DEPARTMENT OF INFORMATION SYSTEMS

SYSTEMS DESIGN & DEVELOPMENT

SYSTEMS SPECIFICATION FOR *[PHUMLAKAMNANDI]*

TEAM MEMBERS

Student Number: VRYOWE001	***Student Name: Owen Vurayai***
Student Number: MHLGIV006	***Student Name : Given Mahlangu***
Student Number: NDZKHO004	***Student Name: Kholisile Ndzube***
Student Number: ZNGSIY012	***Student Name : Siyabonga Zungu***

PLAGIARISM DECLARATION

1. We know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. This Systems Specification is our own work.
3. We have not allowed and will not allow anyone to copy our work with the intention of passing it off as their own work.

Full Name: Kholisile Ndzube **Signed:** K.N **Date:** 04 /10/2024

Full Name: Owen Vurayai **Signed:** O.V **Date:** 04/10/2024

Full Name: Given Mahlangu **Signed**: G.M **Date:** 04/10/2024

Full Name: Siyabonga Zungu **Signed:** S.Z **Date**: 04/10/ 2024

TABLE OF CONTENTS

1. INTRODUCTION

1.1. OVERVIEW OF SPECIFICATION

This document outlines the system specifications for the **Phumla Kamnandi Hotel Management System**. We are creating the system to provide easy customer service and to increase customer satisfaction. The document follows the standard format provided in the project documentation, including details on corporate background, IT infrastructure, operations, screen layout, interface flow diagrams, test data definitions, and future requirements for the system. The document also consists of the detailed designed sequence diagrams, the report diagram, integrity controls and our implementation plan.

1.2. CONTEXT & SCOPE OF SYSTEM SPECIFICATION

The Phumla Kamnandi Hotel group is a recently established empowerment enterprise formed last year to manage a collection of small hotels located across South Africa. The group consists of multiple independently operating hotels, which benefit from being part of a larger organization in terms of marketing and brand recognition. Despite differences in size, services, and appearance, a key strategic objective is to provide a consistent guest experience across all hotels.

The executive team is committed to delivering high-end services and luxurious accommodations at competitive prices. To achieve this, the company is moving towards standardized processes, from check-in to check-out procedures, across all its properties. One critical element of this strategy is the introduction of a unified hotel management system, which will help ensure consistent business operations, a uniform presentation of customer-facing documents, and streamlined management and financial reporting. Additionally, the system will help reduce overall costs related to hardware, software, and employee training.

