Group Project: Software Requirements

PA2550: Seminar Series in Software Engineering

16th September 2024

Group ID: Group 1

Bed and Breakfast Reservation System

Student 1	Name	Makarudze, Anna
	E-Mail	anmk24@student.bth.se
	Swedish ID	860630T441
Student 2	Name	Mahapatra, Payel
	E-Mail	pamh24@student.bth.se
	Swedish ID	930618T729
Student 3	Name	Kotapati, Jammithri
	E-Mail	jako22@student.bth.se
	Swedish ID	20010610T223
Student 4	Name	Rokaya, Sanjeeb
	E-Mail	saro24@student.bth.se
	Swedish ID	010702T339
Student 5	Name	Arakkal Sudesh, Meenakshi
	E-Mail	meaa24@student.bth.se
	Swedish ID	951124R029

1 Project Description

1.1 Title

Bed and Breakfast Reservation System (BnB)

1.2 Project Aim and Objectives

The goal of this project is to develop a web application that allows users to make reservations for breakfast, accommodations, and social events like Concerts, Karaoke Night, and Scavenger Hunt offered by the BnB. The application can also be accessed by unregistered users who can view the rooms and events available in the system. They can also sign up and find all the information about the Bed and Breakfast. The users of the website are the Guests, Manager, and Front-desk Staff. The application will provide guests with an online booking platform to reserve rooms, breakfast, and social events. It will also display up-to-date information on room availability and upcoming social events at the BnB. Upon completing a reservation, guests will receive an email confirmation of the reservation. Guests can also cancel reservations. Payments are not handled by the system. It is handled in person by the Front-Desk Staff. Additionally, the system will notify the manager and front desk staff of all reservations and cancellations made through the website. The system will also allow the BnB manager and front desk staff to make reservations for guests. The system will also generate statistical reports, allowing the BnB manager to analyze booking trends and performance over time.

1.3 Objectives:

- To allow guests to make reservations for breakfast, accommodations, and social events.
- To provide an online booking platform that displays up-to-date information on room availability and social events.
- To send email confirmations to guests upon successful reservations.
- To allow guests to modify or cancel their reservations.
- To notify the manager and front desk staff of all reservations and cancellations made through the system.
- To enable the manager to modify the room's availability.
- To generate statistical reports on reservations made at various times. Reports would include the total number of guests at the BnB, total number of children at the BnB, total number of events at the BnB, total revenue incurred at the BnB, and total reservations made during a particular period of time. These reports will be viewed by the BnB Manager.

2 User Descriptions

Name	Description	Responsibilities
Guest	Any person who would use the system to make a reservation at the BnB must be registered. Unregistered persons can use the system but cannot make a reservation.	Search for available rooms, View available rooms, View social events, Sign Up, Make a reservation, Modify reservation, Cancel reservation.
Front Desk Staff	BnB staff who interact with guests at the front desk and manage room allocations and social events through the system.	View available rooms, Allocate rooms, Update availability of rooms, Create social events, Update social events, View social events, Enter user details, Make a reservation, Modify reservation, Cancel reservation, Confirm bookings, Check-in customers, Update payment status, Check-out customers.
Manager	The person who manages the BnB.	View available rooms, Allocate rooms, Update availability of rooms, Create social events, Update social events, View social events, Enter user details, Make a reservation, Modify reservation, Cancel reservation, Confirm bookings, Check-in customers, Update payment status, Check-out customers, View reports, Add room details, Create new users, Modify users, Modify room details.

Table 1: User Descriptions

3 Software Features and Requirements

A guest looking to book a room visits the BnB's website. The guest opens the website and searches for available rooms by entering their desired check-in and check-out dates, the number of adults, and the number of children. The guest views a list of available rooms and clicks on an individual room to see more details, including amenities and pricing. The guest clicks the "Reserve" button to save the reservation. A sign-up page appears, prompting the guest to enter their details: first name, last name, email, phone number, address, city, country, state(which is optional), ZIP Code and password. After successfully signing up, the guest is redirected to the reservation page, where they confirm the reservation details and submit the information. The system generates a unique booking reference. The guest receives a confirmation of their reservation along with a unique booking reference. The guest can log in at any time to view, modify, or cancel their reservation through their account.

Front desk staff log in to the system through the website to manage reservations. They search for available rooms to assist guests, create new reservations, or modify existing ones. The staff can also add or update information about social events offered by the BnB. When guests arrive, the front desk staff checks them in and updates their payment status in the system. Upon departure, the staff checks out the guests and marks the rooms as available for future reservations. The front desk staff receives notifications of all online reservations, modifications, and cancellations to keep them updated on the current reservations.

The Manager of the BnB logs in to access all the functionality available to the front desk staff, including managing reservations and room availability. In addition, the manager can view detailed reports on reservations and room utilization to monitor the BnB's performance. The manager also has administrative control to add or remove user accounts ensuring proper access control within the system.

3.1 Requirements

- REQ1: The system shall allow guests to sign up to the system.

  The guests shall provide the following details:

  - First name
  - Last name
  - Email address
  - Phone number
  - Address
  - City
  - State (optional)
  - Country

- REQ2: The system shall allow registered guests to log in and make reservations.

- REQ3 The system shall allow guests to search for available rooms based on the number of adults, number of children, and check-in and check-out dates.

- REQ4 The system shall allow guests to view individual rooms so that they can select rooms.
- REQ5 The system shall allow guests to reserve selected rooms and save the information.
- REQ6 The system shall allow guests to make reservations for social events they would like to participate in during their stay at the BnB.
- REQ7 The system shall send an email notification to the guest along with a unique booking code to confirm the reservation has been made with the BnB.
- REQ8 The system shall allow guests to modify or cancel their reservations.
- REQ9 The system shall send email notifications on modification or cancellation of the reservation to the guest, front desk staff, and manager.
- REQ10 The system shall allow the creation of BnB staff accounts by the BnB Manager. The following details shall be required:
  - First name
  - Last name
  - Email address
  - Phone number
  - Role
  - Address
  - ZIP Code
  - Country
  - City
  - State(optional)
- REQ11 The system shall allow the manager to submit room information.
- REQ12 The system shall allow the manager to modify room information.
- REQ13 The system shall allow BnB staff to add information on social events.
  The following details shall be required:
  - Name
  - Date
  - Start time
  - End time
  - Venue
  - Host
  - Minimum Number of people
  - Maximum number of people
  - Age restrictions

  - Additional Information

- REQ14 The system shall allow BnB staff to update social events.
- REQ15 The system shall allow the BnB staff to make reservations on behalf of guests.
- REQ16 The system shall send notifications to BnB staff on reservations and cancellations made by guests.
- REQ17 The system shall allow BnB staff to modify reservations, check-in and check out guests, and update payment status.
- REQ18 The system shall allow guests and BnB staff to log out of their accounts.
- REQ19 The system shall allow the manager to get reports on BnB reservations.
- REQ20 The system shall allow the BnB manager to modify staff accounts.

