

**Universiti Tunku Abdul Rahman**

**UECS2354 Software Testing**

**Group 20**

Test Plan

Assignment

# **Document Control**

|  |  |
| --- | --- |
| **Document Name** | Test Plan |
| **Reference Number** | - |
| **Version** | V1.0 |
| **Project Code** | UECS2354 Software Testing Assignment |
| **Status** | [to be updated] |
| **Date Released** | [to be updated] |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Position** | **Signature** |
| Prepared By: | **Kok Yu San** | **Leader** |  |
| **Hwang Jen Fung** | **Member** |  |
| **Tham Jing Yi** | **Member** |  |
| **Wong Yong Jun** | **Member** |  |
| Reviewed By: |  |  |  |
| Approved By: |  |  |  |
| Approved By: |  |  |  |

# **Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Release Date** | **Section** | **Amendments** |
| 1 | 20/4.2024 |  | First edition |
| 2 | 24/4/2024 |  | Changing of all the methods of the first edition according to the latest code. |
|  |  |  |  |

Table of Content

[Document Control 2](#_Toc127346001)

[Version History 3](#_Toc127346002)

[1.0 Introduction 5](#_Toc127346003)

[1.1 Objective 5](#_Toc127346004)

[1.2 Scope 5](#_Toc127346005)

[1.3 Reference 5](#_Toc127346006)

[2.0 Test Plans 6](#_Toc127346007)

[2.1 Test Items 6](#_Toc127346008)

[2.2 Features to be tested. 7](#_Toc127346009)

[2.3 Features not to be tested. 8](#_Toc127346010)

[2.4 Test Basis 8](#_Toc127346011)

[2.5 Test Conditions 9](#_Toc127346012)

[2.6 Entry Criteria 11](#_Toc127346013)

[2.7 Exit Criteria 12](#_Toc127346014)

# Introduction

## 1.1 Objective

## We are developing as well as testing a Hotel Room Booking System. This is system to assign a user which can either be a VIP member, normal member or non-member to a room which also consists of VIP, Deluxe and Standard. Furthermore, an exclusive reward can be entitled to a VIP or normal member occasionally.

The objective is to make sure the hotel room booking system can run as smoothly as possible which is able to assign the correct number and type of room based on the condition stated, assigning the user to the waiting list if the number of rooms required is insufficient as well as cancel the booking of the user. By doing software testing we can find the bugs in the software and fix them. Besides that, we can also make sure that the software can return the expected result for the users.

## 1.2 Scope

For the test scope of the Hotel Room Booking System, we will focus on all the class which are User, Booking, WaitingList, Room and Printer. However, we are excluding the Room and Printer classes from the test scope as they are not ready for testing yet.

1. Parameterized Testing

* By implementing the parameterized testing, it enables us to write parameters to test on multiple scenarios with different inputs using the same method under test. This helps in covering a wide range of possibilities for each functionality without the need to repeat the testing process 1 by 1.

1. Mockito

* In our testing we will be implementing the Mockito components. It allows us to create mock objects, which is the substitute for the real component for the dependency testing. In our system the room class and the printer class are not ready for testing therefore we use mock objects to return a fixed value to make our testing more predictable and check for the behaviors of the system. We also use mock for the testing for dependency with more than 1 classes. This helps to verify the interaction for both the classes without the need of relying on the actual objects and value.

1. Integration Testing

* In the testing, we will carry out integration testing which validates all the components of the system work successfully with each other when seamlessly deliver end-to-end functionality. This includes the interactions between the booking class and other classes for example when doing the setBooking() method, if availability of room are insufficient for the amount needed, the user are properly pass to the WaitingList class.

## 1.3 References

# 2.0 Test Plans

## 2.1 Test Items

## Hotel Room Booking System

|  |  |
| --- | --- |
| Classes Name | Methods |
| Booking | Booking(User, Room, WaitingList, Printer)Booking(User, int[])void addBookedList(User, int[])ArrayList<Booking> getBookedList()void setBookedList(ArrayList<Booking>)User getUser()void setBooking()int[] getUserRoom()void cancelBooking(User) |
| User | User(String name, String, boolean, int)User()void setName(String)String getMember\_type()void setMember\_type(String)boolean getExcl\_reward()void setExcl\_reward(boolean)int getNumRoom()void setNumRoom(int) |
| WaitingList | WaitingList()void addWaiting(User)ArrayList<User> getWaiting(String)void removeWaiting(User) |
| Room | int getVip()void setVip(int)int getDeluxe()void setDeluxe(int)int getStandard()int getStandard()boolean checkRoom(String)void bookRoom(String) |
| Printer | void printInfo(String) |

