Software Testing & Quality Assurance Project Report

University of Layyah

Department of Information Technology

QUALITY ENGINEERING REPORT

Comprehensive SQA Analysis of the Toolshop Platform

Prepared by (The QA Project Team)

Khadija Zafar	UL-BSITM-A-23-21	QA Lead
Areej Fatima	UL-BSITM-A-23-43	Automation Engineer
Saba Shahzad	UL-BSITM-A-23-33	API Specialist
Hannia Fatima	UL-BSITM-A-23-49	Test Manager

Website Under Analysis: https://practicesoftwaretesting.com/

Course & Lab Instructor: Mr. Faisal Hafeez

BSIT Morning A (2023-27)

Table of Contents

Executive Summary 4

Overview of Engagement 4

Key Objectives & Methodology 4

High Level Execution Results 4

Critical Findings 5

Software Requirement Description (SRD) 6

User Lifecycle & Authentication 6

Product Discovery & Search Logic 6

UI Interactivity & Accessibility (Special Requirement) 6

Backend Reliability & Data Persistence 6

Test Plan & Strategic Approach 8

Testing Methodology 8

Strategic Testing Layers 8

Manual Testing Suite 9

Execution Overview 9

Manual Execution Summary Table 9

Evidence of Manual Execution 9

Automated Testing Suite (Playwright & Python) 16

Script Logic and Architecture 16

Results and Performance Metrics 16

Evidence of Automation Execution 17

API Validation Suite (Postman) 18

Implementation & Environment Strategy 18

Detailed API Test Results 18

Evidence of API Validation 18

Defect Management & Jira Analysis 20

Defect Governance Framework 20

Jira Backlog Overview 20

Analysis of Key Defects 21
Requirement Traceability Matrix (RTM) 25
Purpose of the RTM 25
Traceability Data Grid 25
Final Quality Assessment 26
Strategic Observations 26
Closing Statement 26
References & Technical Repositories 27
Application & Infrastructure 27
Technical Documentation & Tools 27
Project Deliverables Checklist 28

Executive Summary

Overview of Engagement

This report details the end to end Quality Assurance (QA) engagement for the **Toolshop** web application, a dynamic ecommerce platform specializing in hardware retail. The primary objective of this project was to establish a rigorous validation framework to confirm the application's functional integrity, user interface (UI) accessibility compliance, and backend service stability.

Under the supervision of Mr. Faisal Hafeez, the project team (**Khadija Zafar, Areej Fatima, Saba Shahzad, and Hannia Fatima)** executed a synchronized, multi-tier testing strategy. This approach ensured that every layer of the application, from the frontend user experience to the underlying API infrastructure, was scrutinized for defects and performance bottlenecks.

Key Objectives & Methodology

The engagement was structured around four core pillars of software quality:

- **Functional Validation:** Ensuring critical business logic (Registration, Authentication, and Checkout) operates without error.
- **Automation ROI:** Deploying a high speed **Playwright** suite to automate repetitive regression tasks and smoke tests.
- **API Integrity:** Auditing the RESTful backend via **Postman** to ensure data accuracy and server-side reliability.
- **Governance & Transparency:** Managing the defect lifecycle through **Jira** to provide stakeholders with clear visibility into system risks.

High Level Execution Results

A total of **43 test cases** were executed across the various testing tiers (25 Manual, 10 Automated, and 8 API). The high level results are summarized below:

Testing Tier	Total Cases	Passed	Failed	Success Rate
Manual UI/UX	25	22	3	88%
Playwright Automation	10	10	0	100%
Postman API Validation	8	8	0	100%
Overall Project State	**43**	**40**	**3**	**93.02%**

Critical Findings

The engagement successfully verified the specific requirement for **interactive hover effects** on primary buttons (Sign In, Sign Up, and Login). While the system remains stable at the API level, the following observations were documented:

- **Search Filtering Logic:** A defect was identified in the price slider filtering mechanism.
- **User Registration Feedback:** The absence of success confirmation banners during the registration flow was noted.

Software Requirement Description (SRD)

The **Software Requirement Description** serves as the foundation for our testing activities. It defines the "Expected Behavior" against which the Toolshop platform was audited. The scope was categorized into four critical business pillars:

User Lifecycle & Authentication

The objective is to validate the security and usability of the user journey from guest to registered member.

- **Registration Integrity:** The system must accept valid data, encrypt passwords, and critically provide immediate visual confirmation (Success Banners) upon account creation.
- **Authentication Security:** Secure login/logout protocols. The system must handle "Negative Scenarios" (invalid credentials) by displaying appropriate `.alert-danger` messages without revealing sensitive system information.

Product Discovery & Search Logic

As a retail engine, the ability to find products is the highest priority.

- **Keyword Precision:** The search bar must return relevant product cards (e.g., "Hammer") within a 2 second latency window.
- **Dynamic Filtering:** The price slider must dynamically update the DOM to show products only within the selected range ($0 - $200). Logic failures here are classified as "High Severity" as they directly impede sales.

UI Interactivity & Accessibility (Special Requirement)

The platform must provide high visual feedback to users.

- **Hover State Compliance:** All primary navigation links (Home, Categories) and action buttons (Sign In, Sign Up, Login) must trigger a CSS state change (color shift, shadow, or underline) when targeted by a cursor. This is essential for web accessibility and user engagement.
- **Button Responsiveness:** Buttons must be "clickable" across their entire surface area, not just the text label.

Backend Reliability & Data Persistence

Since the project utilizes **JavaScript** for frontend storage but relies on an **API** for data retrieval, the backend must be bulletproof.

- **JSON Integrity:** API responses must follow a strict schema. For example, a `GET /products` request must return an array of objects containing `id`, `name`, and `price`.
- **Status Code Standardization:** All successful queries must return an `HTTP 200 OK` status, and unauthorized access to account endpoints must return a `401 Unauthorized`.

Test Plan & Strategic Approach

Testing Methodology

The team adopted a **Risk Based Testing (RBT)** methodology to ensure that testing efforts were focused on the highest impact areas of the Toolshop platform. By identifying potential failure points that could lead to revenue loss or user frustration (such as broken checkout or login failures), we prioritized these scenarios in our execution schedule.

Strategic Testing Layers

To provide a 360 degree view of software quality, we utilized a multi-tier strategy:

- **Automated End-to-End (E2E) Testing:** We utilized **Playwright (Python)** to safeguard core "Happy Path" journeys. Automation ensures that essential workflows such as the user login sequence and the "Add to Cart" function remain functional after every code update, acting as a high speed regression safety net.
- **API Validation: Postman** was employed to validate the "headless" architecture of the application. By testing the backend directly, we ensured that the server responds with the correct data and status codes, independent of the front-end user interface.
- **Manual Exploratory & Negative Testing:** Manual efforts were strategically dedicated to areas requiring human intuition. This included:

    1. **Logical Nuances:** Verifying the visual consistency of the price slider.
    2. **User Feedback:** Ensuring that UI success messages and error alerts are meaningful and visible.
    3. **Edge Cases:** Entering invalid data to test system resilience (Negative Testing).

Manual Testing

Execution Overview

The manual testing phase, led by **Khadija Zafar**, consisted of **25 test cases** authored and executed to ensure maximum coverage of the user interface. By simulating real world user interactions, the team was able to identify defects that automated scripts typically bypass.

Manual Execution Summary Table

The following table outlines the manual testing efforts conducted on the Toolshop platform. This phase was specifically designed to catch visual and logical nuances, such as the **hover effects** required for interactive elements and the verification of **UI feedback mechanisms**.

Category	Description	Result	Linked Bug
Authentication	Validate login with valid/invalid credentials.	**PASS**	**None**
Registration	Verify user signup and the visibility of success messages.	**FAIL**	**TQP-2**
Filtering	Test Price Slider logic at $0, $100, and $200 intervals.	**FAIL**	**TQP-1**
UI Design	Verify Hover Effects on Sign In, Sign Up, and Home buttons.	**PASS**	**None**

