# Software Test Specifications (STS)
# John and Jane Bed and Breakfast Manager (BBRM)

CMIS 330 6380 Software Engineering Principles and Techniques (2215)
STS Project 3

Connor Aaron

Lauren King

06-29-2021

# Table of Contents

# 1. Introduction

## 1.1. Purpose

This document shall be a guide for testers of John and Jane's bed and breakfast reservation manager system. This document shall maintain an Integrity level 1 and shall include a level test plan, level test design, and four level test cases. This document shall be used by testers to find and report errors in the structure and functionality of the system. This document shall be read by designated testers of the structures and functionality to be tested, as well as project managers to oversee the reporting process.

## 1.2. Scope

This software test specifications document shall test the accounting page and the calendar page of the reservation manager system, and the reservation guarantee functionality of the reservation manager system.

## 1.3. Definitions, Acronyms, Abbreviations

| **Name** | **Description** | **Acronym** |
|---|---|---|
| Test Planning | Overview of test cases | |
| Test Case | Four features to be tested | |
| Anomaly Report | Error log | AR |
| White Box testing | Structured testing approach of softwares logic paths | |
| Black Box Testing | Testing focused on functional requirements of the software | |

## 1.4. References

Institute of Electrical and Electronics Engineers. IEEE Standard for Software Test Documentation, IEEE Std 829-1998. New York: IEEE, 1998.

# 2. Test Planning

## 2.1. Test Items

1. Accounting Page, Identifier: UI_TC_1
    a. Referenced from software design document Figure 4.1.1, Architectural Context Diagram
2. Calendar Page, Identifier: UI_TC_2
    a. Referenced from software design document Figure 4.1.1, Architectural Context Diagram
3. User Scenario 1, guaranteed hold status, Identifier: US_TC_1
    a. Referenced from Software Requirements Specifications document Figure 3.0, Use Case Scenario 1
4. User Scenario 2, non-guaranteed hold status, Identifier: US_TC_2
    a. Referenced from Software Requirements Specifications document Figure 3.1, Use Case Scenario 2

## 2.2. Test Traceability Matrix

| **Category** | **Description** | **System Req. #** | **Use Case #** | **Software Req. #** | **Test Case #** | **Pass/Fail** |
|---|---|---|---|---|---|---|
| Accounting | Expenses and Profit | 1.0 | 1.0 | 1.0 | 3.1 | |
| Calendar | Reservation List Display | 1.0 | 1.0 | 1.0 | 3.2 | |
| New Reservation | User Input | 1.0 | 1.0 | 1.0 | 3.3, 3.4 | |

## 2.3. Features to be tested

1. Accounting page of reservation manager system
2. Calendar page of reservation manager system
3. New reservation entry option on Calendar page
4. Create, Read, Update, Delete on Calendar page
5. Calendar display on calendar page

## 2.4. Features not to be tested

1. Reservation manager log-in system
    a. Not implemented, still under consideration

## 2.5. Approach

| **Name** | **Identifier** | **Source Document** | **Summary** | **Test Method** |
|---|---|---|---|---|
| Accounting Page | UI_TC_1 | SDD page 9 | Requirement to monitor expenses and calculate profits | White Box |
| Calendar Page | UI_TC_2 | SDD page 9 | Requirement to manage reservation and display vacancies | White Box |
| New Reservation | US_TC_1, US_TC_2 | SRS pages 13, 14 | Requirement to Create, Read, Update, Delete reservations | Black Box |

## 2.6. Item pass/fail criteria

1. UI_TC_1
    a. 1 or fewer anomalies reported
2. UI_TC_2
    a. 1 or fewer anomalies reported
3. US_TC_1
    a. No anomalies reported
4. US_TC_2
    a. No anomalies reported

## 2.7. Suspension criteria and resumption requirements

Suspend all testing activity if total anomalies reported is greater than 3. Suspend test cases 2 through 4 if anomalies reported is greater than 2 in any one of these test cases.

## 2.8. Test deliverables

- Level Test Plan
- Level Test Cases
    - Accounting
    - Calendar
    - User Scenario 1
    - User Scenario 2
- Anomaly Reports

## 2.9. Testing tasks

1. Tasks shall be dependent on availability of hardware described in the next section.
2. Tasks shall be dependent on the installation of the reservation manager prototype
3. US_TC_1 and US_TC_2 testing shall start after UI_TC_2 has been performed with no anomalies reported

## 2.10. Environmental needs

Tests shall require a computer, monitor, mouse and keyboard setup with a windows operating system. Database system comparable to the system to be implemented by the users John and Jane shall be desirable.

## 2.11. Responsibilities

1. Responsibility of Developers:
    a. Perform the first two test cases for the user interface of the reservation system: the accounting page and the calendar page
    b. Shall provide the test case prompts for the testers
    c. Shall resolve any anomalies reported
2. Responsibility of Testers:
    a. Perform the third and fourth test cases for the user scenarios of the reservation systems Create, Read, Update, Delete functionality.
    b. Shall prepare anomaly reports for the test cases

## 2.12. Training

Training shall be provided to the testers and user representatives via mentoring by knowledgeable staff members

## 2.13. Schedule

White Box Test cases shall be allotted one week to reach one or fewer anomalies milestones. Black Box Testing shall begin after the first week of testing and shall be allotted one week of testing to reach zero anomalies milestone.

