University of Science – Vietnam National University
Faculty of Information Technology

CS423 – Software Testing
HW02 Report
Domain Testing

Student's Name	Student ID	Class
Tran Nhat Thanh	22125093	22TT

Ho Chi Minh City, November 4th, 2025

Contents

1. Glossary of Terms

- **SUT:** Software Under Test (the Toolshop application)
- **TC:** Test Case (A specific set of steps to validate a function)
- **EP:** Equivalence Partitioning (The technique of dividing inputs into groups)
- **VP:** Valid Partition (An equivalence class of *valid* inputs)
- **IP:** Invalid Partition (An equivalence class of *invalid* inputs)
- **BVA:** Boundary Value Analysis (The technique of testing the "edges" of a partition)

2. The Step-by-Step Methodology

My testing process followed the formal methodology presented in the "Domain Testing" slides. This 5-step process was applied to each of the three features.

Step 1: Feature & Use Case Selection

First, I selected three high-priority features that are critical to the application's success and represented each of the main user roles:

1. **Guest: Search & Sort & Filter**
2. **User: Checkout Flow**
3. **Admin: Add/Edit Product**

These features were chosen because their Use Cases are rich with user inputs, making them ideal for Domain Testing.

Step 2: Input & Domain Identification

For each feature, I performed a Use Case Analysis to identify all possible user inputs. Each input represents a "domain" to be tested.
For example, in the "Add Product" feature, the inputs are:

- Name (Text Field)
- Description (Text Area)
- Stock (Number Field)
- Price (Number Field)
- Brand (Dropdown)
- ...and so on.

For each input, I identified its specific rules (the "domain") from the application's behavior and specifications (Stock must be an integer, Price must be a decimal, Name is mandatory, etc.).

Step 3: Equivalence Partitioning (EP)

Once the domains were defined, I applied **Equivalence Partitioning (EP)**. I divided the domain for each input into a set of Valid Partitions (VP) and Invalid Partitions (IP).

Example: For the **Search** field (3-40 char rule), I identified partitions based on their characteristics, just as seen in the test case tables:

- **VP: Valid, Exists:** Strings with a valid length (3-40 chars) that exist in the database ("Pliers").
- **VP: Valid, Not-Exists:** Strings with a valid length (3-40 chars) that do *not* exist ("Spaceship").
- **IP: < 3 chars:** Strings that are too short.
- **IP: > 40 chars:** Strings that are too long.
- **IP: Special Chars:** Strings that contain special characters ("Hammer#").

Step 4: Boundary Value Analysis (BVA)

For domains with numerical or length-based rules, I applied **Boundary Value Analysis (BVA)**. The principle is that errors are most likely to occur at the "edges" of a partition.
I designed test cases to test:

- On the boundary (3 chars)
- Just below the boundary (2 chars)
- Just above the boundary (4 chars)

Example: For the **Search** field (3-40 char rule), my BVA tests Ire:

- **Min Boundary:** 2 (Invalid), 3 (Valid), 4 (Valid)
- **Max Boundary:** 39 (Valid), 40 (Valid), 41 (Invalid)

Step 5: Test Case Creation

Finally, I created two distinct sets of test cases for each feature, as seen in my test-tracking Excel file:

1. **EP Test Cases:** A minimum set of tests designed to cover every single VP and IP at least once. From the slides, I tested invalid partitions one at a time to prevent one error from masking another.
2. **BVA Test Cases:** A specific set of tests targeting the boundaries (Min, Min+1, Max, Max-1, etc.) of all numerical and length-based fields.

This two-table approach ensures achieving both broad (EP) and deep (BVA) test coverage.

3. Application - Feature 1: Guest: Search & Sort & Filter

a. Use Case & Inputs

- **Use Case:** A Guest user (or any user) filters, sorts and searches the product list to find specific items.
- **Inputs Identified:**
    - Search (Text Field)
    - Price Range (Slider)
    - Sort (Dropdown)

  - Category (Checkboxes)
  - Brand (Checkboxes)
  - Sustainability (Checkbox)

b. Partition & Boundary Analysis

- **Search:** Domain was identified as 3-40 characters. BVA was applied to 2, 3, 4 and 39, 40, 41.
- **Price:** Domain was a range from $1 to $200. BVA was applied to the boundaries ($1-$1, $200-$200, $1-$2, etc.).
- **Sort, Category, Brand:** These are sets of values. EP was applied by creating one test case for each valid option (one test for "Sort by Name A-Z", one for "Filter by Hammer", etc.).

c. Final Test Case Summary

A total of **31 test cases** were designed for this feature.

Equivalence Partitioning (EP) Test Cases (19 TCs)

- **SSF-EP-01:** Valid Search (Existing Term) - **Result: Passed**
- **SSF-EP-02:** Valid Search (Non-Existing Term) - **Result: Passed**
- **SSF-EP-03:** Invalid Search (Too Short) - **Result: Failed**
- **SSF-EP-04:** Invalid Search (Too Long) - **Result: Failed**
- **SSF-EP-05:** Invalid Search (Special Chars) - **Result: Failed**
- **SSF-EP-06:** Valid Price Filter (Nominal) - **Result: Failed**
- **SSF-EP-07:** Sort: Name (A - Z) - **Result: Failed**
- **SSF-EP-08:** Sort: Name (Z - A) - **Result: Failed**
- **SSF-EP-09:** Sort: Price (High - Low) - **Result: Failed**
- **SSF-EP-10:** Sort: Price (Low - High) - **Result: Failed**
- **SSF-EP-11:** Sort: CO2 (Best First) - **Result: Failed**
- **SSF-EP-12:** Sort: CO2 (Worst First) - **Result: Failed**
- **SSF-EP-13:** Filter: Single Category - **Result: Passed**
- **SSF-EP-14:** Filter: Multiple Categories - **Result: Passed**
- **SSF-EP-15:** Filter: Single Brand - **Result: Passed**
- **SSF-EP-16:** Filter: Multiple Brands - **Result: Passed**
- **SSF-EP-17:** Filter: Sustainability - **Result: Passed**
- **SSF-EP-18:** Combined (Valid) - **Result: Failed**
- **SSF-EP-19:** Combined (No Results) - **Result: Passed**

Boundary Value Analysis (BVA) Test Cases (12 TCs)

- **SSF-BVA-01:** Search Length (Min - 1) - **Result: Failed**
- **SSF-BVA-02:** Search Length (Min) - **Result: Passed**
- **SSF-BVA-03:** Search Length (Min + 1) - **Result: Passed**
- **SSF-BVA-04:** Search Length (Max - 1) - **Result: Passed**
- **SSF-BVA-05:** Search Length (Max) - **Result: Passed**
- **SSF-BVA-06:** Search Length (Max + 1) - **Result: Failed**

- **SSF-BVA-07:** Price Filter (Min Range) - **Result: Failed**
- **SSF-BVA-08:** Price Filter (Max Range) - **Result: Failed**
- **SSF-BVA-09:** Price Filter (Min + 1) - **Result: Failed**
- **SSF-BVA-10:** Price Filter (Max - 1) - **Result: Failed**
- **SSF-BVA-11:** Price Filter (Full Range) - **Result: Failed**
- **SSF-BVA-12:** Price Filter (Nominal) - **Result: Failed**

d. Bug Summary

The 19 failed test cases point to 4 major bugs:

1. **Bug 1 (Serious): Price Filter is Non-Functional**
    - **Description:** The Price Range slider does not filter the product list. The list continues to show all products regardless of the selected price range.
    - **Failed Tests:** SSF-EP-06, SSF-BVA-07, SSF-BVA-08, SSF-BVA-09, SSF-BVA-10, SSF-BVA-11, SSF-BVA-12 *(7 TCs)*

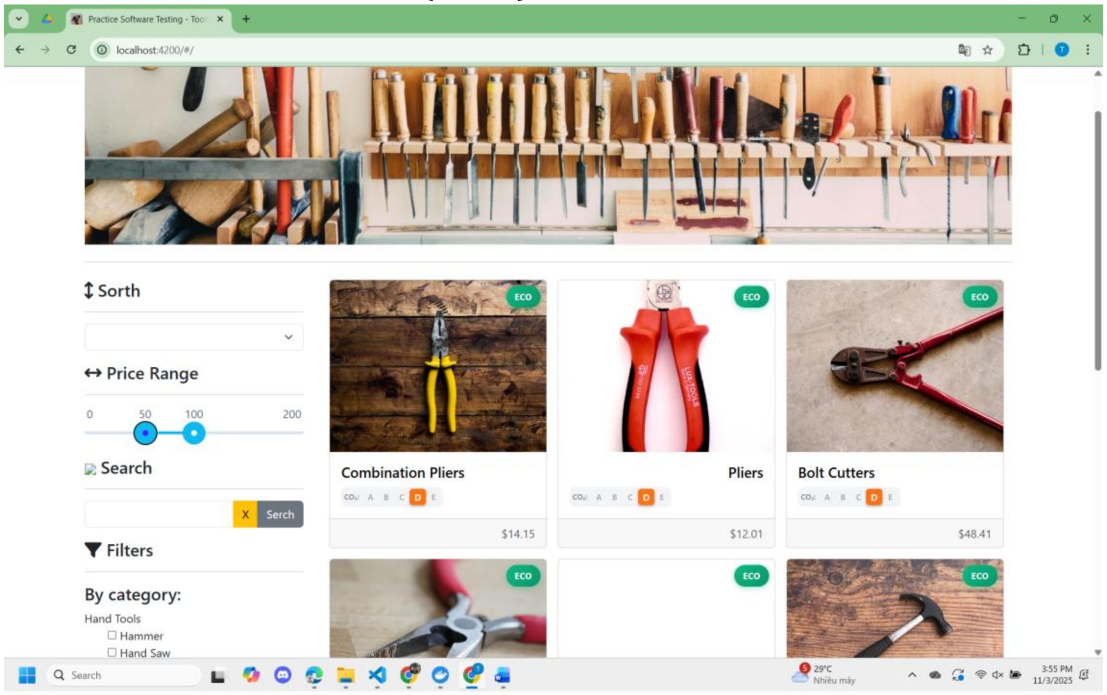


*Figure 1: The price filter is set to a range of **50 to 100**, but the results incorrectly display products priced outside this range.*

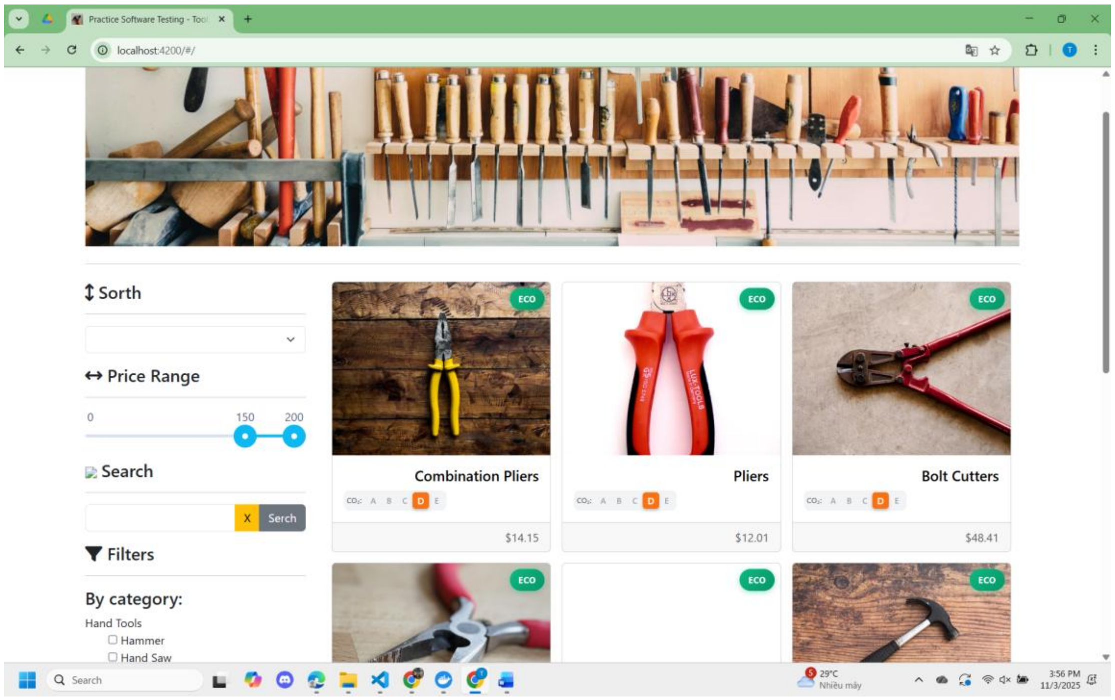


*Figure 2: The filter is adjusted to **150 to 200**, yet the product list remains unchanged, failing to filter out the same items that are all priced below this new range.*

2. **Bug 2 (Serious): Search Validation is Broken**
    - **Description:** The search box fails to validate input. Queries that are too short (< 3 chars), too long (> 40 chars), or contain special characters are processed instead of being blocked with an error message.
    - **Failed Tests:** SSF-EP-03, SSF-EP-04, SSF-EP-05, SSF-BVA-01, SSF-BVA-06 *(5 TCs)*

