Software Test Specifications(STS)

Bed & Breakfast(BnBAS)

Kevin Gole, Dan Beck, Christopher Mair

CMIS330

University of Maryland Global Campus

Professor Shash

2/23/2021

Table of Contents

1. Introduction

1.1 Objectives

The purpose of this Software Test Specifications (STS) document is to develop a testing structure for the main systems of the Bed-and-Breakfast Administrations Software(BnBAS). The document will provide an outline for black-box, and white-box testing to verify the design and requirements of the software.

1.2 Background

The BnBAS will be used to manage a small bed-and-breakfast business and will help manage and schedule reservations. In addition, the software will be used to make payments, store guest information, and store the businesses financial records. The testing included in this document will help verify that the software design is functional and efficient.

1.3 Scope

The testing outline is scoped to three specific modules of the BnBAS's Architectural Context Diagram. These modules include:

1. User Interface - Section 2.1
2. Services - Section 2.2
3. Domain Objects - Section 2.3

1.4 References

References for this document include:

1. Software Design Document (SDD)
2. Software Requirements Specifications (SRS)
3. IEEE Standard for Software Test Documentation, standard 829-1998

1.5 Test Environment

The BnBAS test environment is light due to the few requirements and minimal interaction between components. The environment should be easy for testers to use and assess.

1.5.1 Software

The BnBAS can run on the three most commonly used operating systems(OS); Windows, IOS(Mac), and Linux. The computer(s) being used for the BnBAS should have one of the listed

OS to function properly. In addition, the BnBAS should be installed on the OS to be verified for use.

1.5.2 Hardware

A computer with usable RAM and physical storage will house and run the BnBAS. Minimal RAM and storage are required as the software is simple. In addition to the computer, a mouse, a keyboard, and a monitor will be required to access the BnBAS.

1.5.3 Tools

A usable computer environment (defined in sections 1.5.1 & 1.5.2) is essential for testing of the BnBAS. The User Interface(UI) and database are private, therefore can only be used by the testers/clients. The method of testing will be completed by using the BnBAS UI. Testing through the database can be achieved through using a tool such as mySQL. The tester will need full access to the database to verify the software is working according to the requirements.

1.5.4 Data

The testers will need the following data to verify the BnBAS:

1. Calendar date ranges
    a. Non-overlapping date ranges to prove successful
    b. Overlapping date ranges to prove unsuccessful
2. Customer information
    a. Names
    b. Phone numbers
    c. Addresses
    d. Credit card info
3. Reservation information
    a. Price per day/night
    b. Guarantee dates
4. Bed-and-Breakfast Room Information
    a. Room numbers and its associated room (checks for vacancies)

2. Architectural Context Diagram(ACD) Mappings

2.1 User Interface

This module is for users of the system, such as managers, or front desk clerks, of the bed and breakfast. They are able to use the system via this interface along with the Square Stand to accept payments. Access to this user interface requires the users to have an iPad with the BnBAS installed and attached to the Square Stand.

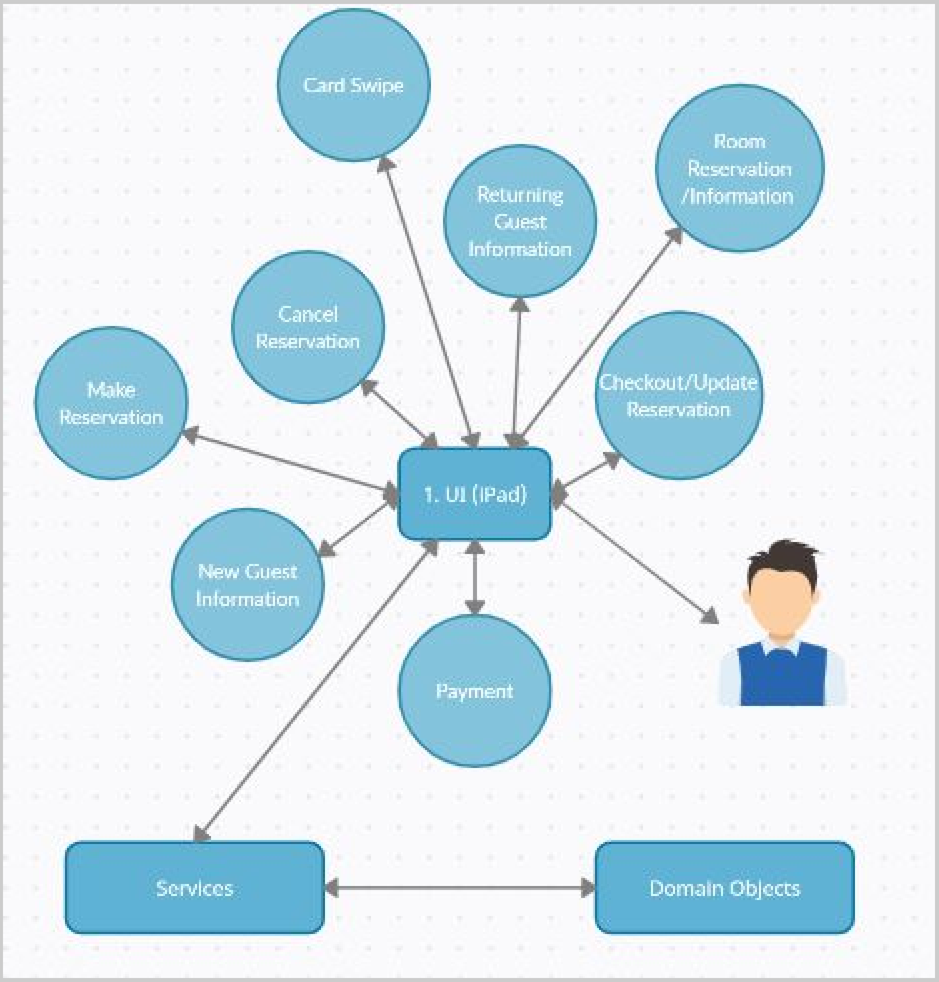


Figure 1, User Interface ACD

2.2 Services

This module contains the processes of the BnBAS where domain objects are involved, such as making a reservation or setting up room cleaning. The service module provides output to the user interface, and input from the user interface is used to manage domain objects where appropriate. This module also sends and retrieves data from the domain objects.