## 2.14. Risk(s) and contingency(s)

Significant error or anomaly reports is a risk that may impact the completion of the planned testing activities. If more than four anomalies are not resolved within the two week time frame, an additional week shall be allotted to the developers and testers to resolve these anomalies.

# 3. Test Case Specification

## 3.1. Accounting Page

### 3.1.1. Test case identifier

Test Case ID: **UI_TC_1**
“User Interface 1 Test Case

### 3.1.2. Test objective

This test case shall be the White Box test cases of the accounting page user interface. This test case shall check for potential errors in the implementation of the accounting page in order to ensure that the user interface meets the requirements specified by the end users John and Jane

### 3.1.3. Input specification

| **Name** | **Description** | **Example** |
|---|---|---|
| Income | Float | “198.50” |
| Expenses | Float | “50.25” |

### 3.1.4. Outputs specification

| **Name** | **Description** | **Example** |
|---|---|---|

| | | |
|---|---|---|
| Expenses | Float | “50.25” |
| Profit | Float | “148.25” |

### 3.1.5. Special environment conditions

This Test Case shall require computer, monitor, mouse and keyboard, database setup comparable to the hardware available to the end users John and Jane. This test case shall require the installation of the Reservation manager prototype on a windows operating system.

### 3.1.6. Special procedural requirements

This test case shall require the reservation manager system to be launched and the accounting page to be selected. This test case shall require a database system comparable to that used by the end users John and Jane

### 3.1.7. Execution procedure steps

1. Launch reservation manager
2. Select accounting page
3. input income and submit
4. Input expenses and submit
5. Observe expenses output
    a. Shall be: current expenses + expenses input
6. Observe Profit Output
    a. Shall be: income input - expenses

### 3.1.8. Dependencies

This test case shall not be dependent on any other test cases executed.

## 3.2. Calendar Page

### 3.2.1. Test case identifier

Test Case ID: **UI_TC_2**
“User interface 2 test case”

### 3.2.2. Test objective

This Shall be the “White Box” test case for the calendar page component of the reservation manager system. This test case shall be used to detect potential errors in the inputs and outputs of new reservations that are generated and displayed on the calendar page user interface. This test case shall detect errors with the create, read, update, and delete functionality between the reservation manager system and the database system.

### 3.2.3. Input specification

| **Name** | **Description** | **Example** |
|---|---|---|
| New Reservation | Button Press, pop-up table of inputs | |
| name | string | “John Smith” |
| address | string | “101 Someplace Drive” |
| phoneNumber | integer | 9164455959 |
| price | float | 100.50 |
| dates | List of strings | <“May 15”,”May 16”,”May 17”,”May 18”> |
| creditCardNumber | integer | 4568424120012548 |
| roomNumbers | Array of boolean | [1,0,0,1] |
| oneDayPayment | Float | 25.50 |
| holdStatus | boolean | 1 |
| timeLimit | string | “May 14” |
| Submit | Button press, adds new reservation | |
| View Reservation | Button Press, Displays selected reservation | |
| Update | Button Press, updates selected reservation | |

New Reservation option shall be dependent on roomNumbers input, which represents room availability. holdStatus shall be dependent on oneDayPayment input. timeLimit shall be dependent on holdStatus input.

### 3.2.4. Outputs specification

New reservations added shall be displayed in the calendar of the calendar page. Reservations on the calendar shall be displayed so that they represent the status of the reservation (I.e. hold status, room availability). Changes made to existing reservations shall be updated on the calendar by the reservation manager system.

### 3.2.5. Special environment conditions

This Test Case shall require computer, monitor, mouse and keyboard, database setup comparable to the hardware available to the end users John and Jane. This test case shall require the installation of the Reservation manager prototype on a windows operating system.

### 3.2.6. Special procedural requirements

This test case shall require the reservation manager system to be launched and the calendar page to be selected. This test case shall require a database system comparable to that used by the end users John and Jane.

### 3.2.7. Execution procedure steps

1. Launch Reservation manager system
2. Select Calendar Page
3. Select new reservation option
4. Input user information and submit
5. Compare reservation output from calendar to reservation inputted in previous step
6. Select Edit reservation option
7. Change reservation data entries and select update
8. Repeat step 5

### 3.2.8. Dependencies

This test case shall not be dependent on any other test cases executed.

## 3.3. User Case Scenario 1

### 3.3.1. Test case identifier

Test Case ID: **US_TC_1**
“User Scenario 1 Test Case”

### 3.3.2. Test objective

This shall be the “Black Box” test case of the User scenario 1 described in the software requirements specifications document. This test case shall be used to determine if the hold status functionality of the reservation manager produces the expected output in a newly generated reservation as described by the user requirements.