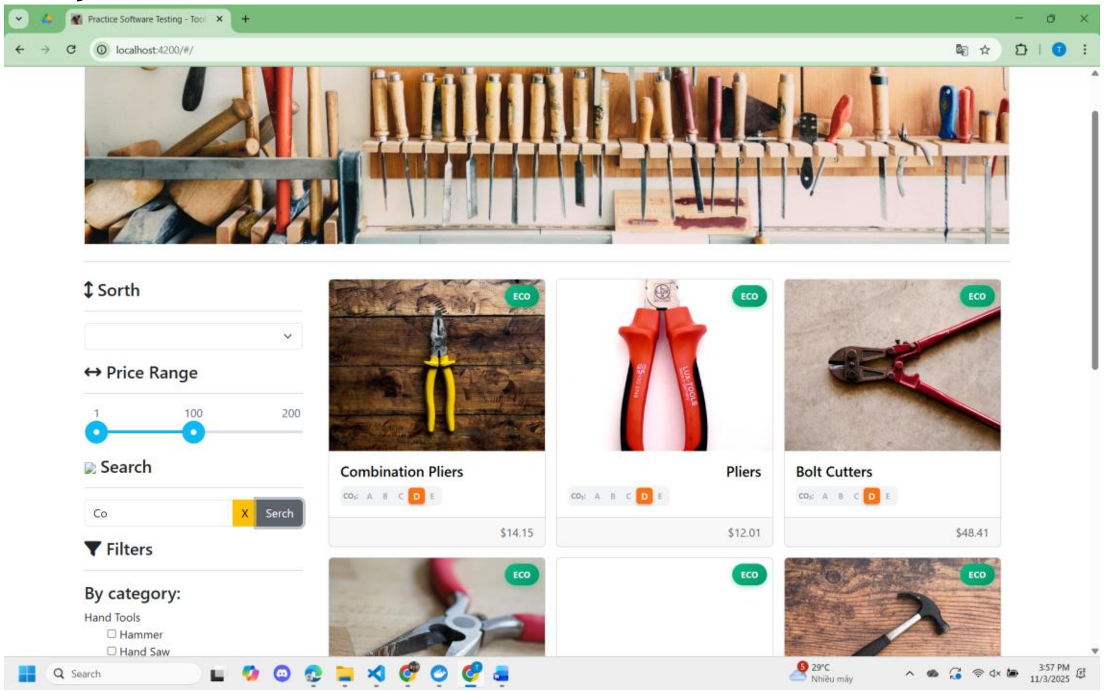


Figure 3: An invalid 2-character query ("Co") is accepted without triggering the expected validation error message or feedback, leaving the user on the same screen.

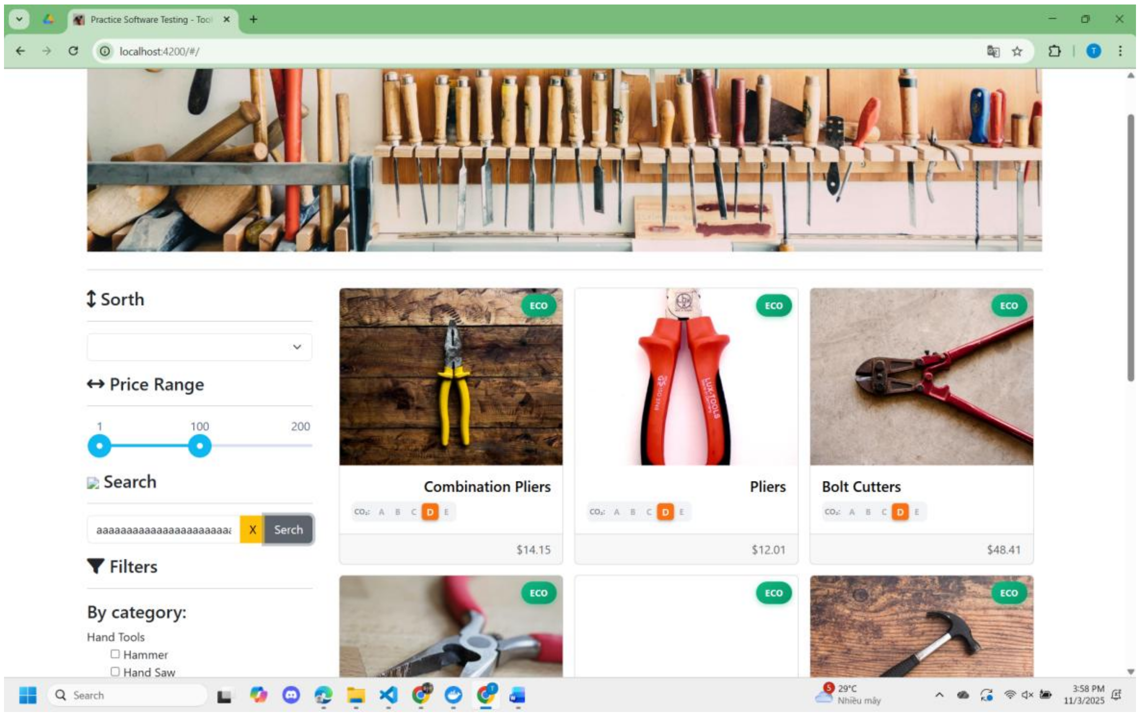
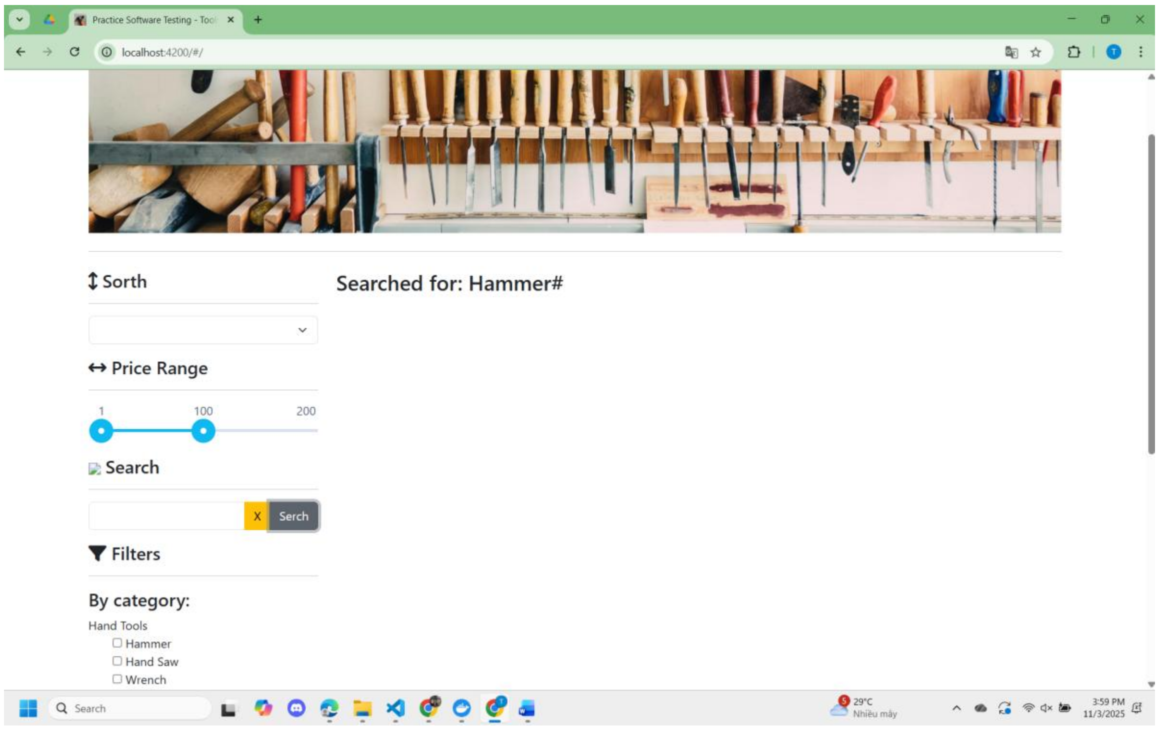


Figure 4: The search box accepts an invalid, excessively long string (41 chars) but provides no error message or feedback, leaving the user on the same screen.



Searched for: Hammer#

Figure 5: The search fails to block special characters, incorrectly processing the invalid input “Hammer#” as a search query.

3. **Bug 3 (Medium): All Sort Functions are Reversed**
   - **Description:** All 6 sort options in the dropdown work, but they sort in the opposite direction of their label.
   - **Failed Tests:** SSF-EP-07, SSF-EP-08, SSF-EP-09, SSF-EP-10, SSF-EP-11, SSF-EP-12 *(6 TCs)*

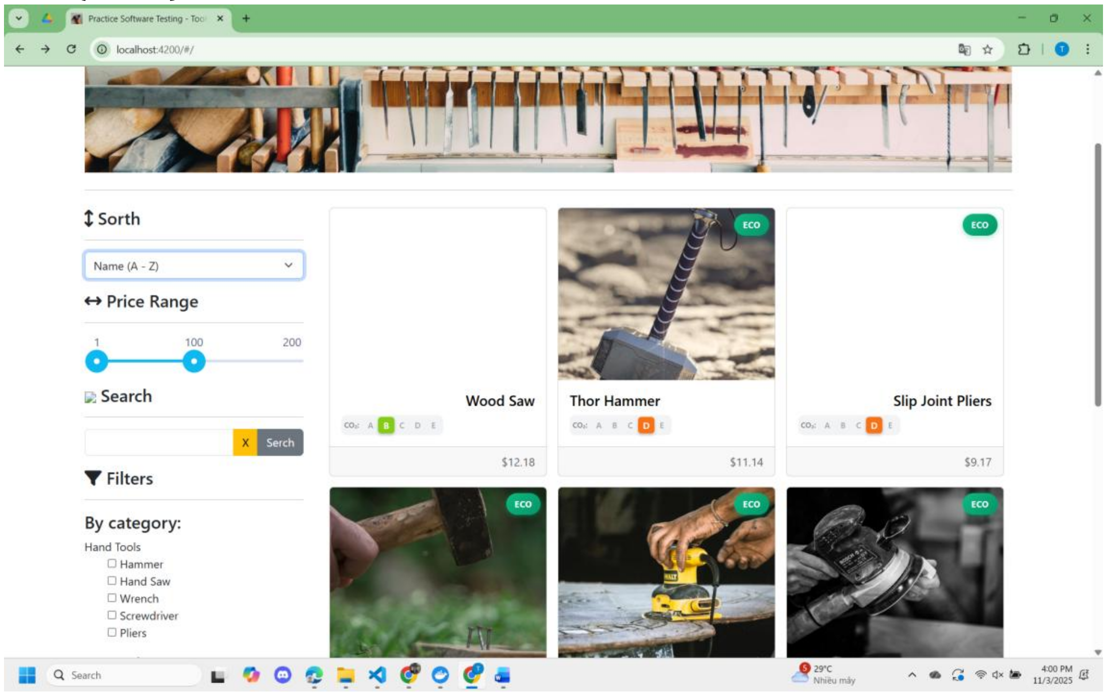


*Figure 6: The sort is set to **"Name (A - Z)"**, but the results are incorrectly sorted in reverse alphabetical order.*

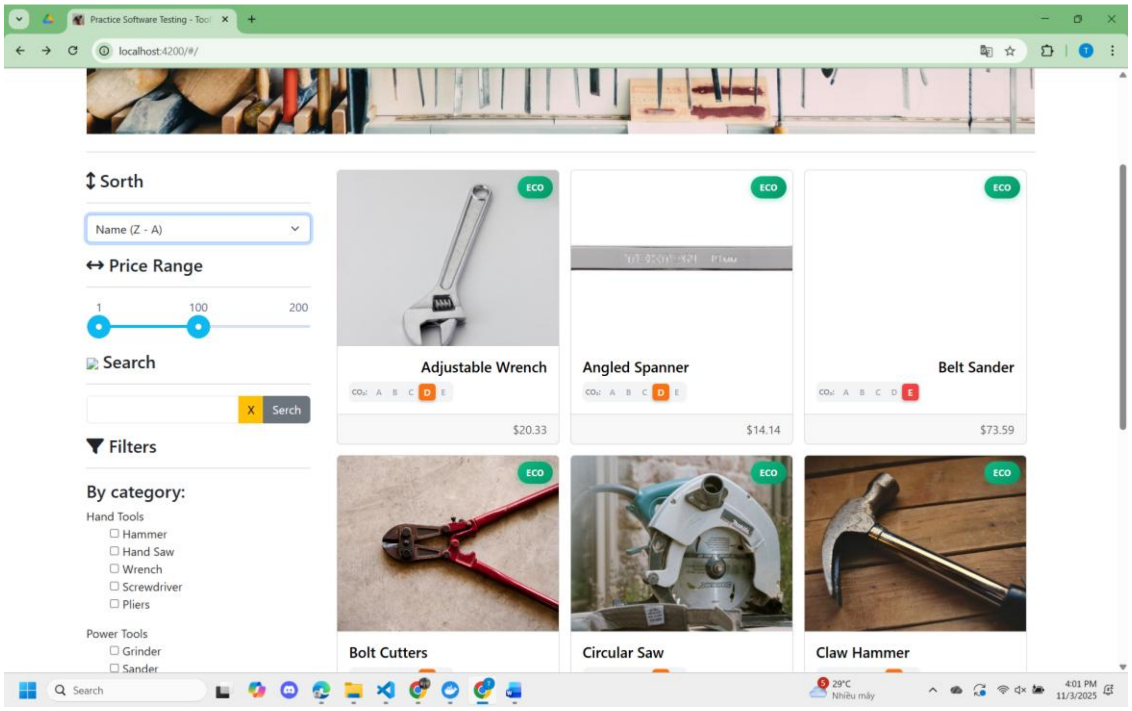


*Figure 7: The sort is set to **"Name (Z - A)"**, but the results are incorrectly sorted in standard alphabetical order.*

