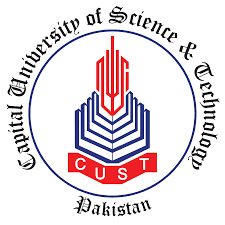
**ASSIGNMENT 3**

****

**Group Members**

**BSE181010 Husnat Khalid Raja**

**BSE181030 Muhammad Usman**

**SOFTWARE QUALITY ENGINEERING**

**SUBMITTED TO: Samir Obaid**

# Table of content

[**CASE STUDY**](#_vr6xpftw5mp6) **4**

[**FUNCTIONS**](#_32g7q6i9bpnk) **4**

[**BVA:**](#_yzkpj6hy4udw) **5**

[Credit Card(PIN)](#_3hgrtmkgibel) 5

[Transaction(ID,PASSWORD)](#_od4fmmidze4x) 6

[**RBVA:**](#_9sppxq1m3my6) **7**

[Transaction(ID,PASSWORD)](#_m2f98vph1d64) 7

[Credit Card(PIN)](#_oj4u5fh1jh78) 8

[**Robust Worst-Case BVA:**](#_xj59rrdri9uo) **9**

[Transaction(ID,PASSWORD)](#_k5wy1jhpf23j) 9

[Credit Card(PIN)](#_9d636e52jhvk) 10

[**Worst-Case BVA:**](#_2nicqkvpoqf6) **11**

[Transaction(ID,PASSWORD)](#_xkdzuiy74ri2) 11

[Credit Card(PIN)](#_2ezkfhe72sbk) 11

[**Equivalence Class Partitioning**](#_h9w41mrwtt73) **12**

[Strong Robust Equivalence Class:](#_aqd0rbu96unt) 12

[Credit Card(PIN)](#_llvp0ez4qtbo) 13

[Transaction(ID,PASSWORD)](#_ch4phg3y6ge8) 13

[**Assignment 3**](#_6pwv7heqm4bz) **14**

[Graph](#_ahfop4gzk9s9) 14

[Identifying Test Cases](#_efulbji13u9x) 16

# CASE STUDY

A company ABC wants to develop an automated sales recording application. The application shall enable the cashier to record the sales of items for the customers. Whenever a customer arrives on counter, the cashier starts new sale process. The customer provides items to the cashier and the cashier enters item identity and quantity. The system describes item description, unit price and subtotal. The cashier continues this activity until customer is finished with the items. Once there are no more items, the cashier enters finish option and the system displays total sale with taxes. The cashier takes cash amount or credit card from the customer whether the customer likes to pay by cash or by credit card. The cashier enters cash amount and the system deduct sales total from amount and shows remaining. The system saves transection to the log, update inventory, generate receipt and the cashier returns remaining amount to customer. If the customer wants to pay by credit card, the cashier enters credit card number along with sale total and the system verifies credit card details and deduct amount via online credit card verification system. Once done, the system logs transection details and update inventory. A company also provides gift cards to their customers. Each gift card worth rupees 100 and a customer can also pay via gift card if he/she has plenty of gift cards and they are available with customer time of shopping. It means the system shall also enable the customer to pay via gift card. In this case, the cashier selects gift card payment option and the system asks for gift card number. The system verifies gift card number and salves transection log and update inventory. A manager shall be able to handle returned item from the customers. In order to return items, the manager starts return item process. The system demands item identifier and manager provides item identifier. The system asks to enter complaint statement and manager writes down complaint statement. The system updates transection details and describes case amount to be returned to the customer. The system also updates inventory details and marks status of an item as rejected.

# FUNCTIONS

System

Customer(NAME,ID)

Credit Card(PIN)

Transaction(ID,PASSWORD)

# BVA:

In BVA testing we test every single possible combination. Cases are calculated by the formula 4(n)+1 where n is the number of variables.

## Credit Card(PIN)

Total test cases 4(1)+1=5, pin range should be 4 digit characters not more or less

min = 000, min+1 =001 , normal =546 , max-1 = 998, max =999

|  |  |  |
| --- | --- | --- |
| **Case** | **Pin** | **Expected output** |
| 1 | 000 | valid |
| 2 | 001 | Valid |
| 3 | 546 | valid |
| 4 | 9991 | invalid |
| 5 | 100 | valid |

## Transaction(ID,PASSWORD)

Total test cases 4(1)+1=5, only binary values will be valid

min = 000, min+1 =001 , normal =546 , max-1 = 999, max =999

|  |  |  |
| --- | --- | --- |
| **Case** | **Pin** | **Expected output** |
| 1 | 000 | valid |
| 2 | 001 | Valid |
| 3 | 5466 | Invalid |
| 4 | 999 | Invalid |
| 5 | 100 | valid |

# RBVA:

In RBVA testing we test every single possible combination. Cases are calculated by the formula 6(n)+1 where n is the number of variables.