3.2 Non-functional Requirements

- The system shall ensure the privacy of guests information and limit access to reservations to logged-in users.
- The website supports operating systems like Windows 8 and above, and macOS 10.5 and above.

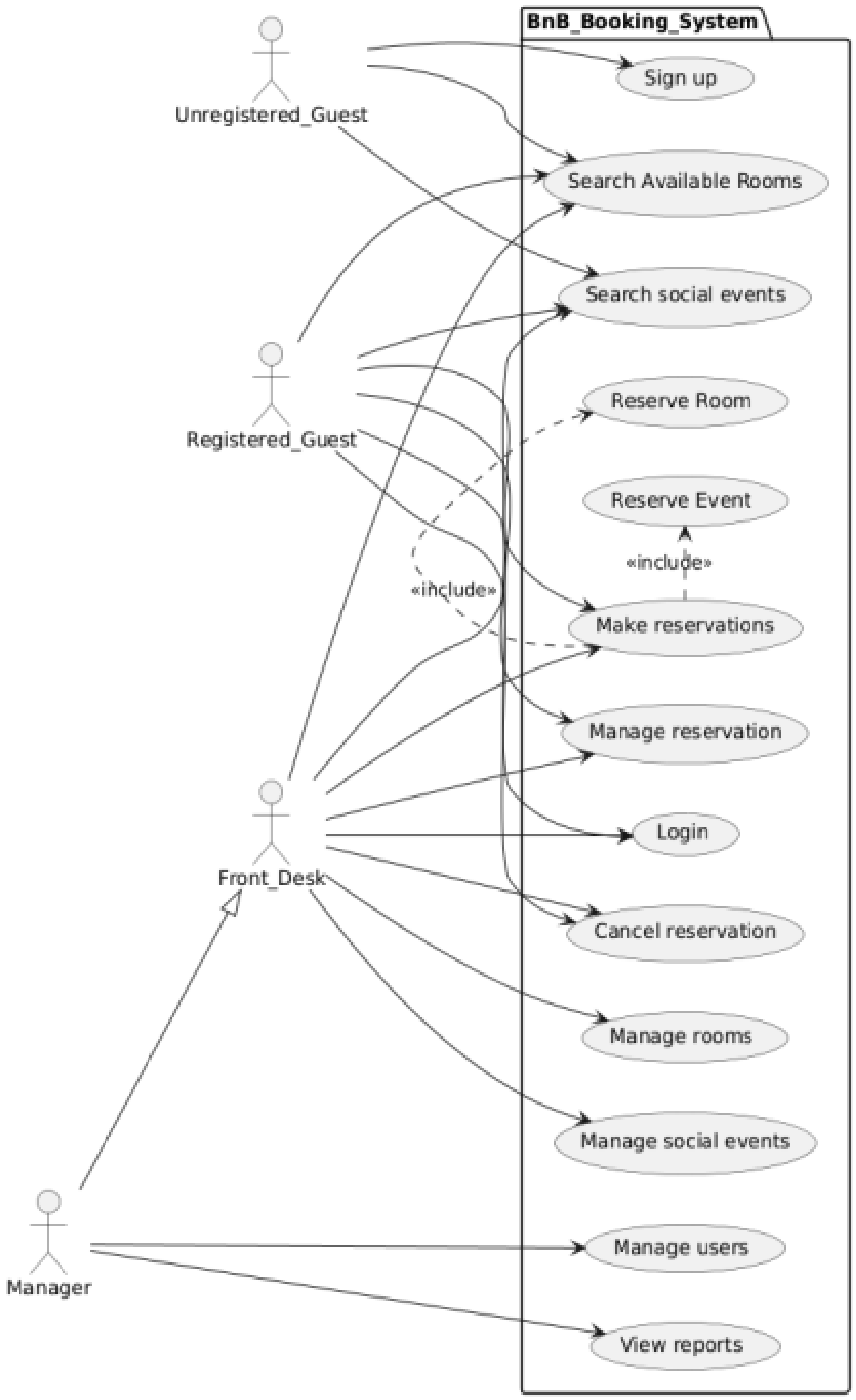
PA2550: Seminar Series in Software Engineering

Group Project: Use Case Analysis

23-09-2024

Student 1	**Name**	Makarudze, Anna
	Email	anmk24@student.bth.se
	Swedish ID	860630T441
Student 2	**Name**	Mahapatra, Payel
	Email	pamh24@student.bth.se
	Swedish ID	930618T729
Student 3	**Name**	Kotapati, Jammithri
	Email	jako22@student.bth.se
	Swedish ID	20010610T223
Student 4	**Name**	Rokaya, Sanjeeb
	Email	saro24@student.bth.se
	Swedish ID	010702T339
Student 5	**Name**	Arakkal Sudesh, Meenakshi
	Email	meaa24@student.bth.se
	Swedish ID	951124R029

1. UML Use Case Diagram



```
@startuml
left to right direction
actor Registered_Guest as rg
actor Unregistered_Guest as ug
actor Front_Desk as f
actor Manager as m
m --|> f

package BnB_Booking_System {
 usecase "Search Available Rooms" as UC1
 usecase "Search social events" as UC2
 usecase "Sign up" as UC3
 usecase "Make reservations" as UC4
 usecase "Cancel reservation" as UC5
 usecase "Login" as UC6
 usecase "Manage rooms" as UC7
 usecase "Manage social events" as UC8
 usecase "Manage reservation" as UC9
 usecase "Manage users" as UC10
 usecase "View reports" as UC11
 usecase "Reserve Room" as UC12
 usecase "Reserve Event" as UC13
}

rg --> UC1
ug --> UC1
f --> UC1
rg --> UC2
ug --> UC2
f --> UC2
ug --> UC3
rg --> UC4
f --> UC4
rg --> UC5
f --> UC5
rg --> UC6
f --> UC6
f --> UC7
f --> UC8
rg --> UC9
f --> UC9
m --> UC10
m --> UC11
UC4 .> UC12 : <<include>>
UC4 .> UC13 : <<include>>
actor Manager as m
@enduml
```