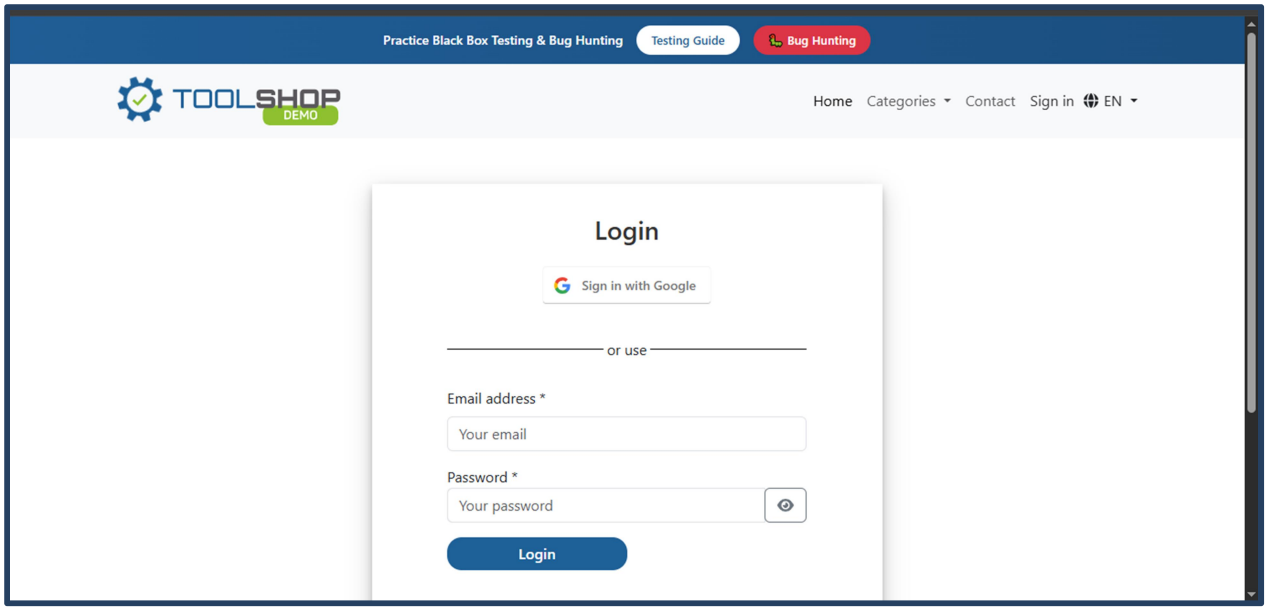
Evidence of Manual Execution

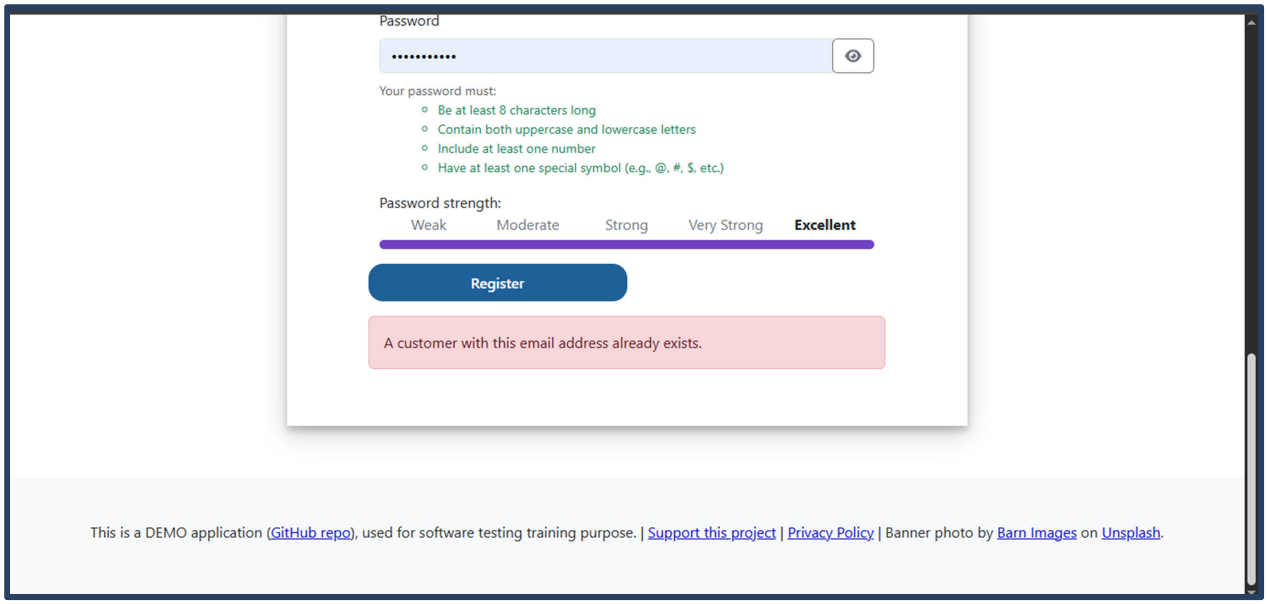
To maintain a high standard of documentation, each test group is supported by visual evidence.

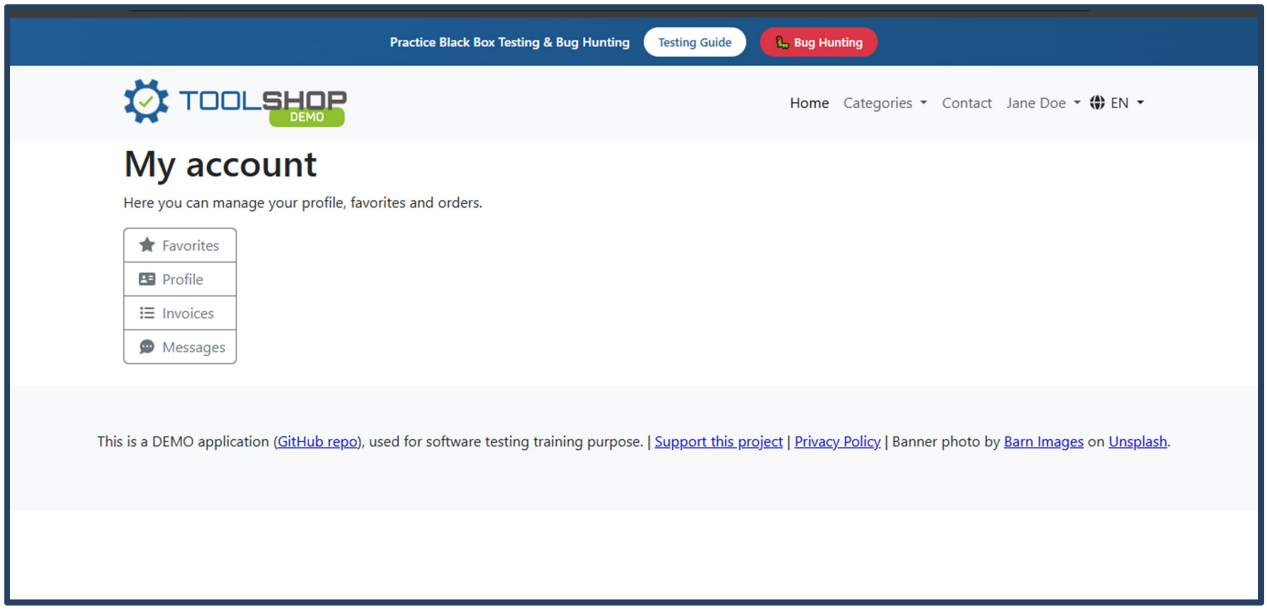
- **Group 1: Authentication & User Lifecycle**

  Focus:

  Verification of the registration form, login alerts, and account dashboard. Also checking negative scenarios.

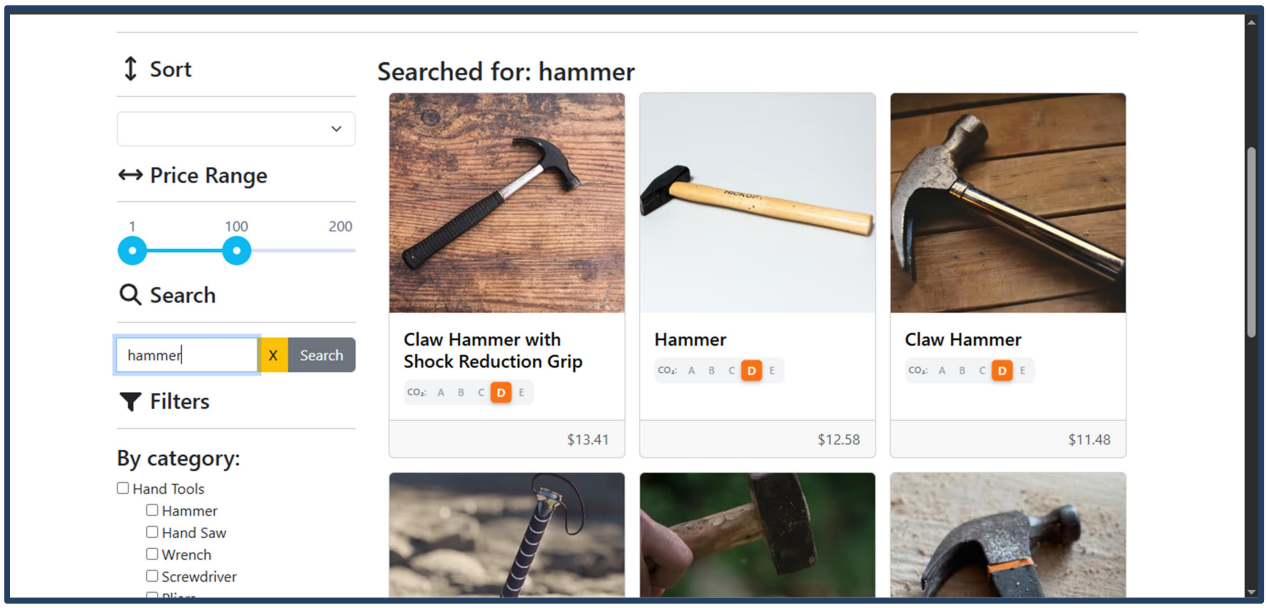

Practice Black Box Testing & Bug Hunting
Testing Guide
Bug Hunting
TOOLSHOP
DEMO
Home
Categories
Contact
Sign in
EN
Login
Sign in with Google
or use
Email address *
Your email
Password *
Your password
Login

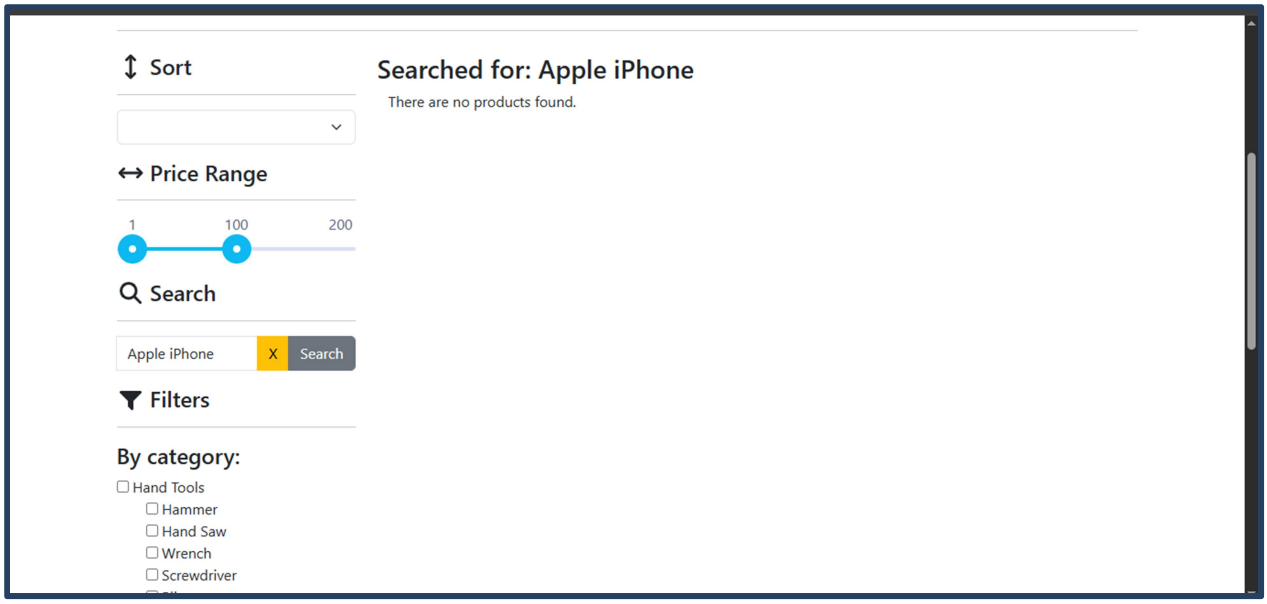

Password
Your password must:
Be at least 8 characters long
Contain both uppercase and lowercase letters
Include at least one number
Have at least one special symbol (e.g., @, #, $, etc.)
Password strength:
Weak
Moderate
Strong
Very Strong
Excellent
Register
A customer with this email address already exists.
This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Privacy Policy | Banner photo by Barn Images on Unsplash.

