Bug Busters

Swag labs

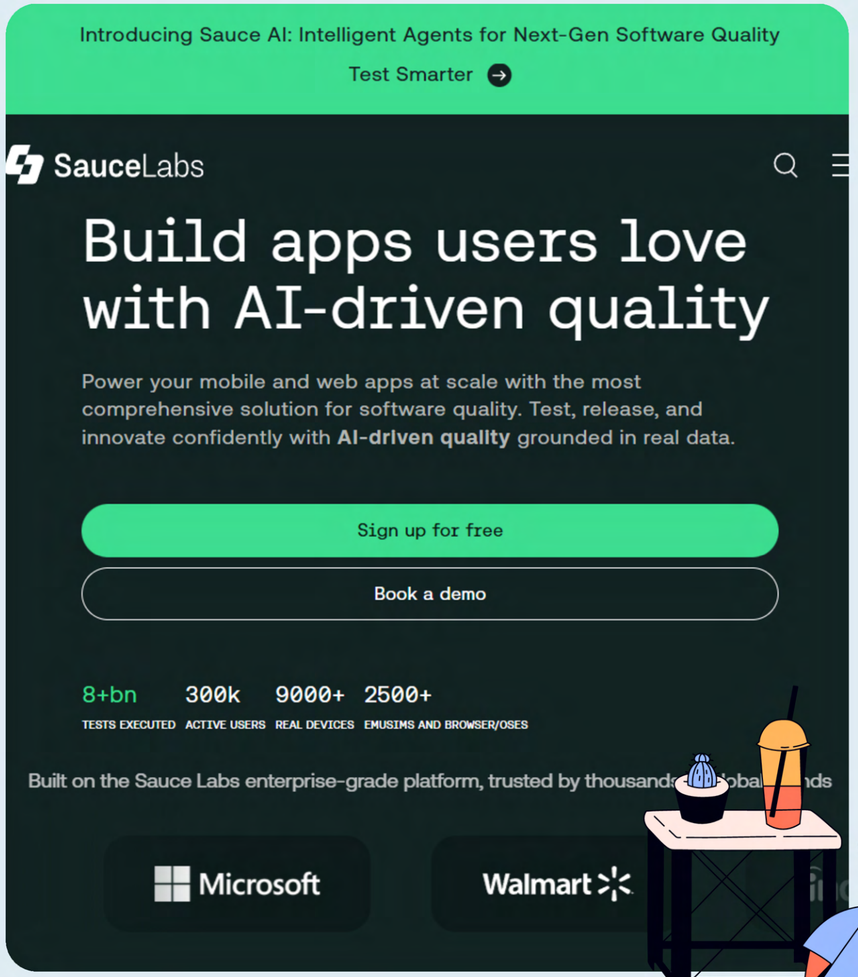
Bug Busters

The Team

Reem Ahmed
Basmala Emad
Sama Abd El Nasser
Mahmoud Ashraf
Abdelrahman Mohamed
Diana Emil

Why Swag Labs?

- demo e-commerce web created specifically for testing practice.
- simple enough for beginners but realistic enough to mimic real-world software behavior.



Project Idea

“Conduct full testing of SwagLabs using a structured QA approach.”

Our testing approach included:

Exploratory Testing

- understand the application flow
- Identify risks and unstable areas

Feature-based Testing

- Divided our tests into major features

Bug Buster

project structure

We organized the entire project into 3 sprints, each focusing on one major testing discipline.

- Sprint 1 — Manual Testing
- Sprint 2 — Automation Testing
- Sprint 3 — API Testing

Bug Buster

Sprint 1 — Manual Testing

Goal: Understand the application thoroughly and document test cases.

