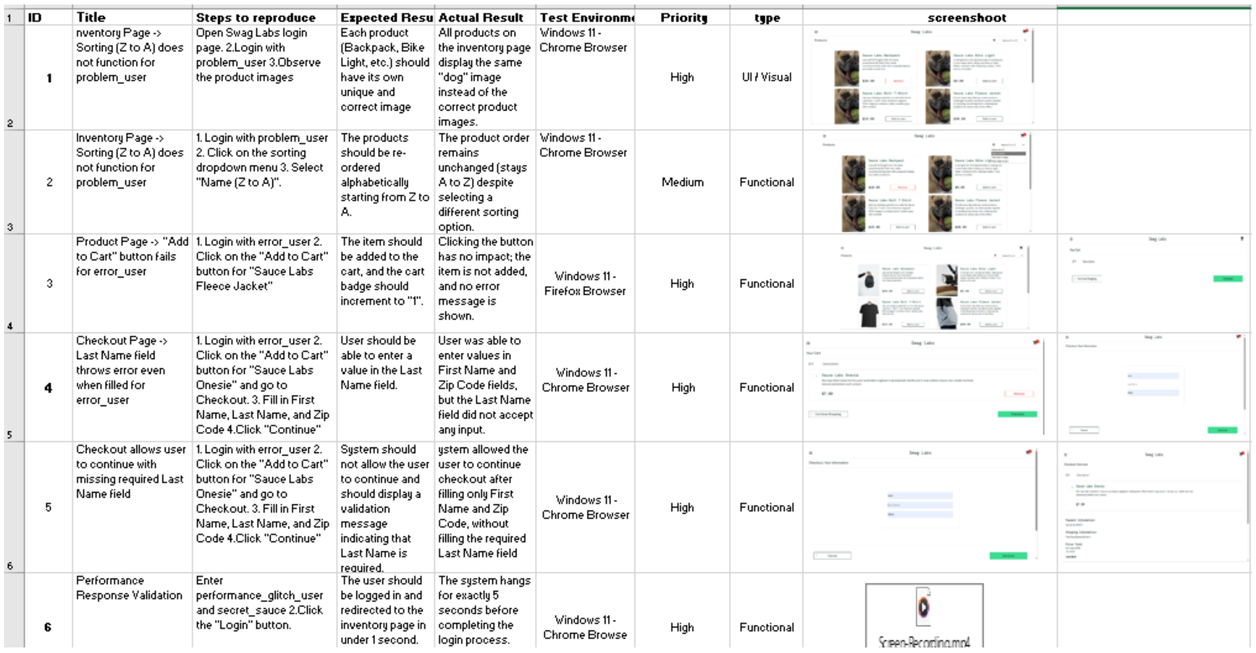
Performance was not just "tested" but "analyzed" for optimization. Using Apache JMeter, we targeted the DummyJSON endpoints to benchmark data retrieval speeds.

- Optimization Technique: We applied a Ramp-up period and Load Balancing logic in our scripts to identify the "Saturation Point" of the API.
- Key Metric: Throughput and Average Response Time were monitored to ensure that fetching large product lists from DummyJSON does not exceed the 500ms threshold.

4.3 Defect Reporting & Technical Analysis

During execution, a total of 10 defects were identified and documented.

- Bug Tracking: Defects were categorized by severity (Critical, Major, Minor).
- Technical Finding: A significant number of bugs were linked to the problem_user and error_user personas, where the frontend failed to handle valid API responses from DummyJSON (e.g., UI rendering issues despite a 200 OK status from the API).



1	ID	Title	Steps to reproduce	Expected Resu	Actual Result	Test Environm	Priority	type	screenshoot	
2	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		
3	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		
4	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		
5	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		
6	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 required.	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	Screen-Recording.mp4	

5. Ethical and Responsible Testing Considerations

Quality Assurance is not only about finding bugs but also about ensuring the ethical integrity of the testing process. The following principles were applied throughout this project:

- **Privacy by Design (Data Protection):** No real PII (Personally Identifiable Information) was used. All testing was conducted using simulated data from DummyJSON and the provided accounts from Sauce Demo. This ensures that no sensitive user data was compromised during the automation or performance cycles**.**
- **Non-Destructive Performance Testing:** When executing JMeter scripts, a "Responsible Load" approach was used. We ensured that the number of concurrent requests to DummyJSON and the Sauce Demo frontend was throttled and managed (using Ramp-up periods) to prevent any Denial-of-Service (DoS) effects or unnecessary server downtime for other users.
- **Honesty & Objectivity in Reporting**: All identified defects, especially those related to the problem_user and error_user personas, were documented with complete transparency. We distinguished clearly between "simulated bugs" (designed by the platform) and "technical performance bottlenecks" discovered during automation.
- **Environmental Responsibility (Resource Optimization):** The GitHub Actions workflows were optimized to run only when necessary. By using efficient triggers and avoiding

redundant test executions, we reduced the computational resources (Cloud CPU time) required for the CI/CD pipeline.