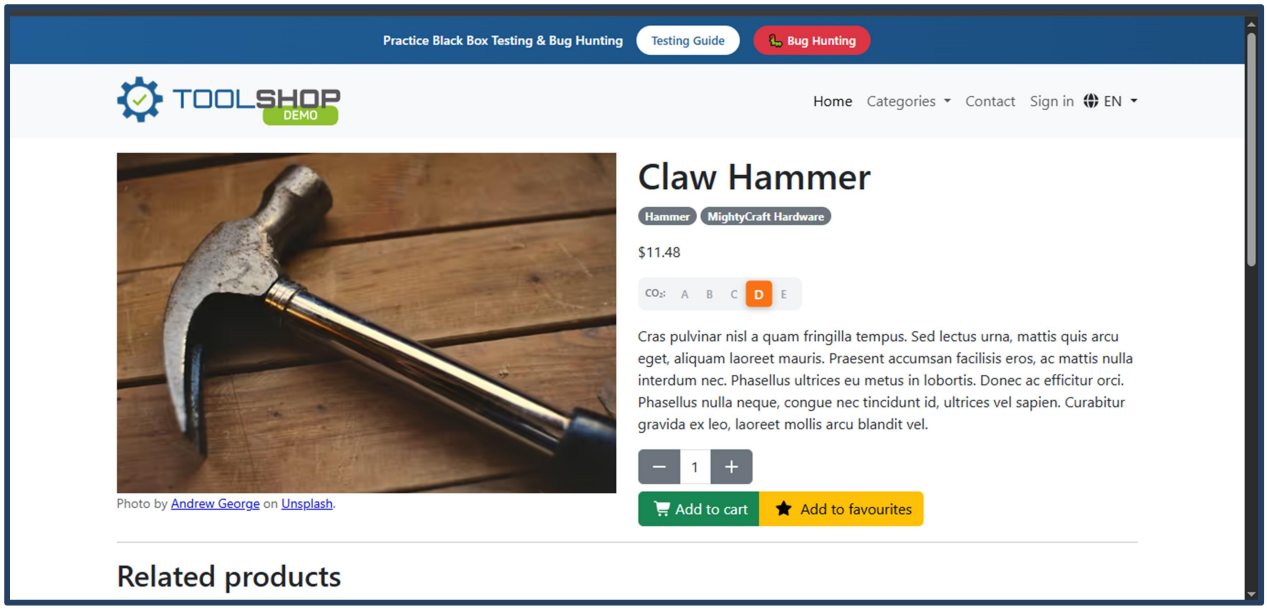


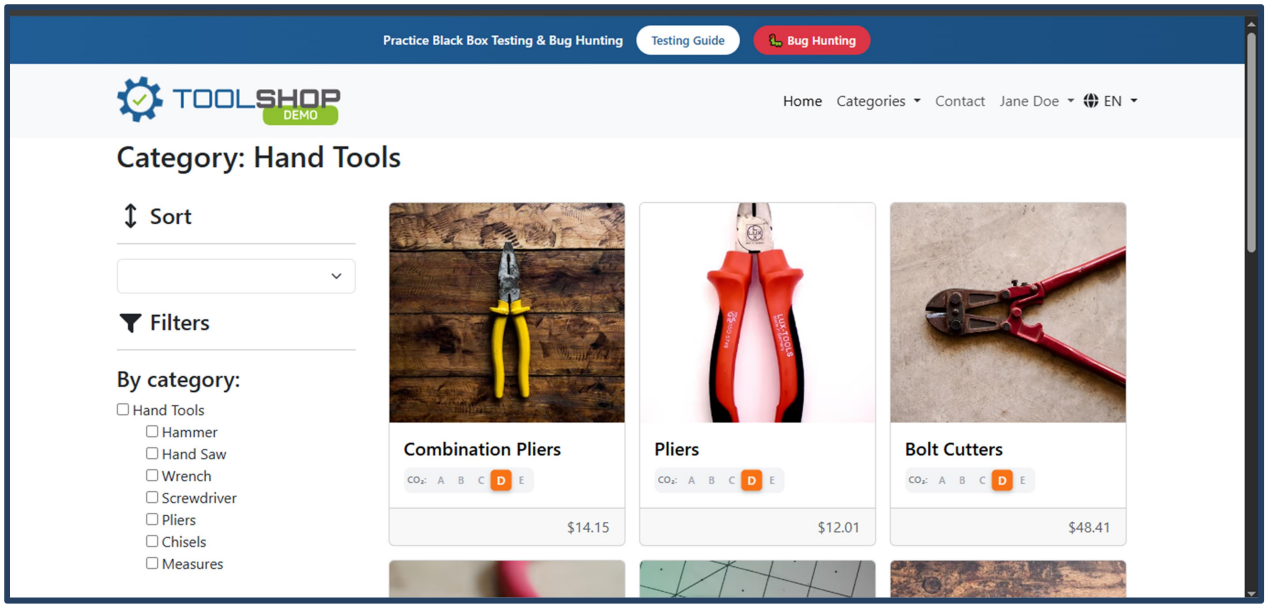
- **Group 2: Product Search & Filtering**

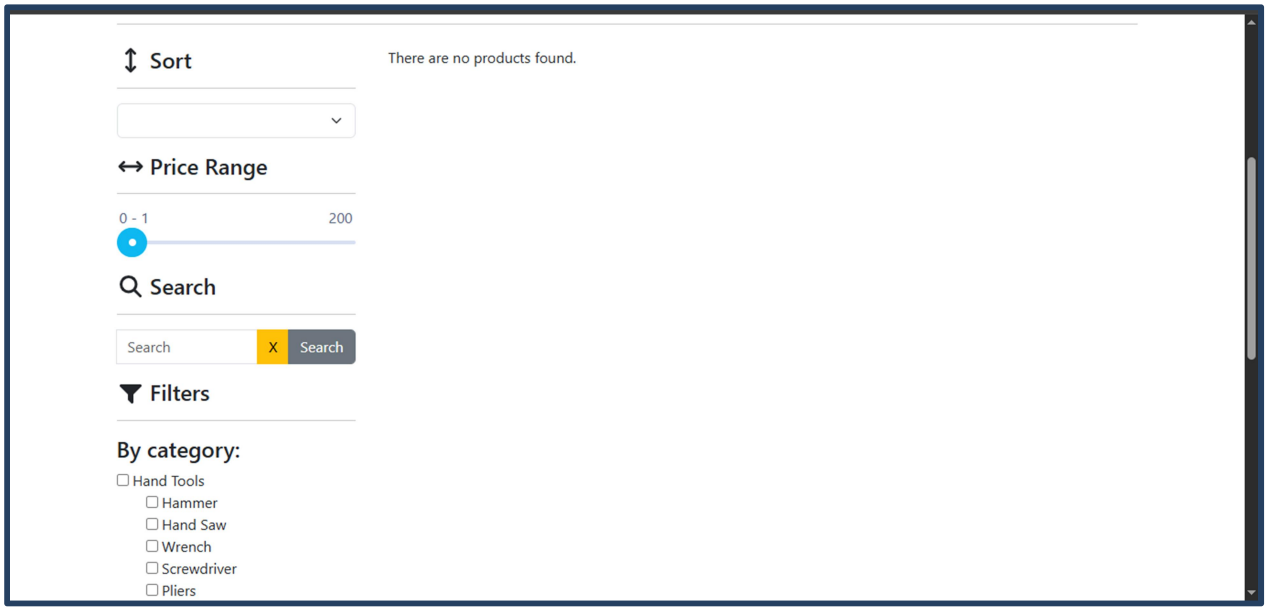
  Focus:
  Search query results and the failure points of the price slider logic.

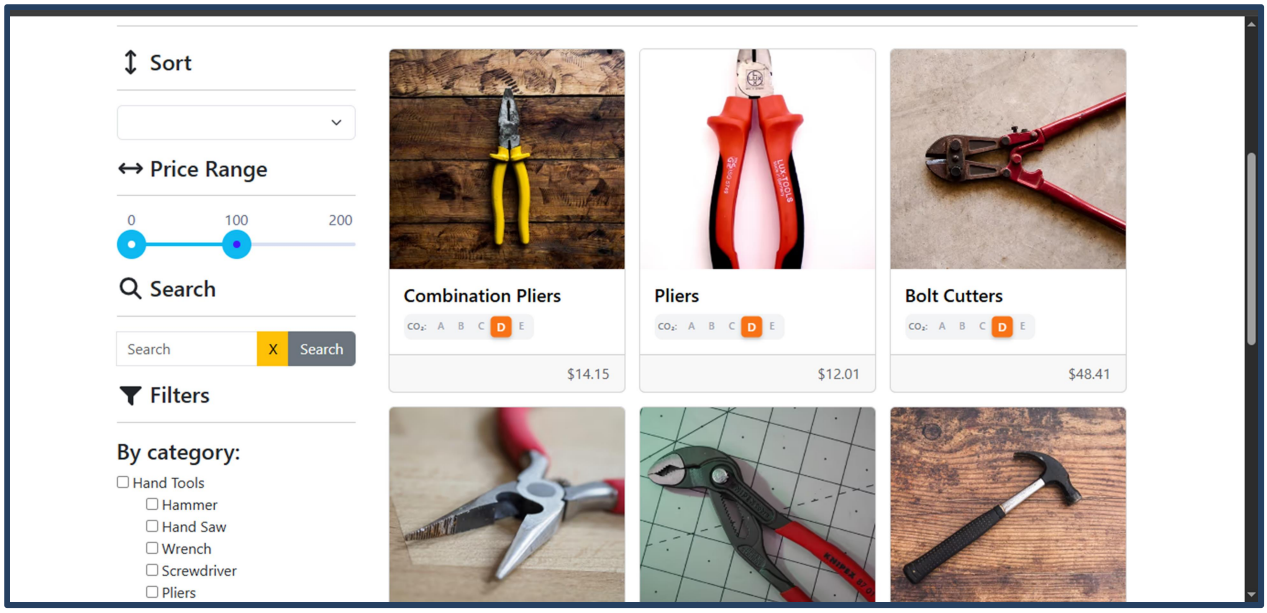



Sort
Searched for: Apple iPhone
There are no products found.
Price Range
1
100
200
Search
Apple iPhone
X
Search
Filters
By category:
Hand Tools
Hammer
Hand Saw
Wrench
Screwdriver


