

Vietnam National University- Ho Chi Minh City  
University of Science

*Faculty of Information Technology*



## HW02.Domain Testing

Course    SOFTWARE TESTING

Class      22KTPM3

Student    22127225 – Trần Thị Thiên Kim

*Ho Chi Minh City, 2025*

# Table of Contents

<b>1 Domain Testing in Toolshop</b>	<b>2</b>
1.1 System Overview . . . . .	2
1.2 Deployment Guide for Toolshop (Short Version) . . . . .	2
1.3 Severity and Priority Analysis . . . . .	2
1.3.1 Role in Domain Testing . . . . .	2
1.3.2 Severity Classification . . . . .	3
1.3.3 Priority Classification . . . . .	3
<b>2 Product Search &amp; Catalog</b>	<b>3</b>
2.1 Description . . . . .	3
2.2 Inputs and Constraints . . . . .	3
2.3 Equivalence Partitioning (EP) . . . . .	3
2.4 Boundary Value Analysis (BVA) . . . . .	4
<b>3 Category Management</b>	<b>4</b>
3.1 Description . . . . .	4
3.2 Inputs and Constraints . . . . .	4
3.3 Equivalence Partitioning (EP) . . . . .	5
3.4 Boundary Value Analysis (BVA) . . . . .	5
<b>4 Shopping Cart Operations</b>	<b>5</b>
4.1 Description . . . . .	5
4.2 Inputs and Constraints . . . . .	5
4.3 Equivalence Partitioning (EP) . . . . .	6
4.4 Boundary Value Analysis (BVA) . . . . .	6
<b>5 Use of AI Tools in Test Design</b>	<b>6</b>
5.1 AI Tool Used . . . . .	6
5.2 Prompts Used . . . . .	6
5.3 Validation and Refinement . . . . .	9
5.3.1 System Behavior Verification . . . . .	9
5.3.2 Functional Specification Alignment . . . . .	10
5.3.3 Manual Review and Formatting . . . . .	10
5.3.4 Deduplication . . . . .	10

# 1 Domain Testing in Toolshop

## 1.1 System Overview

Toolshop is an e-commerce platform focused on selling and managing tools. The platform includes features such as product search, category management, and shopping cart operations. To ensure reliability and robustness of these functionalities, domain testing methods like Equivalence Partitioning (EP) and Boundary Value Analysis (BVA) are applied.

## 1.2 Deployment Guide for Toolshop (Short Version)

- **Prepare:** Install Git, Node.js (v14+), and npm.
- **Clone repository:**

```
git clone https://github.com/testsmith-io/practice-software-testing.git
cd practice-software-testing/sprint5-with-bugs
```

- **Install dependencies:**

```
npm install
```

- **Configure:** Create a .env file with PORT=4200 and DATABASE\_URL=sqlite:///./toolshop.db.
- **Run application:**

```
npm start
```

Access <http://localhost:4200>.

- **Test:** Try searching or adding a product, and record results in test cases.

## 1.3 Severity and Priority Analysis

### 1.3.1 Role in Domain Testing

During domain testing, defects are identified through equivalence partitioning (EP) and boundary value analysis (BVA). Once detected, each defect is evaluated based on:

- **Severity** – the level of impact the defect has on system functionality or user experience.
- **Priority** – the urgency with which the defect needs to be fixed.

### 1.3.2 Severity Classification

- **High** – Defects that cause critical failures or break core functionalities (mapped from “Fatal” and “Serious”).
- **Medium** – Defects that cause minor deviations from requirements, such as layout inconsistencies or unexpected behavior with valid inputs.
- **Low** – Cosmetic issues with minimal or no functional impact, such as missing tooltips or alignment issues.

### 1.3.3 Priority Classification

- **Critical (Immediate)** – Must be fixed within 1 day due to severe impact.
- **High** – Should be fixed within 2–4 days; affects core functions.
- **Medium** – Can be addressed within 5–8 days; minor impact or frequency.
- **Low** – Can be deferred; negligible effect on user experience or operations.

## 2 Product Search & Catalog

### 2.1 Description

- Search products by query and display product information
- Role: Guest/Customer
- Priority: MEDIUM
- Domain: Product discovery and listing

### 2.2 Inputs and Constraints

This feature allows users to search for products using keywords and filters by price.

