Capital University Of Science & Technology

Group Members

BSE181009 ALEENA AKHTAR
BSE181017 SANA MUNIR

ASSIGNMENT 2

SOFTWARE QUALITY ENGINEERING

SUBMITTED TO: Samir Obaid

Capital University Of Science & Technology

Table Of Content

Capital University Of Science & Technology

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 login to the system, Once the user gets logged to

their ID. 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. One main feature if this system is that the user will have to add at least one cold drink to his menu.

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

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	Expected output
1	123	invaild
2	1111	valid
3	10000	invaild

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	Expected output
1	123	invaild
2	1211	valid
3	10000	invaild

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	Expected output
1	123,0,123	invalid
2	123,10,123	invalid
3	123,0,0000	invalid
4	123,10,0000	invalid
5	0000,10,0000	Valid
6	0000,0,123	invalid
7	0000,10,123	invalid
8	0000,0,0000	invalid

9	10000,13,10000	invalid
10	10000,1,10000	invalid
11	10000,13,10000	invalid
12	0000,13,10000	invalid
13	0000,13,0000	invalid
14	10000,1,0000	invalid
15	0000,1,10000	invalid

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	Expected output
1	0000	valid
2	0001	Valid
3	5466	valid
4	99991	invalid
5	1000	valid

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 = 9999, max =9999

Case	Order ID	Expected output
1	0000	valid
2	0001	Valid
3	5466	Invalid
4	9999	Invalid
5	1000	valid

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

Total test cases 4(3)+1=13,
For Quantity min=1 normal=5 max=10
min = 0000, min+1 =0001 , normal =5466 , max-1 = 9998, max =9999

Case	OrderID	Quantity	Password	Expected output
1	0000	1	0000	valid
2	0001	4	0021	invalid
3	0002	3	0025	invaild
4	0003	3	0028	invaild
5	0004	7	0025	invaild
6	0005	4	1121	invaild
7	0006	5	1123	invalid
8	0007	3	1125	invalid
9	0008	2	1223	invalid
10	0009	12	2342	invalid

11	0010	4	5321	vaild
12	0011	14	4325	invalid
13	0012	3	1246	invalid

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 = 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

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

Case	OrderID	Expected output
1	0000	valid
2	0001	Valid
3	5466	Invalid
4	9999	Invalid
5	1000	valid
6	1124	valid
7	4444	valid

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

Total test cases 6(3)+1=19,

For Quantity min=1 normal=5 max=10

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

Case	OrderID	Quantity	Password	Expected output
1	0000	1	0000	valid
2	0001	4	0021	invalid
3	0002	3	0025	invaild
4	0003	3	0028	invaild
5	0004	7	0025	invaild
6	0005	4	1121	invaild
7	0006	5	1123	invalid
8	0007	3	1125	invalid
9	0008	2	1223	invalid

10	0009	12	2342	invalid
11	0010	4	5321	vaild
12	0011	14	4325	invalid
13	0012	3	1246	invalid
14	0013	5	4334	invalid
15	0014	7	2322	invalid
16	0015	4	2345	invalid
17	0016	2	1111	invalid
18	0017	6	6784	invalid
19	0018	8	9999	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).

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	Expected output
1	0000	valid
2	0001	Valid
3	5466	valid

4	99991	invalid
5	1000	valid

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**	**Expected output**
1	0000	valid
2	0001	Valid
3	5466	Invalid
4	9999	Invalid
5	1000	valid

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

Case	**OrderID**	**Quantity**	**Password**	**Expected output**
1	0000	1	0012	valid
2	0001	4	0032	invalid
3	0002	3	0012	invaild
4	0003	3	0023	invaild
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

22	0021	7	0012	invalid
23	0022	4	0032	invalid
24	0023	3	0012	invalid
25	0024	1	0023	invalid
26	0025	4	0023	invalid
27	0026	1	0023	invalid
28	0027	1	0023	invalid
29	0028	5	0023	invalid
30	0029	7	0034	invalid
31	0030	4	0033	invalid

32	0031	2	0021	invalid
33	0032	4	0014	invalid
34	0033	5	0017	invalid
35	0034	6	0018	invalid
36	0035	7	0062	invalid
37	0036	9	0071	invalid
38	0037	0	0019	invalid
39	0038	54	0023	invalid
40	0039	3	3434	invalid
41	0040	5	3453	invalid

42	0041	3	0012	invalid
43	0042	5	0032	invalid
44	0043	7	0012	invalid
45	0044	8	0023	invalid
46	0045	11	0023	invalid
47	0046	34	0023	invalid
48	0047	6	0023	invalid
49	0048	5	0023	invalid
50	0049	7	0034	invalid
51	0050	4	0033	invalid
52	0051	2	0021	invalid
53	0052	7	0014	invalid
54	0053	0	0017	invalid
55	0054	4	0018	invalid
56	0055	2	0062	invalid
57	0056	6	0071	invalid

58	0057	2	0019	invalid
59	0058	1	0023	invalid
60	0059	6	3434	invalid
61	0060	7	3453	invalid
62	0061	3	0012	invalid

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 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

2. ***Void CancelOrder(Int OrderID):***
   Total test cases 7^1=7, only binary values will be valid
   Min-1 = -0001, min = 0000, min+1 =0001 , normal =5466 , max-1 = 9999, max =1000
   max+1=1001

Case	Order ID	Expected output
1	-0001	Invalid
2	0000	valid
3	0001	Valid
4	5466	Invalid
5	9999	Invalid
6	1000	valid
7	10000	Invalid