## Transaction(ID,PASSWORD)

Total test cases 5^3=125,

min = 000, min+1 =001 , normal =546 , max-1 = 998, max =999

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Case** | **OrderID** | **Quantity** | **Password** | **Expected output** |
| 1 | 000 | 1 | 0012 | valid |
| 2 | 001 | 4 | 0032 | valid |
| 3 | 002 | 3 | 0012 | valid |
| 4 | 003 | 3 | 0023 | valid |
| 5 | 0004 | 7 | 0023 | invaild |
| 6 | 0005 | 4 | 0023 | invaild |
| 7 | 0006 | 5 | 0023 | invalid |
| 8 | 0007 | 3 | 0023 | invalid |
| 9 | 0008 | 2 | 0034 | invalid |
| 10 | 0009 | 12 | 0033 | invalid |
| 11 | 0010 | 4 | 0021 | invaild |
| 12 | 0011 | 14 | 0014 | invalid |
| 13 | 0012 | 3 | 0017 | invalid |
| 14 | 0013 | 5 | 0018 | invalid |
| 15 | 0014 | 7 | 0062 | invalid |
| 16 | 0015 | 4 | 0071 | invalid |
| 17 | 0016 | 2 | 0019 | invalid |
| 18 | 0017 | 6 | 0023 | invalid |
| 19 | 0018 | 8 | 3434 | invalid |
| 20 | 0019 | 5 | 3453 | invalid |
| 21 | 0020 | 14 | 2342 | invalid |

In Robust Worst-Case BVA testing we test every single possible combination. Cases are calculated by the formula 7^n (7 power n) where n is the number of variables. To generate test cases first we choose 5 numbers between the given boundary values (min-1, min, min+1, normal, max-1, max, max+1 ).

## Credit Card(PIN)

Total test cases 6(1)+1=7, password range should be 4 digit characters not more or less

min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999

|  |  |  |
| --- | --- | --- |
| **Case** | **Pin** | **Expected output** |
| 1 | 0000 | valid |
| 2 | 0001 | Valid |
| 3 | 5466 | valid |
| 4 | 99991 | invalid |
| 5 | 1000 | valid |
| 6 | 1124 | valid |
| 7 | 4444 | valid |

# Robust Worst-Case BVA:

In Robust Worst-Case BVA testing we test every single possible combination. Cases are calculated by the formula 7^n (7 power n) where n is the number of variables. To generate test cases first we choose 5 numbers between the given boundary values (min-1, min, min+1, normal, max-1, max, max+1 ).

## Transaction(ID,PASSWORD)

Total test cases 7^3=343,

min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999d

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Case** | **OrderID** | **Quantity** | **Password** | **Expected output** |
| 1 | -0001 | 3 | 0012 | Invalid |
| 2 | 0000 | 1 | 0032 | valid |
| 3 | 0001 | 4 | 0012 | invalid |
| 4 | 0002 | 3 | 0023 | invaild |
| 5 | 0003 | 3 | 0023 | invaild |
| 6 | 0004 | 7 | 0023 | invaild |
| 7 | 0005 | 4 | 0023 | invaild |
| 8 | 0006 | 5 | 0023 | invalid |
| 9 | 0007 | 3 | 0034 | invalid |
| 10 | 0008 | 2 | 0033 | invalid |
| 11 | 0009 | 12 | 0021 | invalid |
| 12 | 0010 | 4 | 0014 | invalid |
| 13 | 0011 | 14 | 0017 | invalid |
| 14 | 0012 | 3 | 0018 | invalid |
| 15 | 0013 | 5 | 0062 | invalid |
| 16 | 0014 | 7 | 0071 | invalid |
| 17 | 0015 | 4 | 0019 | invalid |
| 18 | 0016 | 2 | 0023 | invalid |
| 19 | 0017 | 6 | 3434 | invalid |
| 20 | 0018 | 8 | 3453 | invalid |
| 21 | 0019 | 5 | 0063 | invalid |
| 22 | 0020 | 14 | 0087 | invalid |

