

UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

22125005 - MAI XUAN BACH

CS423 - Software Testing

HW#02:

Domain Testing

HCMC – August, 2025

Table of contents

Table of contents.....	2
1. Introduction.....	3
2. Feature Selection.....	3
3. Feature 1: User Registration.....	3
3.1 Step 1: Identify Input & Output Variables.....	3
Input Variables:.....	3
Output Variables:.....	3
3.2 Step 2: Identify Equivalence Classes.....	4
For Email Address Field:.....	4

1. Introduction

This report applies domain testing techniques to analyze three main features from the Practice Software Testing application (Sprint 5 - with bugs version: <https://with-bugs.practicewaretesting.com>). The analysis focuses on three different user types and follows the 4-step domain testing approach: identifying inputs/outputs, identifying equivalence classes, selecting test cases, and performing boundary value analysis.

2. Feature Selection

The three features selected for testing represent different user interactions:

1. User Registration (New Customer) - Testing account creation with input validation
2. Create A Product
3. Product Search (Guest User) - Testing the search functionality available to all visitors

3. Feature 1: User Registration

3.1 Step 1: Identify Input & Output Variables

Input Variables:

- First name (text)
- Last name (text)
- Email address (text)
- Password (text)
- Date of birth (date)

Output Variables:

- Success message "Registration successful"
- Error message "Invalid input"
- Error message "Email already exists"

3.2 Step 2: Identify Equivalence Classes

For Email Address Field:

Condition	Valid Equivalence Classes	Invalid Equivalence Classes
Email format	Valid email format (user@domain.com)	Missing @ symbol, Missing domain, Special characters
Email length	$5 \leq \text{length} \leq 100$ characters	$\text{length} < 5$, $\text{length} > 100$
Email uniqueness	Email not in database	Email already registered

For Password Field:

Condition	Valid Equivalence Classes	Invalid Equivalence Classes
Password length	$8 \leq \text{length} \leq 50$ characters	$\text{length} < 8$, $\text{length} > 50$
Password complexity	Contains uppercase, lowercase, number	Missing required character types

For Date of Birth Field:

Condition	Valid Equivalence Classes	Invalid Equivalence Classes
Age requirement	$\text{Age} \geq 18$ years	$\text{Age} < 18$ years
Date validity	Valid date format (DD/MM/YYYY)	Invalid date (e.g., 32/13/2025)

3.3 Step 3: Selecting Test Cases

In this step, test cases are selected to ensure complete coverage of all identified equivalence classes while minimizing redundancy. According to the domain testing principle, at least one test case should be chosen from each equivalence class, as tests within the same class are expected to produce the same outcome. For the *Register New User* feature, valid classes—such as correctly formatted email addresses, acceptable password lengths, and valid dates of birth—are represented by test cases that together verify the successful registration process. Invalid classes—such as missing “@” in an email, passwords that are too short, or users under 18 years old—are each represented by one test case to confirm the system properly rejects those inputs. This systematic selection strategy ensures that every significant input condition, both valid and invalid, is tested at least once without duplicating effort. As a result, the test suite effectively balances coverage and efficiency, focusing testing effort on the most meaningful input variations.

3.4 Step 4: Boundary Value Analysis

Boundary value analysis (BVA) focuses on testing the edges of input domains, as software systems are most likely to fail at these boundaries rather than within the middle of valid ranges. For the *Register New User* feature, this approach is applied to input fields that have explicit limits or thresholds, such as password length and user age. For example, if the password requirement is between 8 and 50 characters, test cases are designed at the lower and upper limits—7, 8, 9, 49, 50, and 51 characters—to ensure that the system correctly enforces these boundaries. Similarly, for the age requirement of 18 years, test cases are selected just below, at, and above the threshold (ages 17, 18, and 19) to verify proper validation. By emphasizing these edge cases, boundary value analysis increases the likelihood of detecting off-by-one or inequality logic errors, which are common in input validation. This ensures that the registration process handles extreme or limit conditions consistently and reliably.

