İZMİR UNIVERSITY OF ECONOMICS

SE 318
SOFTWARE VERIFICATION AND VALIDATION
SPRING 2020

HOTEL MANAGEMENT SYSTEM

BARIŞ GÖÇ

IHSAN KATMER

KAAN GÖKÇEK

TAYLAN ERYİĞİT

UNIT TEST DOCUMENT

Version <*3.0*>

<05/21/2020>

VERSION HISTORY

VERSION 1.0 (date)
-Staff management(adding , deleting and viewing info) -Room management(room allocation, status of occupancy indication with colors) -Login
VERSION 2.0 (date)
-Login and Register Screen added.(Customer side and admin side) -Test cases added. -Some UI modifications
VERSION 3.0 (date)
-Customer Screen added. -Some modifications on UI -Some modifications on admin side(room management part) -New test cases added.

INTRODUCTION

1.1 PURPOSE OF THE TEST CASE DOCUMENT

The purpose of this document is to test if the system conforms to its specifications or not. By testing this system we can ensure that we can achieve higher level of confidence with the system.
This document demonstrates the tests of Hotel Management System. The users of the system are Customer, Admin(Top authorized person), and Staff(Receptionist).

1.2 CONSTRAINT

To implement this system, Java programming language is mandatory. Also the knowledge of JUnit is necessary. It is important that to choose good test conditions to make sure that the system functions correctly.

UNIT TEST FRAMEWORK: *JUNIT*

Junit is used as unit test framework. It helps to write code faster because of its less complexity. Also JUnit is open source framework.

TEST CASES

Test Case 1	
Test Definition	
testAdminCheck_Positive():It tests if the user is an admin or not.	
Input Value	
“ihsan” “123456”	
Expected Value	**Actual Value**
true	**true**
Result of Test Case	***successful***
Test Script	
assertTrue(d.adminLoginCheck("ihsan","123456"));	

Test Case 2	
Test Definition	
testAdminCheck_Negative:It tests if the user is an admin or not.	
Input Value	
“ihs” “123456”	
Expected Value	**Actual Value**
false	**false**
Result of Test Case	***successful***
Test Script	
assertFalse(d.adminLoginCheck("ihs", "123456"));	

Test Case 3	
Test Definition	
testCheckDates_Positive():It tests if there are overlapping bookings or not.	
Input Value	
1, "2020/01/15", "2020/01/25"	
Expected Value	**Actual Value**
“error”	**“error”**
Result of Test Case	***successful***
Test Script	
assertEquals("error", d.CheckDate(1, "2020/01/15", "2020/01/25"));	

Test Case 4	
Test Definition	
testCheckDates_Positive():It tests if there are overlapping bookings or not.	
Input Value	
2, "2020/02/02", "2020/02/22"	
Expected Value	**Actual Value**
“error”	**“error”**
Result of Test Case	***successful***
Test Script	
assertEquals("error", d.CheckDate(2, 2020/02/02", "2020/02/22"));	

Test Case 5	
Test Definition	
testCheckDates_Positive():It tests if there are overlapping bookings or not.	
Input Value	
"error", d.CheckDate(3, "2020/01/22", "2020/01/24")	
Expected Value	**Actual Value**
“error”	**“error”**
Result of Test Case	***successful***
Test Script	
assertEquals("error", d.CheckDate(3, "2020/01/22", "2020/01/24"));	

Test Case 6	
Test Definition	
testCheckDates_Positive():It tests if there are overlapping bookings or not.	
Input Value	
"error", d.CheckDate(2, "2020/01/19", "2020/01/23"));	
Expected Value	**Actual Value**
“error”	**“error”**
Result of Test Case	***successful***
Test Script	
assertEquals("error",d.CheckDate(2,"2020/01/19", "2020/01/23"));	

Test Case 7	
Test Definition	
testCheckDates_Negative():It tests if there are overlapping bookings or not.	
Input Value	
"room is full", d.CheckDate(1,"2020/01/07", "2020/02/25")	
Expected Value	**Actual Value**
“room is full” <not>	“room is full”
Result of Test Case	***successful***
Test Script	
assertNotEquals("room is full", d.CheckDate(1,"2020/01/07","2020/02/25"));	

Test Case 8	
Test Definition	
testCheckDates_Negative():It tests if there are overlapping bookings or not.	
Input Value	
“room is full", d.CheckDate(2, "2020/01/23", "2020/02/25")	
Expected Value	**Actual Value**
“room is full” <not>	“room is full”
Result of Test Case	***successful***
Test Script	
assertNotEquals("room is full", d.CheckDate(2,"2020/01/23", "2020/02/25"));	

