HOTEL RESERVATION SYSTEM

Test Plan

Team Members:

Mücahit Tozal: Developer (Team Leader)

Doğan Özbayrak: Developer

Emre Ulusoy: Analyst

Ferhat Erduran: Designer, Tester

Uğur Eren Temel: Tester

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Document Purpose

The Test Plan (TP) document shall provide detailed descriptions of the testing procedures that will be applied to the Hotel Reservation System. The document shall act as a guideline for our team to implement the necessary tests and ensure customer satisfaction. We aim to create a high quality, easy-to-use system for guests and staff alike since it will give the guests their first impression of the hotel.

1.2. Document Conventions

This TP document has been prepared in accordance with the *IEEE 829 Test Plan Template*.

1.3 Scope

The "Reservation", "People", and "Room" Management modules are the subsystems included in the testing. Some non-functional requirements as well as the actual connections to banks and e-mail billing shall be assumed but not implemented or tested due to time and budget constraints.

1.4 Objectives and Goals

The testing process intends to make sure that all the requirements specified in the SRS document are met. It also aims to uncover and fix as many system defects as possible.

1.5 References

[1] Ian Somerville, *Software Engineering*, Pearson (9th Edition, 2010).

[2] Ulf Eriksson, "How to write a test plan with the IEEE 829 standard". https://reqtest.com/testing-blog/how-to-write-a-test-plan-2/

2. TEST ITEMS

2.1. Reservation Management Module

This module is concerned with creating, canceling and modifying reservations, calculating total price, and payment operations such as checking credit card validity, generating a bill and mailing it to the customer.

2.2. People Management Module

This module is concerned with creating and deleting customer and employee records as well as verifying the admin account.

2.3. Room Management Module

This module is concerned with adding, deleting and modifying rooms.

3. FEATURES TO BE TESTED

3.1. For the Customer

3.1.1. Viewing Rooms: The customer is able to view all rooms.

3.1.2. Personal Information: The customer is able to enter personal information and reservation details.

3.1.3. Payment Information: The customer is able to enter payment information.

3.1.4. Price Calculation: The system calculates and displays total price correctly.

3.1.5. Payment Verification: The system verifies customer's payment information.

3.1.6. Successful Recording: The system records customer and payment information correctly.

3.1.7. Generating Reservation Number: The system generates a unique reservation number.

3.1.8. Mailing Receipt: The customer is mailed the receipt (if the function returns 1, the customer shall be assumed to have sent the receipt successfully).

3.1.9. Viewing Reservation: The customer is able to view an active reservation by entering their unique reservation number.

3.1.10. Changing Reservation: The customer is able to change an active reservation by entering their unique reservation number.

3.1.11. Canceling Reservation: The customer is able to cancel an active reservation by entering their unique reservation number.

3.2. For the Admin

3.2.1. Login: The admin is able to login to the panel with a unique username and password.

3.2.2. Managing Reservations: The admin is able to add, delete and modify reservations, which include customers' personal and payment information.

3.2.3. Managing Rooms: The admin is able to add, delete and modify rooms.

3.3. General Features

3.3.1 Browser Window: The browser window resizes automatically when played with.

3.3.2 Readability: The information on the web pages is readable and written in understandable English.

3.3.3 Performance: Given a stable internet connection, the system does not have more than 5 seconds of delay.

4. FEATURES NOT TO BE TESTED

Due to time and budget constraints, the following features shall be assumed to be working but not be implemented or tested.

4.1. Bank Connectivity: The system's connection to banks will not be tested but assumed. Credit card verification shall be done according to the credit card number.

4.2. E-mail Connectivity: Mailing their receipt to the customers shall not be tested but assumed. Function testing shall be done according to its return value.

4.3. System Recoverability: It shall be assumed that the database backs itself up every 3 hours.

4.4. System Availability: It shall be assumed that the system is up and running 24/7.

4.5. System Security: It shall be assumed that the system is secure from outside attacks.

4.6. System Capacity: It shall be assumed that the system supports up to 100 people at any one time.

5. APPROACH

5.1. Validation Testing

The following is a successful test scenario that covers all the requirements. The steps are assumed to be taken sequentially.

5.1.1. Given a working internet connection, the system displays the home page correctly.

5.1.2. The browser resizes automatically when played with.

5.1.3. The system loads and displays the "Rooms" page when clicked on the "Rooms" tab.

5.1.4. The system displays available and unavailable room types. When a certain room type has no more available rooms left, the system correctly displays that type as unavailable. When at least one room of that type is freed, the system correctly displays it as available again.

5.1.5. The system loads and displays the "Reservation" page when clicked on the "Make Reservation" button.

5.1.6. When the "Confirm" button is clicked, the system notifies the user if any of the fields were left empty/not filled correctly (more on this in **5.2**). If not, the system displays the total price and loads the payment fields.

5.1.7. When the "Make Transaction" button is clicked, the system notifies the user if any of the fields were left empty/not filled correctly (more on this in **5.2**). If not, the system asks the user to confirm. If confirmed, the system generates a unique reservation number, notifies the user with a success message and records the data in the database.