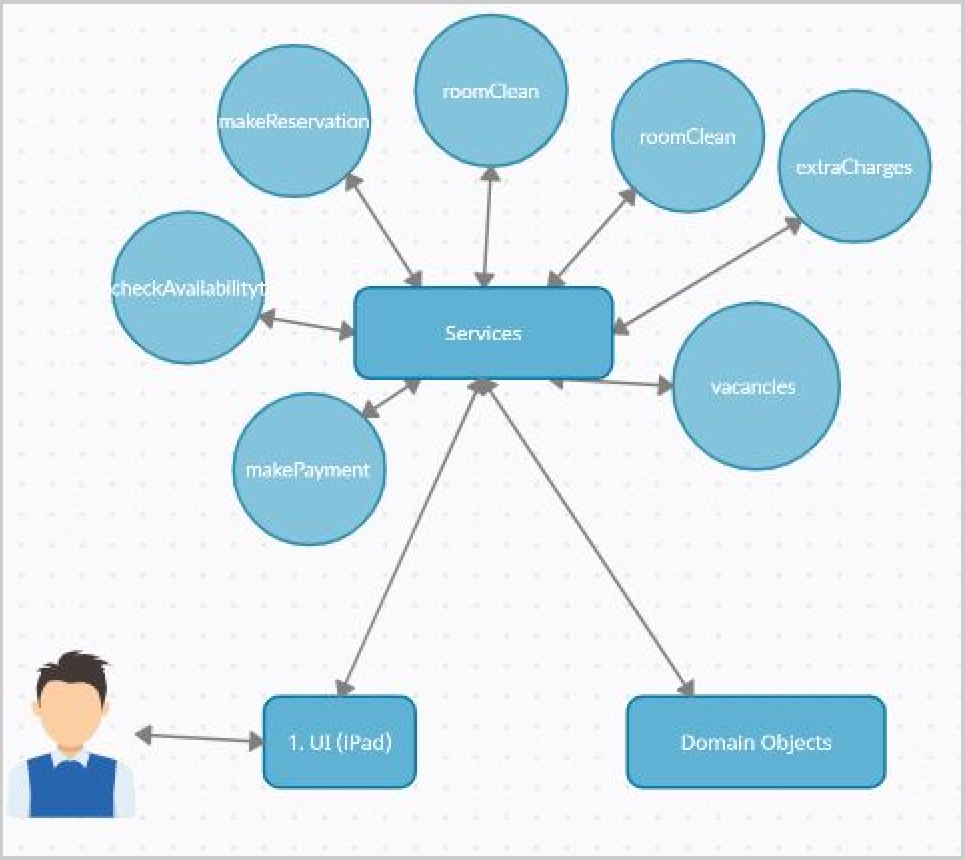


Figure 2, Services ACD

2.3 Domain Objects (Database)

This module covers all domain objects that hold data about a particular function in the BnBAS system.

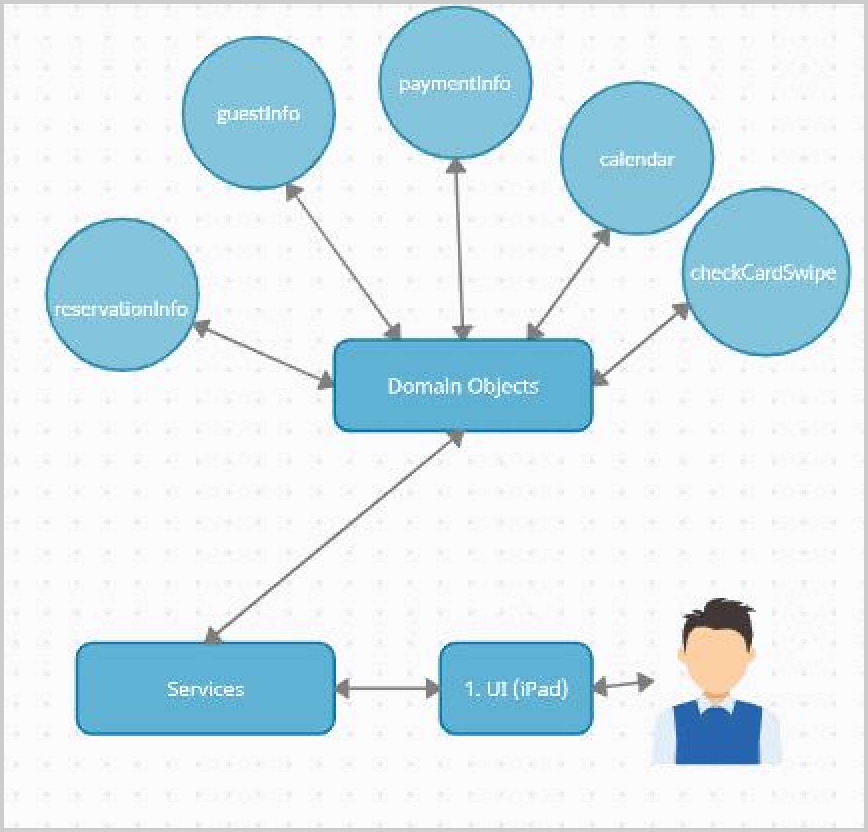


Figure 3, Domain Objects ACD

3. Traceability Matrix

The traceability matrix will provide a connection between the Software Requirements Specifications(SRS) and the Software Design Document(SDD) to the Software Testing Specifications(STS) cases defined below. The matrix will also indicate whether sections of the framework passed/failed their respective tests.

Category	Description	System Requirements	Use Case	Software Requirements	Test Case	Pass/Fail
User Interface(UI)	New Guest Information	3.1.1.5	Figure 5.	5.1	4.1.1	
User Interface(UI)	Make Reservation	3.1.1.4	Figure 6.	5.2	4.1.2	
User Interface(UI)	Cancel Reservation	3.1.1.4	Figure 7.	5.3	4.1.3	
User Interface(UI)	Returning Guest Information	3.1.1.5	Figure 5.	5.3	4.1.4	
User Interface(UI)	Payment	3.1.1.3	Figure 8.	5.4	4.1.5	
User Interface(UI)	Card Swipe	3.1.1.3	Figure 8.	5.4	4.1.6	
User Interface(UI)	Checkout/Update Reservation	3.1.1.2	Figure 9.	5.2	4.1.7	
User Interface(UI)	Room Reservation/Information	3.1.1.2	Figure 10.	5.2	4.1.8	
Services	makePayment	3.3.4	N/A	5.4	4.2.1	
Services	checkAvailability	3.3.2	N/A	5.2	4.2.2	
Services	makeReservation	3.3.3	N/A	5.5	4.2.3	
Services	roomClean	3.3.2	N/A	5.3	4.2.4	
Services	extraCharges	3.3.4	N/A	5.4	4.2.5	
Services	vacancies	3.3.2	N/A	5.3	4.2.6	

Database /Domain Objects	reservationInfo	3.3.3	N/A	5.5	4.3.1	
Database /Domain Objects	guestInfo	3.3.1	N/A	5.6	4.3.2	
Database /Domain Objects	paymentInfo	3.3.4	N/A	5.4	4.3.3	
Database /Domain Objects	calendar	3.3.3	N/A	5.2	4.3.4	
Database /Domain Objects	checkCardSwipe	3.3.5	N/A	5.4	4.3.5	