2. High-Level Use Cases

Use Case Name	Make Room Reservation
Use Case ID	UC01
Actor(s)	Registered Guests, Manager, Front Desk Staff
Type	Primary
Basic Flow	**Guest**, **Manager**, and **Front Desk Staff** who want to make a reservation for a room will visit the Bed and Breakfast website. The **Manager** and the **Front Desk Staff** log in to use the reservation system. **Guest**, **Manager**, and **Front Desk Staff** can choose the desired check-in and check-out dates and the number of guests. The website will display the available rooms on the specified dates. The **Guest**, **Manager**, and **Front Desk Staff** view individual rooms and click on the "Reserve" button against a desired room. The **Manager** and **Front Desk Staff** are redirected to a booking page where they enter the guest's details and book a room on behalf of the guest. The **Guest, Manager and Front-Desk Staff** will receive the booking confirmation and a unique booking reference number through email.

Use Case Name	Manage Social Events
Use Case ID	UC02
Actor(s)	Manager, Front Desk Staff
Type	Primary
Basic Flow	**Manager** and **Front Desk Staff** log in to the system and go to the Manage Event page by clicking the "Manage event" button on the homepage. The system displays a web page with the fields Name, Date, Start time, End time, Venue, Host, Minimum Number of people, Maximum number of people, Age restrictions, Additional Information. **Manager** and **Front Desk Staff** can enter the details of new social events and submit them. **Manager** and **Front Desk Staff** can view all the social events. **Manager** and **Front Desk Staff** can edit or cancel social events.

Use Case Name	Sign Up Guest
Use Case ID	UC03
Actor(s)	**Guest**
Type	Primary
Basic Flow	The **Guest** opens a sign-up page by clicking the "Sign Up" button on the homepage. The System displays the sign-up form with input fields; Name, Email, Contact Number, City, Zip code, State (optional), Country, Password, and Confirm password. The **Guest** fills out the form and submits the information and a new account is created.

Use Case Name	Cancel Reservation
Use Case ID	UC04
Actor(s)	Guest, Manager, Front Desk Staff
Type	Primary
Basic Flow	**Guest**, **Manager**, and **Front Desk Staff** log in to the system and enter a unique booking reference number. **Guest**, **Manager**, and **Front Desk Staff** view the booking that was made against the booking reference number. **Guest**, **Manager** and **Front Desk Staff** clicks on the “Cancel” checkbox against their booking. Booking is considered as cancelled and the details are updated. The **Guest**, **Manager**, and **Front Desk Staff** see a pop-up notification saying the booking is cancelled. **Guest**, **Manager**, and **Front Desk Staff** receive an email confirmation of the cancelled booking.

Use Case Name	Add Room
Use Case ID	UC05
Actor(s)	Manager and Front Desk Staff
Type	Primary
Basic Flow	**Manager** and **Front Desk Staff** who want to add a room will log in into the Bed and Breakfast website and click on “Add Room” button in the dashboard page. A form to fill room details will appear and after submitting the room details like Name, Description, Price, Room Type, and Room capacity, the room will be added in the website.

Use Case Name	View Reports
Use Case ID	UC06
Actor(s)	Manager
Type	Primary
Basic Flow	**Manager** can view performance report of the BnB by clicking on “View Reports” link on the homepage. The **Manager** enters the period for which he wants to view the reports and clicks on “Search” button. The Report page displays reports like number of guests who made reservations at the BnB, Total Revenue incurred by the BnB and Summary report like Occupancy Rate based on Room types.

3. Primary Use Cases

Use Case Name	**Reserve a Room**
Use Case ID	UC01
Actor(s)	Guest, Manager, and Front Desk Staff
Purpose	To make a reservation for a Room at BnB
Overview	The **Guest** can search and select a room by entering check-in, check-out dates, and number of guests. The **Guest** enters their details to make a reservation. The **Manager** and **Front Desk Staff** can do the same on behalf of the guest if needed. The **Guest** receives a booking confirmation email along with a unique booking reference number.
Type	Primary
Cross Reference	REQ3, REQ4, REQ5, REQ7
Pre-Condition(s)	**Manager** and **Front Desk Staff** should be logged in to the system to be able to reserve a room on behalf of the guest.
Main success scenario	
Actor's Actions	**System's Response**
1) **Guest**, **Manager**, and **Front Desk Staff** enter check-in date, check-out date, and number of guests in the search field present on the BnB homepage and click on the "Search" button.	2) The System displays a list of available rooms based on the search criteria provided.
3) **Guest**, **Manager**, and **Front Desk Staff** click on individual rooms to view in detail.	4) The System opens a new page with information on the selected room.
5) **Guest**, **Manager**, and **Front Desk Staff** click on the "Reserve" Button to book a room.	6) The System opens a new page with form fields: name, email, contact number, city, zip code, state(optional), and country.

7) **Guest**, **Manager**, and **Front Desk Staff** enter the details of the guest in the form and click submit.	8) The System saves the details it has received and sends a booking confirmation email to the guest with a unique booking reference number.
Post-Condition(s):	The System updates the information on room availability in the database.
Extensions (if any)	
Actor‘s Actions	**System’s Response**
1) The **Guest**, **Manager**, and **Front Desk Staff** enter the check-in date and check-out date which are earlier than the current date or the check-out date is earlier than the check-in date.	2) The System throws an error message saying “Invalid Dates”.

Use Case Name	**Viewing reports**
Use case ID:	UC02
Actor(s):	Manager
Purpose:	To allow the manager to view reports on BnB reservations
Overview:	The **Manager** can view periodic reports on the BnB reservations to determine business performance.
Type:	Primary
Cross Reference:	REQ19
Pre-Conditions:	The **Manager** should be logged in.
Main success scenario	
Actor's Actions	**System's Response**
1) The **Manager** selects the period for which he wants to view the report	2) The System processes the request and returns a summary of the BnB reservations based on the selected period.
Post-Condition(s):	The **Manager** can view a detailed report generated by the system.

