Capital University Of Science & Technology

Group Members

BSE181009 ALEENA AKHTAR
BSE181017 SANA MUNIR

ASSIGNMENT 3

SOFTWARE QUALITY ENGINEERING

SUBMITTED TO: Samir Obaid

Capital University Of Science & Technology

Table Of Content

Explanation:

As industries are fast expanding, people are seeking more ways to purchase products with much ease and still maintain cost-effectiveness. Food can be ordered through the internet and payment made without going to the restaurant or the food vendor. For this system, the User will get sign up from his Number and can enter his menu ID, Quantity and password then bill will be added to their cart

Case Study:

An XYZ food company wants to develop an app that provides food delivery at your door in very little time and with the best packaging. Providing food from every famous food place near you. Order food with the best user experience. The online food ordering system is one of the latest servicers most fast-food restaurants in the western world are adopting. With this method, food is ordered online and delivered to the customer. This is made possible through the use of an electronic payment system. Customers pay with their credit cards, although credit card customers can be served even before they make payment either through cash or cheque. So, the system designed in this project will enable customers to go online and place an order for their food.

Due to the great increase in the awareness of the internet and the technologies associated with it, several opportunities are coming up on the web. So many businesses and companies now venture into their business with ease because of the internet. One such business that the internet introduced is an online food ordering system. In today's age of fast food and take out, many restaurants have chosen to focus on quick preparation and speedy delivery of orders rather than offering a rich dining experience. Until recently, most of these delivery orders were placed over the phone.

This application helps the restaurants to do all functionalities more accurately and faster way. Food Ordering System reduces manual works and improves the efficiency of restaurants. This application is helping Food Ordering s to maintain the stock and cash flows and there are many more functionalities, like.

- To store records.
- Control orders and services.
- Billings.
- Control staff and their shifting.
- Control multiple branches.
- Helps Manager to control each part of the restaurant.

For the System user will have to register to the app and will and to sign up. He has to add his credentials ***(PhoneNO, password)*** in order to register to the system, Once the user gets register to th system then they can only login by there password , system will remember there number. If a user is new he has to add his number first. Users can select Food ID from the menu and can Add it to the cart and will have to specify his food quantity.

Functions:

1. ***Void CancelOrder(Int OrderID):***
   This function will help the user to cancel the food ID that has been added to the food cart. User will have to enter a 4-digit ID

2. ***Void CreateOrder(int OrderID,int Quantity,int Password)***
   In this function, the user will enter a 4-digit of food ID that he wants to order then he will enter the quantity of the food that wants to order that should not greater than 10 and in the last, he will enter his password again to confirm his order

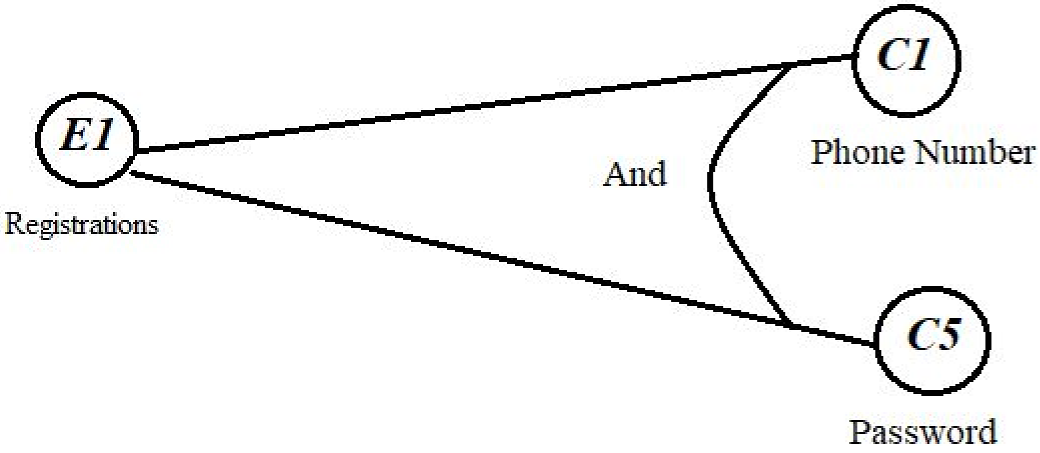
3. **Void EnterQuantity(int Pin)**
   This function will ask the user to enter a pin then a bill will be added to their cart. User will have to enter a 4-digit pin