Sprint 1 — Manual Testing

Entry Criteria

- Application is stable and accessible
- User accounts and product data available
- Team assigned to features
- Test case and bug report templates ready
- Tools configured
- Exploratory testing scope defined

Exit Criteria

- 100% test cases written & executed
- All bugs logged with details
- Feature Analysis
- Test summary report completed
- Manual testing phase signed off

Sprint 1 — Manual Testing

Login

- All possible scenrios

TC_ID	Feature	Title	Description
TC_001	Login	Valid login_001	Loging in using valid username & password
TC_002	Login	Invalid password_001	Loging in using valid username & invalid password
TC_003	Login	Invalid username	Loging in using invalid username & valid password
TC_004	Login	CAPS username	Loging in using valid username in Caps & valid password
TC_005	Login	CAPS password	Loging in using valid username & valid password in CAPS
TC_006	Login	Leaving username blank	Loging in using password only
TC_007	Login	Leaving password blank_001	Loging in using username only

Inventory

- Cart Functionality
- UI response

TC_ID	Feature	Title	Description
TC_INVENTORY_001	Add_To_Cart (Functionality)	Verify "Add To Cart" Button	verify when Clicking on "Add To Cart" Button,the item adds successfully to shopping card.
TC_INVENTORY_002	Add_To_Cart (Functionality)	Verify "Remove" button	verify when Clicking on "Remove" Button,the item removes successfully from shopping card.
TC_INVENTORY_003	Add_To_Cart (Functionality)	Verify Adding Mutiple Item to The Cart	verify when Clicking on "Add To Cart" Button for multiple items, each item adds successfully to shopping card.
TC_INVENTORY_004	Add_To_Cart (Functionality)	Verify Removing Mutiple Item from The Cart	verify when Clicking on "Remove" Button for multiple items, each item removes successfully from shopping card.

Sprint 1 — Manual Testing

Sort

- Alphabetically
- Price

TC_ID	Feature		Title	Description
Tc_01	Sorting Function		Verify Sorting A → Z	Ensure products are sorted alphabetically from A to Z
TC_02	Sorting Function		Verify Sorting Z → A	Ensure products are sorted alphabetically from Z to A
TC_03	Sorting Function		Verify Sorting Low → High	Ensure products appear from lowest to highest price
TC_04	Sorting Function		Verify Sorting High → Low	Ensure products appear from highest to lowest price
TC_05	Sorting Function		Verify Default Sorting	Check default sorting on first load
TC_06	Sorting Function		Verify Sorting Persists After Refresh	Ensure sorting stays after refresh

Product Page

- Information
- Cart Functionality
- Navigation
- UI Response

TC_ID	Feature		Title	Description
TC_001	Product_page (info)		Inventory Item Page Load Verification	Verify Inventory item page loads
TC_002	Product_page (info)		Product Image Load Verification	Verify image loads perfectly
TC_003	Product_page (info)		Product title Verification	Verify the product title matches Inventory Page
TC_004	Product_page (info)		Product description Verification	Verify the product description matches Inventory Page

Sprint 1 — Manual Testing

Cart

All possible scenrios

TC_ID	Feature		Title	Description
TC_001	Cart		Verify that the "Add to Cart" button works properly	Check if the "Add to Cart" button successfully adds the selected product to the shopping cart.
TC-002	Cart		Verify that the "Remove" button removes an item from the cart	Check if the "Remove" button successfully removes the selected product from the shopping cart
TC-003	Cart		Verify that the cart badge number updates correctly	check if the number displayed on the cart badge changes according to the number of items added
TC-004	Cart		Verify that the cart page displays the added products	Check if the products added to the cart are correctly displayed on the cart page .
TC-005	Cart		Verify that the "Continue Shopping" button works correctly	check if you click "continue shooping" it's back to product page

Checkout Overview

Text Verfication

TC_ID	TC_ID	Req ID	Title	Description
TC_01	Checkout_overview		succesful checkout	Verify that a user can successfully complete checkout process using valid payment and shipping info.
TC_02	Checkout_overview		choose one item and monitor the checkout	choose one item and monitor if item its price and appearance shown up correctly and total price shown up correctly
TC_03	Checkout_overview		choose two item and monitor check out	choose two item and monitor if item its price and appearance shown up correctly and total price shown up correctly

Sprint 1 — Manual Testing

Checkout Info

Valid and Invalid Fields

TC_ID	Feature		Title	Description
check_info_001	check_info		enter vaild firstname,lastname,zipcode	check functionality of check_info feature by enter vaild data
check_info_002	check_info		enter invaild firstname and vaild lastname,zipcode	enter firstname as number not string (invaild),vaild lastname and zipcode
check_info_003	check_info		enter firstname field only	enter vaild firstname and leave lastname,zipcode empty