Practice Black Box Testing & Bug Hunting
Testing Guide
Bug Hunting
TOOLSHOP
DEMO
Home
Categories
Contact
Sign in
EN
Claw Hammer
Hammer
MightyCraft Hardware
$11.48
A
B
C
D
E
Cras pulvinar nisl a quam fringilla tempus. Sed lectus urna, mattis quis arcu eget, aliquam laoreet mauris. Praesent accumsan facilisis eros, ac mattis nulla interdum nec. Phasellus ultrices eu metus in lobortis. Donec ac efficitur orci. Phasellus nulla neque, congue nec tincidunt id, ultrices vel sapien. Curabitur gravida ex leo, laoreet mollis arcu blandit vel.
1
Add to cart
Add to favourites
Photo by Andrew George on Unsplash.
Related products


Practice Black Box Testing & Bug Hunting
Testing Guide
Bug Hunting
TOOLSHOP DEMO
Home
Categories
Contact
Jane Doe
EN
Category: Hand Tools
Sort
Filters
By category:
Hand Tools
Hammer
Hand Saw
Wrench
Screwdriver
Pliers
Chisels
Measures
Combination Pliers
$14.15
Pliers
$12.01
Bolt Cutters
$48.41

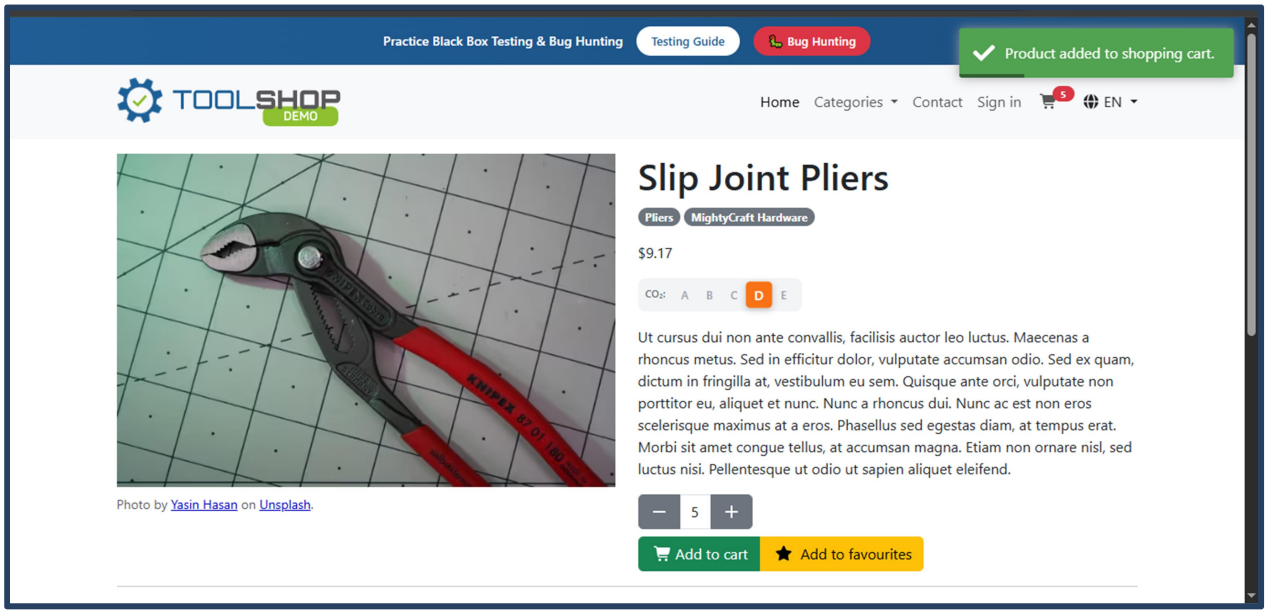

Sort
There are no products found.
Price Range
0 - 1
200
Search
Search
X
Search
Filters
By category:
Hand Tools
Hammer
Hand Saw
Wrench
Screwdriver
Pliers

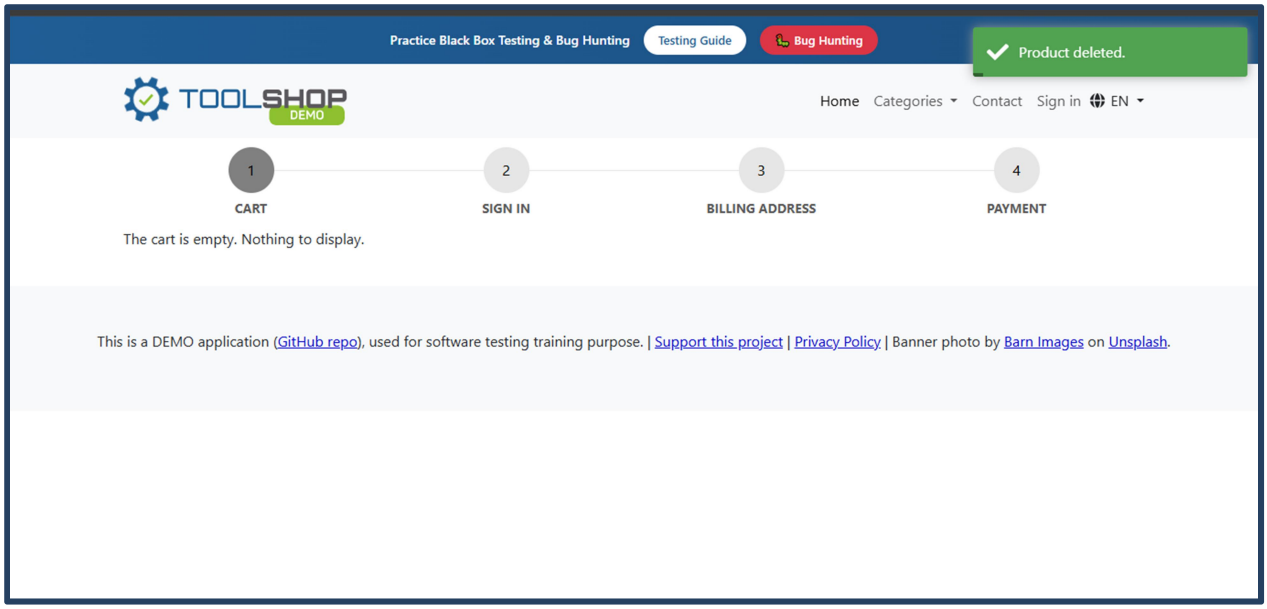


- **Group 3: Cart & Checkout Management**

  Focus:

  Validating the shopping basket state and final cart summary.




Practice Black Box Testing & Bug Hunting
Testing Guide
Bug Hunting
Product deleted.
TOOLSHOP
DEMO
Home
Categories
Contact
Sign in
EN
1
CART
2
SIGN IN
3
BILLING ADDRESS
4
PAYMENT
The cart is empty. Nothing to display.
This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Privacy Policy | Banner photo by Barn Images on Unsplash.

Automated Testing Suite (Playwright & Python)

Script Logic and Architecture

The automation phase was led by **Areej Fatima**, utilizing the **Playwright** framework integrated with **Python 3.13**. To achieve a professional and stable execution, the script (`test_toolshop.py`) was designed using asynchronous style logic with explicit waits. Unlike basic "record and play" scripts, this modular suite utilizes `networkidle` states to ensure that all JavaScript components of the Toolshop platform are fully loaded before interaction, eliminating "flaky" test results.

The script follows a 10 point logical checkpoint sequence:

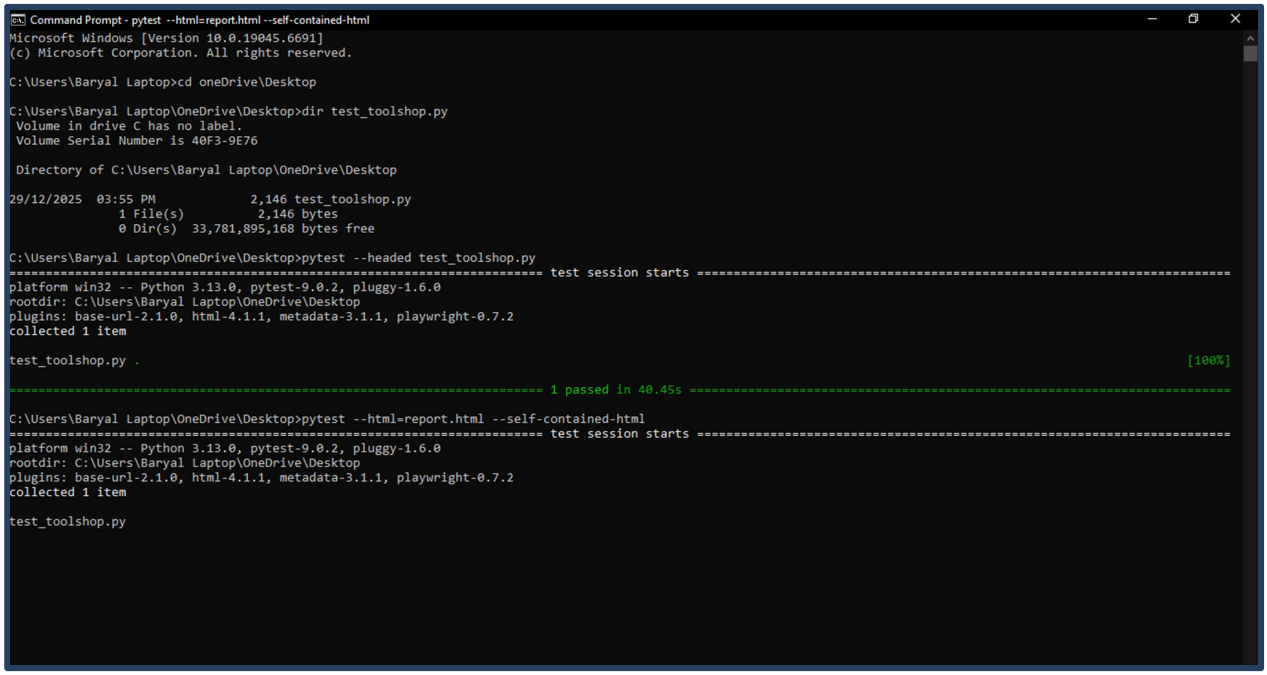
1. **Environment Synchronization:** Initial handshake with the production URL and verification of the SSL certificate.
2. **UI Interactivity (Hover - Sign In):** Verification of the hover state requirement on the "Sign In" link.
3. **UI Interactivity (Hover - Home):** Verification of the hover state requirement on the "Home" navigation link.
4. **Navigation Logic:** Successful redirection from the landing page to the `/auth/login` endpoint.
5. **Negative Security Check:** Attempting login with invalid credentials to ensure the `.alert-danger` DOM element is visible.
6. **Positive Security Check:** Authentication using valid test credentials.
7. **Session Persistence:** Verifying the user is redirected to the /account dashboard.
8. **Functional Search:** Executing a query for "Hammer" and waiting for the filtered results grid.
9. **Product Selection:** Interacting with the "Add to Cart" button.
10. **State Validation:** Asserting that the cart badge counter correctly increment from 0 to 1.