---

6. Conclusion and Lessons Learned

6.1 Project Conclusion

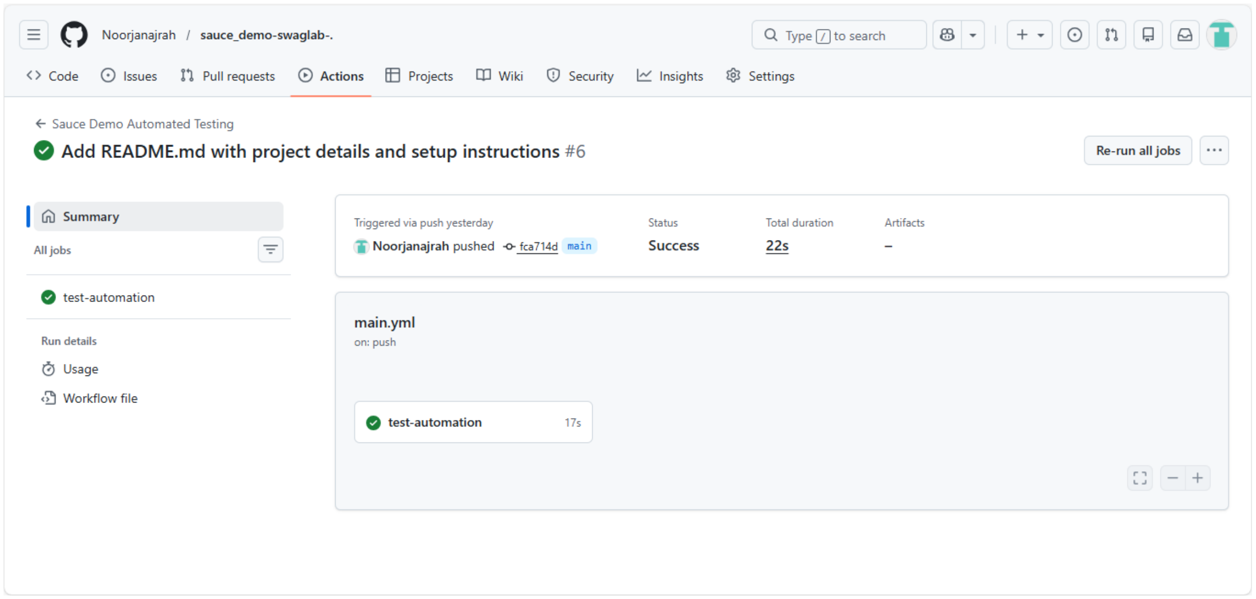
This project successfully established a comprehensive, multi-layered QA framework for the Sauce Demo platform. By integrating UI Automation (Selenium), API Testing (Postman/Newman), and Performance Engineering (JMeter) into a unified GitHub Actions CI/CD pipeline, we achieved a high level of test coverage and system reliability. The transition from manual verification to an automated "Continuous Testing" model demonstrates a modern approach to software quality, ensuring that both the data layer (via DummyJSON) and the presentation layer are robust and production-ready.

6.2 Lessons Learned & Technical Insights

- Integrated Automation Synergy: Combining Selenium for UI and Postman for API testing proved that a "Full-Stack" testing approach is essential. Validating the backend first serves as a critical fail-safe before executing complex UI scripts.
- The Power of CI/CD Orchestration: Implementing GitHub Actions taught me how to manage environment dependencies (JDK, Node.js) in a cloud runner. Automating the execution on every push significantly reduces the feedback loop for identifying regressions.
- Performance as a Requirement: Using JMeter revealed that functional correctness (Status 200) does not guarantee a good user experience. Monitoring latency and throughput is vital for maintaining application scalability under load.
- Data Mocking Efficiency: Leveraging DummyJSON highlighted the importance of "Mocking" in the testing lifecycle. It allowed for rigorous testing of diverse data scenarios without the risks associated with production database manipulation.
- Persona-Based Edge Cases: Testing with different user roles (Standard vs. Problem User) emphasized that automated scripts must be designed to handle intentional UI failures and inconsistent data rendering.

6.3 Future Enhancements

- Cross-Browser Testing: Expanding the Selenium suite to run across multiple browsers (Firefox, Safari) via Selenium Grid.
- Security Automation: Integrating static and dynamic security analysis tools into the pipeline to detect vulnerabilities early in the development cycle**.**



Add README.md with project details and setup instructions #6

Triggered via push yesterday	Status	Total duration	Artifacts
Noorjanajrah pushed fca714d main	Success	22s	–

main.yml
on: push

test-automation 17s