Sprint 1 — Manual Testing

Findings:

✓ Exploratory testing helped discover missing scenarios
✓ Highest bugs found in Checkout Information & Inventory
✓ Most stable feature: Login
✓ Test cases refined during execution
✓ Bug % per feature guided our next Sprint

Feature	Covered	Bug percentage
Login	25	20%
Inventory	20	**10%**
Sort	6	0%
Product_Page	29	10.34%
Cart	15	20%
Checkout_Info	17	70.59%
Checkout_Overview	6	33.33%

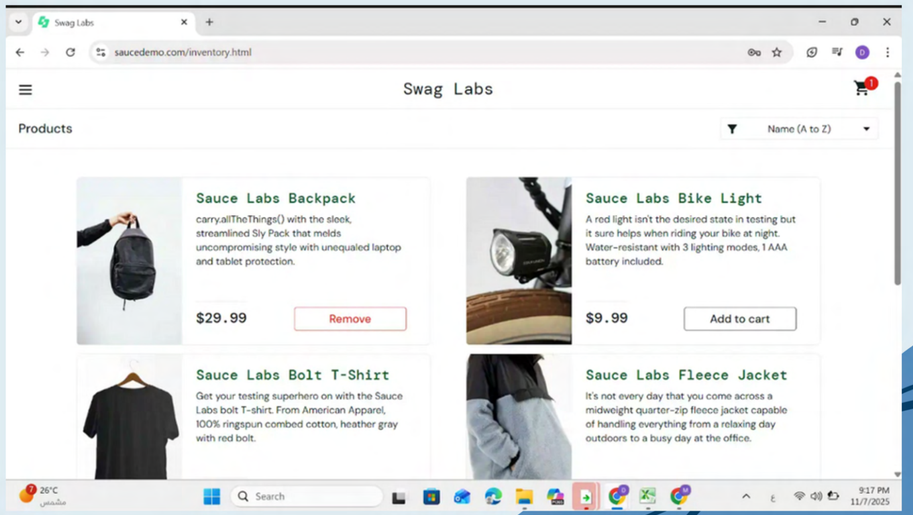
Sprint 1 — Manual Testing

BUG REPORT

Bug ID	Module / Feature	Title / Summary	Reported By	Detected In / Test Cycle	Severity	Priority	Environment	TC_ID
UG_018	Add to Cart	System does not allow adding more than one unit of the same product	Diana	Sprint 1	Medium	Low	Tested on laptop using Chrome browser (Windows 10)	Cart_008

Description	Steps to Reproduce	Expected Result	Actual Result	Attachments / Evidence
When the user tries to add the same product to the cart more than once, the system prevents it by changing the “Add to Cart” button to “Remove.” This behavior may confuse users, as most e-commerce systems allow adding multiple quantities of the same product.	1-open https://www.saucedemo.com/inventory.html 2-Select a product that is out of stock in the backend/database. 3-Try to add the same product again.	The system should allow adding more than one unit of the same product or provide an option to increase quantity in the cart.	After adding the product once, the “Add to Cart” button changes to “Remove,” and the user cannot add more units.	https://drive.google.com/file/d/1Ht19se51eD9x3JbwCd5OWAAV5KwJXXdn/view?usp=drive_link

Evidence



Sprint 1 — Manual Testing

Feedback:

- We checked each other’s test cases to make sure everything was correct.
- We shared direct feedback to improve the test cases.
- This helped us find mistakes and make the test cases stronger.

Bug Buster

Sprint 2 — Automation Testing

Goal: Automate the most important test cases from Sprint 1.