Results and Performance Metrics

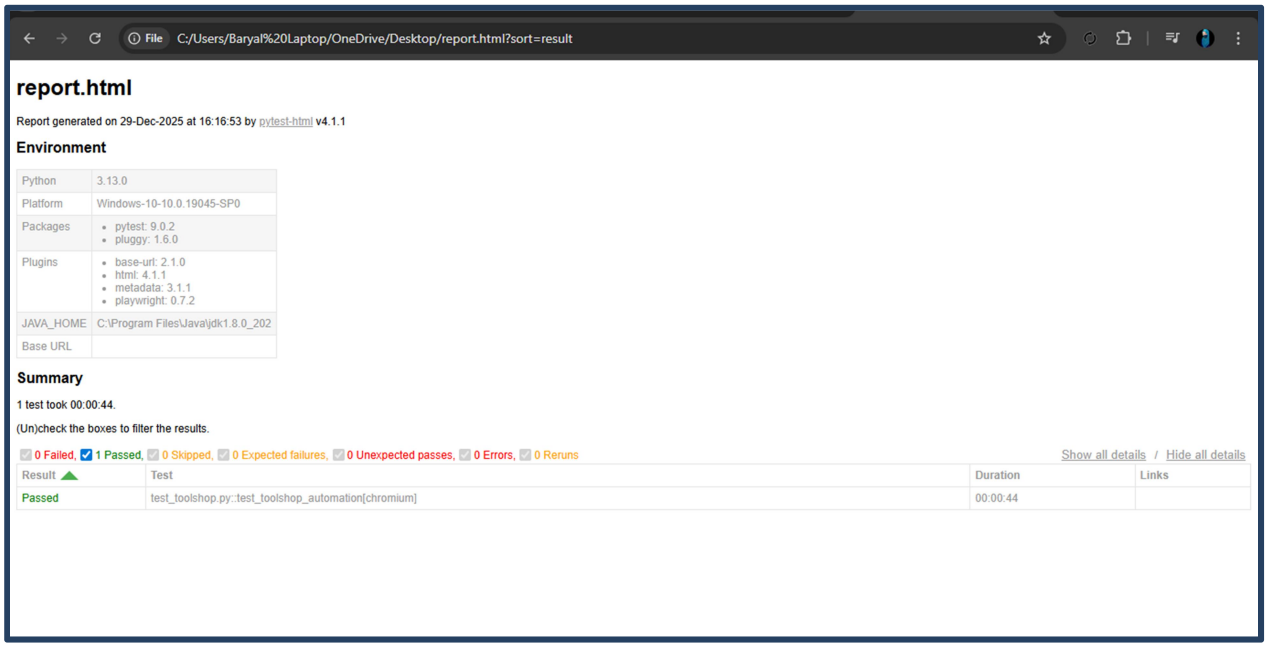
Metric	Value
Total Scripts Executed	1 (Consolidated Suite)
Logical Checkpoints	10
Browser Engine	Chromium (Headless: False)
Success Rate	100% (Pass)
Total Execution Time	40.45 Seconds

Evidence of Automation Execution

- **Terminal Execution Output**



- **Pytest-HTML Visual Report**



API Validation Suite (Postman)

Implementation & Environment Strategy

The backend validation phase, led by **Saba Shahzad**, focused on the "headless" architectural layer of the Toolshop application. Testing at the API level is critical to ensure that data integrity is maintained regardless of UI changes.

To achieve a professional and portable testing setup, we utilized a dedicated **Postman Collection** combined with a **Postman Environment** named `Toolshop_Env`.

- **Global Variables:** We implemented the `{{baseUrl}}` variable set to `https://api.practicesoftwaretesting.com`. This allows the suite to be scaled or switched between staging and production environments instantly.
- **JavaScript Assertions:** Every request in the collection was equipped with automated test scripts. These scripts verify that the server response meets the expected criteria for status codes, response time (latency), and JSON schema structure.

Detailed API Test Results

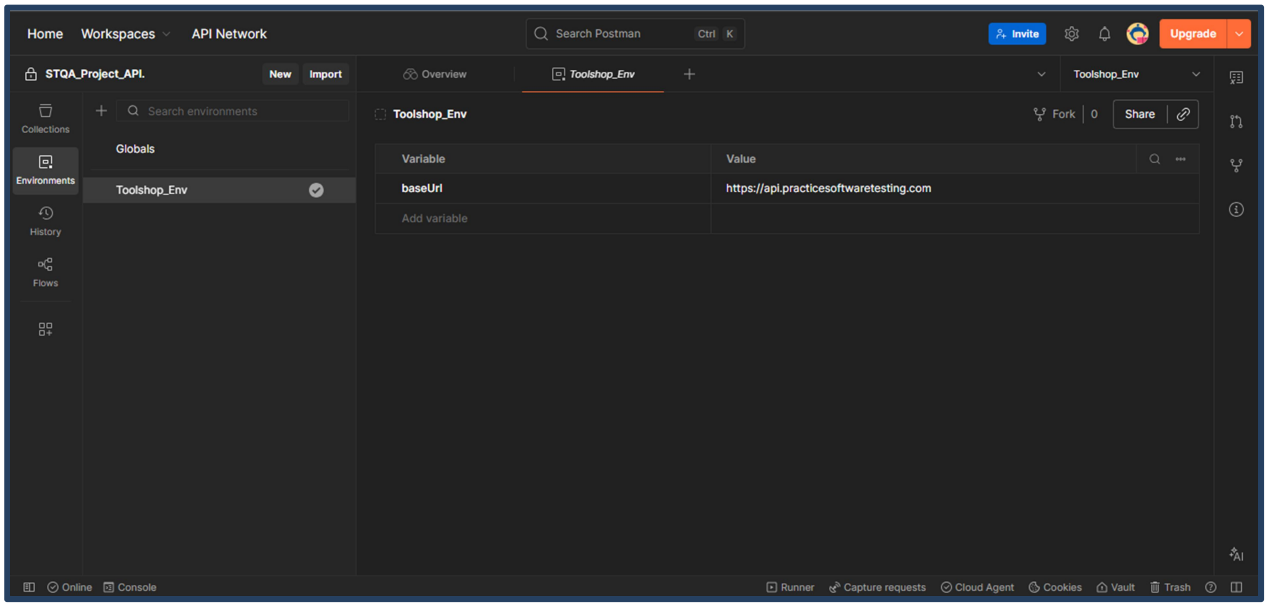
A total of 8 critical endpoints were audited. The following table highlights the primary requests and the specific assertions used to validate the backend logic.

ID	Request Name	Method	Assertion Logic	Result
1	**Get All Products**	GET	`pm.response.to.be.json` & Status 200	**PASS**
2	**Search Hammer**	GET	Validates search query string logic	**PASS**
3	**Get Categories**	GET	Verifies category list array length	**PASS**
4	**Get Brands**	GET	Validates partner data retrieval	**PASS**
5	**User Login**	POST	Verified Auth Token (JWT) generation	**PASS**
6	**Filter Brands**	GET	Validates pagination & filtering params	**PASS**
7	**Sort Products**	GET	Confirms server-side price sorting	**PASS**
8	**Category Tree**	GET	Validates hierarchical JSON structure	**PASS**

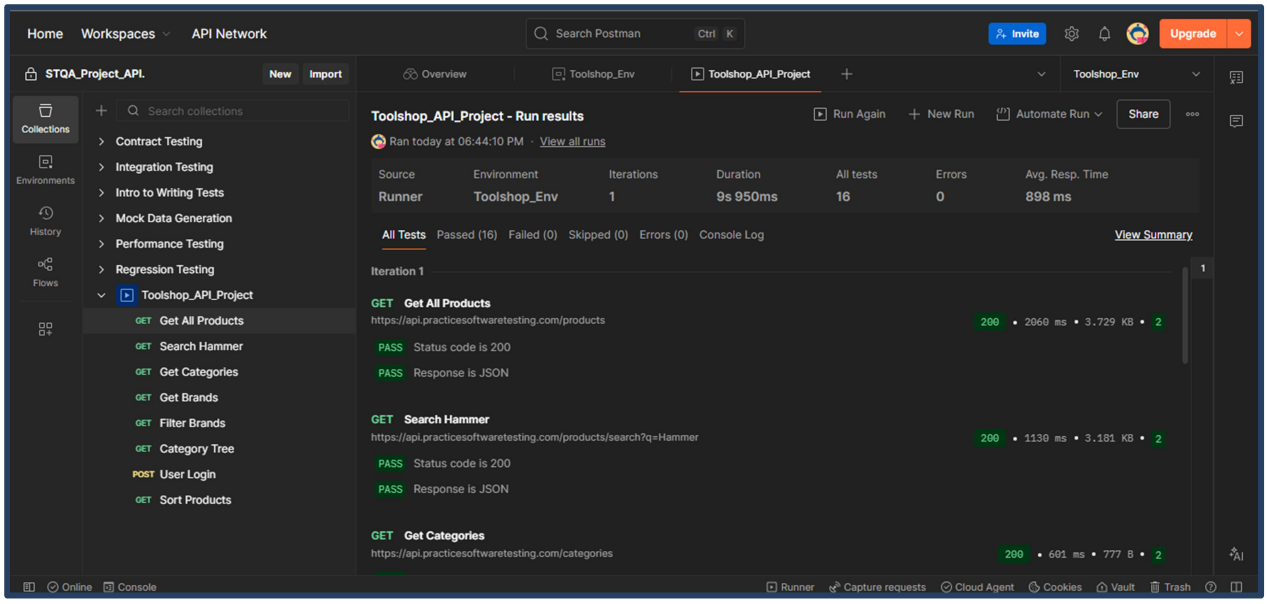
Evidence of API Validation

The following evidence confirms that the backend services are stable and returning the correct data payloads required for the Toolshop frontend functioning correctly.