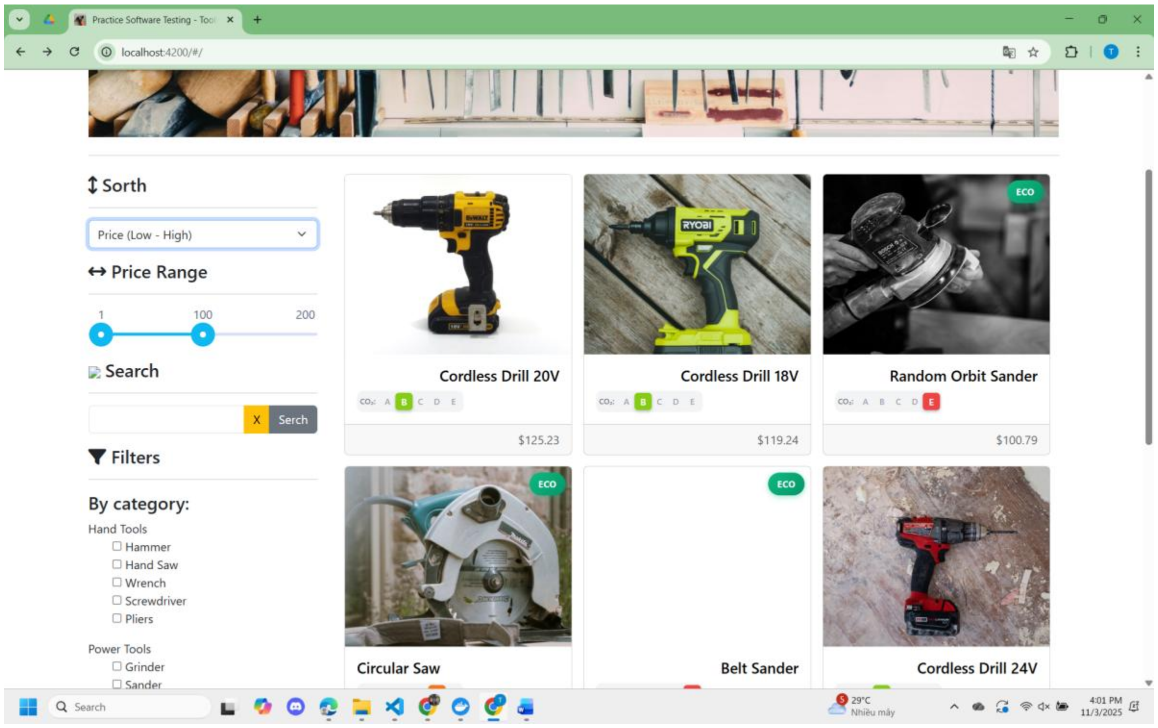


*Figure 8: The sort is set to “**Price (Low - High)**”, but the results are incorrectly sorted from high to low.*

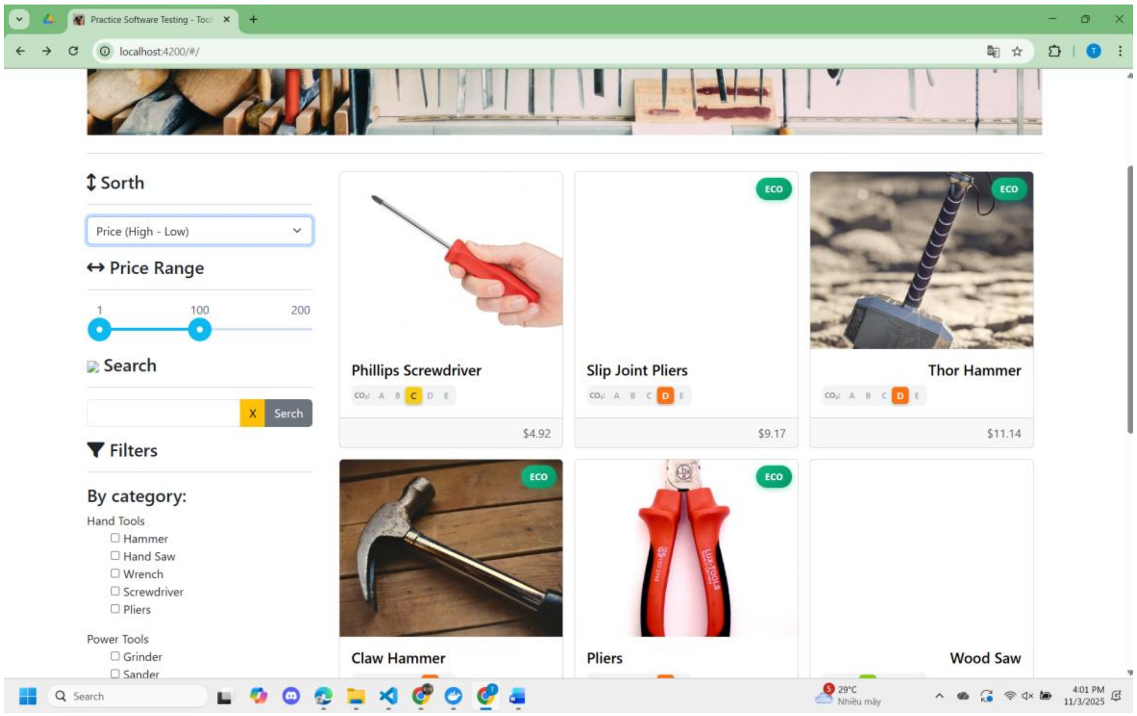
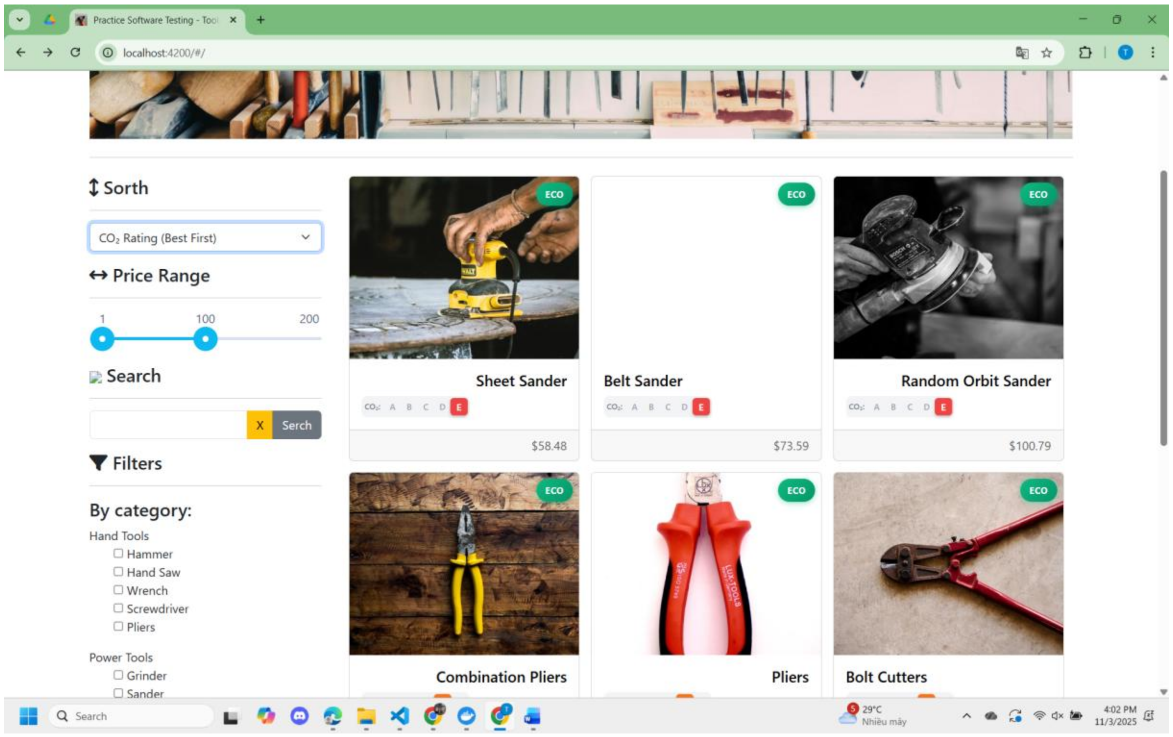
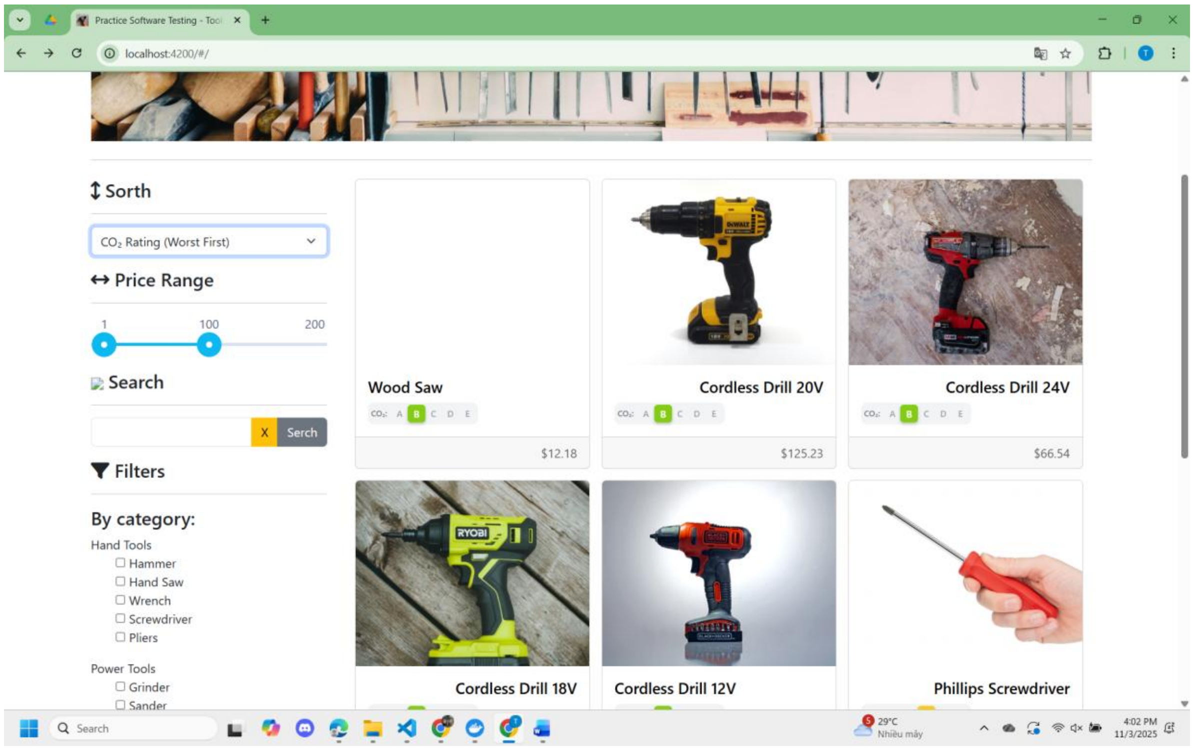


*Figure 9: The sort is set to “**Price (High - Low)**”, but the results are incorrectly sorted from low to high.*



Sorth: CO₂ Rating (Best First)

Figure 10: The sort is set to ***"CO_2 Rating (Best First)"***, but it incorrectly shows items with the worst ratings 'E' and 'D') first.



Sorth: CO₂ Rating (Worst First)

Figure 11: The sort is set to ***"CO_2 Rating (Worst First)"***, but it incorrectly shows items with the best ratings ('B' and 'C') first.

4. **Bug 4 (Serious): Filter Combination Logic is Incorrect**
   - **Description:** The logic for combining filters is broken. When combining "Brand" and "Price" filters, the price filter is ignored.