- **query:** User’s search string (1–255 characters, alphanumeric, no special characters or emojis)
- **min\_price:** Must be  $\geq 0$
- **max\_price:** Must be  $\geq \text{min\_price}$ ; typically, maximum is 200

### 2.3 Equivalence Partitioning (EP)

#### Search Query:

- **Valid Classes:**
  - V1: Alphanumeric (e.g., “hammer”, “drill”)
  - V2: Partial keywords (e.g., “ham”)

- V3: Multi-word queries (e.g., “screwdriver”)
- V4: Capital letters (e.g., “HAMMER”)
- V5: Numbers only (e.g., “123”)

- **Invalid Classes:**

- I1: Empty string (“”)
- I2: Whitespace only
- I3: Special characters (“!@# \$ %”)
- I4: Non-matching keywords
- I5: Emojis or Unicode (e.g., *Unicode symbols like emojis*)
- I6: Leading/trailing spaces

### Price Filter:

- Valid: Range within 0–200,  $\min\_price \leq \max\_price$
- Invalid: Negative prices, min > max, missing inputs

## 2.4 Boundary Value Analysis (BVA)

- **Query Length:** 1 (min), 255 (max), 256+ (invalid)
- **`min_price`:** 0 (free product)
- **`max_price`:** 200 (valid), 200.01 (invalid)

## 3 Category Management

### 3.1 Description

- Organize products into hierarchical categories with CRUD operations
- Role: Admin
- Priority: LOW
- Domain: Product categorization system

### 3.2 Inputs and Constraints

- **`name`:** Unique, non-empty string, max length 120
- **`slug`:** Alphanumeric + hyphen, no spaces or special characters
- **`parent_id`:** Optional; must refer to an existing category
- **`category_id`:** Must refer to an existing category

### 3.3 Equivalence Partitioning (EP)

Name:

- Valid: “Mallet”, “Angle Grinder”
- Invalid: Empty, whitespace only, special characters, duplicate names

Slug:

- Valid: “hand-tools”
- Invalid: Contains space or special characters (e.g., “tools!”)

Parent ID:

- Valid: Refers to an existing category
- Invalid: Non-existent or negative IDs

### 3.4 Boundary Value Analysis (BVA)

- **Name length:** 1 (min), 120 (max), 121+ (invalid)
- **Slug length:** Same as name
- **category\_id:** Valid  $\geq 1$  ; invalid = 0, -1

## 4 Shopping Cart Operations

### 4.1 Description

- Add/remove products from cart with quantity controls
- Role: Customer
- Priority: HIGH
- Domain: Cart quantity and item management

### 4.2 Inputs and Constraints

- **product\_id:** Must exist and be in stock
- **quantity:** Integer from 1 to 10
- **action:** add/update/remove
- **user\_session:** Determines cart ownership and behavior

## 4.3 Equivalence Partitioning (EP)

**Quantity:**

- Valid: 1–10 (e.g., 1, 5)
- Invalid: 0, negative, >10, non-integer (e.g., “abc”, “1.5”)

**Product ID:**

- Valid: Existing product
- Invalid: Non-existent or out of stock

**User Session:**

- Valid: Logged-in user
- Invalid: Guest user

## 4.4 Boundary Value Analysis (BVA)

- **Quantity:** 0 (invalid), 1 (min), 10 (max), 11 (invalid)
- **Cart Capacity:** 10 items max; adding 11th fails
- **Stock Limit:** Stock = 5; test input quantity = 6

# 5 Use of AI Tools in Test Design

## 5.1 AI Tool Used

**Tool Name:** ChatGPT (OpenAI, GPT-4)

## 5.2 Prompts Used

- **Prompt:** Generate detailed test cases for the Product Search & Catalog feature, including search by name, filter by price, and keyword relevance. Include edge cases and expected results.

Generate detailed test cases for the Product Search & Catalog feature, including search by name, filter by price, and keyword relevance. Include edge cases and expected results.

Here are detailed test cases for the **Product Search & Catalog** feature, covering **search by name**, **filter by price**, and **keyword relevance**. Each test case includes the test scenario, inputs, expected results, and any edge cases.

### SECTION A: Search by Product Name

Test Case ID	Scenario	Input	Expected Result
TC-A1	Exact match search	"iPhone 13"	Product(s) named "iPhone 13" appear at the top of the list
TC-A2	Partial match search	"iPhone"	All products with "iPhone" in the name appear (e.g., "iPhone 13", "iPhone SE")
TC-A3	Case-insensitive search	"IPHONE 13"	Same result as "iPhone 13"
TC-A4	Typo-tolerant search	"iphon 13" ↓	System suggests or matches "iPhone 13" if fuzzy search is supported

Hỏi bất kỳ điều gì

+ Công cụ

0

Figure 1: Prompt Used to Design Test Cases

- **Prompt:** Generate test cases for the Shopping Cart feature including add to cart, update quantity, remove item. Include test cases for invalid product IDs and max quantity limits.