Sprint 2 — Automation Testing

Entry Criteria

- Stable build available
- Manual test cases selected for automation
- Framework (Selenium + Java) is set up
- Test data and reusable functions prepared

Exit Criteria

- Core flows automate
- Scripts run successfully without false failures
- Automation report generated
- Code reviewed and committed

Sprint 2 — Automation Testing

Tools used:

- Selenium WebDriver
- Java
- TestNG
- Allure

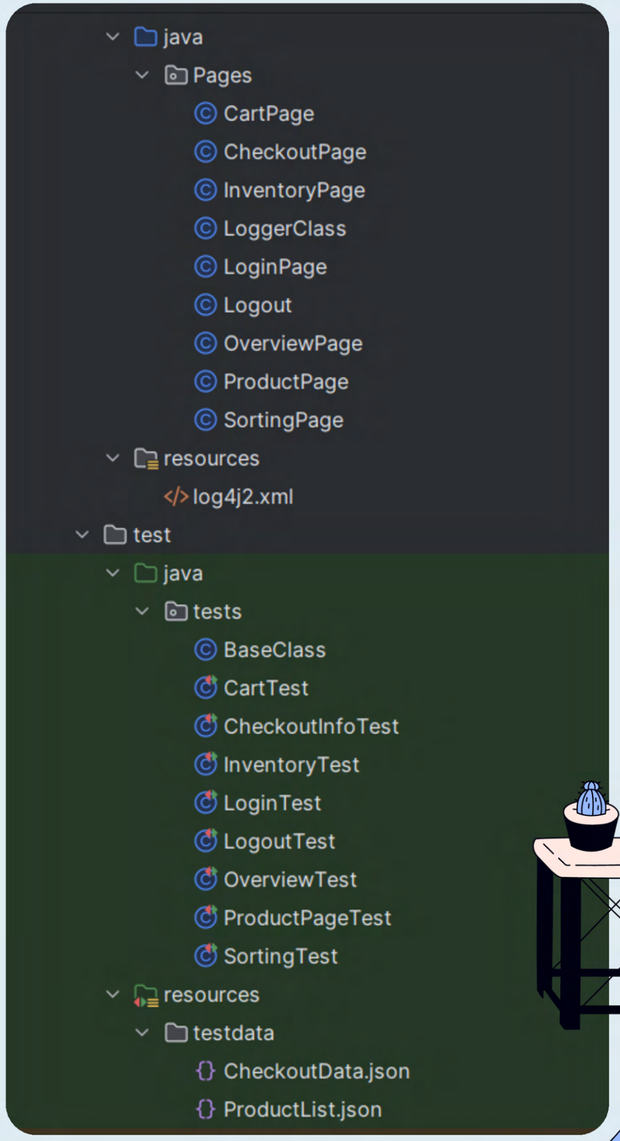
Approach:

- Data Driven
- POM(Design Pattern)

Sprint 2 — Automation Testing

Automated areas:

- Login tests
- Add to cart / remove from cart
- Cart updates
- Product detail validations
- Checkout flow
- Sorting
- Inventory



Sprint 2 — Automation Testing

Test Execution Flow

- Launch browser using Selenium WebDriver.
- Navigate to Swag Labs URL.
- Login using credentials from DataProvider.
- Perform actions: add products to cart, checkout, verify totals.
- Assertions to validate expected results.
- Log execution steps using log.info().
- Generate report in Allure after execution.

```
public class BaseClass {  8 inheritors

    WebDriver driver;  62 usages

    @BeforeMethod
    public void setup() {
        driver = new FirefoxDriver();      // Edit
        driver.manage().window().maximize();
        driver.get("https://www.saucedemo.com/");
    }
```