5.1.8. The system mails the receipt to the user (mailCust() function returns 1).

5.1.9. The user enters their reservation number in the home page and clicks "View Reservation".

5.1.10. The system correctly displays the relevant information (user name, room number, check-in/check-out dates).

5.1.11. The user changes their reservation dates and chooses a different room. The system gives a success message and updates the database.

5.1.12. The user deletes their reservation. The system gives a success message and updates the database.

5.1.13. The system loads and displays the "Admin Panel" when clicked on the "Admin Login" button.

5.1.14. The user enters username and password and clicks "Login". If entered incorrectly, the system notifies the user (more on this in **5.2**). If entered correctly, the system loads the admin panel.

5.1.15. Active reservations are displayed under "Reservations" and all rooms are displayed under "Rooms".

5.1.16. The user selects a reservation then changes its customer name, room number and payment method. The system updates the database and gives a success message.

5.1.17. The user selects a reservation then removes it from the system. The system updates the database and gives a success message.

5.1.18. The user adds a new room to the system then removes it. The system updates the database and gives a success message on both occasions.

5.1.19. The user clicks "Logout" and the system notifies the user that they have successfully logged out.

5.2. Defect Testing

The following are test cases that will be applied. Receiving the expected outputs shall be considered a success.

5.2.1. **TC:** Admin Login Success
Test Data: Admin [username] and [password]
Input: **[**username]: "ferduran" **AND** [password]: "123456"
Output: Successful redirection to the admin panel.

5.2.2. **TC:** Admin Login Error #1
Test Data: Admin [username] and [password]
Input: [username]: empty **OR [**password]: empty
Output: Error message: "[Field] cannot be left empty."

5.2.3. **TC:** Admin Login Error #2
Test Data: Admin [username] and [password]
Input: [username]: "asd" **OR [**password]: "asd"
Output: Error message: "[Field] is incorrect."

5.2.4. **TC:** Customer Personal Info Success
Test Data: Customer [first_name], [last_name], [date_of_birth], [e_mail], [phone_number], [room_type], [number_of_rooms], [number_of_guests], [check_in], [check_out], [additional_wishes]
Input: [first_name] = string, [last_name] = string, [date_of_birth] = date, [e_mail] = string, [phone_number] = string, [room_type] = int, [number_of_rooms] = int, [number_of_guests] = int, [check_in] = date, [check_out] = date, [additional_wishes] = string
Output: Display total price and load the payment fields.

5.2.5. **TC:** Customer Personal Info Error #1

Test Data: Customer [first_name], [last_name], [date_of_birth], [e_mail], [phone_number], [room_type], [number_of_rooms], [number_of_guests], [check_in], [check_out], [additional_wishes]
Input: At least one of the fields above left empty
Output: Error message: "[Field] cannot be left empty."

5.2.6. **TC:** Customer Personal Info Error #2
Test Data: Customer [first_name], [last_name], [date_of_birth], [e_mail], [phone_number], [room_type], [number_of_rooms], [number_of_guests], [check_in], [check_out], [additional_wishes]
Input: At least one of the fields above entered a wrong type (refer to **5.2.4** for the correct types).
Output: Error message: "[Field] is incorrect. Please enter a valid input." (Valid inputs will be given to the user next to the fields).

5.2.7. **TC:** Customer Payment Info Success
Test Data: Customer [card_no], [card_type], [holder_name], [exp_date], [cvv]
Input: [card_no] = (int) 1XXX XXXX XXXX XXX9 **AND** [card_type] = string, [holder_name] = string, [exp_date] = date, [cvv] = (int) XXX
Output: Display confirmation query.

5.2.8. **TC:** Customer Payment Info Error #1
Test Data: Customer [card_no], [card_type], [holder_name], [exp_date], [cvv]
Input: At least one of the fields above left empty
Output: Error message: "[Field] cannot be left empty."

5.2.9. **TC:** Customer Payment Info Error #2
Test Data: Customer [card_no], [card_type], [holder_name], [exp_date], [cvv]
Input: At least one of the fields above entered a wrong type (refer to **5.2.7** for the correct types) **OR** [card_no] does not start with 1 and end with 9.
Output: Error message: "[Field] is invalid. Please enter a valid input."

- Note that the input fields shall be bounded, so buffer overflow shall not be possible. For example, the [card_no] field will not allow typing after (4*4+3 =) 19 digits. Similarly, the [cvv] field shall be bounded at 3 digits. All string types shall be bounded at 50 characters.

6. SUSPENSION CRITERIA AND RESUMPTION REQUIREMENTS

If any of the test cases fail to give the expected output, the developers shall be notified, and the test shall be put on hold until the underlying problems are fixed. Then the test case shall be reapplied to the item in question. If the expected output is received, the testing procedure shall continue. If not, the steps above shall be repeated until it is.

7. SCHEDULE

The testing process shall take place between April 16 and April 30, 2019.

8. RISKS AND CONTINGENCIES

In order to prevent damaging or losing the existing system, it shall be backed up in a separate device before starting the testing activities.

9. APPENDIX

Refer to the SRS and SDD documents for further understanding of the system and the user requirements.