## 2.2 Features to be tested.

|  |  |
| --- | --- |
| Class | Features |
| Booking | * To assign different types of room according to the user required number of rooms and availability of rooms. * To subtract the number of rooms booked from the number of available rooms. * To assign users to a waiting list if the number of rooms required is not sufficient. * To assign users to booking list if rooms are sufficient. * To allow users to be removed from the waiting list when cancelling booking. * To allow users to remove from the booking list when cancel booking. * To throw exceptions when conditions are not met. |
| User | * To set username to the system. * To set the member type of a user * To assign exclusive rewards to VIP members and normal members. * To get the number of rooms to book at a time by the user. * To throw exceptions when conditions are not met. |
| WaitingList | * To add the user into the waiting list according to member type. * To retrieve users from the waiting list according to the member type * To remove user from the waiting list according to the user detail. * To throw exceptions when conditions are not met. |

## 2.3 Features not to be tested.

|  |  |
| --- | --- |
| Class | Features |
| Booking | * To create a booking list to store all the bookings. |
| User | * To create an object user to store all the user information. (Constructor) |
| WaitingList | * To create array list to store all the users in a waiting list when the room is insufficient for booking. |
| Room | * To check the available rooms for VIP, Deluxe and Standard * To book the room for the user. |
| Printer | * To print the booking details |

## 2.4 Test Basis

The test basis which is the sources of getting the test item. In conducting the planning for the Hotel Room Booking System, we are collecting information to create a test case based on several sources such as:

1. Decision table

* Created based on the condition stated in the project details provided when setBooking()

1. Test case

* According to the condition set in the decision table, a test case is designed to better create the test code.

1. Use case diagram

* Use case diagram is a diagram reflecting the real-world uses of a product which creates a simple interactive diagram which shows the interaction of the “actors” including users, customers and system.

1. Class diagram

* The class diagram shows all the relationships between the classes as well as showing all the attributes of the class diagram and method to have a better overview of the system to make the design of the system more logical.

## 2.5 Test Conditions

|  |  |
| --- | --- |
| Class | Features |
| Booking | * To verify VIP members can only make a maximum of 3 rooms at a time. * To verify Normal member can only make a maximum of 2 rooms at a time. * To verify non-members can only book a room at a time. * To verify invalid exception is thrown when number of rooms to book at a time less than or equal to 0 or exceed the maximum rooms to book at a time. * To verify the correct number of VIP, Deluxe and Standard room is assigned to the user. * To verify the member type of users must be equal to VIP, member, normal. * To verify invalid exception is thrown if member type is not equal to VIP, member, normal. * To verify when insufficient amount of room as requested, VIP member will be assigned to the VIP waiting list. * To verify when insufficient amount of room as requested, Normal member will be assigned to the member waiting list. * To verify when insufficient amount of room as requested, non-member will be assigned to the normal waiting list. * To verify the booking is saved in the booking list when the booking is successfully made. * To verify the user can cancel their booking from the booking list after successful booking. * To verify the user can cancel their booking by removing the user from the waiting list. * To verify invalid exception is thrown when user is not found in waiting list or booking list. |
| User | * To verify if the username is set using setName() and pass using method getName(). * To verify invalid exception is thrown if invalid username is set. * To verify if the member type is set using setMember\_Type() and is pass using method getMember\_type(). * To verify invalid exception is thrown if invalid member type is set. * To verify if the number of rooms wanted is set using setNumRoom() and is passed using method getNumRoom(). * To verify invalid exception is thrown if invalid number of rooms set for the user for each room type is passed. * To verify if the exclusive reward status of normal member is set using setExcl\_reward () and is passed using method getExcl\_reward(). |
| WaitingList | * To verify user is added to the into the waiting list. * To verify invalid exception is thrown if invalid member type is passed to the method getWaiting(String member\_type) * To verify the user is removed from the waiting list after cancelling the booking. |
| Room and Printer | * These two classes are not ready for testing, yet room class checking for room availability and printer which prints the booking information. Therefore, the features within the method will not be tested, but we will be implementing the test double to test for the integration between the classes. |

## 2.6 Entry Criteria

1. Test objects and test items ready and testable.
2. Test cases and condition table are created.
3. After importing all the external library files which is necessary for the environment set up such as:

JUnitParams-1.0.2.jar, byte-buddy-1.12.8.jar, byte-buddy-agent-1.12.8.jar,

mockito-core-4.4.0.jar, objenesis-3.2.jar, Junit 4

**A computer program with text

Description automatically generated with medium confidence**

1. Tester teams are ready to test for the testing.
2. After receiving the approval from the Project Management Team to proceed.

## 2.7 Exit Criteria

1. No future changes to the design, codes and features.
2. No crash, panic, unexpected process termination or other stoppage of processing occurred.
3. Tester team has executed all the planned tests according to the test plan.
4. All “must-fix” bugs that are notified are resolved.
5. After the integration test is carried out to show the interaction between the classes.
6. The system stability and reliability, completion of all the planned tests as required by the Project Management Team.
7. The Project Management Team agrees that we have completed the testing/