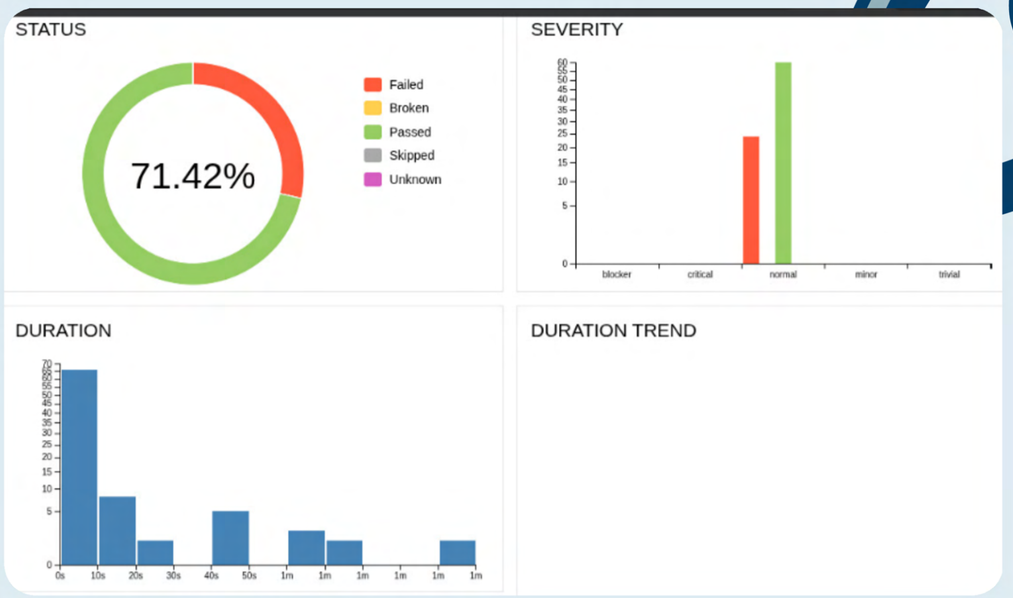
```
@AfterMethod
public void teardown() { driver.quit(); }
```

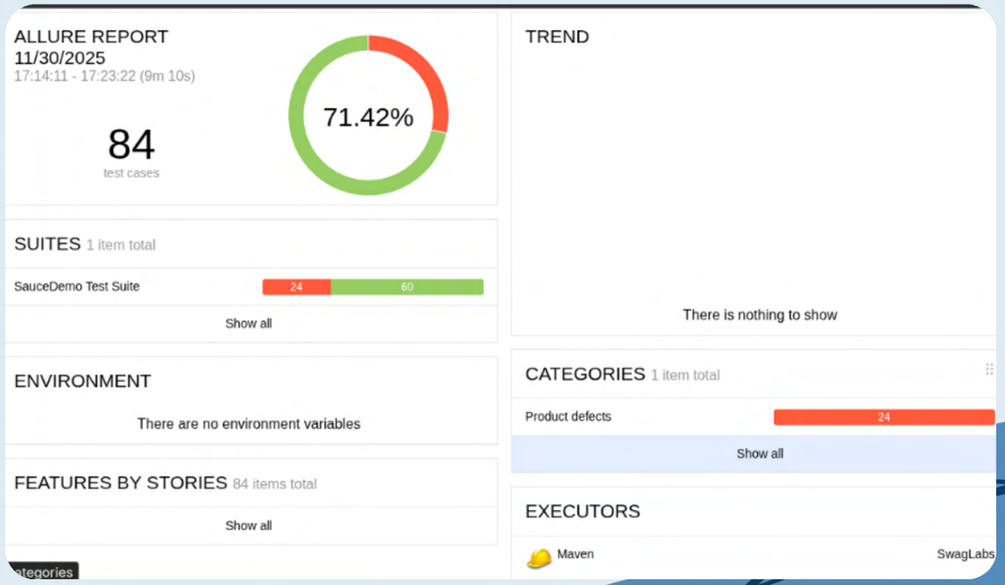
```
@DataProvider(name = "ValidLoginData")
public Object[][] validDataProvider() {
    return new Object[][]{
            { "standard_user", "secret_sauce" },  // Standard user
            { "problem_user", "secret_sauce" },  // Problem user
            { "performance_glitch_user", "secret_sauce" },  // Performance glitch user
            { "error_user", "secret_sauce" },  // Error user
            { "visual_user", "secret_sauce" },  // Visual user
    };
}
```

Sprint 2 — Automation Testing

Allure Report:

- Dashboard Overview
- Detailed Test Case View
- Suites View
- Timeline View
- Severity Labels
- History Tracking



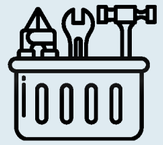


Sprint 2 — Automation Testing

Page Classes

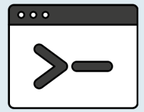
Test Classes

Base Class



Benefits of This Approach

- Clean, maintainable code using POM.
- Reusable test methods and data-driven tests.
- Easy debugging with logs.
- Professional reports for stakeholders.