3. ***Void CreateOrder(int OrderID,int Quantity,int Password)***
   Total test cases 7^3=343,
   For Quantity min=1 normal=5 max=10
   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

23	0021	7	0021	invalid
24	0022	4	0056	invalid
25	0023	3	0005	invalid
26	0024	1	0015	invalid
27	0025	4	0045	invalid
28	0026	1	0033	invalid
29	0027	1	0091	invalid
30	0028	5	0068	invalid

31	0029	7	0023	invalid
32	0030	4	0087	invalid
33	0031	2	0012	invalid
34	0032	4	0032	invalid
35	0033	5	0012	invalid
36	0034	6	0023	invalid
37	0035	7	0023	invalid
38	0036	9	0023	invalid
39	0037	0	0023	invalid
40	0038	13	0023	invalid
41	0039	3	0034	invalid
42	0040	5	0033	invalid
43	0041	3	0012	invalid
44	0042	5	0032	invalid
45	0043	7	0012	invalid
46	0044	8	0023	invalid
47	0045	11	0023	invalid
48	0046	12	0023	invalid
49	0047	6	0023	invalid
50	0048	5	0023	invalid
51	0049	7	0034	invalid
52	0050	4	0033	invalid
53	0051	2	0021	invalid
54	0052	7	0014	invalid
55	0053	2	0017	invalid
56	0054	4	0018	invalid

Capital University Of Science & Technology

57	0055	2	0062	invalid
58	0056	6	0071	invalid
59	0057	2	0019	invalid
60	0058	1	0023	invalid
61	0059	6	3434	invalid
62	0060	7	3453	invalid
63	0061	3	0012	invalid
64	0052	4	1232	invalid
65	0053	7	0234	invalid
66	0054	3	3343	invalid
67	0055	6	1257	invalid
68	0056	1	6326	invalid
69	0057	4	0013	invalid
70	0058	8	0089	invalid
71	0059	2	3478	invalid
72	0060	3	3672	invalid
73	0061	4	0047	invalid
74	0071	1	0034	invalid
75	0071	7	0012	invalid
76	0021	2	0034	invalid
77	0012	3	0042	invalid
78	0071	5	0015	invalid
79	0058	2	8954	invalid
80	0083	7	9012	invalid
81	0082	2	6345	invalid
82	0086	8	7823	invalid
83	0087	3	4512	invalid

84	0088	4	0018	invalid
85	0089	5	0062	invalid
86	0055	3	0043	invalid

87	0023	2	3232	invalid
88	0024	1	0012	invalid
89	0025	4	0032	invalid
90	0026	9	0012	invalid
91	0027	5	0023	invalid
92	0028	2	0023	invalid
93	0029	1	0023	invalid
94	0030	7	0023	invalid
95	0031	3	0023	invalid
96	0032	8	0034	invalid
97	0033	9	0033	invalid
98	0034	6	0021	invalid
99	0056	5	0014	invalid
100	0057	4	0017	invalid
101	0058	1	0018	invalid
102	0059	3	0062	invalid
103	0060	9	0071	invalid
104	0061	1	0019	invalid
105	0071	5	0023	invalid
106	0071	4	3434	invalid

107	0043	1	0014	invalid
108	0044	3	0017	invalid

Capital University Of Science & Technology

109	0045	8	0018	invalid
110	0046	4	0062	invalid
111	0047	5	0071	invalid
112	0048	6	0019	invalid
113	0049	2	0023	invalid
114	0050	4	3434	invalid
115	0081	9	0012	invalid
116	0083	7	0032	invalid
117	0097	6	0012	invalid
118	0089	4	0023	invalid
119	0072	5	0023	invalid
120	0022	3	0023	invalid
121	0076	2	0023	invalid
122	0089	1	0023	invalid
123	0088	8	0034	invalid
124	0082	9	0033	invalid
125	0084	4	0021	invalid
126	0086	3	0083	invalid
127	0058	3	0032	invalid
128	0059	5	0012	invalid
129	0060	7	0023	invalid
130	0061	8	0023	invalid
131	0071	11	0023	invalid
132	0071	34	0023	invalid
133	0021	6	0023	invalid
134	0012	5	0034	invalid
135	0071	7	0033	invalid

Capital University Of Science & Technology

136	0058	4	0021	invalid
137	0083	2	0014	invalid
138	0082	7	0017	invalid
139	0086	2	0032	invalid
140	0087	4	0012	invalid
141	0088	2	0023	invalid
142	0002	6	0054	invalid
143	0003	2	0072	invalid
144	0004	3	0022	invalid
145	0005	1	0076	invalid
146	0006	4	0089	invalid
147	0007	3	0088	invalid
148	0008	3	0082	invalid
149	0009	7	0084	invalid
150	0010	4	0086	invalid
151	0011	5	0058	invalid
152	0012	3	0059	invalid
153	0013	2	0060	invalid
154	0014	12	0061	invalid
155	0015	3	0071	invalid
156	0016	1	0071	invalid
157	0017	4	0021	invalid
158	0002	3	0012	invalid
159	0003	7	0071	invalid
160	0004	4	0072	invalid
161	0005	3	0025	invalid
162	0023	1	0026	invalid

163	0043	4	0027	invalid
164	0023	1	0028	invalid
165	0012	1	0029	invalid
166	0098	5	0030	invalid
167	0065	7	0031	invalid
168	0045	4	0032	invalid
169	0011	2	0041	invalid
170	0032	4	0054	invalid
171	0045	5	0063	invalid
172	10000	6	0078	invalid