4. Test Case Specifications

4.1 User Interface Test Cases

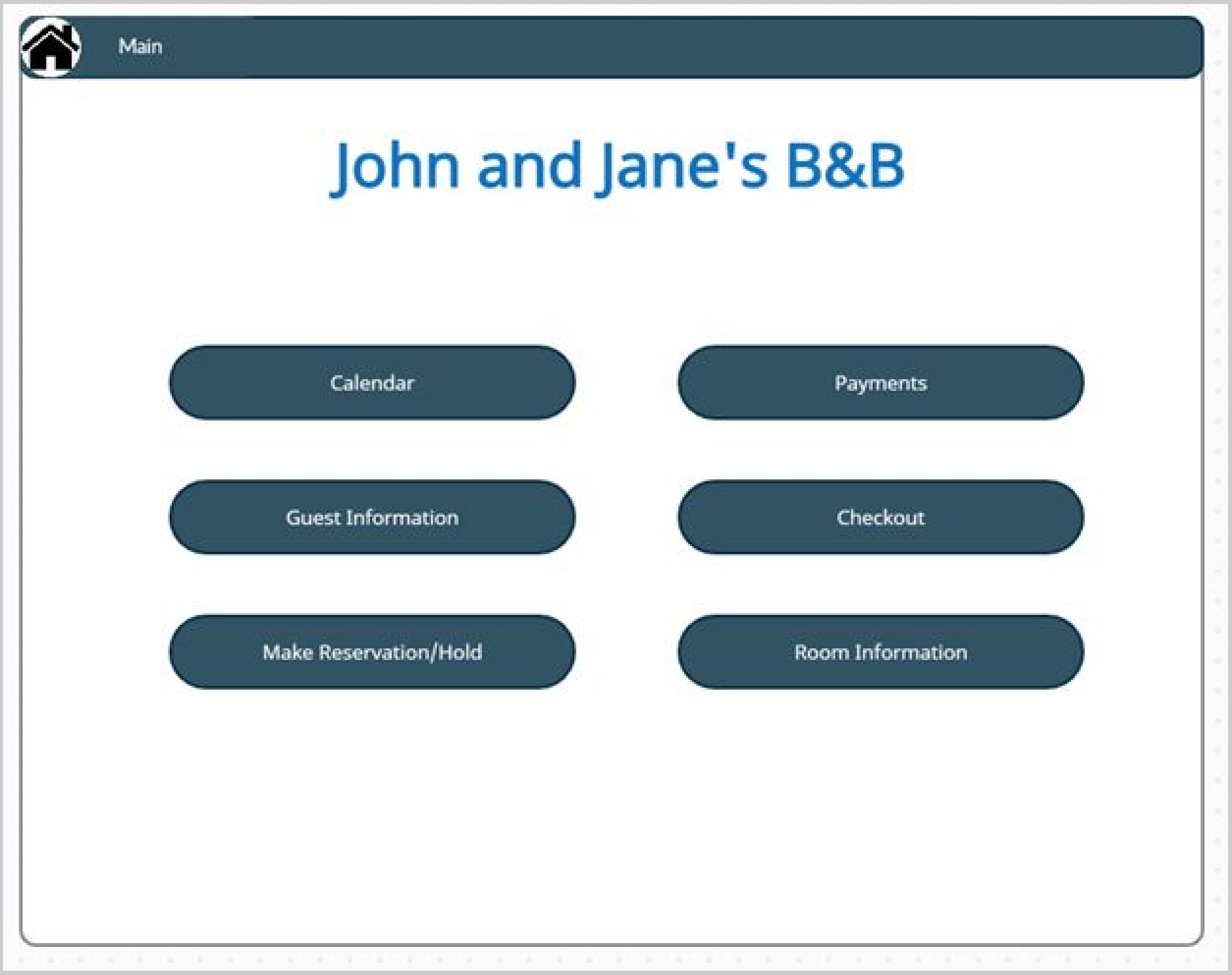
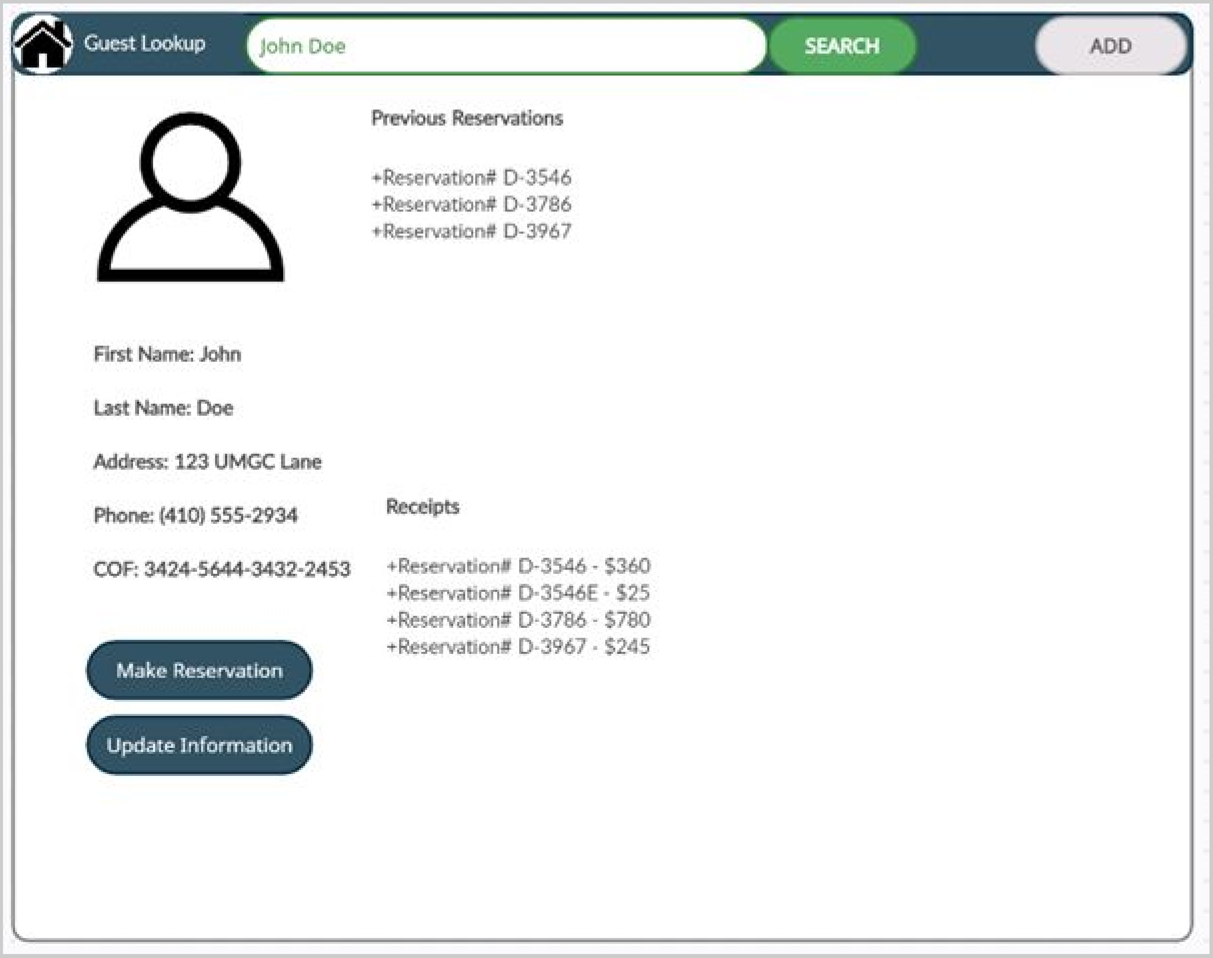