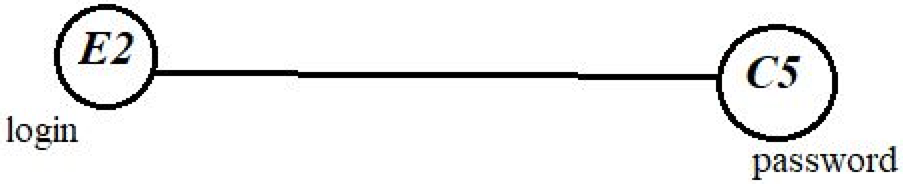
Capital University Of Science & Technology

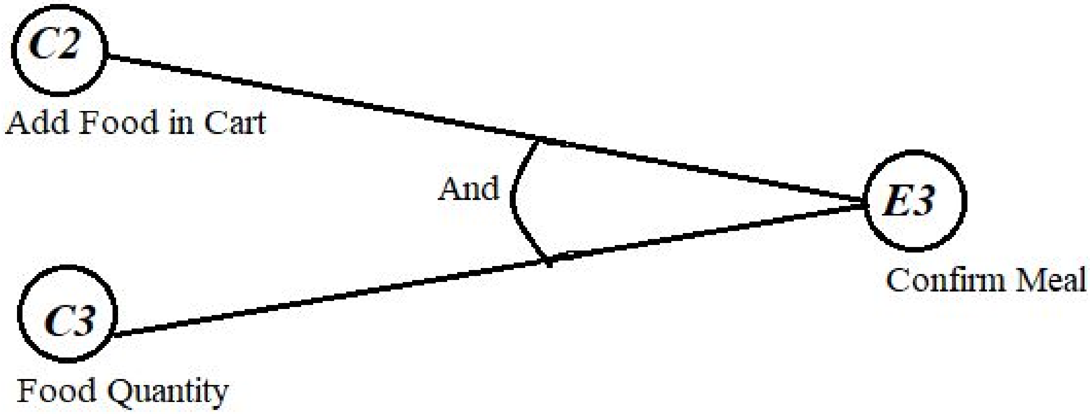
Assignment 3

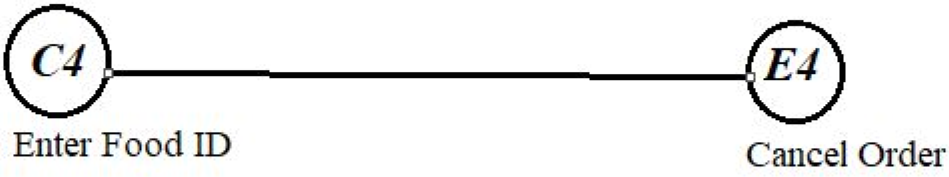
Cause	***Effects***
C1: Phone Number	E1: Registration
C2: Add Food in Cart	E2: Login
C3: Food Quantity	E3: Confirm Meal
C4: Enter Food ID	E4: Cancel Order
C5: Password	E5: Confirm Bill
C6: Credit Card	
C7: Cash on delivery	