Test Case 9	
Test Definition	
testCheckDates_Negative():It tests if there are overlapping bookings or not.	
Input Value	
"room is full", d.CheckDate(3, "2020/03/12", "2020/03/19")	
Expected Value	**Actual Value**
“room is full”<not>	**“room is full”**
Result of Test Case	***successful***
Test Script	
assertNotEquals("room is full", d.CheckDate(3,"2020/03/12","2020/03/19"));	

Test Case 10	
Test Definition	
testCheckDates_Negative():It tests if there are overlapping bookings or not.	
Input Value	
"room is full", d.CheckDate(2, "2020/01/02", "2020/01/25")	
Expected Value	**Actual Value**
“room is full” <not>	**“room is full”**
Result of Test Case	***successful***
Test Script	
assertNotEquals("room is full", d.CheckDate(2, "2020/01/02", "2020/01/25"));	

Test Case 11	
Test Definition	
testRoomIsFull():It tests if the room is full or not.	
Input Value	
2, d.getPoint(roomIndex2)	
Expected Value	**Actual Value**
2	**2**
Result of Test Case	***successful***
Test Script	
assertEquals(2, d.getPoint(roomIndex2));	

Test Case 12	
Test Definition	
testRoomIsEmpty():It tests if room is full or not.	
Input Value	
3, d.getPoint(roomIndex3)	
Expected Value	**Actual Value**
3<not>	**3**
Result of Test Case	***successful***
Test Script	
assertNotEquals(3,d.getPoint(roomIndex3));	

Test Case 13	
Test Definition	
testGetDates_Positive():It checks for the occupancy status of the room between two dates.	
Input Value	
"Room is available", d.getDate(3)	
Expected Value	**Actual Value**
“Room is available”	**“Room is available”**
Result of Test Case	***successful***
Test Script	
assertEquals("Room is available", d.getDate(1));	

Test Case 14	
Test Definition	
testGetDates_Positive():It checks for the occupancy status of the room between two dates.	
Input Value	
"Room is available", d.getDate(3)	
Expected Value	**Actual Value**
“Room is available”	**“Room is available”**
Result of Test Case	***successful***
Test Script	
assertEquals("Room is available", d.getDate(3));	

Test Case 15	
Test Definition	
testGetDates_Negative():It checks for the occupancy of the room between two dates or not.	
Input Value	
"Room is available", d.getDate(2)	
Expected Value	**Actual Value**
“Room is available <not>	**“Room is available”**
Result of Test Case	***successful***
Test Script	
assertNotEquals("Room is available",d.getDate(2));	

Test Case 16	
Test Definition	
testLoginDetails_Positive():It checks for users id uniqueness.This is the first time this user registers so the user should be able to register.	
Input Value	
d.CheckCustomerUserName(“kaan”)&&d.CheckCustomerPassword(“abcst”)	
Expected Value	**Actual Value**
true	**true**
Result of Test Case	***successful***
Test Script	
d.registerEmployee(12,”kaan”,”absct”,”kaan”,”gokcek”,33,”male”,”xx”); **assertTrue(d.CheckCustomerUserName(“kaan”)&& d.CheckCustomerPassword(“absct”));**	

Test Case 17	
Test Definition	
testLoginDetails_Positive():It checks for users id uniqueness.This is the first time this user registers so the users should be able to register.	
Input Value d.CheckCustomerUserName(“bgoc”)&&d.CheckCustomerPassword(“23456)	
Expected Value	**Actual Value**
true	true
Result of Test Case	*successful*
Test Script	
d.registerCustomer(1234,”bgoc”,”23456”,”baris”,”goc”, “age”, “gender”, “address”); assertTrue(d.CheckCustomerUserName("bgoc")&& d.CheckCustomerPassword("23456"));	

Test Case 18	
Test Definition	
testLoginDetails_Negative():It checks for users id uniqueness.This time, the user shouldn’t be able to register because id is not unique.	
Input Value	
d.CheckCustomerUserName("kaan”) && d.CheckCustomerPassword("nasvt"),	
Expected Value	**Actual Value**
true <not>	false
Result of Test Case	*successful*
Test Script	
d.registerEmployee(12,"kaan","nasvt","kaan","gokcek",42,"male","xxx"); assertFalse(d.CheckCustomerUserName("kaan")&&d.CheckCustomer Passod("nasvt"));	