### 3.3.3. Input specification

| **Name** | **Description** | **Example** |
|---|---|---|
| New Reservation | Button Press, pop-up table of inputs | |
| name | string | “John Smith” |
| address | string | “101 Someplace Drive” |
| phoneNumber | integer | 9164455959 |
| price | float | 100.50 |
| dates | List of strings | <“May 15”,”May 16”,”May 17”,”May 18”> |
| creditCardNumber | integer | 4568424120012548 |
| roomNumbers | Array of boolean | [1,0,0,1] |
| oneDayPayment | Float | 25.50 |
| holdStatus | boolean | 1 |
| timeLimit | string | |

holdStatus shall not be set to 0 for this test case. timeLimit shall not receive an input for this test case.

### 3.3.4. Outputs specification

Expected output shall be a new reservation entry displayed on the calendar in the calendar page of the reservation manager. The new reservation shall be displayed on the calendar with the appropriate color determined by the holdStatus and roomNumber inputs (I.e. green reservation for holdStatus set to 1, red reservation for roomNumbers set to [1,1,1,1].

### 3.3.5. Special environment conditions

This Test Case shall require computer, monitor, mouse and keyboard, database setup comparable to the hardware available to the end users John and Jane. This test case shall require the installation of the Reservation manager prototype on a windows operating system. Testers shall not require training on the use of the system nor knowledge of the internal systems.

### 3.3.6. Special procedural requirements

This test case shall require the reservation manager system to be launched and the calendar page to be selected. This test case shall require a database system comparable to that used by the end users John and Jane.

### 3.3.7. Execution procedure steps

1. Launch Reservation manager system
2. Select Calendar Page
3. Select new reservation option
4. Input user information and submit
5. Check calendar for correct display of reservation based on inputs

### 3.3.8. Dependencies

Test case UI_TC_2 must be executed prior to this test case to ensure that the connection between the reservation manager system and the database system are established.

## 3.4. User Case Scenario 2

### 3.4.1. Test case identifier

Test Case ID: **US_TC_2**
“User Scenario 2 Test Case”

### 3.4.2. Test objective

This shall be the “Black Box” test case of the User scenario 2 described in the software requirements specifications document. This test case shall be used to determine if the hold status and time limit functionality of the reservation manager produces a non-guaranteed reservation as described in the user requirements.

### 3.4.3. Input specification

| **Name** | **Description** | **Example** |
|---|---|---|
| New Reservation | Button Press, pop-up table of inputs | |
| name | string | “John Smith” |
| address | string | “101 Someplace Drive” |
| phoneNumber | integer | 9164455959 |
| price | float | 100.50 |
| dates | List of strings | <“May 15”,”May 16”,”May 17”,”May 18”> |
| creditCardNumber | integer | 4568424120012548 |
| roomNumbers | Array of boolean | [1,0,0,1] |
| oneDayPayment | Float | 0.0 or blank |
| holdStatus | boolean | 0 |
| timeLimit | string | “May 15” |

oneDayPayment input shall be set to 0.0 or left blank for this test case. timeLimit Shall receive an input for this test case.

### 3.4.4. Outputs specification

Expected output shall be a non-guaranteed new reservation entry displayed on the calendar in the calendar page of the reservation manager. The new reservation shall be displayed on the calendar with the appropriate color determined by the holdStatus (I.e. purple reservation for holdStatus set to 0). Non-guaranteed reservation shall be deleted from the reservations list if timeLimit condition is met and oneDayPayment is 0.0.

### 3.4.5. Special environment conditions

This Test Case shall require computer, monitor, mouse and keyboard, database setup comparable to the hardware available to the end users John and Jane. This test case shall require the installation of the Reservation manager prototype on a windows operating system. Testers shall not require training on the use of the system nor knowledge of the internal systems.

### 3.4.6. Special procedural requirements

This test case shall require the reservation manager system to be launched and the calendar page to be selected. This test case shall require a database system comparable to that used by the end users John and Jane. This test case shall require a time commitment in order to test the reservation cancel functionality specified by the user requirements.

### 3.4.7. Execution procedure steps

1. Launch reservation manager system and select calendar page
2. Select new reservation
3. Input required fields
4. Set oneDayPayment to 0.0 or leave blank
5. holdStatus shall automatically be set to 0
6. Set timeLimit and submit reservation
7. Check reservation display on the calendar
8. Wait until timeLimit is reached or edit reservation oneDayPayment
9. Check calendar page when timeLimit is reached or when oneDayPayment is inputted

### 3.4.8. Dependencies

Test case UI_TC_2 must be executed prior to this test case to ensure that the connection between the reservation manager system and the database system are established.

# 4. Anomaly Reports

| Test Case # | Inputs | Expected results | Actual Results | Unexpected outcomes | Procedure steps | Environment | Attempts to repeat | Testers | Observers |
|---|---|---|---|---|---|---|---|---|---|
| 3.1 | | | | | | | | | |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 3.2 | | | | | | | | | |
| 3.3 | | | | | | | | | |
| 3.4 | | | | | | | | | |