Logging

Reporting

Sprint 2 — Automation Testing

Coding sessions

- Support each other while writing automation scripts
- Resolve coding issues in real-time
- Ensure consistent approach across the team
- Share best practices and reusable functions

Bug Buster

Sprint 3 — API Testing

Goal: Validate API

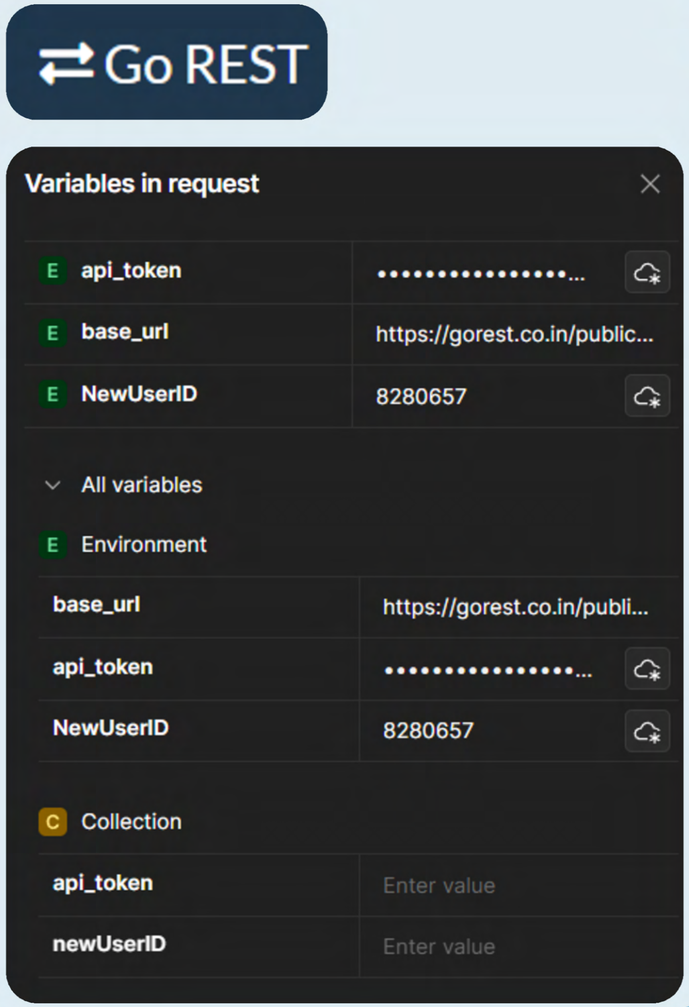
Sprint 3 — API Testing

Sadly Swag Lab doesn’t have API we can test on so we used a small Project called GORest to test their API

Sprint 3 — API Testing

RESTful API Testing (Postman)

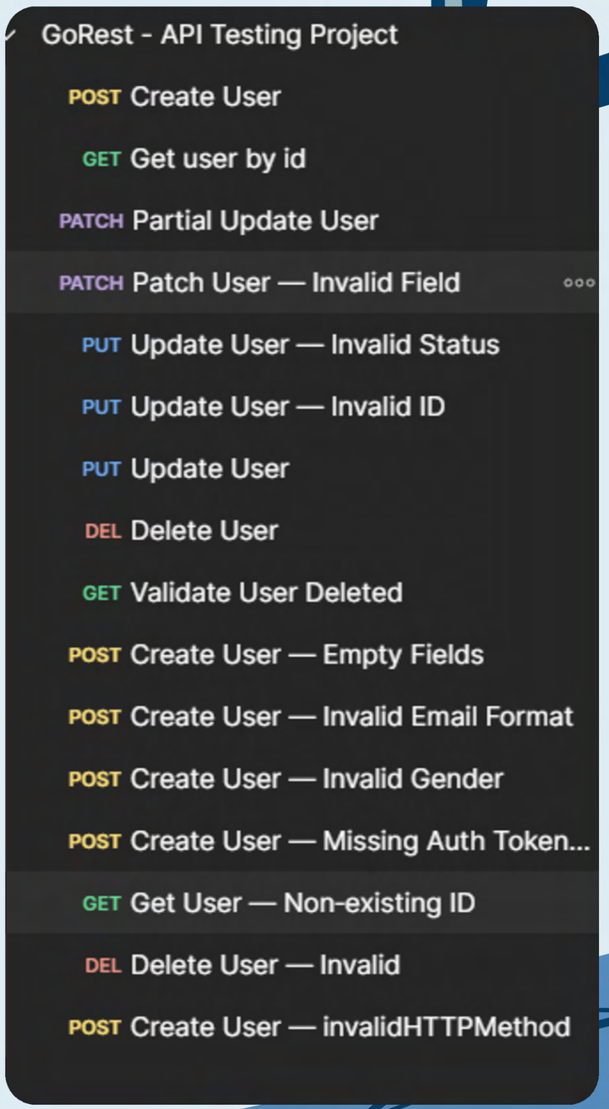
- Tested RESTful APIs (GET, POST, PUT, DELETE)
- Used Postman to send requests & manage collections
- Bearer Token for authentication
- Saved dynamic variables (e.g., user ID)
- Validated status codes & response fields
- Used Pre-request & Test scripts for automation