PA2550: Seminar Series in Software Engineering

Group Project: Design Documentation

30-09-2024

Student 1	**Name**	Makarudze, Anna
	Email	anmk24@student.bth.se
	Swedish ID	860630T441
Student 2	**Name**	Mahapatra, Payel
	Email	pamh24@student.bth.se
	Swedish ID	930618T729
Student 3	**Name**	Kotapati, Jammithri
	Email	jako22@student.bth.se
	Swedish ID	20010610T223
Student 4	**Name**	Rokaya, Sanjeeb
	Email	saro24@student.bth.se
	Swedish ID	010702T339
Student 5	**Name**	Arakkal Sudesh, Meenakshi
	Email	meaa24@student.bth.se
	Swedish ID	951124R029

1. UML Interaction Diagrams

Use Case Name:	Make Room Reservation
Use Case ID:	UC01
Actor(s):	Guests, Manager, Front Desk Staff
Type:	Primary
Basic Flow:	**Guest, Manager**, and **Front Desk Staff** who want to make a reservation for a room will visit the Bed and Breakfast website. The logged in **Guest, Manager** and the **Front Desk Staff** can choose the desired check-in and check-out dates and the number of guests. The website will display the available rooms on the specified dates. The **Guest**, **Manager** and **Front Desk Staff** view individual rooms and click on the "Reserve" button against a desired room. The **Guest**, **Manager** and **Front Desk Staff** are redirected to a booking page where they enter the guest's details and book a room. The **Guest** will receive the booking confirmation and a unique booking reference number through email.

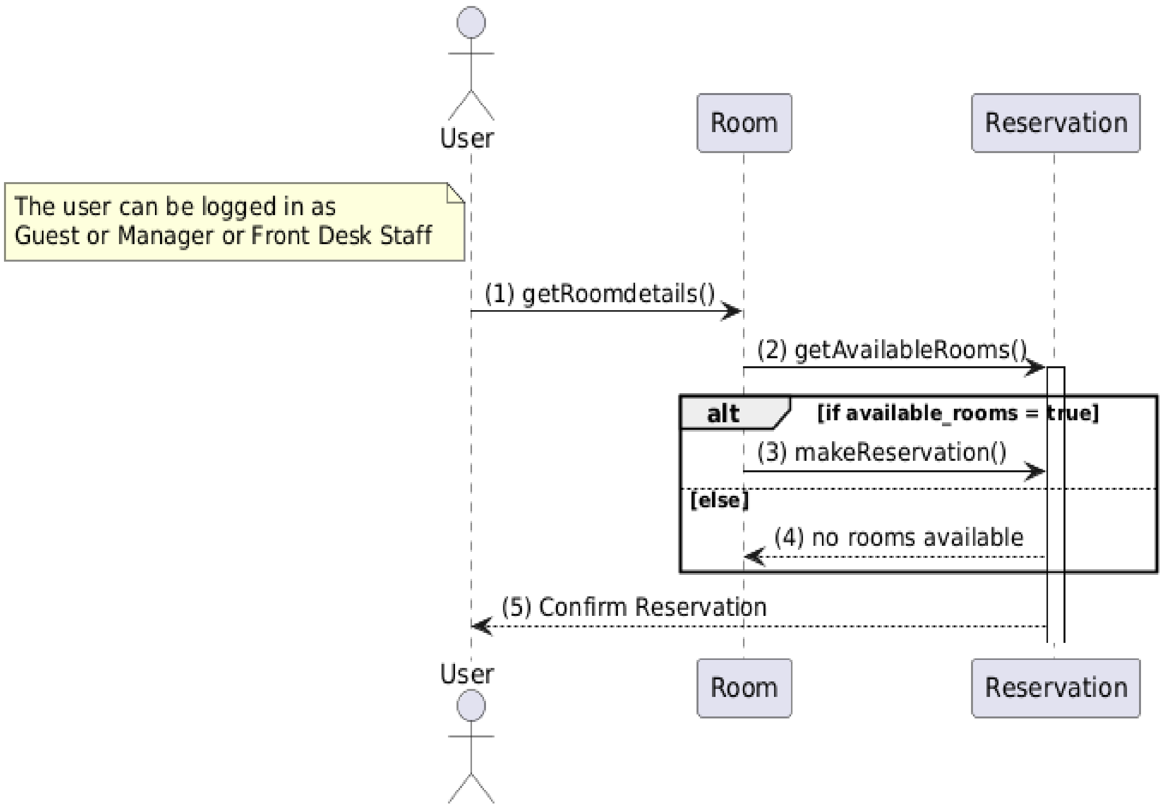


Figure 1: Reserve a Room Sequence Diagram

```
@startuml
autonumber "(#)"
actor User
note left of User
The user can be logged in as
Guest or Manager or Front Desk Staff
end note
User -> Room: getRoomdetails()
Room -> Reservation: getAvailableRooms()
activate Reservation
alt if available_rooms = true
 Room -> Reservation: makeReservation()
else else
 Reservation --> Room: no rooms available
end
 Reservation --> User: Confirm Reservation
@enduml
```

Use Case Name	Viewing reports
Use Case ID	UC02
Actor(s)	Manager
Type	Primary
Basic Flow	The **Manager** selects the period for which he wants to view the report. The System processes the request and returns a summary of the BnB reservations based on the selected period.