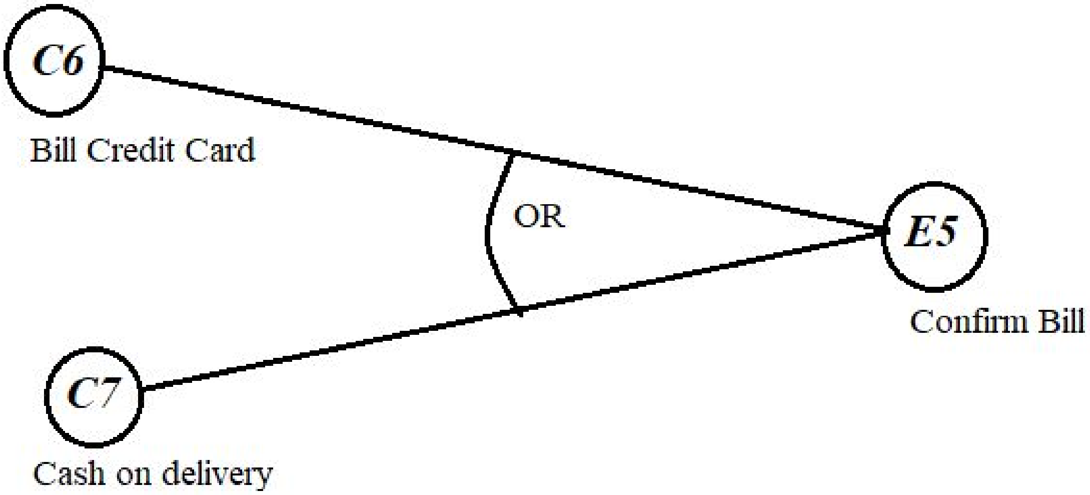
Graphs


E1
Registrations
C1
Phone Number
And
C5
Password


E2
login
C5
password


C2
Add Food in Cart
And
C3
Food Quantity
E3
Confirm Meal


C4
Enter Food ID
E4
Cancel Order


C6
Bill Credit Card
OR
E5
Confirm Bill
C7
Cash on delivery

Decision Table

Test cases	**T1**	**T2**	**T3**	**T4**	**T5**
C1: Phone Number	1	0	0	0	0
C2: Add Food in Cart	0	0	0	0	0
C3: Food Quantity	0	0	1	0	0
C4: Enter Food ID	0	0	1	1	0
C5: Password	1	1	0	0	0
C6: Credit Card	0	0	0	0	1
C7: Cash on delivery	0	0	0	0	1
E1: Registration	1	0	0	0	0
E2: Login	0	1	0	0	0
E3: Confirm Meal	0	0	1	0	0
E4: Cancel Order	0	0	0	1	0
E5: Confirm Bill	0	0	0	0	1

Identifying Test Cases

Test cases	**Input(cause)**	**Expected Output (effects)**
1	Phone Number **&** Password	Get Registered
2	Password	Get Login
3	Food Quantity **&** Enter Food ID	Meal Confirmed
4	Enter Food ID	Meal Canceled
5	Credit Card **Or** Cash On delivery	Confirm Bill

Test Cases

We are using boundary value analysis from which we get our expected results

Test cases	**Input(cause)**		**Expected Output (effects)**
1	Phone number	<=0000 & >=9999	Get Registered
2	<=0000 & >=9999		Get Login
3	<=1 & >=10		Meal Confirmed
4	<=0000 & >=9999		Meal Canceled
5	<=0000 & >=9999	Cash On delivery	Confirm Bill

Capital University Of Science & Technology

Assignment no 2

Mistake of assignment 2	Did not find any mistake in assignment 2

Equivalence Class Partitioning

Strong Robust Equivalence Class:

In Equivalence Class Partitioning we will see working of Strong Robust Equivalence Class. In this function we test beyond the boundaries. Strong robust testing produces test cases for all valid and invalid elements of the product of the equivalence class. However, it is incapable of reducing the redundancy in testing.

1. ***Void CancelOrder(Int OrderID):***

   For Order Id and Password

   - Below boundary (less than 0000)
   - Within boundary (0000-9999)
   - Beyond boundary (greater than 9999)

Case	**Order ID**
1	123
2	1111
3	10000

2. **Void EnterQuantity(int Pin)**

   For Order Id and Password

   - Below boundary (less than 0000)
   - Within boundary (0000-9999)
   - Beyond boundary (greater than 9999)

Case	**Pin**
1	123
2	1211
3	10000

3. ***Void CreateOrder(int OrderID,int Quantity,int Password)***

For Order Id and Password

- Below boundary (less than 0000)
- Within boundary (0000-9999)
- Beyond boundary (greater than 9999)

For Quantity

- Below boundary (less 1)
- Within boundary (1-10)
- Beyond boundary (greater than 10)

No of Case	**Test cases**
1	123,0,123
2	123,10,123
3	123,0,0000
4	123,10,0000
5	0000,10,0000
6	0000,0,123
7	0000,10,123
8	0000,0,0000
9	10000,13,10000
10	10000,1,10000
11	10000,13,10000
12	0000,13,10000
13	0000,13,0000

14	10000,1,0000
15	0000,1,10000

Assignment no 1

Mistake in Assignment 1	
1	In **BVA** we added random numbers, but have add within Boundaries
2	In **RBVA** we added completely wrong and random test cases
3	In **worst case BVA** we added completely wrong and random test cases
4	In **Robust worst case BVA** we added completely wrong and random test cases

BLACK BOX TESTING

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.

1. ***Void EnterPhoneNo(Int Password)***
   Total test cases 4(1)+1=5, 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**
1	0000

2	0001
3	5466
4	9998
5	9999

2. ***Void CancelOrder(Int OrderID):***
   Total test cases 4(1)+1=5, only binary values will be valid
   min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999

Case	Order ID
1	0000
2	0001
3	5466
4	9998
5	9999

3. ***Void CreateOrder(int OrderID,int Quantity,int Password)***
   Total test cases 4(3)+1=13,
   For Quantity min=1, min+1=2 normal=5 max-1=9, max =10
   For Password min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999
   For OrderID min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999

No Of Cases	Test cases
1	5,0000,0000
2	5,0001,0001

3	5,5466,5466
4	5,9998,9998
5	5,9999,9999
6	0001,1,0000
7	0001,2,0001
8	0001,5,5466
9	0001,9,9998
10	0001,10,9999
11	5466,2,0001
12	5466,9,9998
13	5466,10,9999

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.