Sprint 3 — API Testing

API Endpoints Tested:

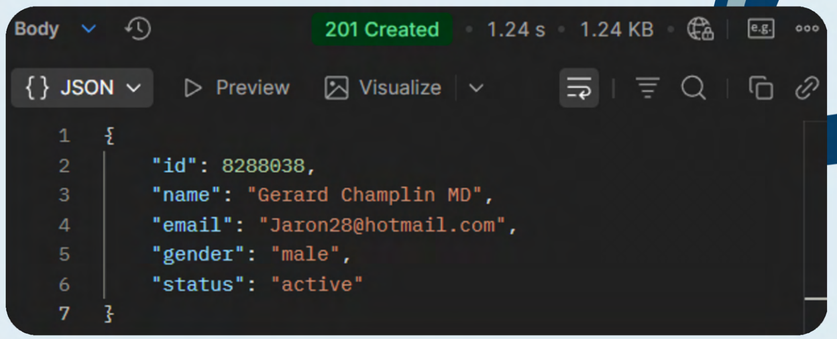
- OST: Create User
- GET: Get User by ID
- PATCH: Partial Update User / Invalid Field
- PUT: Update User / Invalid ID / Invalid Status
- DELETE: Delete User / Invalid
- GET: Validate User Deleted
- POST (Negative Tests): Empty Fields / Invalid Email / Invalid Gender / Missing Auth Token / Invalid HTTP Method
- GET (Negative Test): Non-existing ID

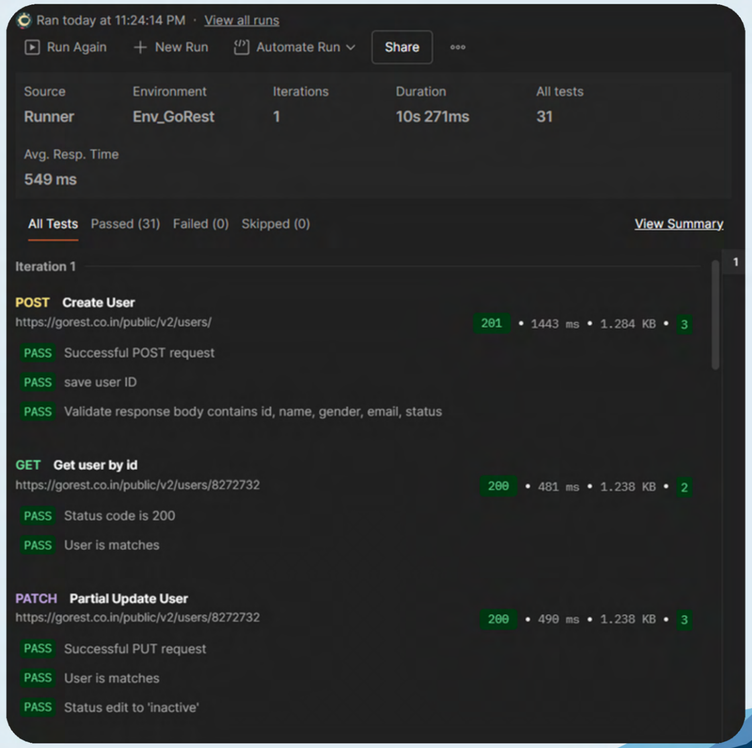


Sprint 3 — API Testing

Deliverables:

- Test Cases – All scenarios covered
- API Documentation – Endpoints & auth
- Execution Report – Results & logs

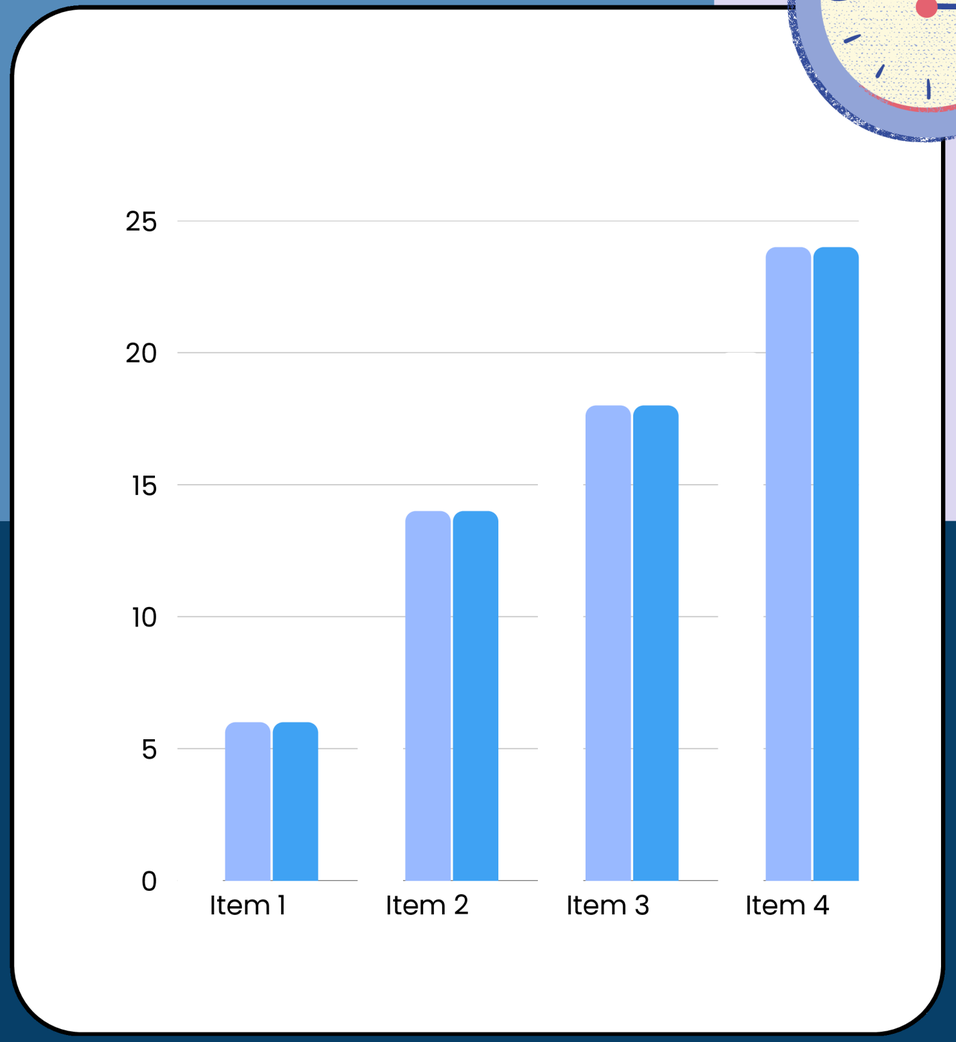




Bug Buster

summary

- We applied testing: manual + automation + API.
- Covered all major functionality of SwagLabs.
- Identified bugs.



Bug Busters

What We learnt

Importance of exploratory testing before designing formal test cases
How to break down a project into workable sprints
Building a structured automation code
Creating clear, maintainable test cases
Understanding real QA workflow from start to finish

Bug Buster

Thank You!

Do you have any questions?