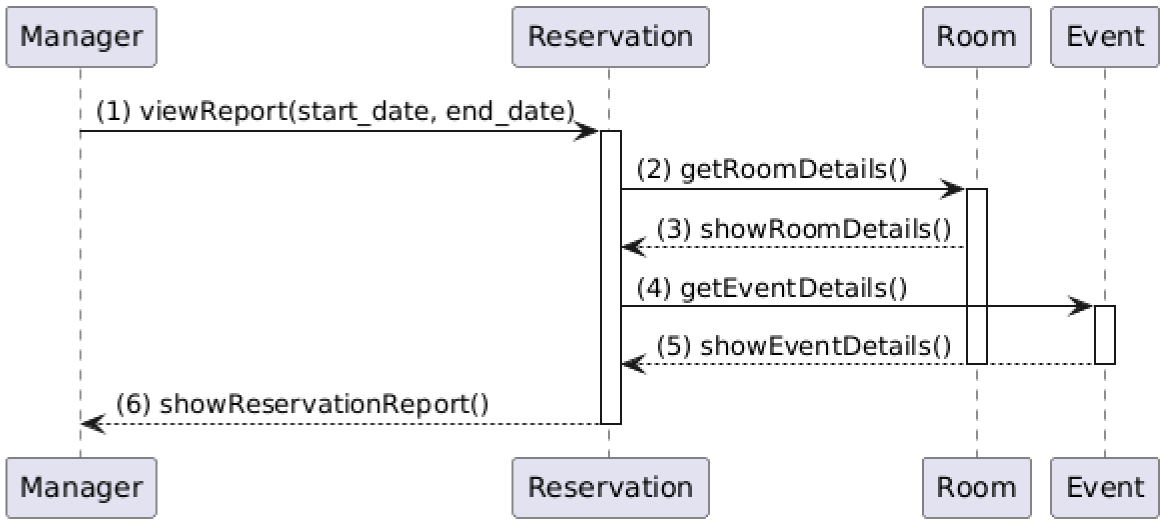


Figure 2: View Reports Sequence Diagram

```
@startuml
participant   Manager    as   Foo1
participant  Reservation   as  Foo2
participant Room              as Foo3
participant Event             as Foo4
```

```
autonumber 1 "(#)"

Foo1 -> Foo2 : viewReport(start_date, end_date)

activate Foo2

Foo2 -> Foo3 : getRoomDetails()

activate Foo3

Foo3 --> Foo2 : showRoomDetails()

Foo2 -> Foo4 : getEventDetails()

activate Foo4

Foo4 --> Foo2 : showEventDetails()

deactivate Foo3

deactivate Foo4

Foo2 --> Foo1 : showReservationReport()

deactivate Foo2

@enduml
```

2. UML Class Diagrams

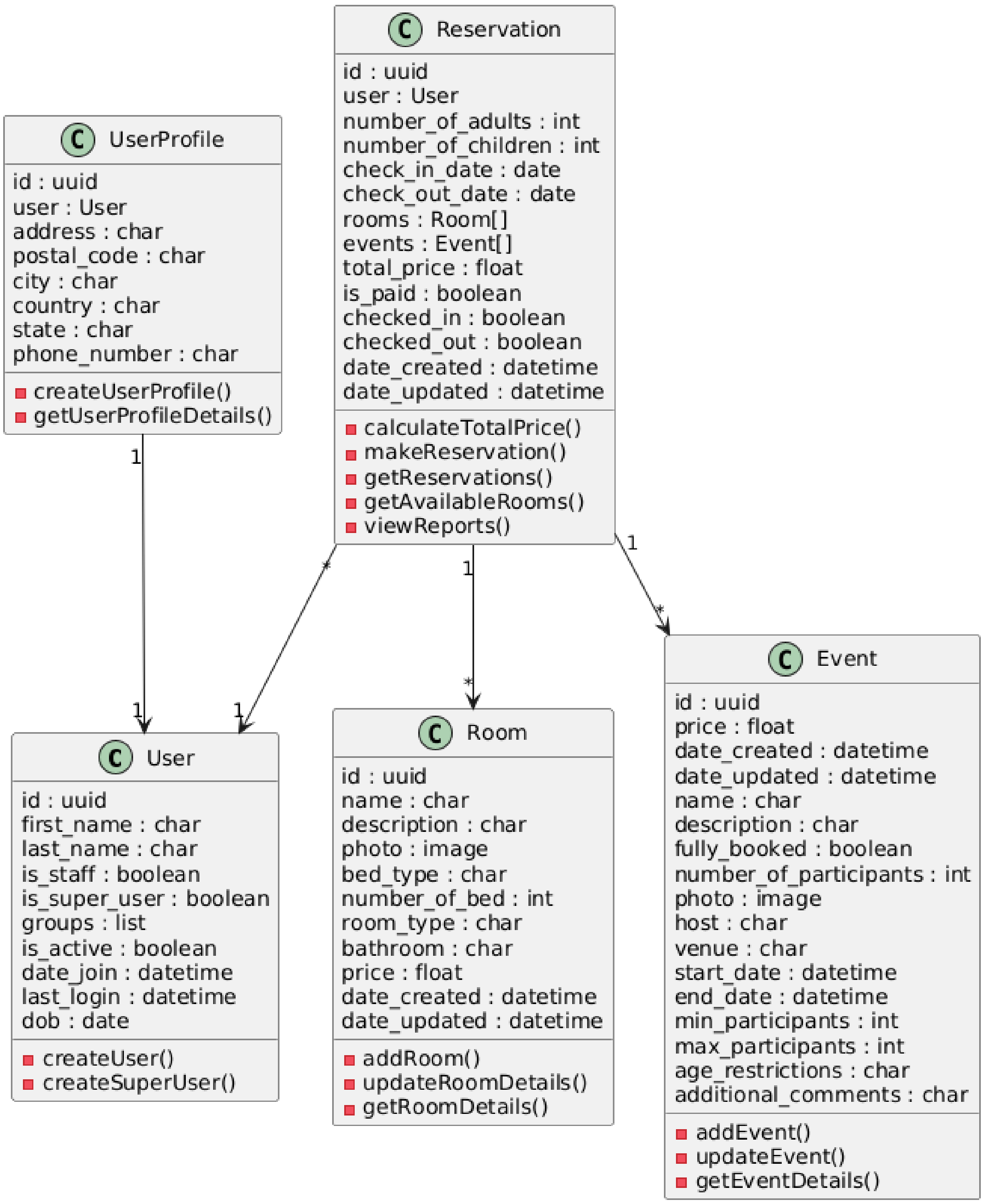


Figure 3: BnB System Class Diagram

```
@startuml

class User {
 id : uuid
 first_name : char
 last_name : char
 is_staff : boolean
 is_super_user : boolean
 groups : list
 is_active : boolean
 date_join : datetime
 last_login : datetime
 dob : date

-createUser()
-createSuperUser()
}

class UserProfile {
 id : uuid
 user : User
 address : char
 postal_code : char
 city : char
 country : char
 state : char
 phone_number : char

-createUserProfile()
-getUserProfileDetails()
}

class Reservation {
 id : uuid
 user : User
 number_of_adults : int
 number_of_children : int
 check_in_date : date
 check_out_date : date
 rooms : Room[]
 events : Event[]
 total_price : float
 is_paid : boolean
 checked_in : boolean
 checked_out : boolean
 date_created : datetime
 date_updated : datetime

-calculateTotalPrice()
-makeReservation()
-getReservations()
-getAvailableRooms()
-viewReports()

}

class Room {
 id : uuid
 name : char
 description : char
 photo : image
 bed_type : char
 number_of_bed : int
 room_type : char
 bathroom : char
 price : float
 date_created : datetime
 date_updated : datetime

-addRoom()
-updateRoomDetails()
-getRoomDetails()

}

class Event{
 id : uuid
 price : float
 date_created : datetime
 date_updated : datetime
 name : char
 description : char
 fully_booked : boolean
 number_of_participants : int
 photo : image
 host : char
 venue : char
 start_date : datetime
 end_date : datetime
 min_participants : int
 max_participants : int
 age_restrictions : char
 additional_comments : char

-addEvent()
-updateEvent()
-getEventDetails()
}

UserProfile "1"--> "1" User
Reservation "*"--> "1" User
Reservation "1" --> "*" Room
Reservation "1" --> "*" Event
@enduml
```