1. ***Void EnterPhoneNo(Int Password)***
   Total test cases 6(1)+1=7, password range should be 4 digit characters not more or less min-1=999 min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999 max+1=10000

Case	**Pin**
1 (below)	999
2	0000
3	0001

4	5466
5	9998
6	9999
7 (above)	10000

2. ***Void CancelOrder(Int OrderID):***
   Total test cases 6(1)+1=7, only binary values will be valid
   min-1=999 min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999
   max+1=10000

Case	OrderID
1 (below)	999
2	0000
3	0001
4	5466
5	9998
6	9999
7 (above)	10000

3. ***Void CreateOrder(int OrderID,int Quantity,int Password)***
   Total test cases 6(3)+1=19,
   For Quantity min-1=0, min=1, min+1=2 normal=5 max-1=9, max =10 max+1=11
   For Password and order Id
   min-1=999 min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999
   max+1=10000

No of Case	**Test cases**
1	5,999,999
2	5,0000,0000
3	5,0001,0001
4	5,5466,5466
5	5,9998,9998
6	5,9999,9999
7	5,10000,10000
8	0001,0,999
9	0001,1,0000
10	0001,2,0001
11	0001,5,5466
12	0001,9,9998
13	0001,10,9999
14	0001,11,10000
15	5466,0,999
16	5466,2,0001
17	5466,9,9998
18	5466,10,9999
19	5466,11,10000

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).

Capital University Of Science & Technology

1. ***Void EnterPhoneNo(Int Password)***

   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**
1	0000
2	0001
3	5466
4	9998
5	9999

2. ***Void CancelOrder(Int OrderID):***

   Total test cases 5^1=5, only binary values will be valid
   min = 0000, min+1 =0001 , normal =5466 , max-1 = 9999, max =9999

Case	**Order ID**
1	0000
2	0001
3	5466
4	9998
5	9999

3. ***Void CreateOrder(int OrderID,int Quantity,int Password)***
Total test cases 5^3=125,
For Quantity min=1 normal=5 max=10
min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999

No of cases	Test cases
1	1,0000,0000
2	1,0001,0001
3	1,5466,5466
4	1,9998,9998
5	1,9999,9999
6	2,0000,0000
7	2,0001,0001
8	2,5466,5466
9	2,9998,9998
10	2,9999,9999
11	5,0000,0000
12	5,0001,0001
13	5,5466,5466
14	5,9998,9998
15	5,9999,9999
16	9,0000,0000
17	9,0001,0001
18	9,5466,5466
19	9,9998,9998
20	9,9999,9999
21	10,0000,0000

22	10,0001,0001
23	10,5466,5466
24	10,9998,9998
25	10,9999,9999
26	0000,2,0001
27	0000,5,5466
28	0000,9,9998
29	0000,10,9999
30	0001,1,0000
31	0001,5,5466
32	0001,9,9998
33	0001,10,9999
34	5466,1,0000
35	5466,2,0001
36	5466,9,9998
37	5466,10,9999
38	9998,1,0000
39	9998,2,0001
40	9998,5,5466
41	9998,10,9999
42	9999,1,0000
43	9999,2,0001
44	9999,5,5466
45	9999,9,9998
46	0000,2,0001
47	0000,5,5466

48	0000,9,9998
49	0000,10,9999
50	0001,1,0000
51	0001,5,5466
52	0001,9,9998
53	0001,10,9999
54	5466,1,0000
55	5466,2,0001
56	5466,9,9998
57	5466,10,9999
58	9998,1,0000
59	9998,2,0001
60	9998,5,5466
61	9998,10,9999
62	9999,1,0000

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).