4. Feature 2: Create A Product

4.1 Step 1: Identify Input & Output Variables Inputs:

Input Variables:

- Product Name (text)
- Description (text)

- Stock (integer)
- Price (decimal)
- Location Offer (checkbox)
- Item for Rent (checkbox)
- Brand (drop-down list)
- Category (drop-down list)
- CO₂ Rating (drop-down list)
- Image (selection field)

Outputs:

- Success message “Product created successfully”
- Error message “Invalid input”
- Field-specific error messages (e.g., “Name is required”, “Price must be greater than zero”, “Stock must be a whole number”)

4.2 Step 2: Identify Equivalence Classes

Condition	Valid Equivalence Classes	Invalid Equivalence Classes
Product Name	$1 \leq \text{length} \leq 100$ characters	length = 0 (empty), length > 100
Description	non-empty text	empty description
Stock	integer $0 \leq \text{stock} \leq 10\,000$	stock < 0, non-integer (e.g. 2.5), exceeds 10 000
Price	decimal > 0 (e.g. 0.01 – 9999.99)	0 or negative value, non-numeric input
Brand	selected from list	missing selection
Category	selected from list	missing selection
CO₂ Rating	selected from defined options (A – E)	missing or undefined rating

Image	valid image file selected	missing image or unsupported file type
-------	---------------------------	--

4.3 Step 3: Selecting Test Cases

In this step, test cases are chosen to represent all identified equivalence classes for the *Create Product* feature while minimizing redundancy. Each input field is analyzed to ensure both valid and invalid conditions are covered at least once. Valid equivalence classes, such as a properly formatted product name, positive numeric price, and integer stock quantity within the allowed range, are represented by test cases that simulate a successful product creation. Invalid equivalence classes, such as an empty name field, negative stock value, or non-numeric price, are each represented by a distinct test case to verify that the system appropriately rejects incorrect inputs. This method follows the principle that executing multiple tests within the same equivalence class would be redundant, as they are expected to yield the same outcome. Therefore, one “best representative” test is selected for each equivalence class to achieve optimal coverage. This ensures that all possible input scenarios—valid, invalid, and edge conditions—are systematically validated while maintaining testing efficiency.

4.4 Step 4: Boundary Value Analysis

Boundary value analysis (BVA) is applied in this step to identify potential defects that may occur at the edges of valid input ranges for the *Create Product* feature. This approach focuses on testing values at, just below, and just above defined boundaries—since errors often arise from incorrect handling of inequality conditions. For example, the **Stock** field, which accepts positive integers, should be tested with values such as 0, 1, and -1 to confirm that the system rejects negative or zero quantities if not allowed. Similarly, for the **Price** field, boundary test cases are created around the lower limit (e.g., 0, 0.01, and 0.02) to ensure that the application properly enforces a minimum positive price. The **Product Name** field, which may have a character limit (e.g., 1–100 characters), should also be tested with boundary inputs of 0, 1, 100, and 101 characters to verify both length validation and user interface constraints. By emphasizing these extreme yet realistic input values, BVA strengthens test coverage and increases the likelihood of detecting off-by-one errors, validation flaws, or incorrect field constraints, ensuring the product creation module performs reliably across all edge conditions.

5. Feature 3: Search And Filter Product

5.1 Step 1: Identify Input & Output Variables Inputs:

Input Variables

- Search query (text string)
- Selected category (checkbox list)
- Selected brand (checkbox list)
- Price range (numeric range: min–max)
- CO₂ rating (radio buttons: A–E)
- Sort option (drop-down)

Output Variables

- List of matching products displayed
- Message “No products found”
- Error message “Invalid input” (if applicable)

5.2 Step 2: Identify Equivalence Classes

Condition	Valid Equivalence Classes	Invalid Equivalence Classes
Search Query	$1 \leq \text{length} \leq 100$ alphanumeric characters	length = 0 (empty), length > 100, special characters only (@#\$%)
Category Selection	one or more categories selected from valid list	invalid category name, no category selected (when required)
Brand Selection	valid brand selected	non-existent brand or none selected (if required)
Price Range	$1 \leq \text{min} < \text{max} \leq 200$	$\text{min} \geq \text{max}$, negative values, non-numeric input

CO₂ Rating	selected from defined options (A–E)	undefined value or missing selection
Sort Option	valid option selected (e.g., price, name)	invalid or undefined sort option

5.3 Step 3: Selecting Test Cases

In this step, test cases are selected to represent each equivalence class while avoiding redundancy. For valid classes, test cases are chosen to ensure the full functionality of the search and filter feature—for example, entering a valid keyword, selecting one or more categories, and defining a valid price range should display the corresponding list of products. For invalid classes, each condition such as an empty search, reversed price range (e.g., min > max), or invalid characters in the search query is represented by one test case to verify that the system handles incorrect inputs gracefully. This approach ensures that each type of input condition is validated at least once while keeping the total number of tests efficient and manageable.