PA2550: Seminar Series in Software Engineering

Group Project: Quality Assurance Activity Report, including unit tests

14-10-2024

Student 1	**Name**	Makarudze, Anna
	Email	anmk24@student.bth.se
	Swedish ID	860630T441
Student 2	**Name**	Mahapatra, Payel
	Email	pamh24@student.bth.se
	Swedish ID	930618T729
Student 3	**Name**	Kotapati, Jammithri
	Email	jako22@student.bth.se
	Swedish ID	20010610T223
Student 4	**Name**	Rokaya, Sanjeeb
	Email	saro24@student.bth.se
	Swedish ID	010702T339
Student 5	**Name**	Arakkal Sudesh, Meenakshi
	Email	meaa24@student.bth.se
	Swedish ID	951124R029

1. **How many unit tests have been written for the project code units? Explain by giving numbers and details about code units and unit tastings. <u>(The code of at least 2 of your unit tests should be part of this report)</u>**

Code for unit testing Login Functionality

```
@pytest.fixture
def valid_login_form(db, guest):
    form = {"username": "elsa@test.com", "password": "pass1234"}
    return form



@pytest.fixture
def missing_email(db):
    form = {"username": "", "password": "pass1234"}
    return form



@pytest.fixture
def missing_password(db):
    form = {"username": "test@123.com", "password": ""}
    return form

def test_missing_email(manager_client, missing_email):
    """Test login with invalid credentials"""
    form = LoginForm(data=missing_email)
    assert form.is_valid() is False
    assert form.errors == {"username": ["This field is required."]}


def test_missing_password(manager_client, missing_password):
    """Test login with invalid credentials"""
    form = LoginForm(data=missing_password)
    assert form.is_valid() is False
    assert form.errors == {"password": ["This field is required."]}


def test_valid_login_form(valid_login_form, manager_client):
    """Test login with valid details"""
    form = LoginForm(data=valid_login_form)
    # assert form.is_valid()
    assert form.errors == {}
    def test_valid_login(manager_client, valid_login_form, client):

    """Test login with valid credentials"""
    response = client.get(reverse("accounts:login"))
    assert response.status_code == 200
    response = client.post(reverse("accounts:login"), data=valid_login_form)
    assert response.status_code == 302
```

Unit Test for Add Room Functionality by Manager and Front desk staff role

```
@pytest.fixture
def valid_room_form(db):
    return {
        "name": "Queen Room",
        "description": "A nice room to share.",
        "bed_type": "Single",
        "number_of_beds": 4,
        "room_type": "Queen",
        "bathroom":  "Ensuite",
        "room_capacity": 4,
        "price":              500.0,
        "can_be_rented": True,
    }
@pytest.fixture
def front_desk_client(client, front_desk):
    """Fixture to create a client to simulate what front desk staff can view while browsing the website."""
    client.force_login(front_desk)
    return client


@pytest.fixture
def manager_client(client, manager):
    """
    Fixture to create a client to simulate what the manager/admin can do while browsing the website.
    This user should be able to access everything.
    """
    client.force_login(manager)
    return client



def test_add_room_view_guest(guest_client):
    response = guest_client.get(reverse("rooms:add_room"))
    assert response.status_code == 403


def test_add_room_view_manager(manager_client, valid_room_form):
    response = manager_client.get(reverse("rooms:add_room"))
    assert response.status_code == 200
    response = manager_client.post(reverse("rooms:add_room"), data=valid_room_form)
    assert response.status_code == 302
def test_add_room_view_staff(front_desk_client, valid_room_form):
    response = front_desk_client.get(reverse("rooms:add_room"))
    assert response.status_code == 200
    response = front_desk_client.post(reverse("rooms:add_room"), data=valid_room_form)
    assert response.status_code == 302
```

Test Add Event Functionality by Manager and Front desk staff

```
@pytest.fixture
def valid_event_form(db):
  return {
      "name": "Concert Day",
      "description": "Concert by Ed Sheeran",
      "host": "Payel",
      "venue": "Multisalen",
      "start_date": "2024-12-05",
      "end_date": "2024-12-06",
      "min_participants": 100,
      "max_participants": 150,
      "num_participants": 130,
      "fully_booked": True,
      "price": 560,
      "age_restrictions": "No age restrictions",
      "additional_information": "No food allowed in concert hall",
  }
def test_add_event_view_guest(guest_client):
  response = guest_client.get(reverse("events:add_event"))
  assert response.status_code == 403


def test_add_event_view_staff(front_desk_client, valid_event_form):
  response = front_desk_client.get(reverse("events:add_event"))
  assert response.status_code == 200
  response = front_desk_client.post(
      reverse("events:add_event"), data=valid_event_form
  )
```

```
    assert response.status_code == 302


def test_add_event_view_manager(manager_client, valid_event_form):
    response = manager_client.get(reverse("events:add_event"))
    assert response.status_code == 200
    response = manager_client.post(reverse("events:add_event"), data=valid_event_form)
    assert response.status_code == 302
```

Test Make Room Reservation by a Guest

```
@pytest.fixture
def room(db):
    return Room.objects.create(
        name="Acacia",
        description="A nice single room.",
        photo="image.jpeg",
        bed_type="Double",
        number_of_beds=1,
        room_type="Single",
        bathroom="Shared",
        price=1000.00,
    )
@pytest.fixture
def valid_reservation_rooms(db, room, guest):
    """Fixture where rooms are missing"""
    return {
        "user": guest,
        "number_of_adults": 2,
        "number_of_children": 1,
        "check_in_date": "2024-12-10",
        "check_out_date": "2024-12-12",
        "rooms": [room.id],
    }
@pytest.mark.django_db
def test_make_reservation_view(guest_client, valid_reservation_rooms):
    response = guest_client.get(reverse("website:make_reservation"))
    assert response.status_code == 200
    response = guest_client.post(
        reverse("website:make_reservation"), data=valid_reservation_rooms
    )
    assert response.status_code == 200
```