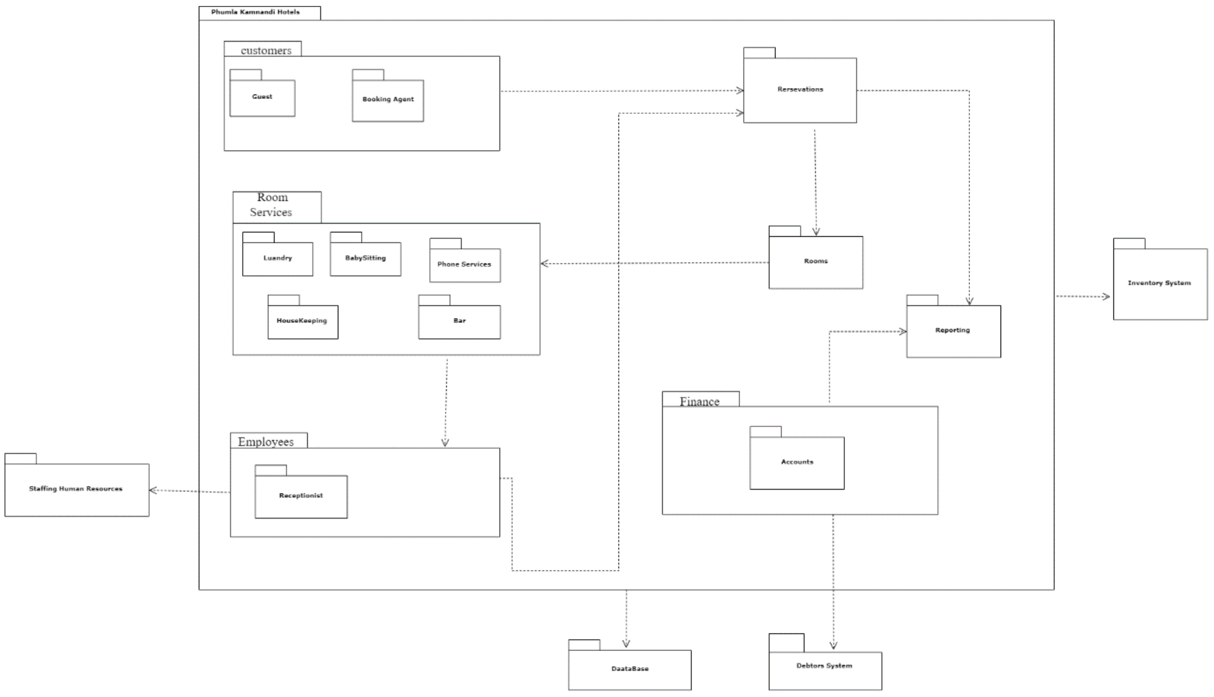
We the IT department at corporate headquarters have been tasked with modernizing the group's technological infrastructure over a three-year period, and the development of a new hotel management system is a key part of this effort. This system will be responsible for handling core hotel functions such as bookings, check-ins, check-outs, room updates, and room removals. The system will be built using C#.Net with a local SQL database, designed initially as a single-user application.

For reporting purposes, the system will be deployed at each hotel and linked to the corporate office system to allow for data aggregation and reporting. It will also generate two essential reports: a dynamic report on occupancy levels and another report that generates a list of expected check-ins on a given date.

- **Main Components and Relationships in the Package Diagram**:
- **Customers**:
  - **Guests** and **Booking Agents** are included in this package. It represents the people who directly book rooms or assist with bookings (agents).
- **Room Services**:
  - This includes specific services such as **Laundry**, **Housekeeping**, **Baby Sitting**, **Phone Services**, and **Bar**. These services are linked to the **Rooms** system, showing that they cater to guest needs during their stay.
- **Reservations**:
  - A key package that links **Customers** and the **Rooms**. Reservations are connected to **Reporting**, allowing the hotel to generate occupancy reports and manage data regarding which rooms are booked and available.
- **Employees**:
  - Specifically focusing on the **Receptionist** role, which manages guest check-ins, reservations, and is the system operator.
- **Finance**:
  - Includes **Accounts**, which likely with guest payments, room charges, and other financial transactions related to the guest's stay.
- **External Systems**:

- **Inventory System**: Tracks room supplies and another inventory.
- **Database**: Stores all the data about guests, rooms, services, guests' reservation sand account records.
- **Debtors System**: tracks outstanding payments or debts from guests.

- **Staffing Human Resources**:
  - This package is connected to employees, to manage staffing, shift schedules, and staff availability.
- **Explanation and Justification for Scope**
- **Scope of Developing**: The primary focus of the project is the **Reservation System**, **Rooms**, and **Reporting**. These   areas need enhancement to provide real-time insights into room availability, occupancy reports, and improve reservation accuracy.
- **Purpose**:
  - The **Reservation System** is the backbone of the hotel's operations. This system first ensures smoother room booking and avoids overbooking issues.
  - **Room Management** ensures real-time visibility to optimize room allocation and guest experiences.
  - **Reporting** allows hotel managers to make data-driven decisions regarding room utilization, occupancy trends, and service performance.
- **Critical Dependencies**:
  - **Reservations** are directly linked to the **Rooms** and **Reporting** systems. These must be aligned, as any changes to reservations impact room availability and reporting tools.
  - The **Finance** package is another dependency, as reservations and room occupancy affect billing and payment systems.
  - The **PhumlaniKamnadi Reservation System** is directly integrated with the **database**, allowing for the storage of all transactional records, guest details, reservation information, and other essential data needed for managing bookings and hotel operations efficiently.