Generate test cases for the Shopping Cart feature including add to cart, update quantity, remove item. Include test cases for invalid product IDs and max quantity limits.

Here's a comprehensive set of test cases for the Shopping Cart feature, covering:

- Add to cart
- Update quantity
- Remove item
- Invalid inputs (e.g. product ID, quantity limits)

Each test case includes a unique ID, test scenario, input conditions, and expected results.

## SECTION A: Add to Cart

Test Case ID	Scenario	Input	Expected Result
TC-A1	Add valid product to cart	Product ID: 1001, Qty: 1	Product added to cart, cart reflects correct item and total
TC-A2	Add same product twice	Product ID: 1001, Qty: 1 (added twice)	Quantity increments to 2, not duplicated
TC-A3	Add multiple different products	Product ID: 1001 & 2002	Both products appear in cart with correct quantities
TC-A4	Add product with quantity > max allowed	Product ID: 1001, Qty: 101 (max = 100)	Validation error or capped at max limit
TC-A5	Add product with quantity = 0	Product ID: 1001, Qty: 0	Error or no item added
TC-A6	Add product with negative quantity	Product ID: 1001, Qty: -2	Error message shown; item not added

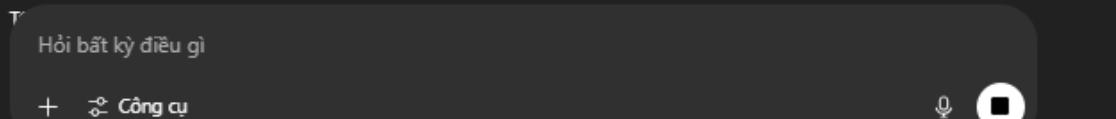


Figure 2: Prompt for Cart Edge and Invalid Cases

- **Prompt:** Design functional and edge test cases for the following features: Product Search & Catalog, Category Management, and Shopping Cart. Include test scenarios, inputs, expected outputs, and identify potential edge cases

Design functional and edge test cases for the following features:  
 Product Search & Catalog, Category Management, and Shopping Cart.  
 Include test scenarios, inputs, expected outputs, and identify potential edge cases

Here is a comprehensive suite of functional and edge test cases for three key e-commerce features:

## 1. Product Search & Catalog

### Functional Test Cases

Test Case ID	Scenario	Input	Expected Output
PS-01	Search by full product name	"iPhone 15 Pro"	Exact product appears in results
PS-02	Search by partial name	"iPhone"	All products with "iPhone" in name returned
PS-03	Search with keyword in description	"ceramic shield"	Products with keyword in description shown
PS-04	Search by product ID	Product ID: 10123	Matching product details returned
PS-05	Filter by price range	Min: \$500, Max: \$1000	Only products within that range shown
PS-06	Sort by price (ascending)	Sort: Low to High	Products ordered correctly
PS-07	Sort by rating	Sort: Top Rated	Products with highest rating appear first
PS-08	View product details	Click product card	Full product page loads with all details
PS-09	Search with no match	"xyz123"	"No results found" message
PS-10	Search with invalid input	"Hỏi bất kỳ điều gì"	Validation error message displayed

+ Công cụ

0



Figure 3: Prompt Design Functional and Edge Test

## 5.3 Validation and Refinement

### 5.3.1 System Behavior Verification

Each AI-generated test case was validated through:

- **Swagger UI:** To verify API parameters, endpoints and response structures.

- **Toolshop frontend interface:** To confirm expected UI behaviors and interactions.
- **Input validation testing:** To ensure the system handles invalid or unexpected inputs gracefully, such as empty fields, incorrect data types, or out-of-range values.

### 5.3.2 Functional Specification Alignment

Test cases were reviewed and aligned with:

- **Course materials:** To confirm consistency with theoretical requirements.
- **Homework specifications:** To meet assignment-specific criteria.
- **Swagger API schemas:** To validate API functionality against documented endpoints and data models.
- **Manual execution in the QA/staging environment:** To verify correct behavior of both frontend and backend components in a controlled environment.

### 5.3.3 Manual Review and Formatting

Test cases were refined to ensure:

- **Clear instructions:** Step-by-step procedures for testers to follow.
- **Accurate inputs and expected outcomes:** Precise test data and anticipated results for each scenario.
- **Consistent terminology:** Standardized terms (e.g., using "quantity" consistently instead of "amount" or "number") across all test cases.

### 5.3.4 Deduplication

Redundant test cases generated by the AI were merged or removed.

*Example:* Multiple boundary value test cases for quantity were combined into a single BVA scenario (e.g., quantity = 1, 10, 11).