Test Case ID	TC_001	**Test Case Description**	Login Functionality in BnB Reservation System		
Created By	Payel	**Reviewed By**	Anna	**Version**	1
<u>QA Tester's Log</u>	All the test cases are passed.				
Tester's Name	Payel	**Date Tested**	6-10-2024	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**
1	Account should be created for Guest, Manager, and Front-desk Staff.
2	
3	

S #	**Test Data**

1	Username = <u>elsa@test.com</u>
	Password = pass1234

2	Username = “”
	Password = pass1234

3	Username = <u>elsa@test.com</u>
	Password = “”

<u>Test Scenario</u>	Test the login functionality of the website with different test data.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Unit Test with valid username and password	200 OK	Assert True	Pass
2	Unit Test with empty username and a valid password	Error Message	Assert True	Pass
3	Unit Test with valid username	Error Message	Assert True	Pass

	and empty password			

Test Case ID	TC_002	**Test Case Description**	Add room functionality by user having Manager and Front-desk Staff Privileges		
Created By	Anna	**Reviewed By**	Jammithri	**Version**	1
QA Tester's Log	All the test cases are passed.				
Tester's Name	Anna	**Date Tested**	6-10-2024	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**
1	Manager and Front-desk Staff should be logged into the system to add rooms

S #	**Test Data**
1	"name": "Queen Room" "description": "A nice room to share" "bed_type": "Single" "number_of_beds": 4 "room_type": "Queen" "bathroom": "Ensuite" "room_capacity": 4 "price": 500.0 "can_be_rented": True

Test Scenario	Test the add room functionality by different users having different privileges.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed /Suspended**
1	Unit test to add room by a user having the Guest privileges	403 Forbidden	Assert True	Pass
2	Unit Test to add room by a user having Manager privileges	200 OK	Assert True	Pass

3	Unit test to add room by a user having Front-desk Staff privileges	200 OK	Assert True	Pass

Test Case ID	TC_003	**Test Case Description**	Add event functionality by user having Manager and Front-desk staff Privileges		
Created By	Jammithri	**Reviewed By**	Meenakshi	**Version**	1
QA Tester's Log	All the test cases are passed.				
Tester's Name	Jammithri	**Date Tested**	8-10-2024	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**
1	Manager and Front-desk Staff should be logged into the system to add events

S #	**Test Data**
1	"name": "Concert Day" "description": "Concert by Ed Sheeran" "host": "Payel" "venue": "Multisalen" "start_date": "2024-12-05" "end_date": "2024-12-06" "min_participants": 100 "max_participants": 150 "num_participants": 130 "fully_booked": True "price": 560 "age_restrictions": "No age restriction" "additional_information": "No food allowed in concert hall"

Test Scenario Test the add event functionality by different users having different privileges.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Unit test to add event by a user having the Guest privileges	403 Forbidden	Assert True	Pass
2	Unit Test to add event by a user having Manager privileges	200 OK	Assert True	Pass

3	Unit test to add event by a user having Front-desk Staff privileges	200 OK	Assert True	Pass

Test Case ID	TC_004	**Test Case Description**	Make room reservation by the Guest		
Created By	Meenakshi	**Reviewed By**	Sanjeeb	**Version**	1
QA Tester's Log	All the test cases are passed.				
Tester's Name	Meenakshi	**Date Tested**	10-10-2024	**Test Case (Pass/Fail/Not Executed)**	Pass

S #	**Prerequisites:**		**S #**	**Test Data**

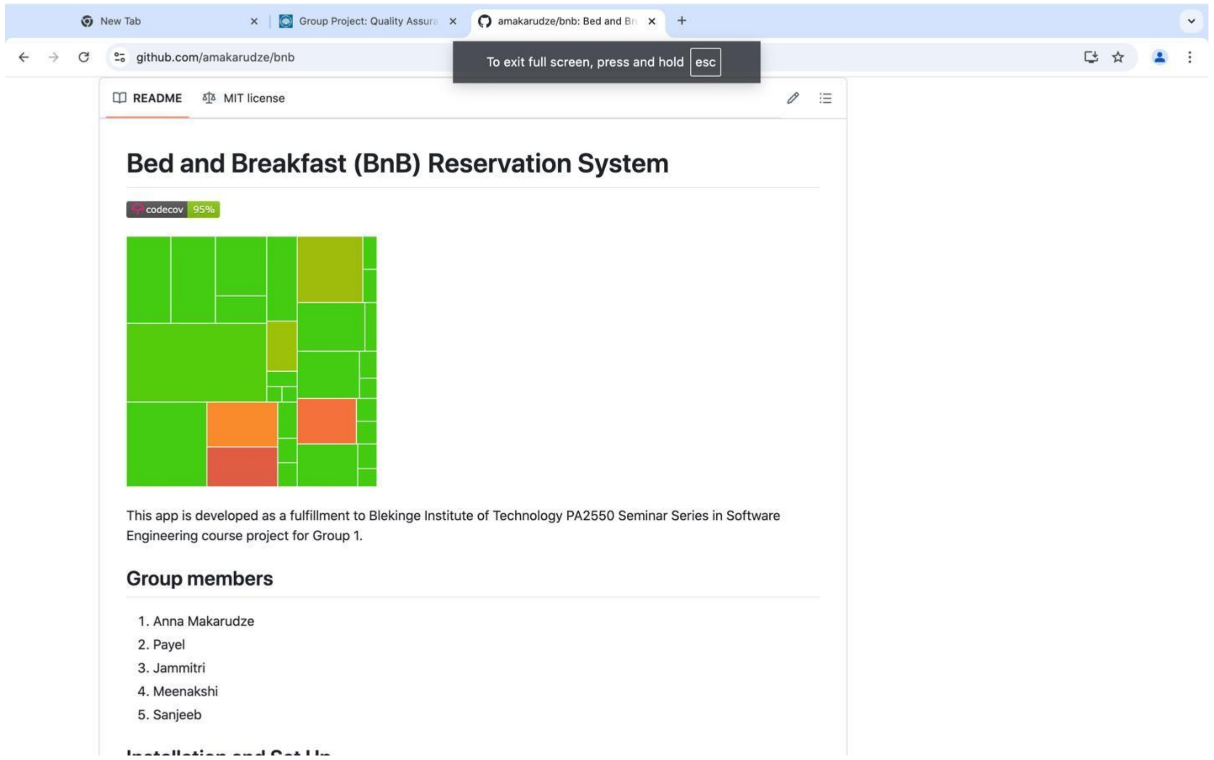
1	The guest should be signed up and logged before making a room reservation		1	"user": guest
				"number_of_adults": 2
				"number_of_children": 1
				"check_in_date": "2024-12-10"
				"check_out_date": 2024-12-12
				"rooms": [room.id]
				
				Room = {name="Acacia"
				description="A nice single room"
				photo="image.jpeg"
				bed_type="Double"
				number_of_beds=1
				room_type="Single"
				bathroom="Shared"
				price=1000.00}

Test Scenario	Test the make room reservation functionality by the Guest.