- **Failed Tests:** SSF-EP-18 *(1 TC)*

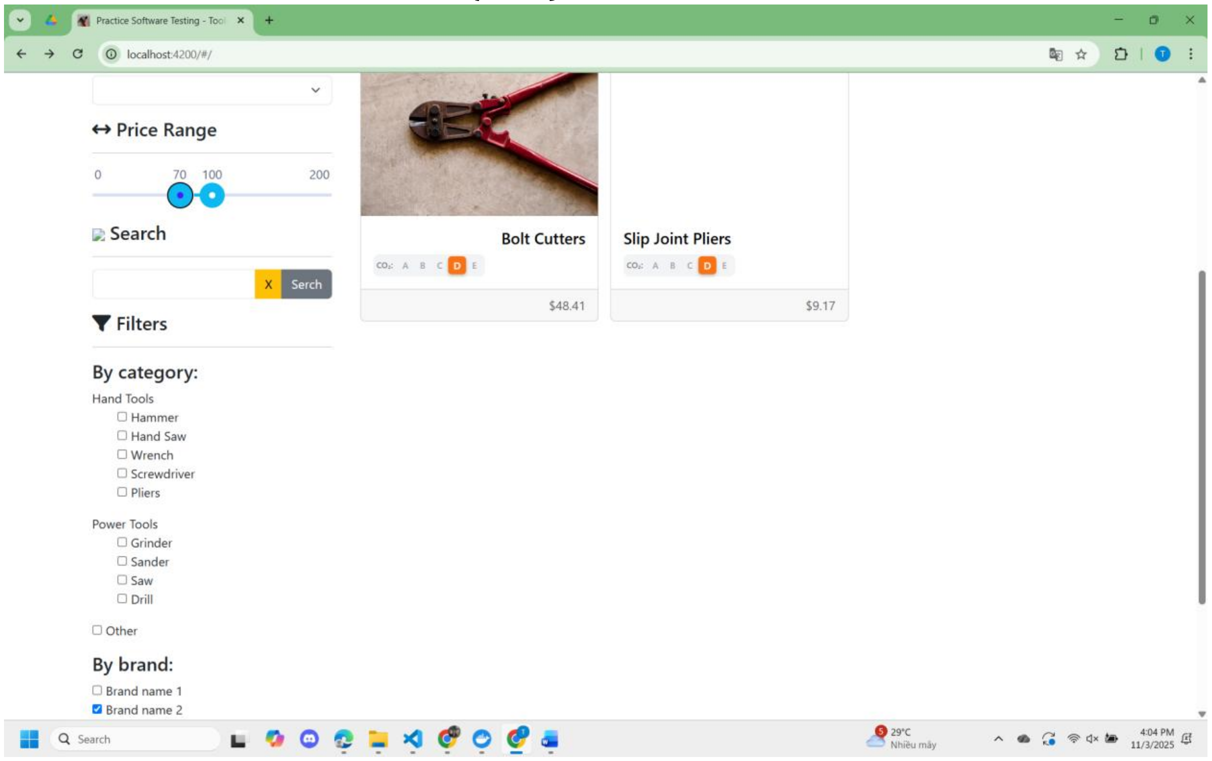


*Figure 12: When "Brand" filter is used, the **Price Range** filter (set 70-100) is ignored. The results incorrectly show items priced at $48.41 and $9.17, both outside the selected range.*

e. Test Outcome Summary

- **Passed:** 12
- **Failed:** 19
- **Total:** 31
- **Analysis:** The feature suffers from severe defects. Core functionality, including validation (Search), filtering (Price), and sorting (all 6 options), is fundamentally broken.

4. Application - Feature 2: User: Checkout Flow

a. Use Case & Inputs

- **Use Case:** A logged-in User proceeds through the 4-step checkout process.
- **Inputs Identified:**
  - Cart: Quantity (Number Field)
  - Address: Address, City, State, Country, Postcode (Text Fields)
  - Payment: Payment Method (Dropdown), Account Name (Text Field), Account Number (Text Field)

b. Partition & Boundary Analysis

- **Quantity:** Domain was 1-10. BVA was applied to 0, 1 and 10, 11.

- **Address Fields:** All fields were identified as Mandatory. The BVA was applied to the boundary between 0 characters (Invalid) and 1 character (Valid).
- **Payment Method:** This was a set of values. EP was applied to test each valid option ("Credit Card") and each invalid option ("Error 304").
- **Account Number:** Assumed a 16-digit rule for "Credit Card". BVA was applied to 15, 16, and 17 digits.

c. Final Test Case Summary

A total of **41 test cases** were designed for this feature.

Equivalence Partitioning (EP) Test Cases (18 TCs)

- **CHK-EP-01:** Cart - Valid Quantity - **Result: Passed**
- **CHK-EP-02:** Cart - Invalid Quantity (Text) - **Result: Passed**
- **CHK-EP-03:** Cart - Verify Total Calculation - **Result: Failed**
- **CHK-EP-04:** Address - Valid (All Fields) - **Result: Passed**
- **CHK-EP-05:** Address - Invalid (Address Empty) - **Result: Passed**
- **CHK-EP-06:** Address - Invalid (City Empty) - **Result: Passed**
- **CHK-EP-07:** Address - Invalid (State Empty) - **Result: Passed**
- **CHK-EP-08:** Address - Invalid (Country Empty) - **Result: Passed**
- **CHK-EP-09:** Address - Invalid (Postcode Empty) - **Result: Passed**
- **CHK-EP-10:** Payment - Invalid (Method Empty) - **Result: Passed**
- **CHK-EP-11:** Payment - Valid (Bank Transfer) - **Result: Passed**
- **CHK-EP-12:** Payment - Valid (Cash on Delivery) - **Result: Failed**
- **CHK-EP-13:** Payment - Valid (Credit Card) - **Result: Passed**
- **CHK-EP-14:** Payment - Valid (Buy Now Pay Later) - **Result: Failed**
- **CHK-EP-15:** Payment - Valid (Gift Card) - **Result: Failed**
- **CHK-EP-16:** Payment - Invalid (Error 304) - **Result: Failed**
- **CHK-EP-17:** Payment - Invalid (Name Empty) - **Result: Passed**
- **CHK-EP-18:** Payment - Invalid (Number Empty) - **Result: Passed**

Boundary Value Analysis (BVA) Test Cases (23 TCs)

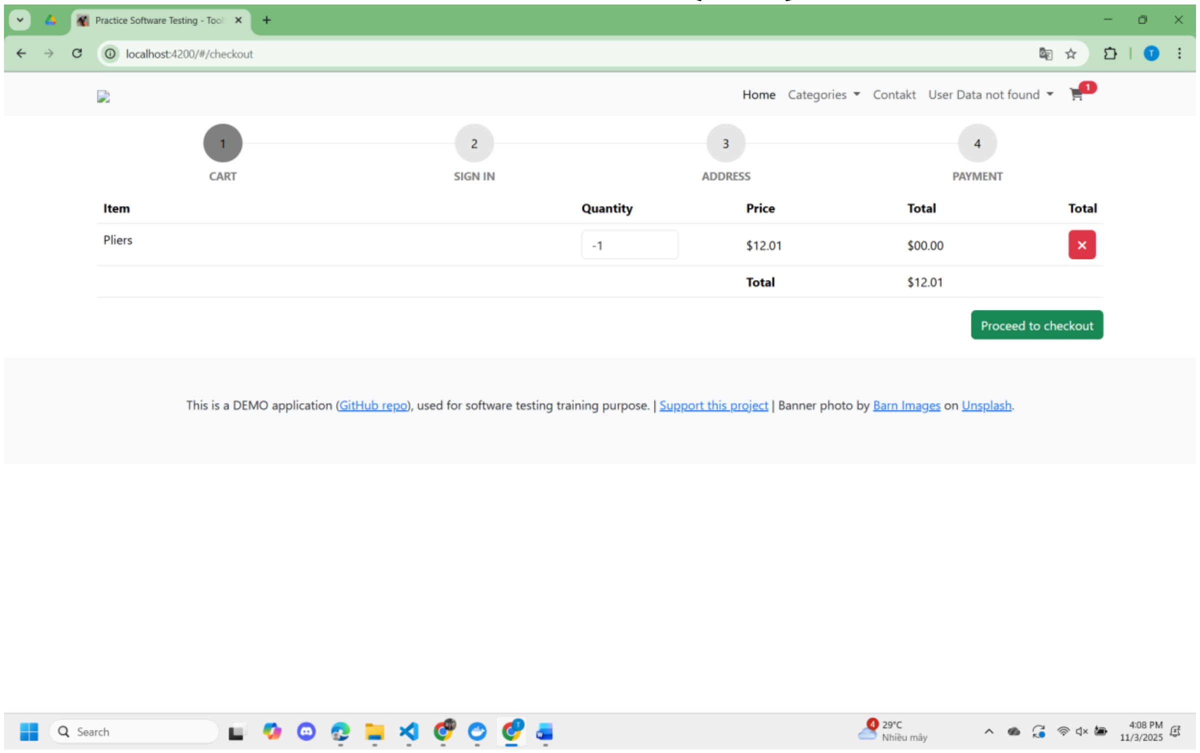
- **CHK-BVA-01:** Cart - Qty (Min - 1) - **Result: Failed**
- **CHK-BVA-02:** Cart - Qty (Min) - **Result: Passed**
- **CHK-BVA-03:** Cart - Qty (Max) - **Result: Passed**
- **CHK-BVA-04:** Cart - Qty (Max + 1) - **Result: Passed**
- **CHK-BVA-05:** Cart - Qty (Negative) - **Result: Failed**
- **CHK-BVA-06:** Address - Address (Min - 1) - **Result: Passed**
- **CHK-BVA-07:** Address - Address (Min) - **Result: Passed**
- **CHK-BVA-08:** Address - City (Min - 1) - **Result: Passed**
- **CHK-BVA-09:** Address - City (Min) - **Result: Passed**
- **CHK-BVA-10:** Address - State (Min - 1) - **Result: Passed**
- **CHK-BVA-11:** Address - State (Min) - **Result: Passed**
- **CHK-BVA-12:** Address - Country (Min - 1) - **Result: Passed**

- **CHK-BVA-13:** Address - Country (Min) - **Result: Passed**
- **CHK-BVA-14:** Address - Postcode (Min - 1) - **Result: Passed**
- **CHK-BVA-15:** Address - Postcode (Min) - **Result: Passed**
- **CHK-BVA-16:** Payment - Name (Min - 1) - **Result: Passed**
- **CHK-BVA-17:** Payment - Name (Min) - **Result: Passed**
- **CHK-BVA-18:** Payment - Number (Min - 1) - **Result: Failed**
- **CHK-BVA-19:** Payment - Number (Min) - **Result: Passed**
- **CHK-BVA-20:** Payment - Number (Max + 1) - **Result: Failed**
- **CHK-BVA-21:** Payment - Number (Empty) - **Result: Passed**
- **CHK-BVA-22:** Payment - Number (Nominal) - **Result: Passed**
- **CHK-BVA-23:** Payment - Number (Text) - **Result: Failed**

d. Bug Summary

The 10 failed test cases point to 4 major bugs:

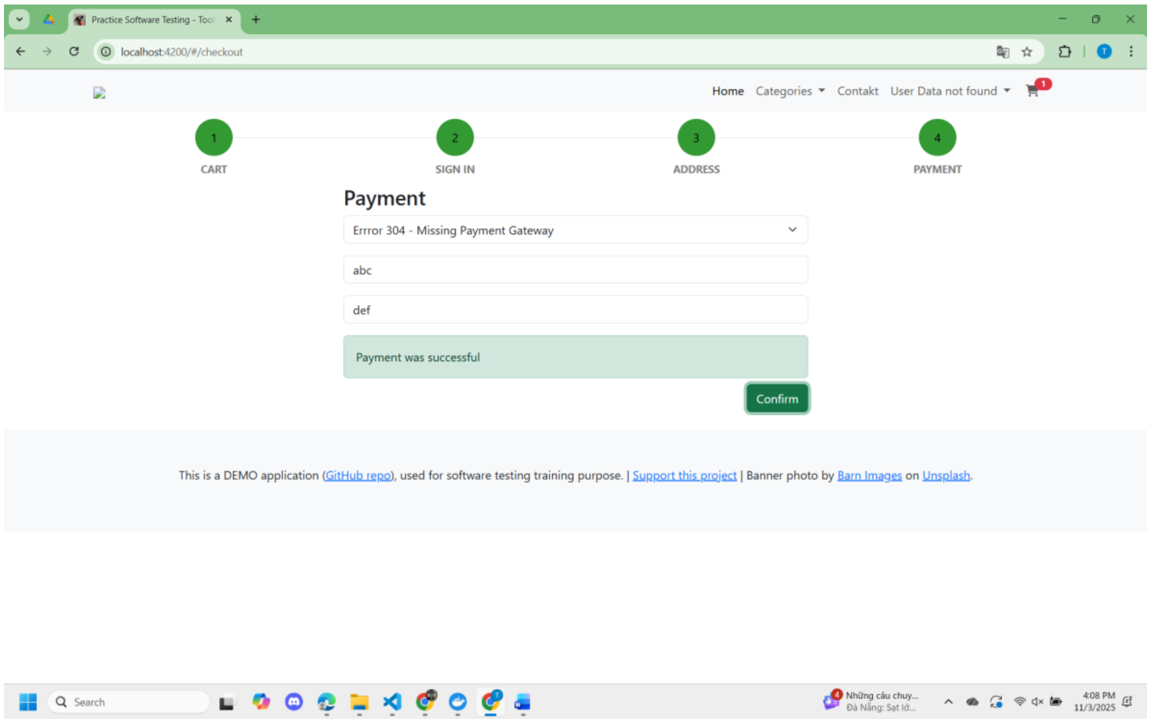
1. **Bug 1 (Fatal): Allows Checkout with Invalid Data**
   - **Description:** The system fails to stop the user from checking out with critical errors, including a negative quantity and a known-broken payment method ("Error 304").
   - **Failed Tests:** CHK-EP-16, CHK-BVA-05 *(2 TCs)*



Item	Quantity	Price	Total	Total
Pliers	-1	$12.01	$00.00	×
		Total	$12.01	

This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

Figure 13: The cart accepts an invalid ***negative quantity (-1)****, yet the "Proceed to checkout" button remains enabled.*