Package Diagram

1.3. DESIGN ASSUMPTIONS & CONSTRAINTS

Our first assumption was that each hotel contains 400 rooms, categorized into four distinct room types: Single, Double, Deluxe, and Suite. The room IDs are divided into specific ranges for easy identification and organization. The breakdown of room types is as follows:

- **Single rooms**: Room IDs range from 1 to 100.
- **Double rooms**: Room IDs range from 101 to 200.
- **Deluxe rooms**: Room IDs range from 201 to 300.
- **Suite rooms**: Room IDs range from 301 to 400.

This allocation ensures that each room type has a designated set of room IDs, streamlining both room management and reservation processes.

We also assumed that each hotel branch maintains its own database to store crucial information, including:

- **Reservations**: All bookings made for the specific branch.
- **Guests**: Details of guests who have stayed at the branch.
- **Staff**: Most importantly, the staff members (e.g., receptionists) who are authorized to access the system being developed.

This decentralized database approach allows each hotel branch to have independent control over its data, ensuring secure and localized management of both guest and staff information.

Additionally, we assumed that one guest can have multiple rooms on their reservation, meaning a single reservation ID can be linked to several rooms. This allows flexibility for guests who wish to reserve more than one room during their stay, such as families or groups traveling together. Each room booked under the same reservation will be associated with the same guest profile, ensuring that billing and stay history remain consistent and accurate.

We also assumed that the system would manage real-time room availability, meaning that as soon as a room is booked, it is instantly marked as unavailable. This ensures that the system prevents any possibility of overbooking, keeping room inventory accurate at all times. The availability status will be updated dynamically, ensuring that other potential guests cannot reserve the same room for overlapping dates.