## Credit Card(PIN)

Total test cases 7^1=7, password range should be 4 digit characters not more or less

Min-1 = -0001, min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999 max+1=10000

|  |  |  |
| --- | --- | --- |
| **Case** | **Pin** | **Expected output** |
| 1 | -0001 | Invalid |
| 2 | 0000 | valid |
| 3 | 0001 | Valid |
| 4 | 5466 | valid |
| 5 | 19999 | invalid |
| 6 | 1000 | valid |
| 7 | 10000 | Invalid |

# Worst-Case BVA:

In Worst-Case BVA testing we test every single possible combination. Cases are calculated by the formula 5^n (5 power n) where n is the number of variables. To generate test cases first we choose 5 numbers between the given boundary values (min, min+1, normal, max-1, max).

## Transaction(ID,PASSWORD)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0015 | 4 | 0071 | invalid |
| 2 | 0016 | 2 | 0019 | invalid |
| 3 | 0017 | 6 | 0023 | invalid |
| 4 | 0018 | 8 | 3434 | invalid |
| 5 | 0019 | 5 | 3453 | invalid |
| 6 | 0020 | 14 | 2342 | invalid |

## Credit Card(PIN)

Total test cases 5^1=5, password range should be 4 digit characters not more or less

min = 0000, min+1 =0001 , normal =5555 , max-1 = 9998, max =9999

|  |  |  |
| --- | --- | --- |
| **Case** | **Pin** | **Expected output** |
| 1 | 0000 | valid |
| 2 | 0001 | Valid |
| 3 | 5466 | valid |

# Equivalence Class Partitioning

## Strong Robust Equivalence Class:

Void Customer (name,id)

min=000

min+1=001

normal=250

max-1=998

max=999

|  |  |  |
| --- | --- | --- |
| Case | Customer id | output |
| 1 | 11 | invalid |
| 2 | 111 | valid |
| 3 | 1000 | invalid |

## Credit Card(PIN)

min = 000,

min+1 =001 ,

normal =555 ,

max-1 = 998,

max =999

|  |  |  |
| --- | --- | --- |
| **Case** | **Pin** | **Expected output** |
| 1 | 0003 | invalid |
| 2 | 098 | valid |
| 3 | 546 | valid |

## Transaction(ID,PASSWORD)

min = 000,

min+1 =001 ,

normal =546 ,

max-1 = 998,

max =999

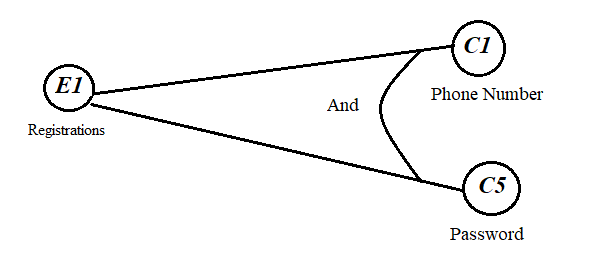
|  |  |  |
| --- | --- | --- |
| **Case** | **Pin** | **Expected output** |
| 1 | 002 | valid |
| 2 | 01 | invalid |
| 3 | 542 | valid |

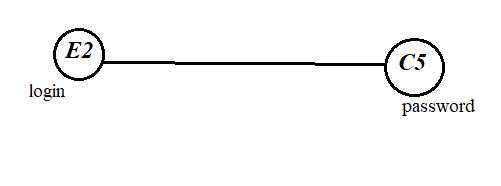
|  |  |  |
| --- | --- | --- |
| 4 | 98 | invalid |
| 5 | 671 | valid |
| 6 | 456 | valid |
| 7 | 671 | valid |

# 

# **Assignment 3**

## **Graph**





|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test cases** | **T1** | **T2** | **T3** | **T4** | **T5** |
| C1: Phone Number | 1 | 0 | 0 | 0 | 0 |
| C5: Password | 1 | 1 | 0 | 0 | 0 |
| C6: Credit Card | 0 | 0 | 0 | 0 | 1 |
| E1: Registration | 1 | 0 | 0 | 0 | 0 |
| E2: Login | 0 | 1 | 0 | 0 | 0 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Identifying Test Cases

|  |  |  |
| --- | --- | --- |
| **Test cases** | **Input(cause)** | **Expected Output**  **(effects)** |
| 1 | Phone Number **&** Password | Get Registered |
| 2 | Password | Get Login |
|  |  |  |
|  |  |  |
|  |  |  |

### 