Payment

Errror 304 - Missing Payment Gateway

abc

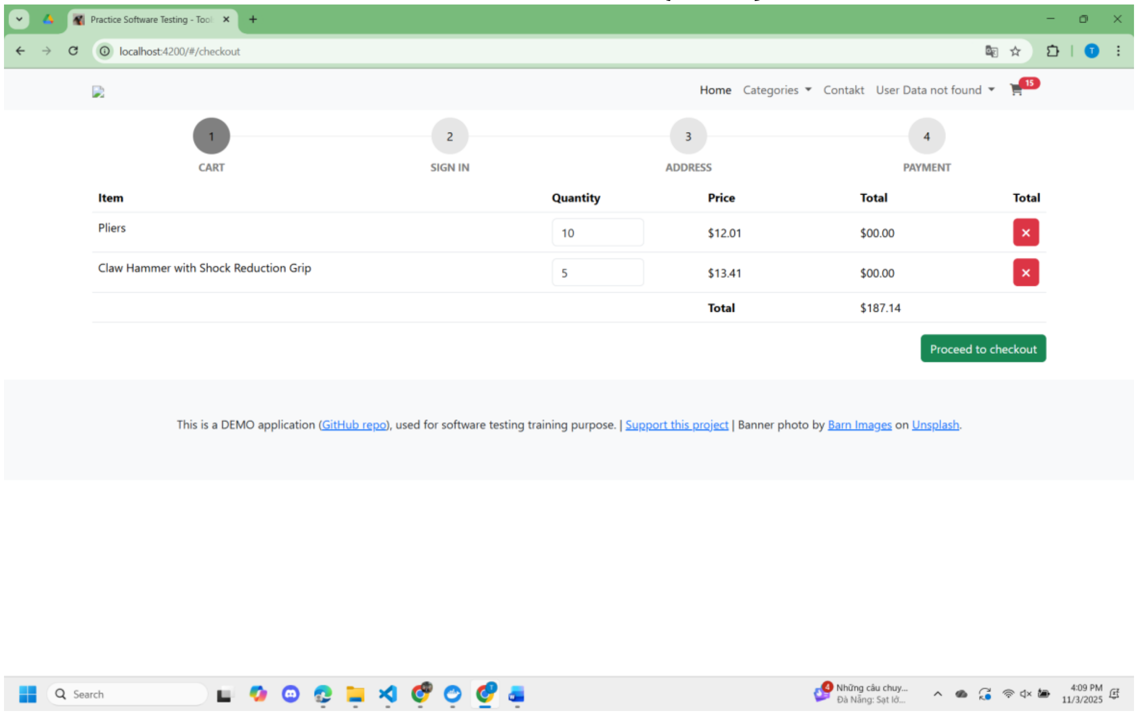
def

Payment was successful

This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

*Figure 14: The payment page processes the broken **"Error 304"** option and incorrectly displays a **"Payment was successful"** message.*

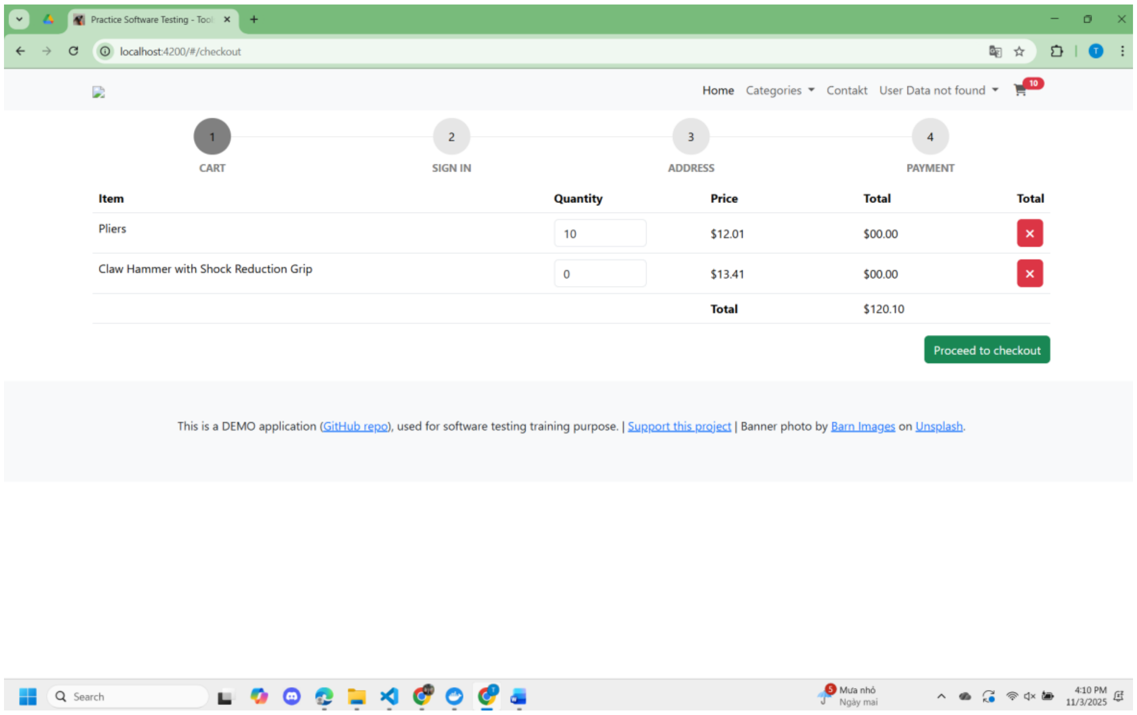
2. **Bug 2 (Serious): Cart Logic is Broken**
   - **Description:** The cart's logic is flawed. The "Total for item" line shows $0.00. Setting quantity to 0 fails to remove the item or show an error.
   - **Failed Tests:** CHK-EP-03, CHK-BVA-01 *(2 TCs)*



Item	Quantity	Price	Total	Total
Pliers	10	$12.01	$00.00	✕
Claw Hammer with Shock Reduction Grip	5	$13.41	$00.00	✕
		Total	$187.14	

This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

*Figure 15: The "Total" column for individual items incorrectly displays **$00.00**, despite the "Pliers" (Qty 10) and "Claw Hammer" (Qty 5) having valid prices.*



localhost:4200/#/checkout

Home Categories Contakt User Data not found 10

1 CART 2 SIGN IN 3 ADDRESS 4 PAYMENT

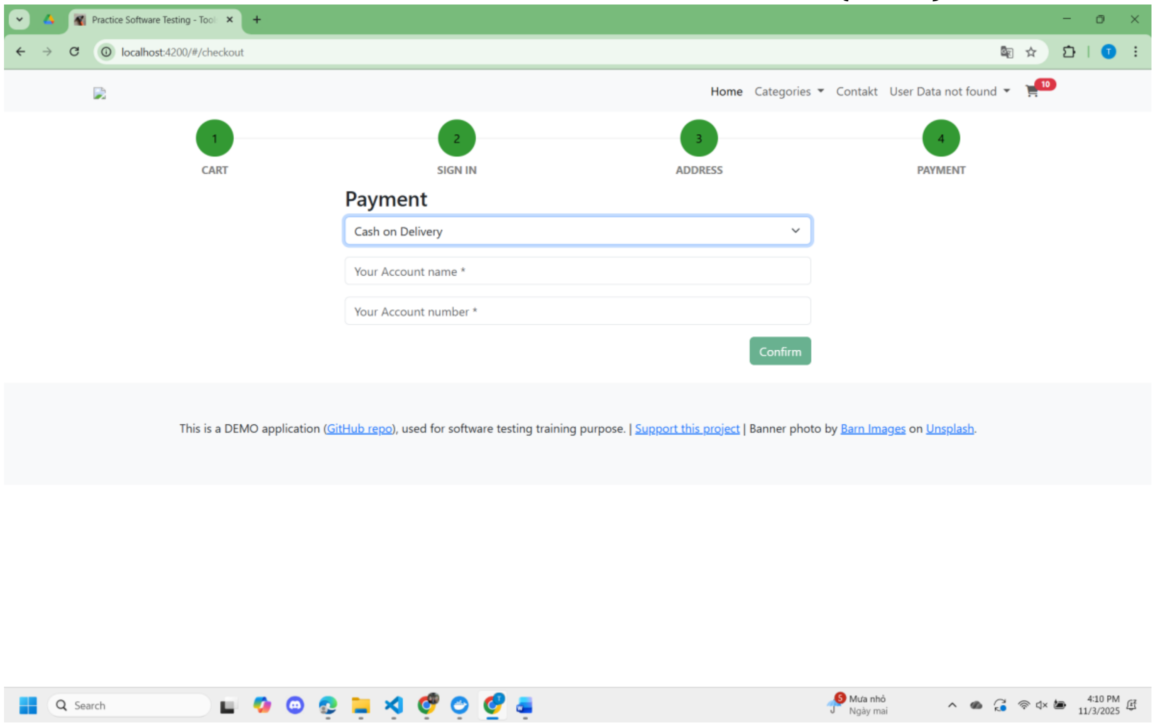
Item	Quantity	Price	Total	Total
Pliers	10	$12.01	$00.00	✕
Claw Hammer with Shock Reduction Grip	0	$13.41	$00.00	✕
		Total	$120.10	

Proceed to checkout

This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

*Figure 16: Setting the "Claw Hammer" quantity to **0** fails to remove the item from the cart.*

3. **Bug 3 (Serious): Payment Form is Not Dynamic**
   - **Description:** The payment form fails to show the correct fields for different payment methods. "Cash on Delivery", "Buy Now Pay Later", and "Gift Card" all incorrectly show the "Account Name" and "Account Number" fields instead of hiding them or showing their own specific fields.
   - **Failed Tests:** CHK-EP-12, CHK-EP-14, CHK-EP-15 *(3 TCs)*



localhost:4200/#/checkout

Home Categories Contakt User Data not found 10

1 CART 2 SIGN IN 3 ADDRESS 4 PAYMENT

Payment

Cash on Delivery

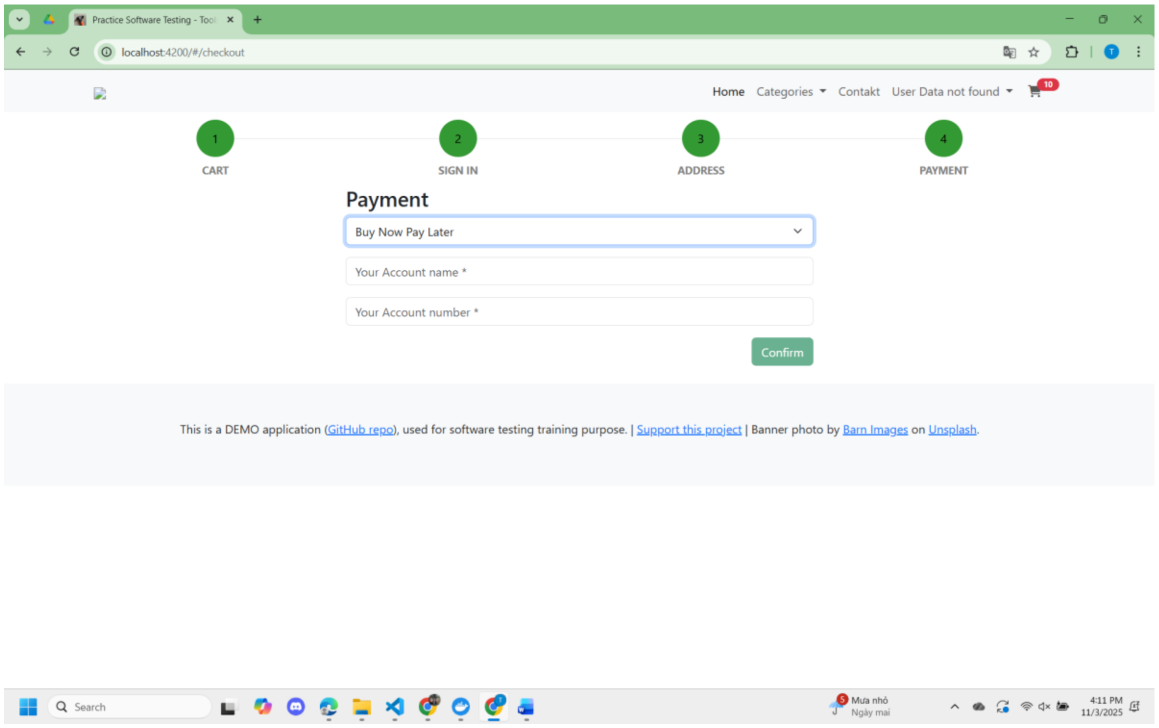
Your Account name *

Your Account number *

Confirm

This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

*Figure 17: Selecting **"Cash on Delivery"** incorrectly displays "Account name" and "Account number" fields, which are irrelevant for this payment type.*