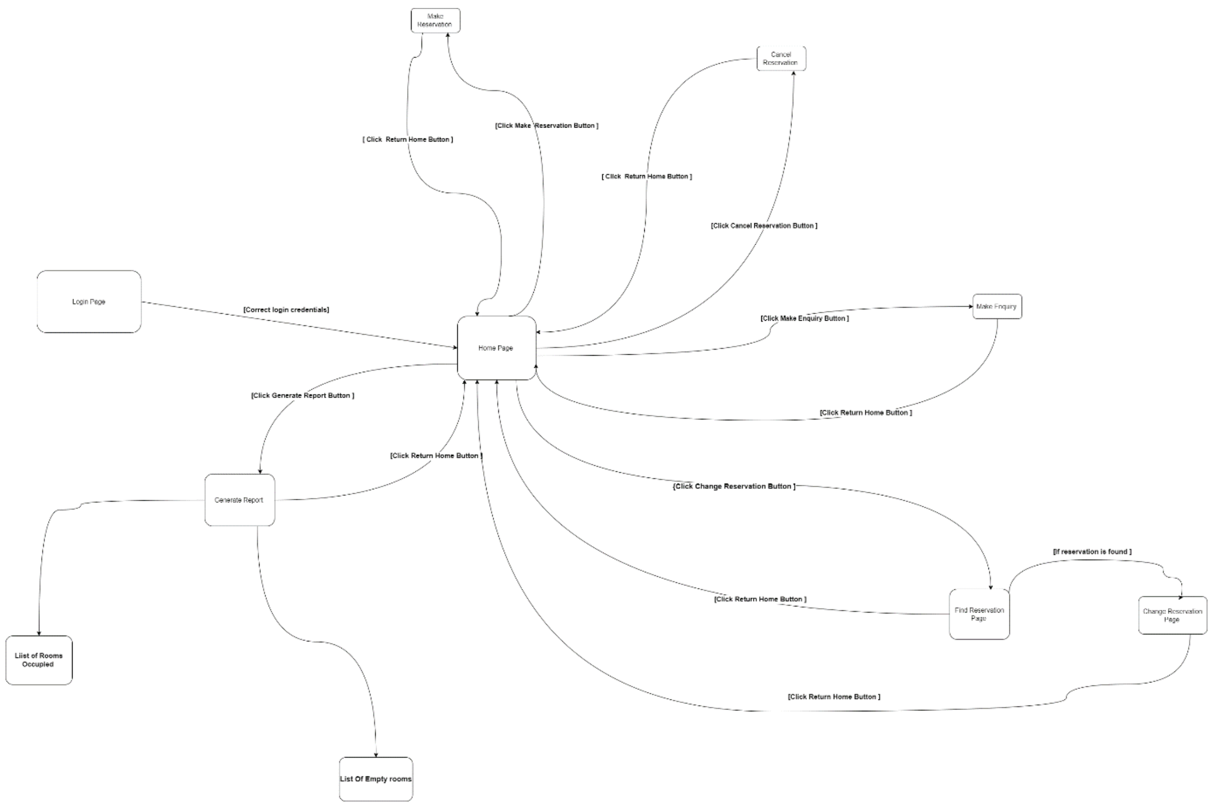
2. USER INTERFACE & DIALOGUE DESIGN

2.1. INTERFACE FLOW DIAGRAMS

- This interface flow diagram represents the key user interactions and navigational paths within the PhumlaniKamnadi Reservation System. Here's a brief breakdown:
- **Login Page**:

- Users first input credentials on the login page. Upon correct login, they are directed to the **Home Page**.

- **Home Page**:
  - The central hub from which users can access different functionalities like making a reservation, cancelling a reservation, making inquiries, generating reports, or viewing room lists.
- **Make Reservation**:
  - From the home page, users can click on the "Make Reservation" button, proceeding to the reservation page to finalize the booking process.
  - There is a path to return to the home page after the reservation is completed.
- **Cancel Reservation**:
  - Users can cancel existing reservations by clicking the "Cancel Reservation" button, which directs them to a dedicated page for cancellations.
- **Make Enquiry**:
  - Users can make inquiries about available rooms or services by clicking the "Make Inquiry" button, which leads to a dedicated inquiry page.
- **Change Reservation**:
  - There is a flow for changing existing reservations. After clicking "Change Reservation," users are directed to a page where they can find an existing reservation, modify it, and confirm the changes.
  - If a reservation is found, the user is directed to a page for editing the reservation details.
- **Generate Report**:
  - Receptionists or staff can generate room-related reports by clicking the "Generate Report" button. This leads to pages displaying either **List of Rooms Occupied** or **List of Empty Rooms**, which are useful for managing room allocation.
- **Return to Home**:
  - For all operations, there is always a "Return Home" button that allows users to navigate back to the **Home Page** easily after completing their tasks.


Make Reservation
Cancel Reservation
[Click Return Home Button]
[Click Make Reservation Button]
[Click Return Home Button]
[Click Cancel Reservation Button]
Login Page
[Correct login credentials]
Home Page
[Click Make Enquiry Button]
Make Enquiry
[Click Generate Report Button]
[Click Return Home Button]
[Click Return Home Button]
Generate Report
[Click Change Reservation Button]
[If reservation is found]
[Click Return Home Button]
Find Reservation Page
Change Reservation Page
Liist of Rooms Occupied
[Click Return Home Button]
List Of Empty rooms