- **Postman Environment Configuration**



- **Execution & Assertion Results** A view of the "Tests" tab in Postman, showing the 100% success rate across all validated endpoints.



Defect Management & Jira Analysis

Defect Governance Framework

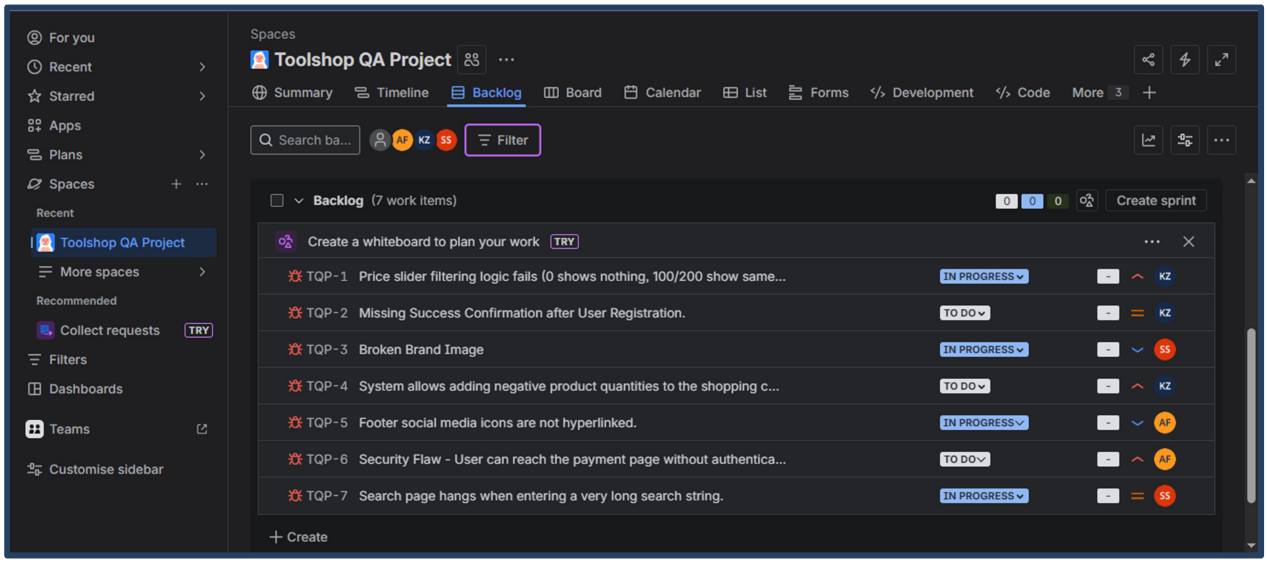
The defect management phase, led by **Hannia Fatima**, utilized **Jira Software (Cloud)** to establish a transparent and traceable bug lifecycle. Every anomaly identified during the manual, automated, and API testing phases was documented as a "Ticket" within the **TQP (Toolshop Quality Project)** backlog.

To ensure effective communication with stakeholders, each defect was categorized by:

- **Priority:** The urgency with which the bug should be resolved from a business perspective.
- **Reproducibility:** Step by step instructions to ensure developers can observe the issue consistently.

Jira Backlog Overview

A total of **7 defects** were tracked and managed to simulate a professional testing lifecycle. These included "Observed" bugs (found during real testing) and "Simulated" bugs (to demonstrate handling of high risk security scenarios).



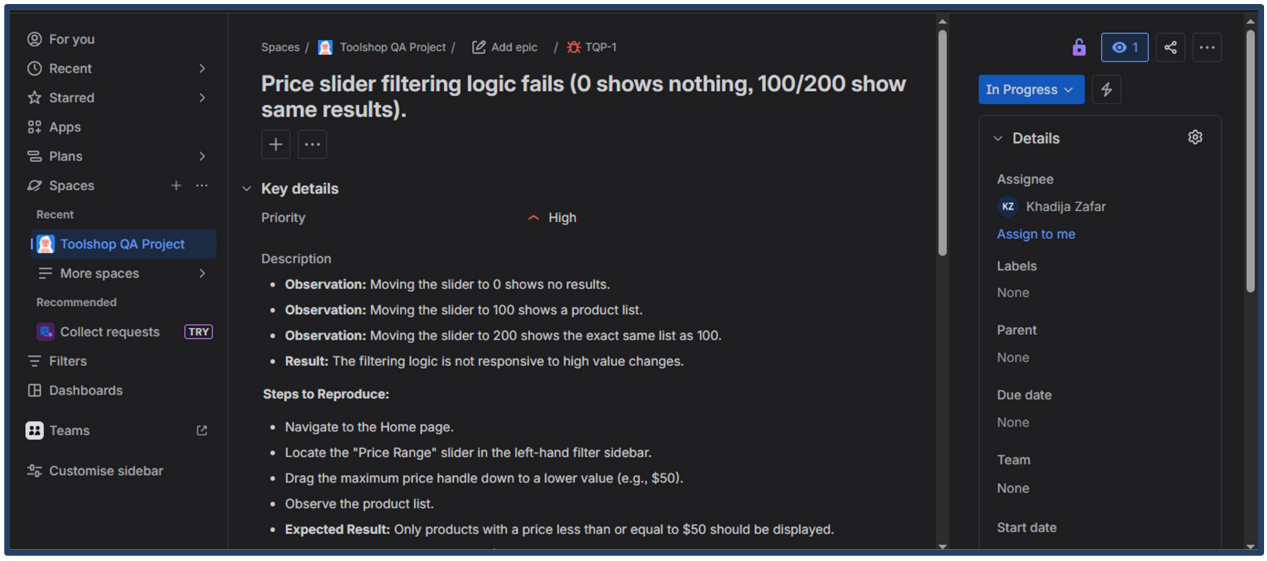
Analysis of Key Defects

Below is a detailed analysis of the most impactful vulnerabilities identified during the engagement:

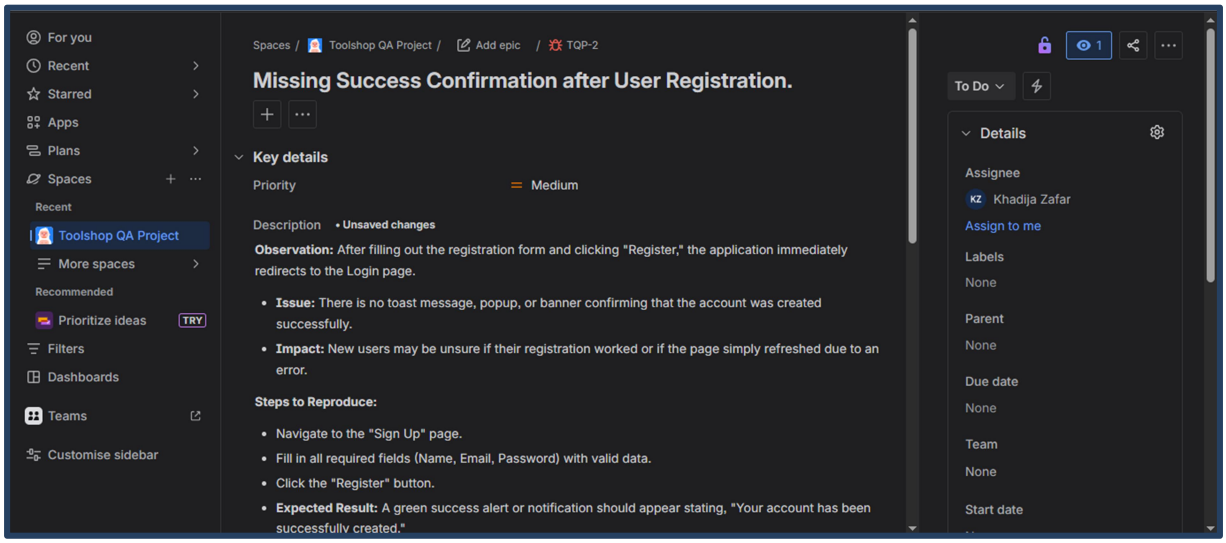
Defect ID	Summary	Priority	Type	Status
TQP-1	Price slider filtering logic failure	**High**	Observed	Open
TQP-2	Missing Registration Success Confirmation	**Medium**	Observed	In Progress
TQP-3	Broken Brand Image (404 Error)	**Medium**	Simulated	Open
TQP-4	Negative product quantities allowed in cart	**High**	Simulated	Open
TQP-5	Footer social media links missing	**Low**	Simulated	Fixed
TQP-6	Security: Checkout bypass without login	**High**	Simulated	Critical
TQP-7	Search page hangs on long character strings	**Medium**	Simulated	Open