Payment

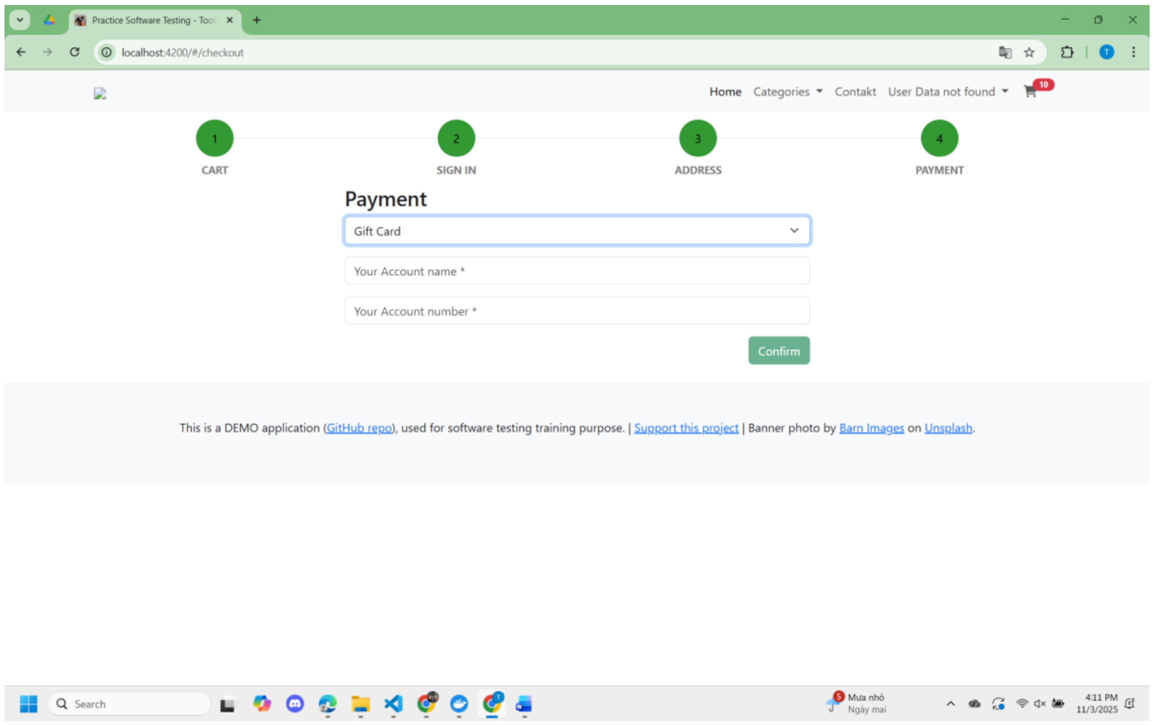
Buy Now Pay Later

Your Account name *

Your Account number *

This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

*Figure 18: The form for **"Buy Now Pay Later"** fails to update, incorrectly showing the same generic "Account name" and "Account number" fields.*



Payment

Gift Card

Your Account name *

Your Account number *

This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

*Figure 19: When **"Gift Card"** is selected, the form incorrectly shows "Account name" and "Account number" instead of the expected "Gift Card Number" field.*

4. **Bug 4 (Serious): Credit Card Validation is Broken**
   - **Description:** The "Account Number" field for Credit Cards has non-existent validation. It accepts numbers that are too short (15 digits), too long (17 digits), and non-numeric text.
   - **Failed Tests:** CHK-BVA-18, CHK-BVA-20, CHK-BVA-23 *(3 TCs)*

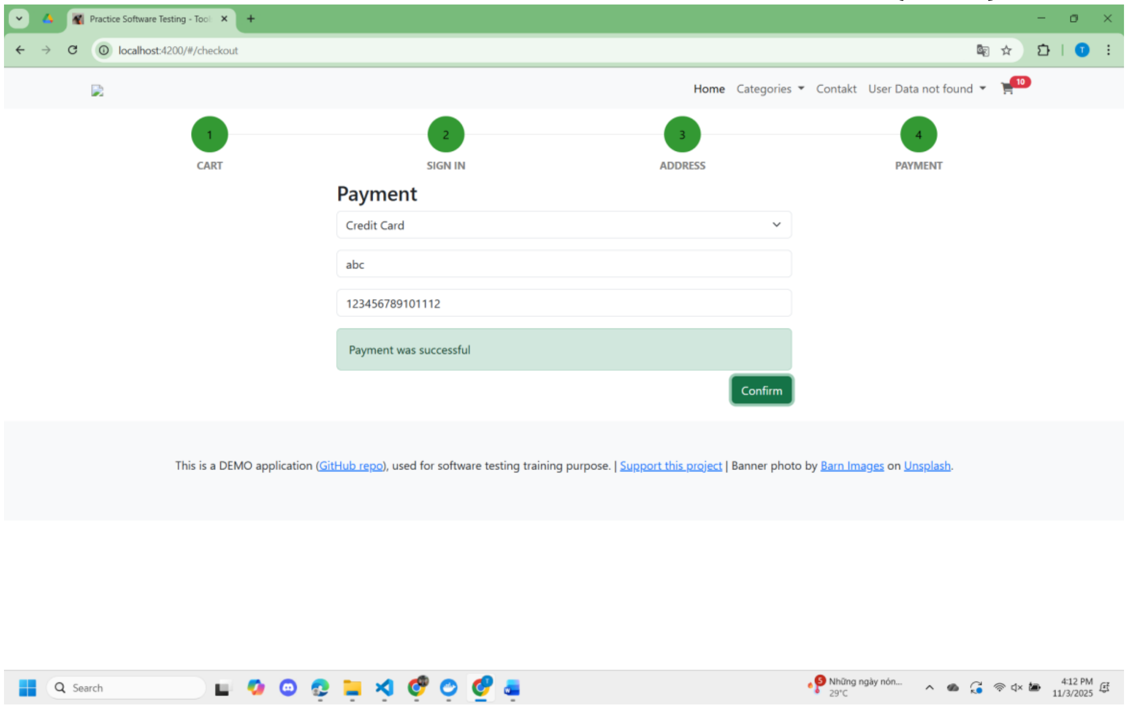

Payment
Credit Card
abc
123456789101112
Payment was successful
Confirm
This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

Figure 20: The form accepts an invalid 15-digit card number (too short) and incorrectly confirms it with a "Payment was successful" message.

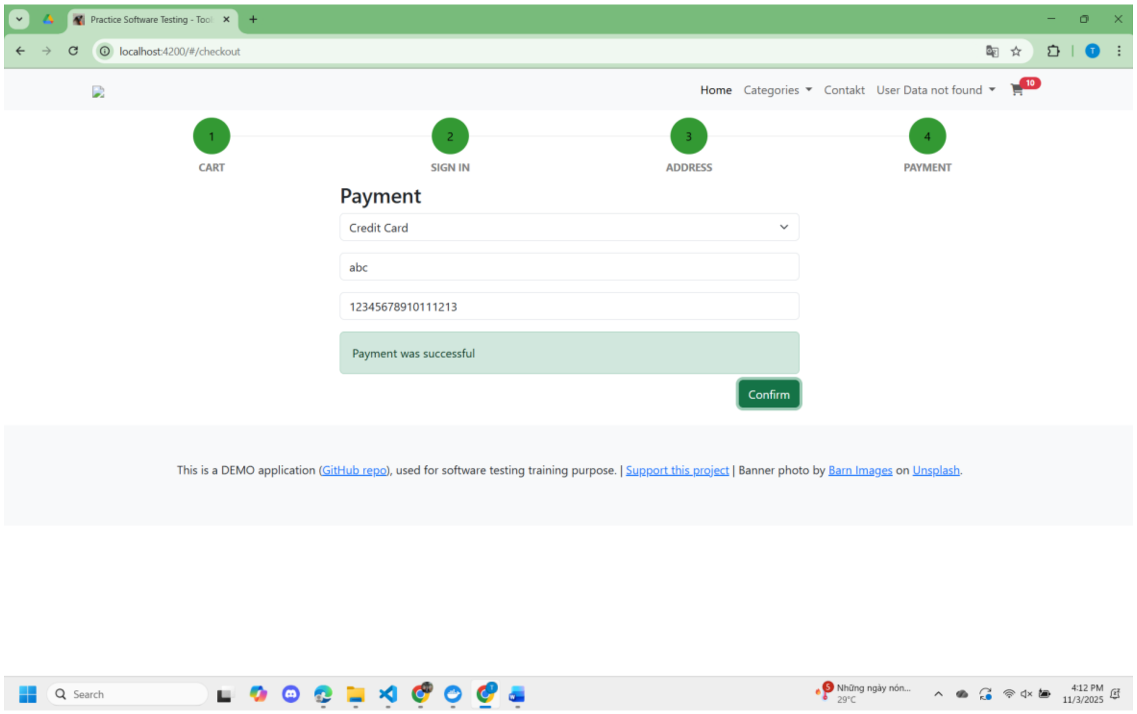
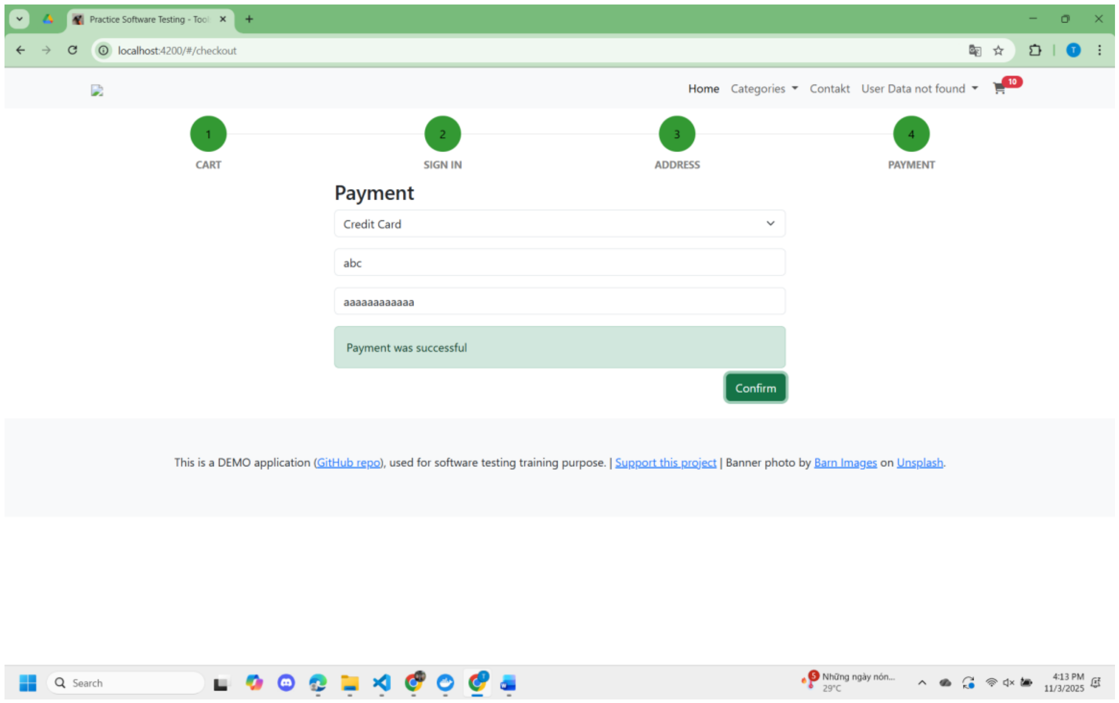

Payment
Credit Card
abc
12345678910111213
Payment was successful
Confirm
This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

Figure 21: The form accepts an invalid 17-digit card number (too long) and incorrectly confirms it with a "Payment was successful" message.



Payment

Credit Card

abc

aaaaaaaaaaaa

Payment was successful

This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

Figure 22: The form fails to validate input, accepting non-numeric text ("aaaaaaaaaa") as a card number and incorrectly showing "Payment was successful".

e. Test Outcome Summary

- **Passed:** 31
- **Failed:** 10
- **Total:** 41
- **Analysis:** The feature suffers from critical defects. The cart calculation is wrong for each item, and the payment validation is broken, allowing users to check out with invalid quantities (negative) and invalid payment methods (Error 304, incorrect digit counts).

5. Application - Feature 3: Admin: Add/Edit Product

a. Use Case & Inputs

- **Use Case:** An Admin adds a new product or edits an existing one.
- **Inputs Identified:** Name, Description, Stock, Price, Brand, Category, etc.

b. Partition & Boundary Analysis

- **Name:** Domain was 1 – 120 chars. BVA was applied.
- **Description:** Domain was 1 – 1250 chars. BVA was applied.
- **Stock:** Domain was ≥ 0. BVA was applied to −1, 0, 1.
- **Price:** Domain was ≥ 0.00. BVA was applied to −0.01, 0.00, 0.01. I also created an invalid partition for 3+ decimal places.

Here we only consider those main fields clearly.

c. Final Test Case Summary

A total of **35 test cases** were designed for this feature.