5.4 Step 4: Boundary Value Analysis

Boundary value analysis focuses on the limits of user inputs where errors are most likely to occur. For the *Search and Filter* feature, boundaries are applied mainly to numeric and text-based fields. The **search query** length is tested at 0, 1, 2, 99, 100, and 101 characters to verify that empty and overly long queries are handled properly. The **price range** input is tested around its numeric limits, such as (1–200), (0–200), (1–201), and (100–100), to confirm correct validation and behavior at boundary edges. For the **filter options**, testing includes selecting none, one, and multiple filters to ensure proper logical handling of selection combinations. Through boundary testing, potential off-by-one or input validation errors can be detected, ensuring that the system performs correctly under both normal and extreme input conditions.

Fit's Mantis System

[Print Reports] [CSV Export] [Excel Export] [Graph]									
[Search] [Apply Filter] [Advanced Filters] [Reset Filter]									
Viewing Issues (1 - 19 / 19) [Print Reports] [CSV Export] [Excel Export] [Graph]									
	P	ID	#		Category	Severity	Status	Updated	Summary
<input type="checkbox"/>	^	0056074			APCS	feature	new	2025-11-05	In the category page, name field, we can inject the sql statement
<input type="checkbox"/>	^	0056072			APCS	feature	new	2025-11-05	The price adjustment bar does not filter the product list based on price
<input type="checkbox"/>	^	0056070			APCS	major	new	2025-11-05	In product list page, sort by name (alphabet) is in reverse order
<input type="checkbox"/>	^	0056042	1		APCS	major	new	2025-11-05	/invoices?in=status,ON_HOLD returns invoices with other statuses
<input type="checkbox"/>	-	0056039	1		APCS	minor	new	2025-11-05	Status filter leaks invoices not in AWAITING_FULFILLMENT.
<input type="checkbox"/>	^	0056033	1		APCS	major	new	2025-11-05	Page 0 and Page 1 responses contain the same product IDs
<input type="checkbox"/>	-	0056032	1		APCS	minor	new	2025-11-05	between=price,abc,10 returns items instead of 4xx or empty set
<input type="checkbox"/>	-	0056031	1		APCS	minor	new	2025-11-05	between=price,abc returns normal product list instead of 4xx or empty
<input type="checkbox"/>	-	0056030	1		APCS	major	new	2025-11-05	between=price,100 still returns many items priced < 100
<input type="checkbox"/>	⌵	0056028	1		APCS	major	new	2025-11-05	between=price,<low> returns items priced below the given lower bound
<input type="checkbox"/>	⌵	0055947			General	major	new	2025-11-04	Admin can set stock and price to negative value
<input type="checkbox"/>	^	0055946			General	major	new	2025-11-04	Sort a-z is showing z-a, sort z-a is showing a-z
<input type="checkbox"/>	^	0055945			General	major	new	2025-11-04	Sort high-low is showing low-high, sort low-high is showing high-low
<input type="checkbox"/>	-	0055944			General	major	new	2025-11-04	Product Title switch to new position when typing new character in the search bar
<input type="checkbox"/>	-	0055943			General	text	new	2025-11-04	Typo "Sorth" instead of "Sort" in product page
<input type="checkbox"/>	^	0055942			General	feature	new	2025-11-04	Password with 9 characters still pass
<input type="checkbox"/>	-	0055941			General	feature	new	2025-11-04	User whose age is 76 can still register account
<input type="checkbox"/>	-	0055940			General	major	new	2025-11-04	Customers who are 17 years old and 364 days can still register account
<input type="checkbox"/>	^	0055915	1		APCS	minor	new	2025-11-04	Login API test fails due to leading whitespace in the email field during domain testing.
<input type="checkbox"/> Select All <input type="button" value="Copy"/> <input type="button" value="OK"/>									
new feedback acknowledged confirmed assigned resolved closed									

6. Self Assessment

Self-Assessment Report

Criteria	Outcomes (Brief description about what you get/trouble from each requirement)	Grade	Self-Assessed Grade
<u>1</u>	<u>Name of Feature 1</u>	<u>40</u>	<u>40</u>
	1.1 Domain Analysis	5	5
	1.2 Test Cases (30)	20	20
	1.3 Boundary Analysis	5	5
	1.4 Bugs	10	10
<u>2</u>	<u>Name of Feature 2</u>	<u>30</u>	<u>30</u>
	2.1 Domain Analysis	5	5
	2.2 Test Cases (30)	15	15
	2.3 Boundary Analysis	5	5
	2.4 Bugs	5	5
<u>3</u>	<u>Name of Feature 3</u>	<u>30</u>	<u>30</u>
	3.1 Domain Analysis	5	5
	3.2 Test Cases (30)	15	15
	3.3 Boundary Analysis	5	5
	3.4 Bugs	5	5
	<u>Total</u>	<u>100</u>	<u>100</u>