Test Case 19	
Test Definition	
testLoginDetails_Negative():It checks for users id uniqueness.This time, the user shouldn't be able to register because id is not unique.	
Input Value	
d.CheckCustomerUserName("barisgoc") &&d.CheckCustomerPassword("23456");	
Expected Value	**Actual Value**
true<not>	false
Result of Test Case	*successful*
Test Script	
d.registerCustomer(1234,"barisgoc", "23456", "baris", "goc", 24,"male ,"Xxx"); assertFalse(d.CheckCustomerUserName("barisgoc")&& d.CheckCustomerPassword("23456"));	

Test Case 20	
Test Definition	
testLoginCheck_Positive():It checks if the customer exists in the database or not.	
Input Value	
"ihs","1"	
Expected Value	**Actual Value**
true	true
Result of Test Case	*successful*
Test Script	
assertTrue(d.loginCheck("ihs", "1"));	

Test Case 21	
Test Definition	
testLoginCheck_Negative():It checks if customer exists in the database or not.	
Input Value	
“ihs” “123456”	
Expected Value	**Actual Value**
true<not>	**false**
Result of Test Case	***successful***
Test Script	
assertFalse(d.loginCheck("ihs","123456"));	

Test Case 22	
Test Definition	
testValidation_P():It checks if the password matches with specific pattern or not.	
Input Value	
“_Brs_1234”.matches(strRegEx)?true:false;	
Expected Value	**Actual Value**
true	**true**
Result of Test Case	***successful***
Test Script	
String strRegEx = "^(?=.*[0-9]).{8,15}$"; assertTrue(“_Brs_1234”.matches(strRegEx)?true:false);	

Test Case 23	
Test Definition	
testValidation_P():It checks if the password matches with specific pattern or not.	
Input Value	
“1234Brs1234”.matches(strRegEx)?true:false	
Expected Value	**Actual Value**
true	**true**
Result of Test Case	***successful***
Test Script	
String strRegEx = "^(?=.*[0-9]).{8,15}$"; assertTrue(“1234Brs1234”.matches(strRegEx)?true:false);	

Test Case 24	
Test Definition	
testValidation2_N():It checks if password matches with specific pattern or not.	
Input Value	
“1234”.matches(strRegEx)?true:false	
Expected Value	**Actual Value**
true <not>	**false**
Result of Test Case	***successful***
Test Script	
String strRegEx = "^(?=.*[0-9]).{8,15}$"; assertFalse(“1234”.matches(strRegEx)?true:false);	

Test Case 25	
Test Definition	
testValidation2_N():It checks if password matches with its specific pattern or not.	
Input Value	
“abcd”.matches(strRegEx)?true:false	
Expected Value	**Actual Value**
true <not>	**false**
Result of Test Case	***successful***
Test Script	
String strRegEx = "^(?=.*[0-9]).{8,15}$"; assertFalse(“1234”.matches(strRegEx)?true:false);	

Test Case 26	
Test Definition	
testValidation2_N():It checks if password matches with specific pattern or not.	
Input Value	
“baris1234”.matches(strRegEx)?true:false	
Expected Value	**Actual Value**
true <not>	**false**
Result of Test Case	***successful***
Test Script	
String strRegEx = "^(?=.*[0-9]).{8,15}$"; assertFalse(“baris1234”.matches(strRegEx)?true:false);	

Test Case 27	
Test Definition	
testRoomOwner_Positive ():It checks if that person staying in the room or not.	
Input Value	
"taylan", d.get(2)	
Expected Value	**Actual Value**
“taylan”	**“taylan”**
Result of Test Case	***successful***
Test Script	
assertEquals("taylan",d.get(2));	

Test Case 28	
Test Definition	
testRoomOwner_Positive (): It checks if that person staying in the room or not.	
Input Value	
"baris",d.get(3)	
Expected Value	**Actual Value**
“baris”	**“baris”**
Result of Test Case	***successful***
Test Script	
assertEquals("baris",d.get(3));	
Test Case 29	

Test Definition	
testRoomOwner_Negative(): It checks if that person staying in the room or not.	
Input Value	
"ihsan",d.get(2)	
Expected Value	Actual Value
“ihsan” <not>	“taylan”
Result of Test Case	*successful*
Test Script	
assertNotEquals("ihsan",d.get(2));	

Test Case 30	
Test Definition	
testRoomOwner_Negative():It checks if that person staying in the room or not.	
Input Value	
"kaan",d.get(3)	
Expected Value	Actual Value
“kaan” <not>	“baris”
Result of Test Case	*successful*
Test Script	
assertNotEquals("kaan",d.get(3));	

4. CONCLUSION

THERE ARE 30 TEST CASES AND ALL OF THEM ARE SUCCESSFUL.