Equivalence Partitioning (EP) Test Cases (12 TCs)

- **PROD-EP-01:** Valid (Happy Path) - **Result: Passed**
- **PROD-EP-02:** Invalid (Name Empty) - **Result: Passed**
- **PROD-EP-03:** Invalid (Stock Empty) - **Result: Passed**
- **PROD-EP-04:** Invalid (Stock Text) - **Result: Passed**
- **PROD-EP-05:** Invalid (Stock Decimal) - **Result: Failed**
- **PROD-EP-06:** Invalid (Price Empty) - **Result: Passed**
- **PROD-EP-07:** Invalid (Price Text) - **Result: Passed**
- **PROD-EP-08:** Invalid (Brand Empty) - **Result: Passed**
- **PROD-EP-09:** Invalid (Category Empty) - **Result: Passed**
- **PROD-EP-10:** Valid (Description optional field) - **Result: Passed**
- **PROD-EP-11:** Valid (Edit Product) - **Result: Passed**
- **PROD-EP-12:** Valid (Delete Product) - **Result: Passed**

Boundary Value Analysis (BVA) Test Cases (23 TCs)

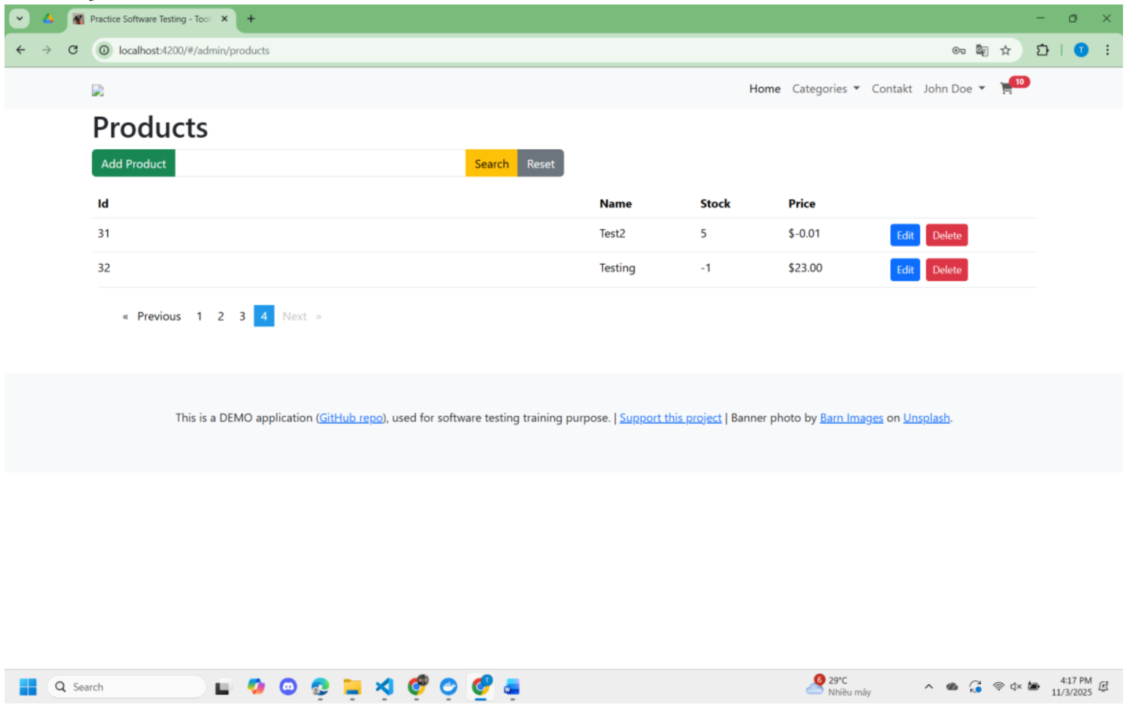
- **PROD-BVA-01:** Name (Min - 1) - **Result: Passed**
- **PROD-BVA-02:** Name (Min) - **Result: Passed**
- **PROD-BVA-03:** Name (Min + 1) - **Result: Passed**
- **PROD-BVA-04:** Name (Max - 1) - **Result: Passed**
- **PROD-BVA-05:** Name (Max) - **Result: Passed**
- **PROD-BVA-06:** Name (Max + 1) - **Result: Passed**
- **PROD-BVA-07:** Description (Min - 1) - **Result: Passed**
- **PROD-BVA-08:** Description (Min) - **Result: Passed**
- **PROD-BVA-09:** Description (Min + 1) - **Result: Passed**
- **PROD-BVA-10:** Description (Max) - **Result: Passed**
- **PROD-BVA-11:** Description (Max + 1) - **Result: Passed**
- **PROD-BVA-12:** Stock (Min - 1) - **Result: Failed**
- **PROD-BVA-13:** Stock (Min) - **Result: Passed**
- **PROD-BVA-14:** Stock (Min + 1) - **Result: Passed**
- **PROD-BVA-15:** Stock (Nominal) - **Result: Passed**
- **PROD-BVA-16:** Price (Min - 0.01) - **Result: Failed**
- **PROD-BVA-17:** Price (Min) - **Result: Passed**
- **PROD-BVA-18:** Price (Min + 0.01) - **Result: Passed**
- **PROD-BVA-19:** Edit - Name (Invalid) - **Result: Passed**
- **PROD-BVA-20:** Edit - Description (Invalid) - **Result: Passed**
- **PROD-BVA-21:** Edit - Stock (Invalid) - **Result: Failed**
- **PROD-BVA-22:** Edit - Price (Invalid) - **Result: Failed**
- **PROD-BVA-23:** Valid (Checkboxes) - **Result: Passed**

d. Bug Summary

The 5 failed test cases point to 2 major bugs:

1. **Bug 1 (Fatal): Allows Negative Stock and Price**

- **Description:** The system's validation is broken and allows an Admin to save a product with a negative stock value (-1) and a negative price (-0.01). This is a critical data integrity failure and exists on both the "Add" and "Edit" forms.
- **Failed Tests:** PROD-BVA-12, PROD-BVA-16, PROD-BVA-21, PROD-BVA-22 *(4 TCs)*

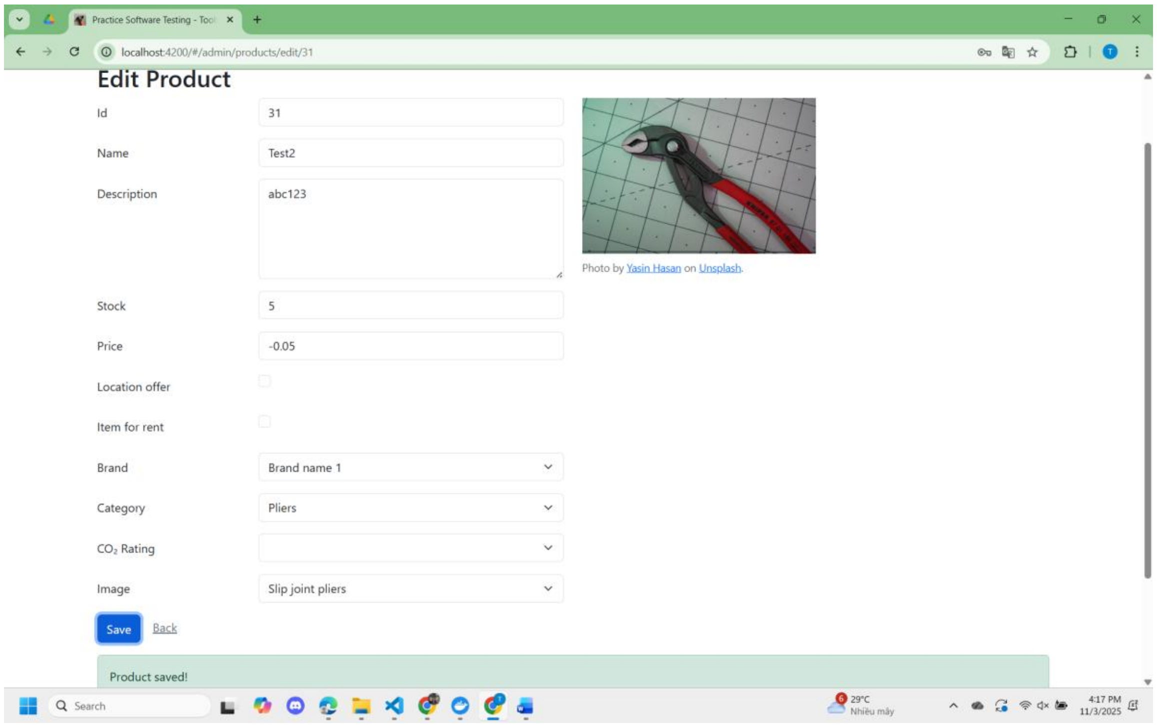


Products

Id	Name	Stock	Price
31	Test2	5	$-0.01
32	Testing	-1	$23.00

This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

*Figure 23: The system allows to save invalid data, which results in Product 31 has a **negative price ($-0.01)**, and Product 32 has a **negative stock (-1)**.*

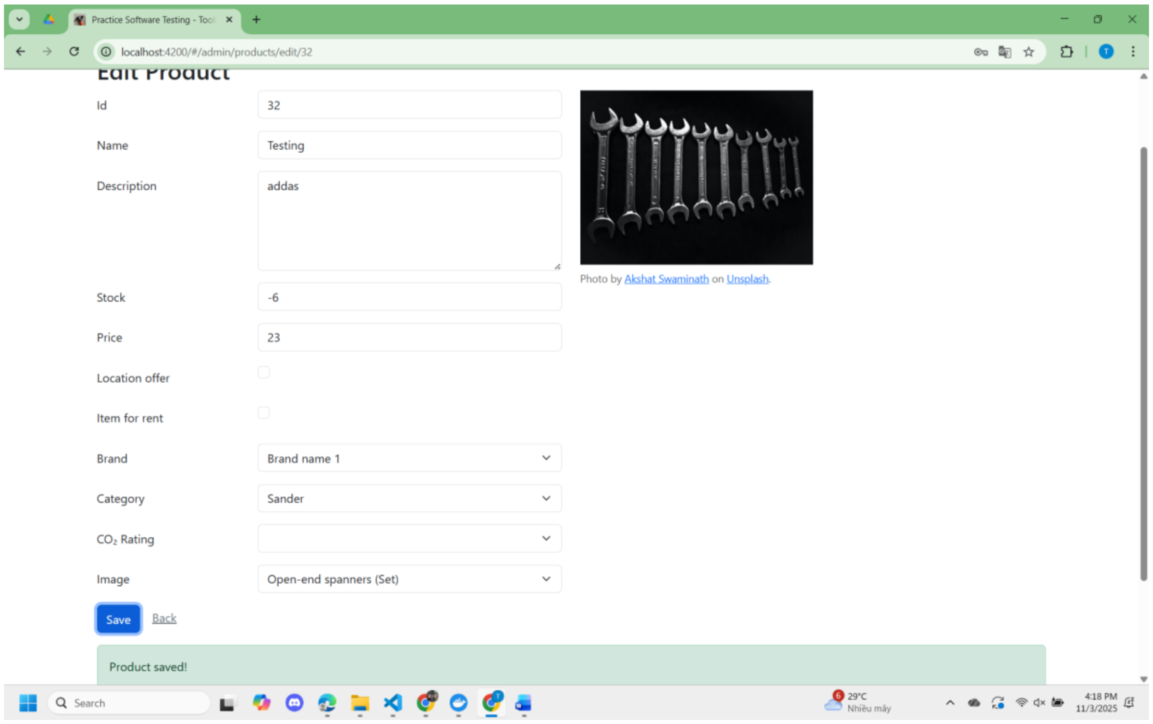

localhost:4200/#/admin/products/edit/31
Photo by Yasin Hasan on Unsplash.
Save
Back

Edit Product

Id	31
Name	Test2
Description	abc123
Stock	5
Price	-0.05
Location offer	
Item for rent	
Brand	Brand name 1
Category	Pliers
CO₂ Rating	
Image	Slip joint pliers

Product saved!

*Figure 24: The "Edit Product" form fails to validate the Price field, allowing a **negative value (-0.05)** to be saved, as confirmed by the "Product saved!" message.*



Practice Software Testing - Tool

localhost:4200/#/admin/products/edit/32

Edit Product

Id	32
Name	Testing
Description	addas
Stock	-6
Price	23
Location offer	
Item for rent	
Brand	Brand name 1
Category	Sander
CO₂ Rating	
Image	Open-end spanners (Set)

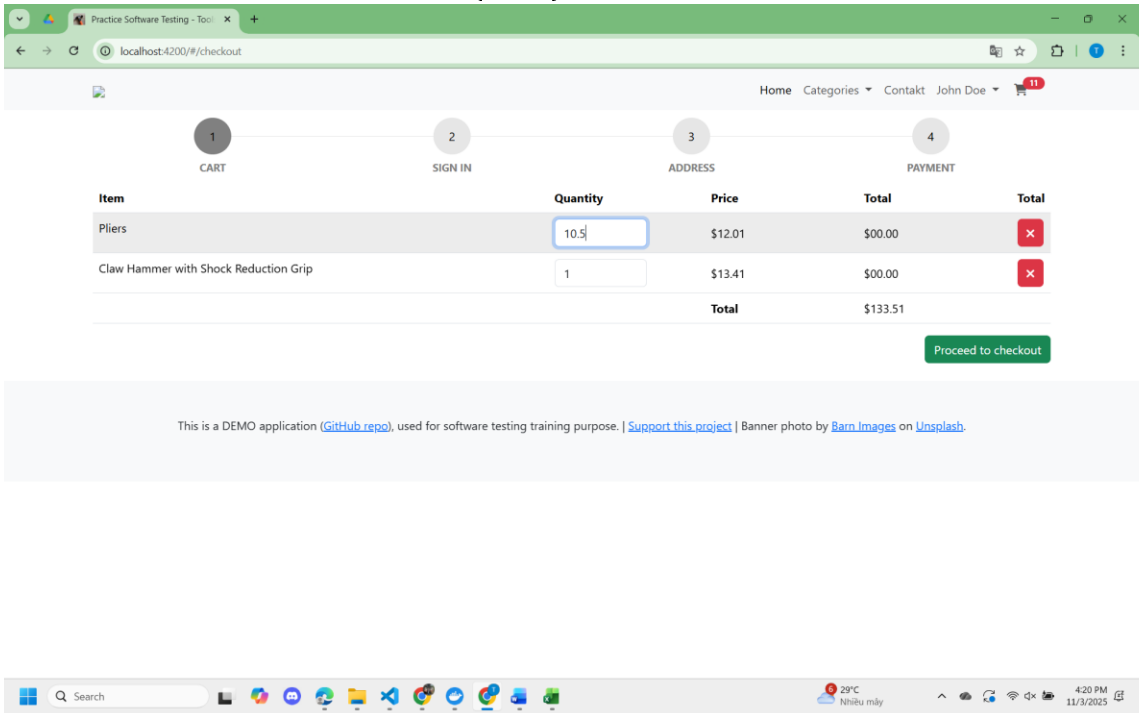
Photo by Akshat Swaminath on Unsplash.