Figure 4, Main Page

4.1.1 ID: BnBAS-UI1 (Black-Box)



+Reservation# D-3546
+Reservation# D-3786
+Reservation# D-3967

First Name: John

Last Name: Doe

Address: 123 UMGC Lane

Phone: (410) 555-2934

COF: 3424-5644-3432-2453

+Reservation# D-3546 - $360
+Reservation# D-3546E - $25
+Reservation# D-3786 - $780
+Reservation# D-3967 - $245

Figure 5, Guest Lookup

Objective:

The objective of this test case is to provide a black-box test for the user to add information for a new guest.

Test Items:

This test will emulate the user adding a guest's information to their database utilizing the newGuest, and newCardSwipe method.

Input Specifications:

1. Guest Information is selected

2. The Add button is selected

3. Click on first name to add first name
4. Client first name is entered
5. Click on last name to add last name
6. Client last name is entered
7. Click on address to add address
8. Client address is entered
9. Click on phone to add phone number
10. Client phone number is entered
11. Click on COF to add card number
12. Client card number is entered
13. Client swipes new credit card
14. Update Information is selected

Output Specifications:

1. Guest Information page is opened
2. Guest Information shows blanks
3. Keyboard appears
4. Client first name is saved
5. Keyboard appears
6. Client last name is saved
7. Keyboard appears
8. Client address is saved
9. Keyboard appears
10. Client phone number is saved
11. Keyboard appears

12. Client card number is saved

13. Client card number is saved

14. All client information is saved to database

Environmental Needs:

1. iPad with BnBAS installed

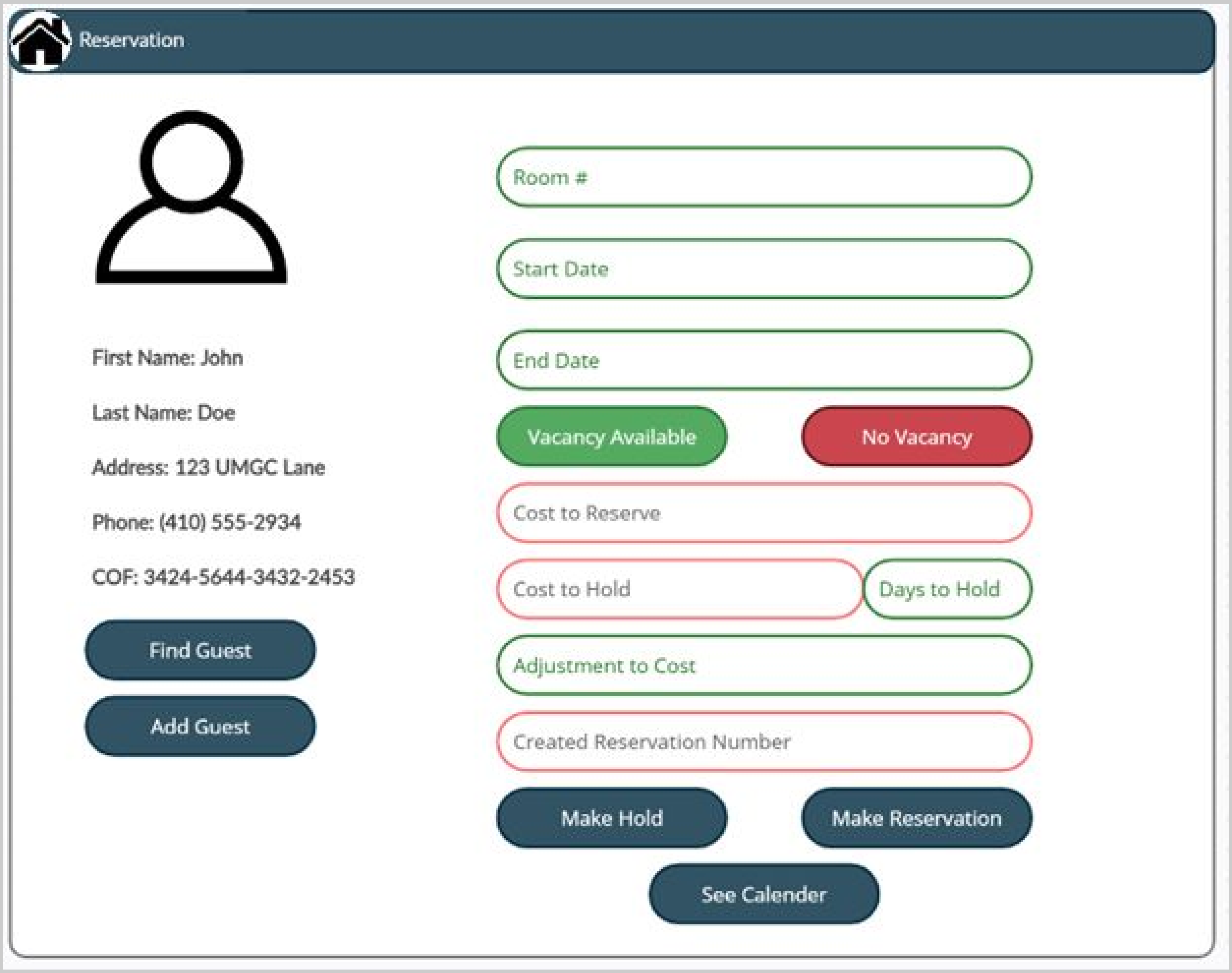
2. Square Stand with iPad attached

3. Power for Square Stand

4. Phone service for receiving calls

5. Wifi for iPad and Square Stand

4.1.2 ID: BnBAS-UI2 (Black-Box)



First Name: John

Last Name: Doe

Address: 123 UMGC Lane

Phone: (410) 555-2934

COF: 3424-5644-3432-2453

Figure 6, User Interface ACD

Objective:

The objective of this test case is to provide a black-box test for the Make Reservation portion of the software.

Test Items:

This test will emulate the user setting up a new reservation with a new guest utilizing the checkAvailability, and setReservation methods.