Step #	**Step Details**	**Expected Results**	**Actual Results**	**Pass / Fail / Not executed / Suspended**
1	Unit Test with valid Guest making room reservation	200 OK	Assert True	Pass

2. What percentage of code in your units has been tested by these unit tests?

We have been able to achieve 95% of code coverage for our codebase by running Python unit test cases(**pytest**).


New Tab
Group Project: Quality Assura
amakarudze/bnb: Bed and Br
github.com/amakarudze/bnb
To exit full screen, press and hold esc
README
MIT license
codecov 95%

Bed and Breakfast (BnB) Reservation System

This app is developed as a fulfillment to Blekinge Institute of Technology PA2550 Seminar Series in Software Engineering course project for Group 1.

Group members

1. Anna Makarudze
2. Payel
3. Jammitri
4. Meenakshi
5. Sanjeeb

Image 1: GitHub code coverage screenshot. GitHub link: *https://github.com/amakarudze/bnb*

3. **Have you written any integration tests (tests to check the functional correctness of task two or more units as they perform a task)? If yes, how many and provide details. If not, why not?**

   We have written one integration test for guests making reservations, starting when they visit the website, search for available rooms, select a room, sign up, and make a reservation.
   The code for this test is shown below:

```
import pytest

from django.conf import settings
from django.core import mail
from django.test import RequestFactory
from django.urls import reverse

from pytest_django.asserts import assertRedirects

from accounts.views import send_email
from website.views import make_reservation

@pytest.mark.transactional_db(True)
def test_guest_reservation_process(
    client,
    guest,
    search_form_valid,
    valid_sign_up_form,
    valid_reservation_rooms,
    reservation,
    reservations,
    room,
    rooms,
    available_rooms,
    check_in_date,
    check_out_date,
    number_of_adults,
    number_of_children,
    upcoming_events,
):
    response = client.get(reverse("website:home"))
    assert response.status_code == 200

    response = client.get(reverse("website:search"), data=search_form_valid)
    assert response.status_code == 200
    assert "rooms" in response.context
    assert len(response.context["rooms"]) == 7

    response = client.get(reverse("website:make_reservation"))
    assert response.status_code == 302
    assertRedirects(response, "/accounts/login/?next=/make_reservation/")

    response = client.get(reverse("accounts:signup"))
    assert response.status_code == 200
    response = client.post(reverse("accounts:signup"), data=valid_sign_up_form)

    to_email = [guest.email]
    send_email(
        "Sign Up Confirmation",
        "Test signup confirmation.",
        settings.DEFAULT_FROM_EMAIL,
        to_email,
    )
    assert len(mail.outbox) == 2
    assert mail.outbox[0].subject, "Signup Confirmation"

    assert response.status_code == 200

    client.force_login(guest)
```

```
factory = RequestFactory()
request = factory.get(reverse("website:make_reservation"))
request.session = {
    "check_in_date": check_in_date,
    "check_out_date": check_out_date,
    "number_of_adults": number_of_adults,
    "number_of_children": number_of_children,
    "rooms": available_rooms,
    "events": upcoming_events,
}
request.user = guest
response = make_reservation(request)

assert response.status_code == 200

response = client.post(
    reverse("website:make_reservation"),
    data=valid_reservation_rooms,
)
assert response.status_code == 302
```

4. **Have you prepared and tested any acceptance tests? If yes, provide details. If not, why not?**

   We have executed a few acceptance tests through manual testing till now. The details are as below.

Test Scenario	**Acceptance Criteria**	**Status**
Test User login functionality	• All users can enter their email address and password • Form validates that email is in the correct format, else displays an error popup message • The password is masked “XXX”	Pass

	• All input fields are required, if not an error message popup is displayed. • The User is logged in after clicking the Submit button	
Test search available rooms functionality	• All users can enter the check-in date and check-out date, number of guests, and number of children in the search bar that is displayed on the homepage. • All available rooms should then be displayed after clicking the Search button. • If the check-in date is earlier than the current date or the check-out date is before the check-in date, the page displays an error message	Pass
Test Make a reservation by a guest	• The guest selects an available room and clicks the Reserve button. • Guest is redirected to a signup page to enter their personal details • Mandatory fields are required to be filled before submission, else an error message pops up. • Guest clicks on the Sign-up button and they are then redirected to the View Room page with reserve button. • Signed up Guest can now click on Reserve button and they are redirected to Reservation Successful page.	Pass

5. How many bugs did you find as your group tested the code? Please share statistics. How many hours of effort during sprints have been spent on bug removal?

We have found around 5-10 bugs during unit testing of different modules. We detected these bugs by preparing thorough test case data that covered all possible scenarios of a functionality. In Python programming, we prepared "**pytest fixtures**" having test data for all possible outcomes of a functionality. We noticed while some test cases passed, others had failed with different test data and this led to us detecting the bugs and finding the resolution for it by fixing and refactoring the main code under test. We have been working in sprints having two weeks duration (80 hours) and in each sprint, we have spent 16 hours (2 days) approximately on fixing bugs detected through unit tests.

6. **Lastly, how confident are you (as a group) that the product is bug/error-free and ready for the demonstration? (Give a brief answer)**

   Each of our python forms and views have been unit tested thoroughly with all possible test data. Our website is able to perform the tasks that is expected from it and this verification has been done manually as well as through unit testing by each of the team member. We have handled all exception scenarios by displaying proper error message whenever the system receives data or user input that is not expected as part of the requirement. Therefore, we believe our website is bug and error free and ready for demonstration. In the unlikely case that any potential bugs arise which had been missed during our testing, will be addressed further by refining our test suit.