Save Back

Product saved!

Figure 25: *The "Edit Product" form fails to validate the Stock field, allowing a* ***negative value (-6)*** *to be saved, as confirmed by the "Product saved!" message.*

2. **Bug 2 (Fatal): Silent Data Corruption on Invalid Input**
   - **Description:** When an invalid decimal ("10.5") is entered for "Stock", the system does not show an error. Instead, it "rounds" the value and saves the product, silently corrupting the inventory data.
   - **Failed Tests:** PROD-EP-05 *(1 TC)*



Practice Software Testing - Tool

localhost:4200/#/checkout

Home Categories Contakt John Doe 11

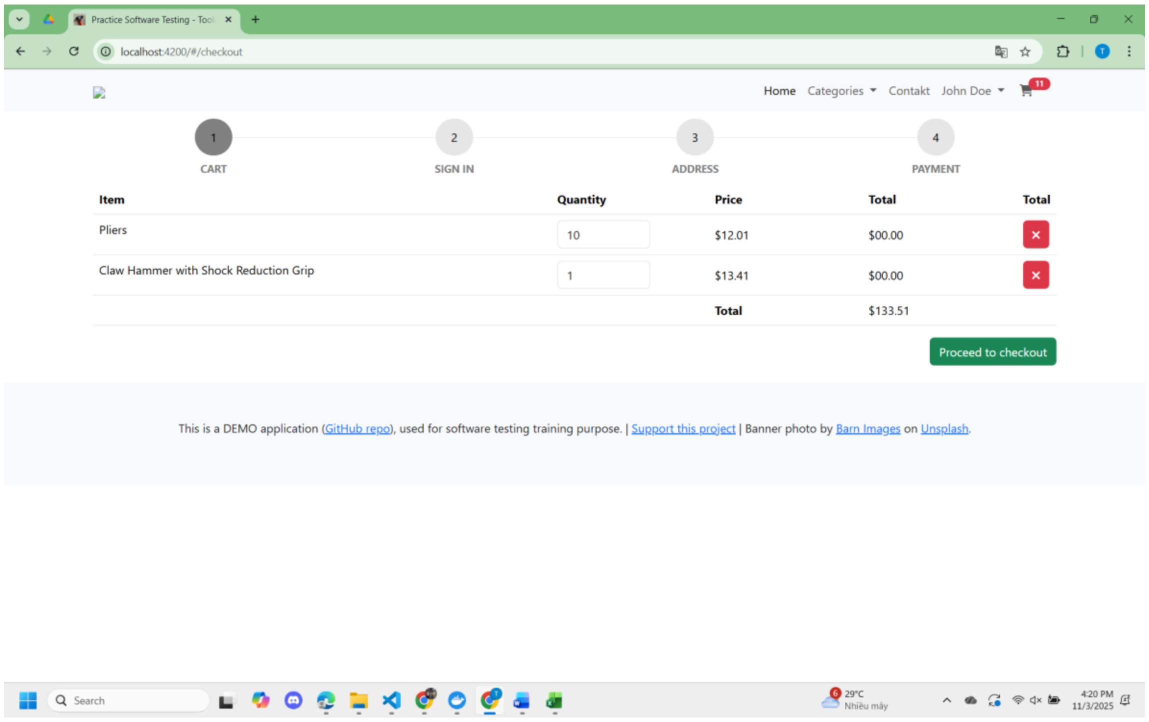
1 CART 2 SIGN IN 3 ADDRESS 4 PAYMENT

Item	Quantity	Price	Total	Total
Pliers	10.5	$12.01	$00.00	×
Claw Hammer with Shock Reduction Grip	1	$13.41	$00.00	×
		Total	$133.51	

Proceed to checkout

This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

Figure 26: An invalid decimal value ("10.5") is entered into the "Quantity" field. The system fails to display a validation error message.



Item	Quantity	Price	Total	Total
Pliers	10	$12.01	$00.00	×
Claw Hammer with Shock Reduction Grip	1	$13.41	$00.00	×
		Total	$133.51	

This is a DEMO application (GitHub repo), used for software testing training purpose. | Support this project | Banner photo by Barn Images on Unsplash.

Figure 27: After processing the invalid input, the system silently truncates the quantity from "10.5" to "10" instead of rejecting it, corrupting the user's intended value.

e. Test Outcome Summary

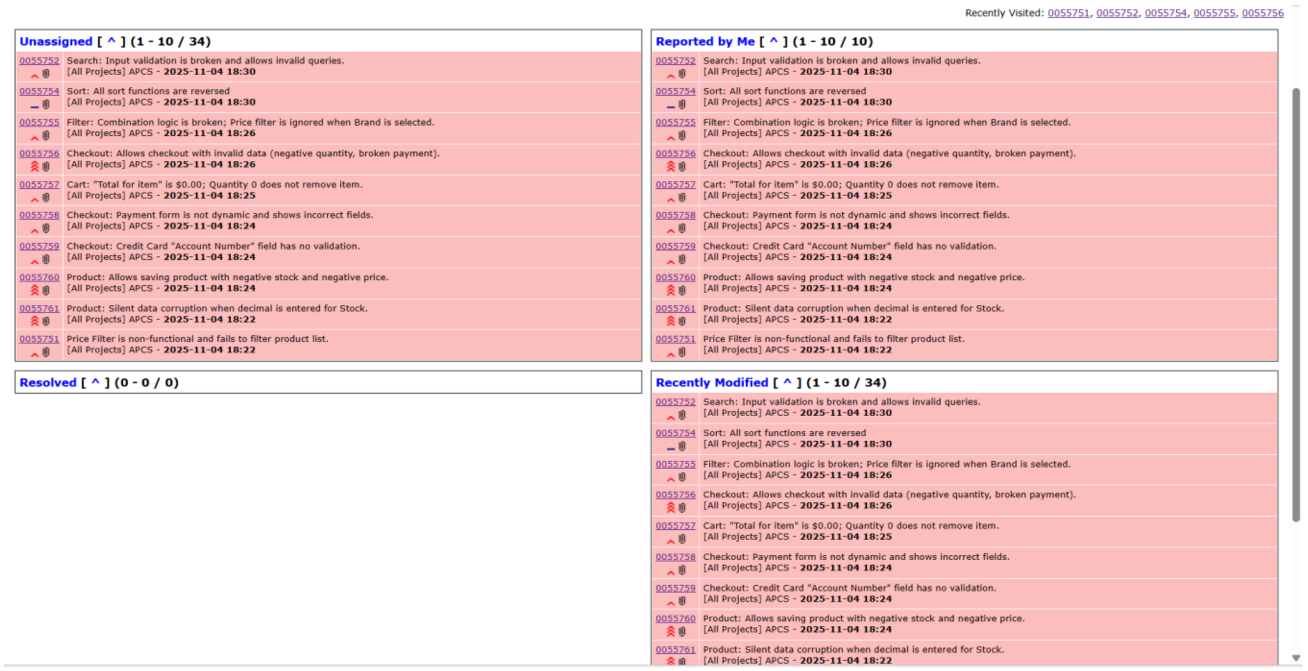
- **Passed:** 30
- **Failed:** 5
- **Total:** 35
- **Analysis:** The feature is largely functional but contains **critical (Fatal)** defects. The validation for core financial and inventory fields is broken, allowing an admin to save products with **negative stock** (PROD-BVA-12), **negative prices** (PROD-BVA-16), and to **corrupt data** by rounding decimals (PROD-EP-05). These data integrity failures are severe.

6. Conclusion

This report details the successful application of the Domain Testing methodology across 107 test cases for three core features of the Toolshop application. The execution of these tests revealed that while some basic functionality is in place, the application suffers from **numerous severe and fatal defects** and is not ready for a production environment.

- **Feature 1 (Search & Sort & Filter):** This feature is **critically broken**. With 19 of 31 tests failing, core user functions like searching, sorting, and price filtering are non-functional or logically incorrect.
- **Feature 2 (Checkout Flow):** This feature is **critically broken**. With 10 of 41 tests failing, it contains fatal flaws that allow users to check out with invalid data, incorrect financial totals, and broken payment methods.

- **Feature 3 (Admin: Add/Edit Product):** This feature is the most stable but contains the most severe **data corruption** bugs. 5 of 35 tests failed, but these failures (accepting negative stock and price) are fatal to data integrity.



Recently Visited: 0055751, 0055752, 0055754, 0055755, 0055756

Unassigned [^] (1 - 10 / 34)

0055752 Search: Input validation is broken and allows invalid queries.
[All Projects] APCS - **2025-11-04 18:30**

0055754 Sort: All sort functions are reversed
[All Projects] APCS - **2025-11-04 18:30**

0055755 Filter: Combination logic is broken; Price filter is ignored when Brand is selected.
[All Projects] APCS - **2025-11-04 18:26**

0055756 Checkout: Allows checkout with invalid data (negative quantity, broken payment).
[All Projects] APCS - **2025-11-04 18:26**

0055757 Cart: "Total for item" is $0.00; Quantity 0 does not remove item.
[All Projects] APCS - **2025-11-04 18:25**

0055758 Checkout: Payment form is not dynamic and shows incorrect fields.
[All Projects] APCS - **2025-11-04 18:24**

0055759 Checkout: Credit Card "Account Number" field has no validation.
[All Projects] APCS - **2025-11-04 18:24**

0055760 Product: Allows saving product with negative stock and negative price.
[All Projects] APCS - **2025-11-04 18:24**

0055761 Product: Silent data corruption when decimal is entered for Stock.
[All Projects] APCS - **2025-11-04 18:22**

0055751 Price Filter is non-functional and fails to filter product list.
[All Projects] APCS - **2025-11-04 18:22**

Resolved [^] (0 - 0 / 0)

Reported by Me [^] (1 - 10 / 10)

0055752 Search: Input validation is broken and allows invalid queries.
[All Projects] APCS - **2025-11-04 18:30**

0055754 Sort: All sort functions are reversed
[All Projects] APCS - **2025-11-04 18:30**

0055755 Filter: Combination logic is broken; Price filter is ignored when Brand is selected.
[All Projects] APCS - **2025-11-04 18:26**

0055756 Checkout: Allows checkout with invalid data (negative quantity, broken payment).
[All Projects] APCS - **2025-11-04 18:26**

0055757 Cart: "Total for item" is $0.00; Quantity 0 does not remove item.
[All Projects] APCS - **2025-11-04 18:25**

0055758 Checkout: Payment form is not dynamic and shows incorrect fields.
[All Projects] APCS - **2025-11-04 18:24**

0055759 Checkout: Credit Card "Account Number" field has no validation.
[All Projects] APCS - **2025-11-04 18:24**

0055760 Product: Allows saving product with negative stock and negative price.
[All Projects] APCS - **2025-11-04 18:24**

0055761 Product: Silent data corruption when decimal is entered for Stock.
[All Projects] APCS - **2025-11-04 18:22**

0055751 Price Filter is non-functional and fails to filter product list.
[All Projects] APCS - **2025-11-04 18:22**

Recently Modified [^] (1 - 10 / 34)

0055752 Search: Input validation is broken and allows invalid queries.
[All Projects] APCS - **2025-11-04 18:30**

0055754 Sort: All sort functions are reversed
[All Projects] APCS - **2025-11-04 18:30**

0055755 Filter: Combination logic is broken; Price filter is ignored when Brand is selected.
[All Projects] APCS - **2025-11-04 18:26**

0055756 Checkout: Allows checkout with invalid data (negative quantity, broken payment).
[All Projects] APCS - **2025-11-04 18:26**

0055757 Cart: "Total for item" is $0.00; Quantity 0 does not remove item.
[All Projects] APCS - **2025-11-04 18:25**

0055758 Checkout: Payment form is not dynamic and shows incorrect fields.
[All Projects] APCS - **2025-11-04 18:24**

0055759 Checkout: Credit Card "Account Number" field has no validation.
[All Projects] APCS - **2025-11-04 18:24**

0055760 Product: Allows saving product with negative stock and negative price.
[All Projects] APCS - **2025-11-04 18:24**

0055761 Product: Silent data corruption when decimal is entered for Stock.
[All Projects] APCS - **2025-11-04 18:22**

Figure 28: All discovered bugs in FIT's Mantis system.

Self – Assessment

Requirement	Outcomes	Grade	Self-Assessed Grade
1	**Guest: Search & Sort & Filter**	**40**	**40**
	1.1 Domain Analysis	5	5
	1.2 Test Cases (31)	20	20
	1.3 Boundary Analysis	5	5
	1.4 Bugs	10	10
2	**User: Checkout Flow**	**30**	**30**
	2.1 Domain Analysis	5	5
	2.2 Test Cases (41)	15	15
	2.3 Boundary Analysis	5	5
	2.4 Bugs	5	5
3	**Admin: Add/Edit Product**	**30**	**30**
	3.1 Domain Analysis	5	5
	3.2 Test Cases (35)	15	15
	3.3 Boundary Analysis	5	5
	3.4 Bugs	5	5
	Total	**100**	**100**