Input Specifications:

1. Make Reservation/Hold is selected
2. Room number, start/end dates are added
3. Adjustment to cost is added (such as discounts)
4. If hold is requested, Days to Hold is filled, then Make Hold is selected
5. Payment has been made
6. If reservation is requested, Make Reservation is selected
7. See Calendar is selected

Output Specifications:

1. Selected guest information is populated, and Make Reservation screen appears
2. If the days and room are available, the Vacancy Available indicator will brighten. Cost to reserve and cost to hold are filled.
3. Cost to reserve/hold is updated
4. Goes to payment screen
5. Calendar fills a hold for those dates in that room
6. Goes to payment screen
7. Calendar fills a reservation for those dates in that room
8. Shows the calendar page

Environmental Needs:

1. iPad with BnBAS installed
2. Square Stand with iPad attached
3. Power for Square Stand
4. Phone service for receiving calls
5. Wifi for iPad and Square Stand

4.1.3 ID: BnBAS-UI3 (Black-Box)

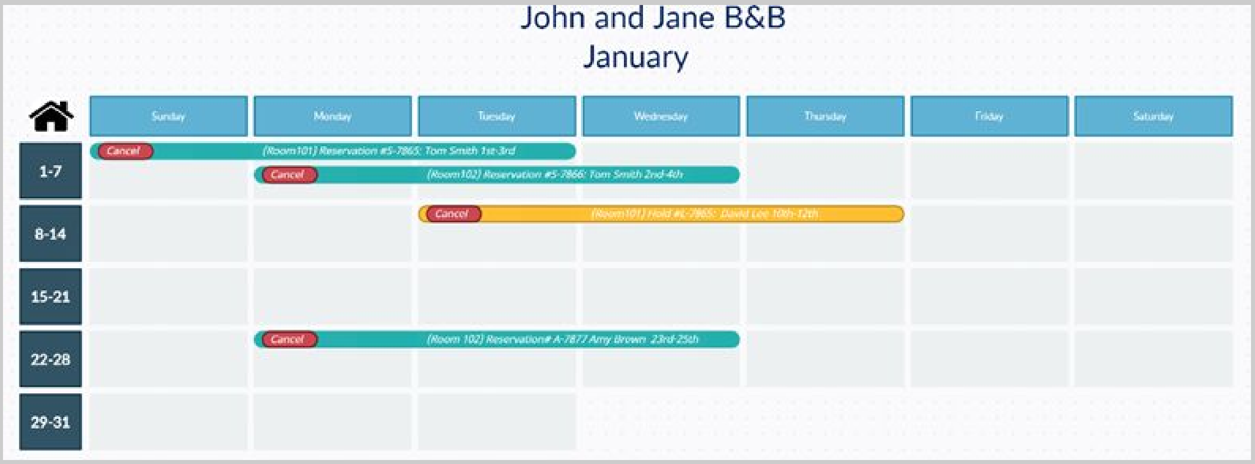

John and Jane B&B
January
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
1-7
8-14
15-21
22-28
29-31
Cancel
(Room101) Reservation #5-7865: Tom Smith 1st-3rd
Cancel
(Room102) Reservation #5-7866: Tom Smith 2nd-4th
Cancel
(Room101) Hold #L-7965: David Lee 10th-12th
Cancel
(Room 102) Reservation# A-7877 Amy Brown 23rd-25th

Figure 7, Calendar

Objective:

The objective of this test case is to provide a black-box test for cancelling a reservation with the software.

Test Items:

This test will emulate the user cancelling a reservation utilizing the cancelReservation method.

Input Specifications:

1. Calendar is selected
2. The reservation that is to be cancelled has its cancel button pressed
3. Yes, is selected in confirmation page

Output Specifications:

1. Shows the calendar page
2. A confirmation prompt shows
3. The reservation is cancelled

Environmental Needs:

1. iPad with BnBAS installed
2. Square Stand with iPad attached
3. Power for Square Stand
4. Phone service for receiving calls
5. Wifi for iPad and Square Stand

4.1.4 ID: BnBAS-UI4 (Black-Box)

Objective:

The objective of this test case is to provide a black-box test for finding a returning guest with the software.

Test Items:

This test will emulate the user finding a guest utilizing the recallGuest method

Input Specifications:

1. Guest Information is selected
2. Search button is pressed
3. Name is entered

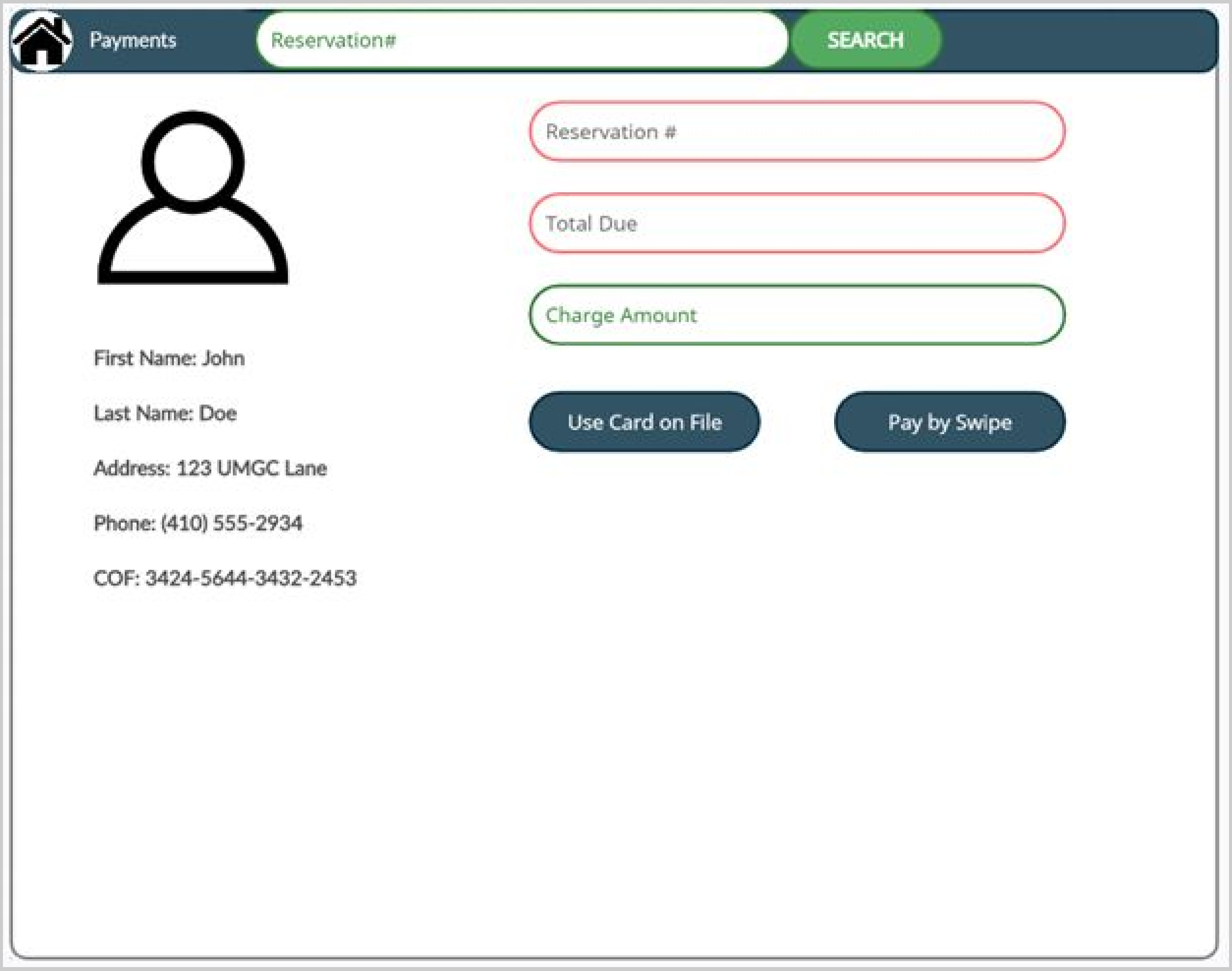
Output Specifications:

1. Shows the Guest Look-up page
2. Keyboard shows
3. Guest information is populated for the rest of the transaction showing previous reservations

Environmental Needs:

1. iPad with BnBAS installed
2. Square Stand with iPad attached
3. Power for Square Stand
4. Phone service for receiving calls
5. Wifi for iPad and Square Stand

4.1.5 ID: BnBAS-UI5 (Black-Box)



First Name: John

Last Name: Doe

Address: 123 UMGC Lane

Phone: (410) 555-2934

COF: 3424-5644-3432-2453

Figure 8, Payments

Objective:

The objective of this test case is to provide a black-box test for making a payment with the card on file

Test Items:

This test will emulate the user making a payment with the card on file utilizing the recallGuest, and guestPayment methods.

Input Specifications:

1. Payment is selected

2. The reservation number is entered or auto-filled based on if new reservation is being made

3. Charge Amount is entered

4. Use Card on File is selected

Output Specifications:

1. Shows the payments page

2. Reservation # and Total Due are auto filled

3. If left blank, the full amount will be charged

4. Guest's card on file is charged the Charge Amount

Environmental Needs:

1. iPad with BnBAS installed

2. Square Stand with iPad attached

3. Power for Square Stand

4. Phone service for receiving calls

5. Wifi for iPad and Square Stand

4.1.6 ID: BnBAS-UI6 (Black-Box)

Objective:

The objective of this test case is to provide a black-box test for making a payment with a new card swipe

Test Items:

This test will emulate the user making a payment with the card on file utilizing the newCardSwipe using the integrated Square Stand.

Input Specifications:

1. Payment is selected
2. The reservation number is entered or auto-filled based on if new reservation is being made
3. Charge Amount is entered
4. Pay By Swipe is selected

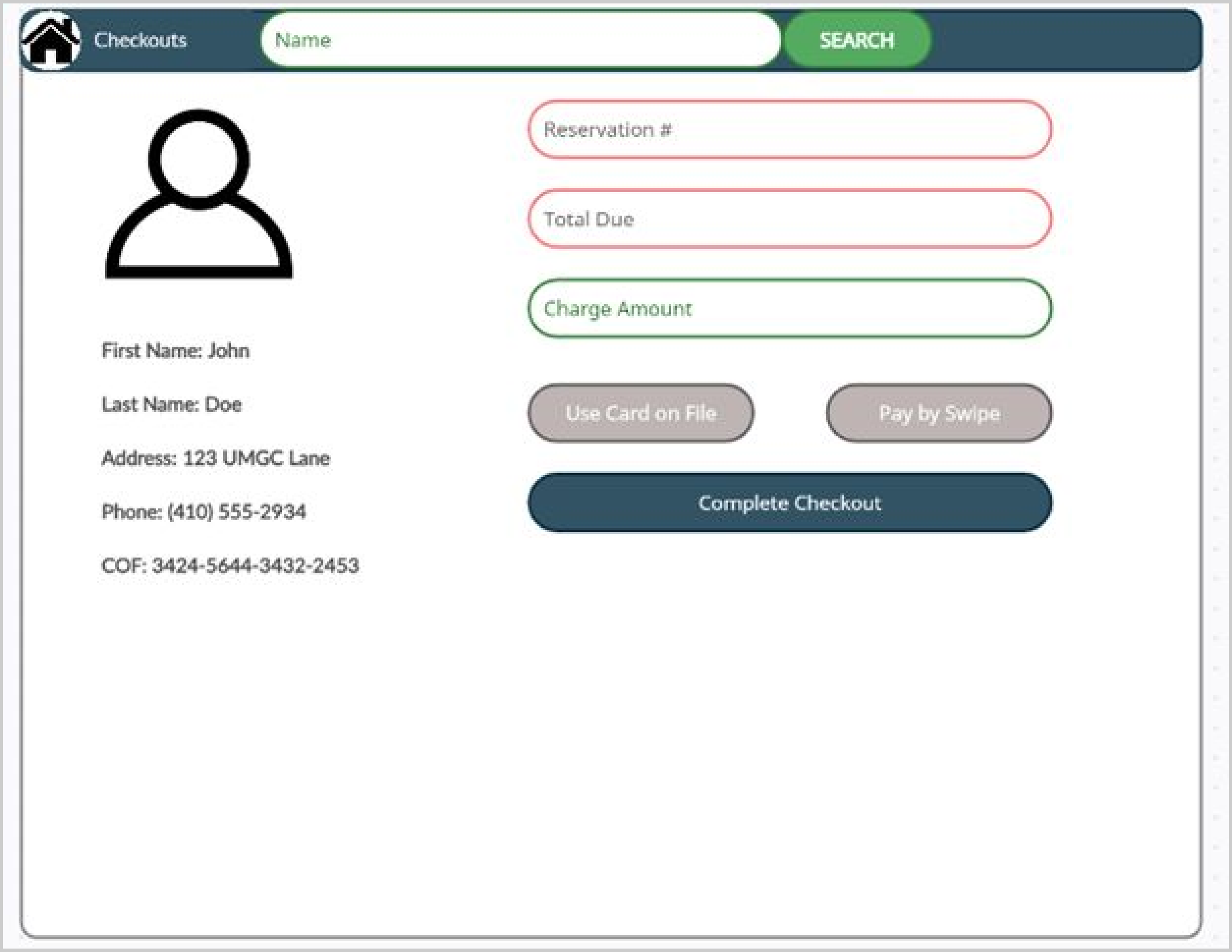
Output Specifications:

1. Shows the payments page
2. Reservation # and Total Due are auto filled
3. If left blank, the full amount will be charged
4. Guest swipes card

Environmental Needs:

1. iPad with BnBAS installed
2. Square Stand with iPad attached
3. Power for Square Stand
4. Phone service for receiving calls
5. Wifi for iPad and Square Stand

4.1.7 ID: BnBAS-UI7 (Black-Box)



First Name: John

Last Name: Doe

Address: 123 UMGC Lane

Phone: (410) 555-2934

COF: 3424-5644-3432-2453

Figure 9, Checkout

Objective:

The objective of this test case is to provide a black-box test for a guest checking out with the software.