1. ***Void EnterPhoneNo(Int Password)***
   Total test cases 7^1=7, password range should be 4 digit characters not more or less
   For Password and order Id
   min-1=999 min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999
   max+1=10000

Case	**Pin**

1	999
2	0000
3	0001
4	5466
5	9998
6	9999
7	10000

2. ***Void CancelOrder(Int OrderID):***
   Total test cases 7^1=7, only binary values will be valid
   Min-1 = 999, min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999
   max+1=10000

Case	**Pin**
1	999
2	0000
3	0001
4	5466
5	9998
6	9999

7	10000

3. ***Void CreateOrder(int OrderID,int Quantity,int Password)***
Total test cases 7^3=343,
For Quantity min-1=0, min=1, min+1=2 normal=5 max-1=9, max =10 max+1=11
For Password and order Id
min-1=999 min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999 max+1=10000

No of cases	**Test cases**
1	0,999,999
2	0,999,0000
3	0,999,0001
4	0,999,5466
5	0,999,9998
6	0,999,9999
7	0,999,10000
8	0,0000,999
9	0,0000,0000
10	0,0000,0001
11	0,0000,5466
12	0,0000,9998
13	0,0000,9999
14	0,0000,10000
15	0,0001,999
16	0,0001,0000
17	0,0001,0001
18	0,0001,5466
19	0,0001,9998

20	0,0001,9999
21	0,0001,10000
22	0,5466,999
23	0,5466,0000
24	0,5466,0001
25	0,5466,5466
26	0,5466,9998
27	0,5466,9999
28	0,5466,10000
29	0,9998,999
30	0,9998,0000
31	0,9998,0001
32	0,9998,5466
33	0,9998,9998
34	0,9998,9999
35	0,9998,10000
36	0,9999,999
37	0,9999,0000
38	0,9999,0001
39	0,9999,5466
40	0,9999,9998
41	0,9999,9999
42	0,9999,10000
43	0,10000,999
44	0,10000,0000
45	0,10000,0001

46	0,10000,5466
47	0,10000,9998
48	0,10000,9999
49	0,10000,10000
50	1,999,999
51	1,999,0000
52	1,999,0001
53	1,999,5466
54	1,999,9998
55	1,999,9999
56	1,999,10000
57	1,0000,999
58	1,0000,0000
59	1,0000,0001
60	1,0000,5466
61	1,0000,9998
62	1,0000,9999
63	1,0000,10000
67	1,0001,999
68	1,0001,0000
69	1,0001,0001
70	1,0001,5466
71	1,0001,9998
72	1,0001,9999
73	1,0001,10000
74	1,5466,999
76	1,5466,0000

77	1,5466,0001
78	1,5466,5466
79	1,5466,9998
80	1,5466,9999
81	1,5466,10000
82	1,9998,999
83	1,9998,0000
84	1,9998,0001
85	1,9998,5466
86	1,9998,9998
87	1,9998,9999
88	1,9998,10000
89	1,9999,999
90	1,9999,0000
91	1,9999,0001
92	1,9999,5466
93	1,9999,9998
94	1,9999,9999
95	1,9999,10000
96	1,10000,999
97	1,10000,0000
98	1,10000,0001
99	1,10000,5466
100	1,10000,9998
101	1,10000,9999
102	1,10000,10000

103	2,999,999
104	2,999,0000
105	2,999,0001
106	2,999,5466
107	2,999,9998
108	2,999,9999
109	2,999,10000
110	2,0000,999
111	2,0000,0000
112	2,0000,0001
113	2,0000,5466
114	2,0000,9998
115	2,0000,9999
116	2,0000,10000
117	2,0001,999
118	2,0001,0000
119	2,0001,0001
120	2,0001,5466
121	2,0001,9998
122	2,0001,9999
123	2,0001,10000
124	2,5466,999
125	2,5466,0000
126	2,5466,0001
127	2,5466,5466
128	2,5466,9998

129	2,5466,9999
130	2,5466,10000
131	2,9998,999
132	2,9998,0000
133	2,9998,0001
134	2,9998,5466
135	2,9998,9998
136	2,9998,9999
137	2,9998,10000
138	2,9999,999
139	2,9999,0000
140	2,9999,0001
141	2,9999,5466
142	2,9999,9998
143	2,9999,9999
144	2,9999,10000
145	2,10000,999
146	2,10000,0000
147	2,10000,0001
148	2,10000,5466
140	2,10000,9998
150	2,10000,9999
151	2,10000,10000
152	5,999,999
153	5,999,0000

154	5,999,0001
156	5,999,5466
157	5,999,9998
158	5,999,9999
159	5,999,10000
160	5,0000,999
161	5,0000,0000
162	5,0000,0001
163	5,0000,5466
164	5,0000,9998
165	5,0000,9999
166	5,0000,10000
167	5,0001,999
168	5,0001,0000
169	5,0001,0001
170	5,0001,5466
171	5,0001,9998
172	5,0001,9999