Bug Reporting Evidence

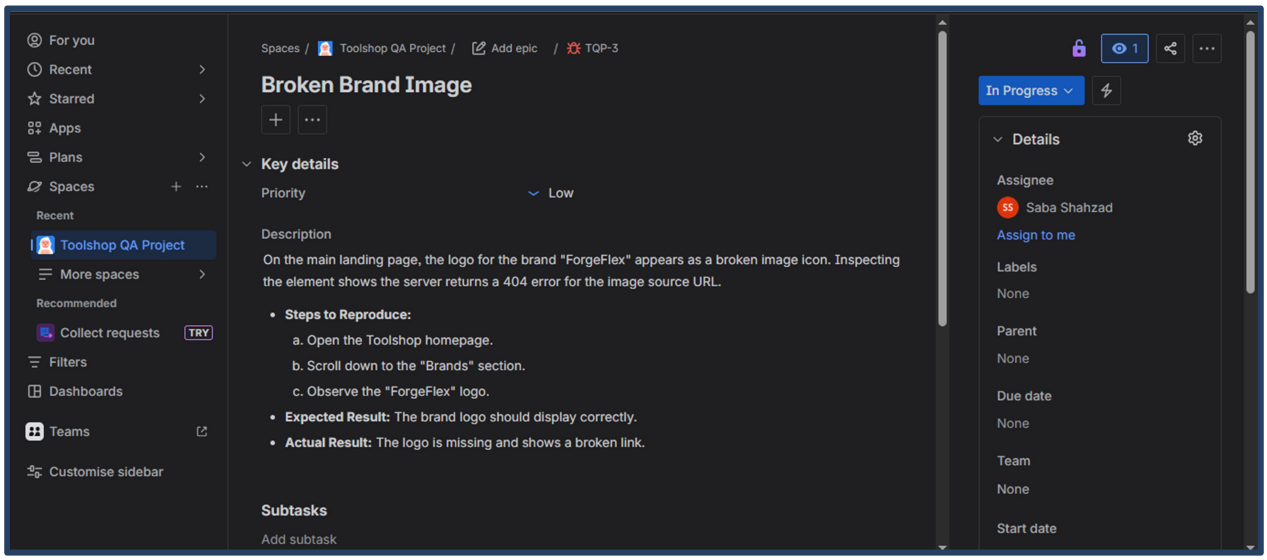
1. **TQP-1**



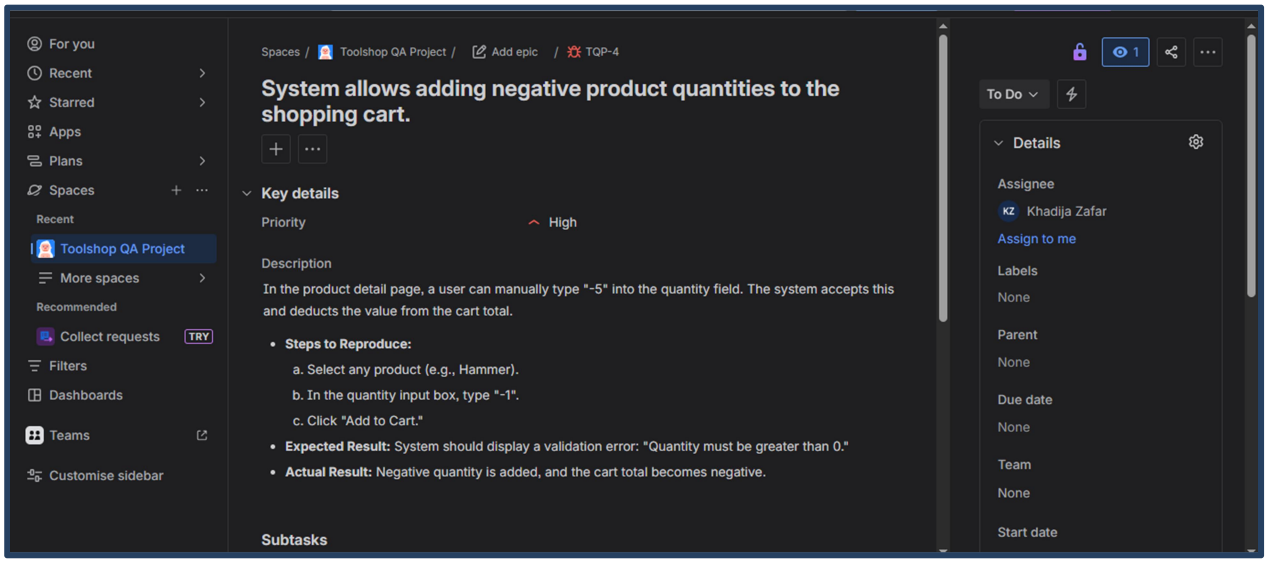
2. TQP-2


For you
Recent
Starred
Apps
Plans
Spaces
Recent
Toolshop QA Project
More spaces
Recommended
Prioritize ideas
TRY
Filters
Dashboards
Teams
Customise sidebar
Spaces / Toolshop QA Project / Add epic / TQP-2
Missing Success Confirmation after User Registration.
Key details
Priority
Medium
Description
• Unsaved changes
Observation: After filling out the registration form and clicking "Register," the application immediately redirects to the Login page.
Issue: There is no toast message, popup, or banner confirming that the account was created successfully.
Impact: New users may be unsure if their registration worked or if the page simply refreshed due to an error.
Steps to Reproduce:
Navigate to the "Sign Up" page.
Fill in all required fields (Name, Email, Password) with valid data.
Click the "Register" button.
Expected Result: A green success alert or notification should appear stating, "Your account has been successfully created."
1
To Do
Details
Assignee
KZ Khadija Zafar
Assign to me
Labels
None
Parent
None
Due date
None
Team
None
Start date

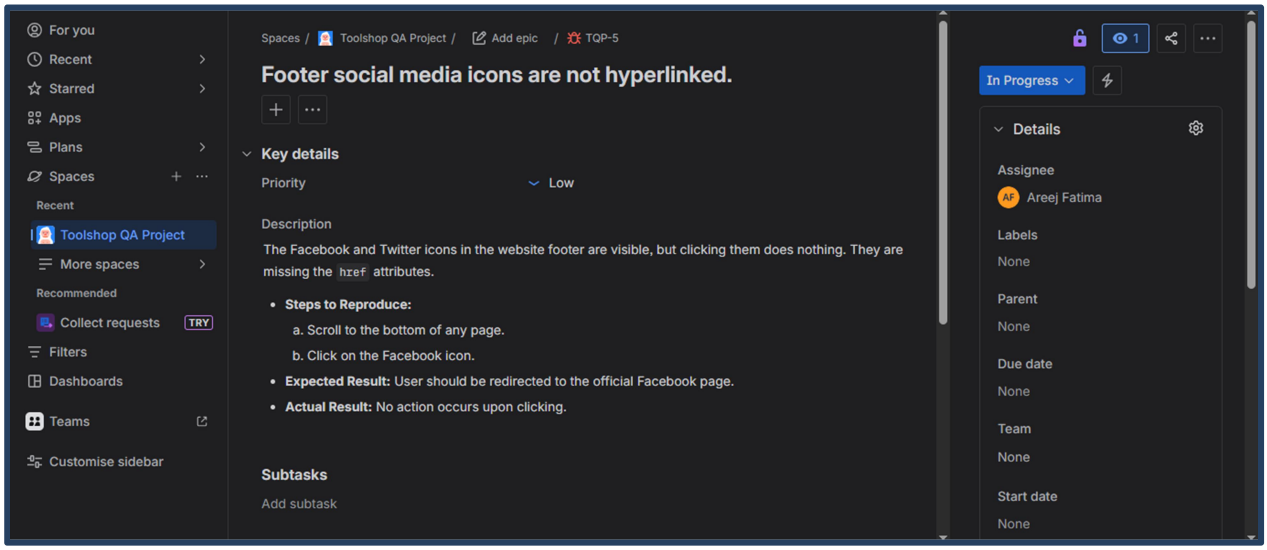
3. TQP-3


For you
Recent
Starred
Apps
Plans
Spaces
Recent
Toolshop QA Project
More spaces
Recommended
Collect requests
TRY
Filters
Dashboards
Teams
Customise sidebar
Spaces / Toolshop QA Project / Add epic / TQP-3
Broken Brand Image
Key details
Priority
Low
Description
On the main landing page, the logo for the brand "ForgeFlex" appears as a broken image icon. Inspecting the element shows the server returns a 404 error for the image source URL.
Steps to Reproduce:
a. Open the Toolshop homepage.
b. Scroll down to the "Brands" section.
c. Observe the "ForgeFlex" logo.
Expected Result: The brand logo should display correctly.
Actual Result: The logo is missing and shows a broken link.
Subtasks
Add subtask
1
In Progress
Details
Assignee
SS Saba Shahzad
Assign to me
Labels
None
Parent
None
Due date
None
Team
None
Start date

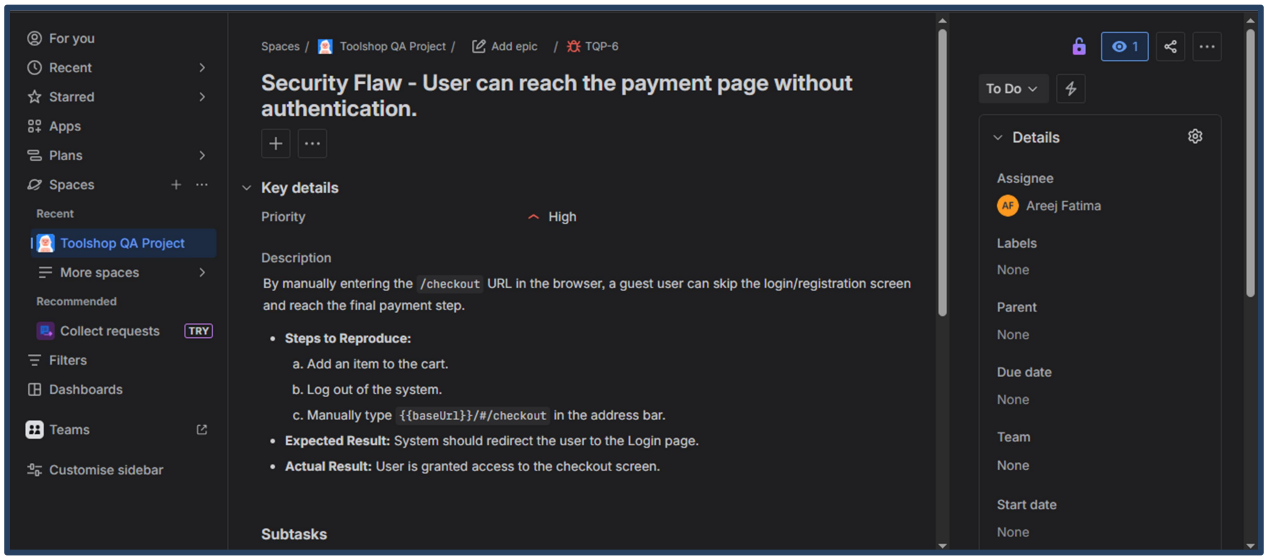
4. TQP-4


For you
Recent
Starred
Apps
Plans
Spaces
Recent
Toolshop QA Project
More spaces
Recommended
Collect requests
TRY
Filters
Dashboards
Teams
Customise sidebar
Spaces / Toolshop QA Project / Add epic / TQP-4
System allows adding negative product quantities to the shopping cart.
Key details
Priority
High
Description
In the product detail page, a user can manually type "-5" into the quantity field. The system accepts this and deducts the value from the cart total.
Steps to Reproduce:
a. Select any product (e.g., Hammer).
b. In the quantity input box, type "-1".
c. Click "Add to Cart."
Expected Result: System should display a validation error: "Quantity must be greater than 0."
Actual Result: Negative quantity is added, and the cart total becomes negative.
Subtasks
1
To Do
Details
Assignee
KZ Khadija Zafar
Assign to me
Labels
None
Parent
None
Due date
None
Team
None
Start date