Test Items:

This test will emulate the user checking a guest out utilizing the roomNeedsCleaning, guestExtraCharge, and checkout methods.

Input Specifications:

1. Checkout is selected

2. Name is entered

3. Extra Charge Amount is entered for any other charges

4. Complete Checkout is pressed

Output Specifications:

1. Shows the checkout page

2. Populates Reservation # and Total Due

3. If entered, Use Card on File and Pay by Swipe buttons become available

4. Guest is charged if Extra Charge Amount is not zero, the room from the reservation’s status changes to Room Needs Clean

Environmental Needs:

1. iPad with BnBAS installed

2. Square Stand with iPad attached

3. Power for Square Stand

4. Phone service for receiving calls

5. Wifi for iPad and Square Stand

4.1.8 ID: BnBAS-UI8 (Black-Box)

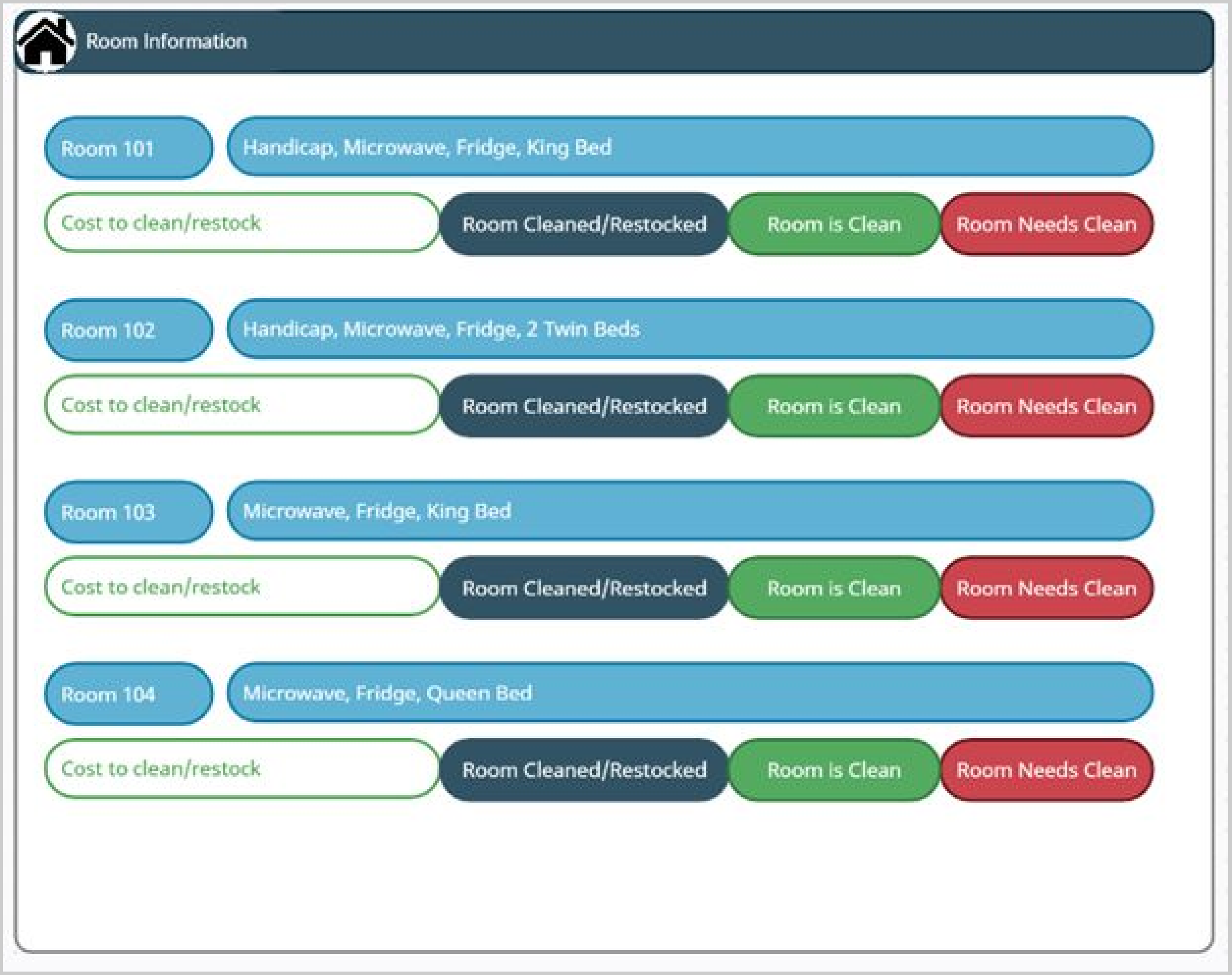


Figure 10, Room Information

Objective:

The objective of this test case is to provide a black-box test seeing the room information within the software.

Test Items:

This test will emulate the user checking the room cleanliness and adding the cost to refresh the room. This will utilize the roomExtraCharges, roomNeedsCleaned, and expenseForRoom methods

Input Specifications:

1. Room Information is selected

2. Cost to clean/restock is entered

3. Room cleaned/Restocked is pressed

Output Specifications:

1. Shows the room information page

2. Accepts the cost and adds to expenses database

3. Room is Clean illuminates, and the room can be allowed for a new reservation

Environmental Needs:

1. iPad with BnBAS installed

2. Square Stand with iPad attached

3. Power for Square Stand

4. Phone service for receiving calls

5. Wifi for iPad and Square Stand

4.2 Service Test Cases

4.2.1 ID: BnBAS-S1 (White-Box)

Objective:

The objective of this test case is to provide a white-box test for making payments with the software

Test Items:
This test will emulate the developer running different scenarios to ensure the purchasing process does not have any flaws/bugs.

Input Specifications:
1. Payment with certified check
2. Payment with Visa credit card
3. Payment with Mastercard credit card
4. Payment with known expired Visa credit card

Output Specifications:

1. Accepts payment with certified check
2. Accepts payment with Visa credit card
3. Accepts payment with Mastercard credit card
4. Declines payment after application communicated with credit card company

Environmental Needs:
1. Veracode to scan application for security flaws
2. iPad with iOS version of 13 or later
3. Path testing
4. Loop testing

4.2.2 ID: BnBAS-S2 (White-Box)

Objective:

The objective of this test case is to provide a white-box test for checking availability with the software.

Test Items:
This test will emulate the developer running program code within application to ensure availability accuracy

Input Specifications:
1. Manually check availability
2. Check availability via application
3. Changing status of room

Output Specifications:
1. 4 rooms available when manually checking
2. 4 rooms available per application
3. Confirmed status change via application immediately

Environmental Needs:
1. Veracode to scan application for security flaws
2. iPad with iOS version of 13 or later
3. Path testing
4. Loop testing

4.2.3 ID: BnBAS-S3 (White-Box)

Objective:

The objective of this test case is to provide a white-box test for making reservations with the software.

Test Items:
This test will emulate the developer stress testing the reservation process by testing different scenarios

Input Specifications:
1. Making a reservation within the application. Requesting one night during weekend
2. Making a reservation within the application. Requesting seven nights
3. Making a reservation within the application. Requesting 3 nights during the week
4. Making a reservation within the application. Requesting 300 nights.

Output Specifications:
1. Application successfully was able to locate available room and complete reservation.
2. Application successfully was able to locate available room and complete reservation.
3. Application successfully was able to locate available room and complete reservation.
4. Application declines request resulting with an error displaying maximum request number.

Environmental Needs:
1. Veracode to scan application for security flaws
2. iPad with iOS version of 13 or later
3. Path testing
4. Loop testing

4.2.4 ID: BnBAS-S4 (White-Box)

Objective:

The objective of this test case is to provide a white-box test for room cleaning with the software.

Test Items:
This test will emulate the developer monitoring accuracy whenever room cleaning has taken place.

Input Specifications:
1. Requesting application to display up to date room statuses
2. Manually adjust room status
3. Request dirty room during reservation process

Output Specifications:
1. Application displays accurate room status information
2. Application successfully updates after manual change of room status
3. Application displays room as unavailable

Environmental Needs:
1. Veracode to scan application for security flaws
2. iPad with iOS version of 13 or later
3. Path testing
4. Loop testing

4.2.5 ID: BnBAS-S5 (White-Box)

Objective:

The objective of this test case is to provide a white-box test for extra charges with the software.

Test Items:
This test will emulate the developer stress testing application and run multiple scenarios which will require extra charges.

Input Specifications:
1. Input room damage charge to application
2. Input valet parking to application
3. Input luxury suite reservation

Output Specifications:
1. Application successfully adds necessary damage charge to test account
2. Application successfully adds necessary valet charge to test account
3. Application successfully adds necessary luxury suite charge to test account

Environmental Needs:
1. Veracode to scan application for security flaws
2. iPad with iOS version of 13 or later
3. Path testing
4. Loop testing

4.2.6 ID: BnBAS-S6 (White-Box)

Objective:

The objective of this test case is to provide a white-box test for hold set-up with the software.

Test Items:
This test will emulate the developer monitoring application in how it reacts to various scenarios

Input Specifications:
1. Change status of room to On Hold
2. Change status of room from On Hold to Ready
3. Change status of room On Hold to Needs Service

Output Specifications:

1. Application successfully updates room On Hold status
2. Application successfully updates room On Hold status
3. Application successfully updates room On Hold status

Environmental Needs:

1. Veracode to scan application for security flaws
2. iPad with iOS version of 13 or later
3. Path testing
4. Loop testing

4.3 Domain Object/Database Test Cases

4.3.1 ID: BnBAS-D1 (White-Box)

Objective:

The objective of this test case is to provide a white-box test for reservations with the software

Test Items:
This test will emulate the developer confirming database accuracy during reservation test scenarios

Input Specifications:

1. Successfully make a reservation for room 101 for 5 days
2. Successfully make a reservation for room 203 for 2 days
3. Successfully fill out a reservation with test account but decline available rooms

Output Specifications:

1. Database populates with Room 101 reservation information
2. Database populates with Room 203 reservation information
3. Database stores test reservation and stores information

Environmental Needs:

1. Veracode to scan application for security flaws
2. iPad with iOS version of 13 or later
3. Path testing
4. Loop testing

4.3.2 ID: BnBAS-D2 (White-Box)

Objective:

The objective of this test case is to provide a white-box test for storing guest information.

Test Items:
This test will emulate the developer monitoring database while guest information is stored.

Input Specifications:
1. Input test guest 1 who will be reserving Room 208 for 3 days
2. Input guest 2 who will be reserving room 111 for 2 days
3. Input guest information who browsed application but did not reserve a room

Output Specifications:
1. Application stored information properly in database
2. Application stored information properly in database
3. Application stored information properly in database

Environmental Needs:
1. Veracode to scan application for security flaws
2. iPad with iOS version of 13 or later
3. Path testing
4. Loop testing

4.3.3 ID: BnBAS-D3 (White-Box)

Objective:

The objective of this test case is to provide a white-box test for the softwares ability to store and organize financial information.

Test Items:
This test will emulate the developer monitoring database to ensure accurate and organized financial information.

Input Specifications:
1. Complete payment for 5 day reservation for Room 303
2. Complete payment for 3 day reservation for Room 106 including adding $100 damage fee
3. Complete payment for 1 day reservation for Room 101 including $10 valet parking fee

Output Specifications:
1. Application stored financial information properly
2. Application stored financial information properly
3. Application stored financial information properly

Environmental Needs:
1. Veracode to scan application for security flaws
2. iPad with iOS version of 13 or later
3. Path testing

4. Loop testing

4.3.4 ID: BnBAS-D4 (White-Box)

Objective:

The objective of this test case is to provide a white-box test for calendar output and organization.

Test Items:
This test will emulate the developer stress testing calendar sync within application

Input Specifications:
1. Add one new reservation for Room 101 for 2 days
2. Add three rooms simultaneously for the upcoming weekend for 3 days
3. Add cleaning request for 4 rooms

Output Specifications:
1. Calendar syncs immediately and updates in application
2. Calendar syncs immediately and updates in application
3. Calendar syncs immediately and updates in application

Environmental Needs:
1. Veracode to scan application for security flaws
2. iPad with iOS version of 13 or later
3. Path testing
4. Loop testing

4.3.5 ID: BnBAS-D5 (White-Box)

Objective:

The objective of this test case is to provide a white-box test for expediting payments.

Test Items:
This test will emulate the developer running test payment scenarios to ensure proper expediting to financial companies. This process efficiency will improve customer satisfaction and security

Input Specifications:
1. Input valid paypal information
2. Input valid Visa credit card information
3. Input invalid Wells Fargo debit card information

Output Specifications:
1. Application successfully validates and completes payment within 3 seconds
2. Application successfully validates and completes payment within 3 seconds

3. Application successfully validates debit card information and returns an error message within 2 seconds

Environmental Needs:

1. Veracode to scan application for security flaws
2. iPad with iOS version of 13 or later
3. Path testing
4. Loop testing