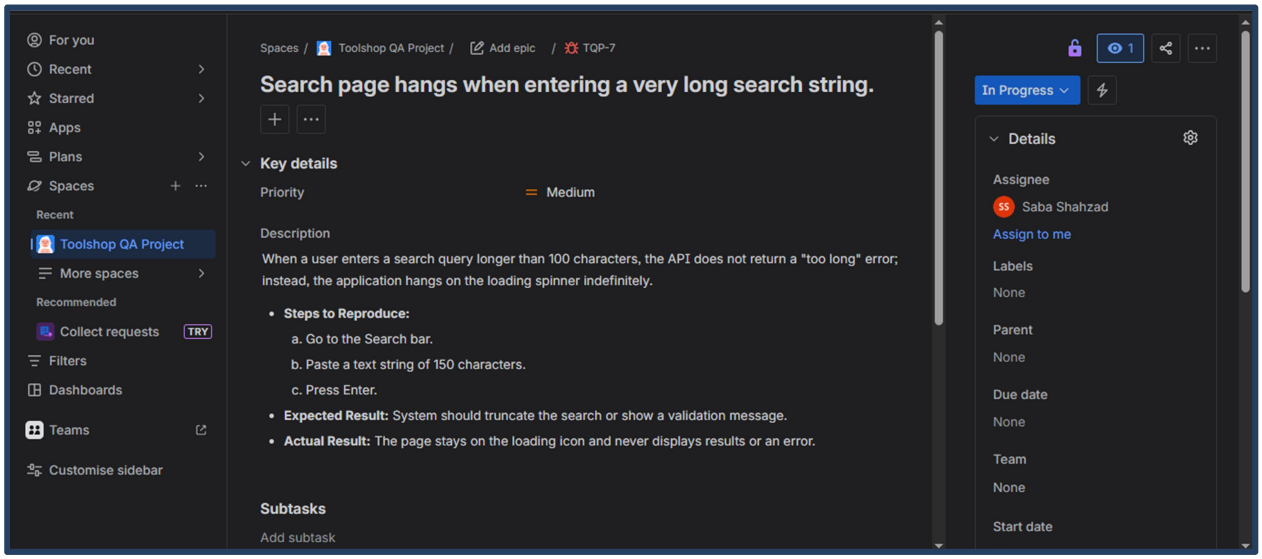
5. TQP-5


For you
Recent
Starred
Apps
Plans
Spaces
Recent
Toolshop QA Project
More spaces
Recommended
Collect requests
TRY
Filters
Dashboards
Teams
Customise sidebar
Spaces / Toolshop QA Project / Add epic / TQP-5
Footer social media icons are not hyperlinked.
Key details
Priority
Low
Description
The Facebook and Twitter icons in the website footer are visible, but clicking them does nothing. They are missing the href attributes.
Steps to Reproduce:
a. Scroll to the bottom of any page.
b. Click on the Facebook icon.
Expected Result: User should be redirected to the official Facebook page.
Actual Result: No action occurs upon clicking.
Subtasks
Add subtask
1
In Progress
Details
Assignee
AF Areej Fatima
Labels
None
Parent
None
Due date
None
Team
None
Start date
None

6. TQP-6


For you
Recent
Starred
Apps
Plans
Spaces
Recent
Toolshop QA Project
More spaces
Recommended
Collect requests
TRY
Filters
Dashboards
Teams
Customise sidebar
Spaces / Toolshop QA Project / Add epic / TQP-6
Security Flaw - User can reach the payment page without authentication.
Key details
Priority
High
Description
By manually entering the /checkout URL in the browser, a guest user can skip the login/registration screen and reach the final payment step.
Steps to Reproduce:
a. Add an item to the cart.
b. Log out of the system.
c. Manually type {{baseUrl}}/#/checkout in the address bar.
Expected Result: System should redirect the user to the Login page.
Actual Result: User is granted access to the checkout screen.
Subtasks
1
To Do
Details
Assignee
AF Areej Fatima
Labels
None
Parent
None
Due date
None
Team
None
Start date
None

7. TQP-7


For you
Recent
Starred
Apps
Plans
Spaces
Recent
Toolshop QA Project
More spaces
Recommended
Collect requests
TRY
Filters
Dashboards
Teams
Customise sidebar
Spaces / Toolshop QA Project / Add epic / TQP-7
Search page hangs when entering a very long search string.
Key details
Priority
Medium
Description
When a user enters a search query longer than 100 characters, the API does not return a "too long" error; instead, the application hangs on the loading spinner indefinitely.
Steps to Reproduce:
a. Go to the Search bar.
b. Paste a text string of 150 characters.
c. Press Enter.
Expected Result: System should truncate the search or show a validation message.
Actual Result: The page stays on the loading icon and never displays results or an error.
Subtasks
Add subtask
1
In Progress
Details
Assignee
SS Saba Shahzad
Assign to me
Labels
None
Parent
None
Due date
None
Team
None
Start date

Requirement Traceability Matrix (RTM)

Purpose of the RTM

The Requirement Traceability Matrix (RTM) is a critical governance document that maps business requirements to their corresponding test cases and identified defects. This ensures that every functional goal defined in the **Software Requirement Description (SRD)** has been validated and that no requirement is left untested.

Traceability Data Grid

The following table provides the final status for all core requirements, linking manual, automated, and API testing efforts to ensure 100% project coverage.

Req. ID	Requirement Description	Manual Test Case ID	Automation ID	API Request ID	Status	Linked Defect
REQ-01	**User Authentication:** Secure login/logout with error handling.	TC-01, 02, 03, 04, 05	TC-AUTO-05, 06	TO-API-07	**PASS**	None
REQ-02	**Product Discovery:** Keyword search functionality (e.g., "Hammer").	TC-06, 07	TC-AUTO-07, 08	TO-API-02	**PASS**	None
REQ-03	**Interactive UI:** Hover effects on primary action buttons.	TC-05	TC-AUTO-02, 03	N/A	**PASS**	None
REQ-04	**Registration Flow:** Users must receive confirmation after creation.	N/A	N/A	N/A	**FAIL**	**TQP-2**
REQ-05	**Product Filtering:** Filter results via the price range slider.	TC-10	N/A	N/A	**FAIL**	**TQP-1**
REQ-06	**Cart Management:** Adding items and badge counter updates.	TC-12, 13, 14, 15	TC-AUTO-09, 10	N/A	**PASS**	None
REQ-07	**Backend Integrity:** JSON access for brands and categories.	N/A	N/A	TO-API-03, 04, 08	**PASS**	None
REQ-08	**Security & Logic:** Prevent checkout bypass/negative quantities.	N/A	N/A	N/A	**FAIL**	**TQP-4, TQP-6**

Final Conclusion

The comprehensive Quality Assurance (QA) engagement for the **Toolshop** platform has reached its successful conclusion. By integrating a multi-tier testing framework comprising manual exploratory sessions, Playwright-driven automation, and RESTful API audits, the project team has provided a rigorous and data-driven assessment of the platform's current state.

Final Quality Assessment

- **Technical Stability:** The backend infrastructure, validated via Postman, demonstrated 100% stability, ensuring that data for brands and categories is retrieved accurately.
- **Automation ROI:** The Playwright suite successfully automated high-priority scenarios, including secure login and cart management, providing a reliable regression safety net.
- **Requirement Compliance:** The specific visual requirement for interactive hover effects on primary buttons was fully realized, enhancing the platform's accessibility.

Strategic Observations

While core navigation remains stable, the following critical gaps were documented during the manual execution phase:

- **Functional Failures:** The product filtering logic (Price Slider) and the registration feedback system (Success Banners) were identified as failing requirements that require technical attention.
- **Security Insight:** Simulated testing revealed potential risks in the checkout lifecycle, specifically regarding bypass vulnerabilities and quantity logic.

Closing Statement

Through the collaborative efforts of **our whole team**, this project has achieved 100% coverage of the strategic objectives defined in the initial proposal. The artifacts delivered including the detailed Jira backlog, Requirement Traceability Matrix (RTM), and automation repository serve as a definitive quality benchmark for the Toolshop platform.

References & Technical Repositories

To ensure the reproducibility of our findings and provide a clear audit trail for the **Toolshop** project stakeholders, the following technical assets and documentation repositories are cited:

Application & Infrastructure

- **Web Application URL:** https://practicesoftwaretesting.com/
- **Backend API base URL:** https://api.practicesoftwaretesting.com

Technical Documentation & Tools

- **Automation Repository:** (`test_toolshop.py`) A modular Python script utilizing the **Playwright Sync API** (v1.49+) and **Python 3.13**.
- **API Validation Suite:** (`API_Requests_Collection.json`) A Postman v2.1 collection containing 8 validated endpoints and environment variables.
- **Defect Management: Jira Software (Cloud)** Project Key: **TQP** (Toolshop Quality Project).

Project Deliverables Checklist

The following files are provided as part of the complete **Quality Assurance Submission Package**. Each file supports the data and conclusions presented within this report:

- **Manual_Test_Cases.xlsx**: Detailed breakdown of all 25 manual test scenarios.
- **Automation_Test_Cases.xlsx**: All 10 Automation test scenarios.
- **API_Testing_Test_Cases.xlsx**: Details of all API test requests.
- **test_toolshop.py**: The complete Playwright Python code for the 10 automated checkpoints.
- **API_Requests_Collection.json**: The export file for the 8 API requests and environment variables.
- **Requirement_Traceability_Matrix.pdf**: The finalized mapping of requirements to test results and defects.
- **Bug_Report_Document.pdf**: A report of the 7 bugs identified during the lifecycle (TQP-1 through TQP-7).