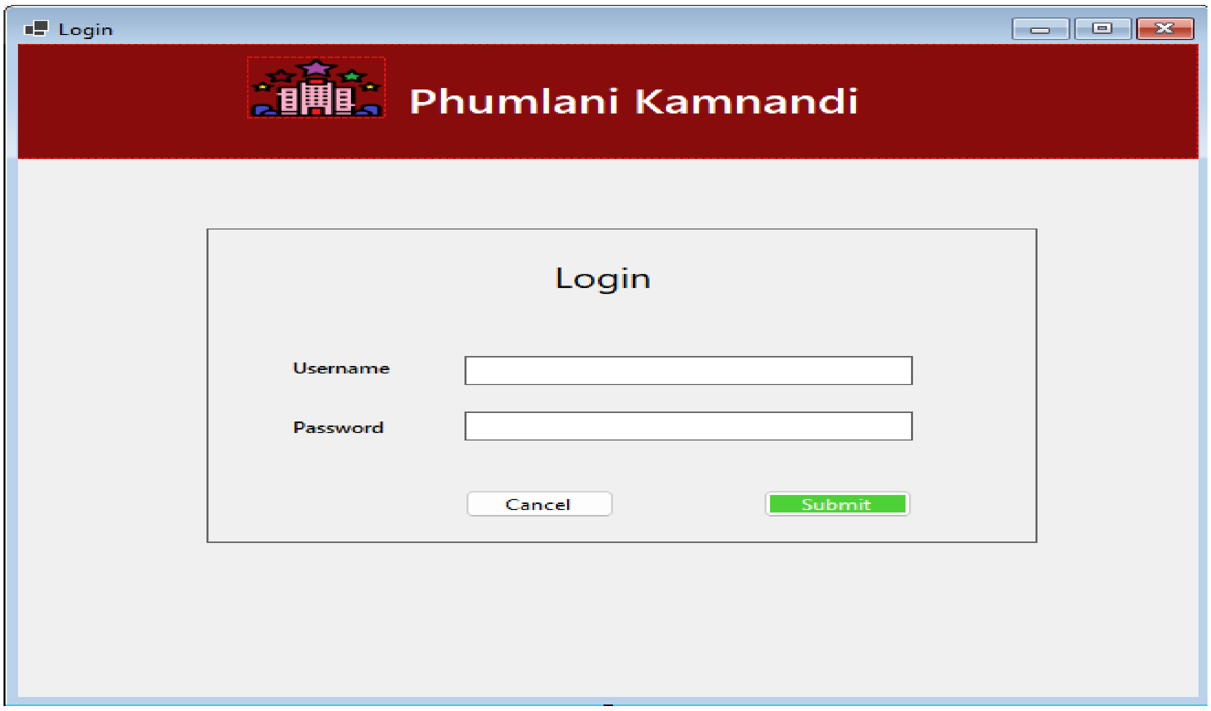
Interface flow diagram

2.2. SCREEN STANDARDS

- **User Interface and Data Entry Process Specification**
    - This section outlines the user interface and data entry processes for the system's login and booking functionalities. The aim is to deliver a clear and structured layout of the input screens and graphical user interfaces (GUIs), along with defining the data elements, associated editing criteria, and data validation controls. Additionally, we will address access restrictions, form copies, and transaction details.
- **User Login Process**
    - The process begins when the actor (user) enters their **username** and **password** into the respective fields. Afterward, the user clicks the **Login** button to authenticate their credentials. If the password is verified as correct, the user will be redirected to the system's **landing page**.
- **Fields:** Username, Password
- **Button:** Login
- **Action:** Authentication and redirection to landing page
    - **Home Page Overview**
    - Upon successful login, the user is directed to the system's main **Landing Page**, as shown in the image below. This page serves as the central hub, providing access to various functionalities including reservation management, inquiries, and reports.
    - **Features on the Home Page:**
- **Make Reservation**: Directs the user to the booking system to create new reservations.
- **Make Enquiry**: Allows the receptionist to inquire about existing bookings.
- **Cancel Reservation**: Enables users to cancel an existing reservation.
- **Change Reservation**: Provides an option to modify the details of an existing reservation.
- **Generate Report**: Allows users to generate a report, possibly containing information about bookings, inquiries, or cancellations.
    - At the middle of the page, a section titled **Expected Check-in** is reserved for displaying the list of guests expected to arrive on a given day, providing the user with a report of upcoming bookings.
- ***Reservation Booking Interface***

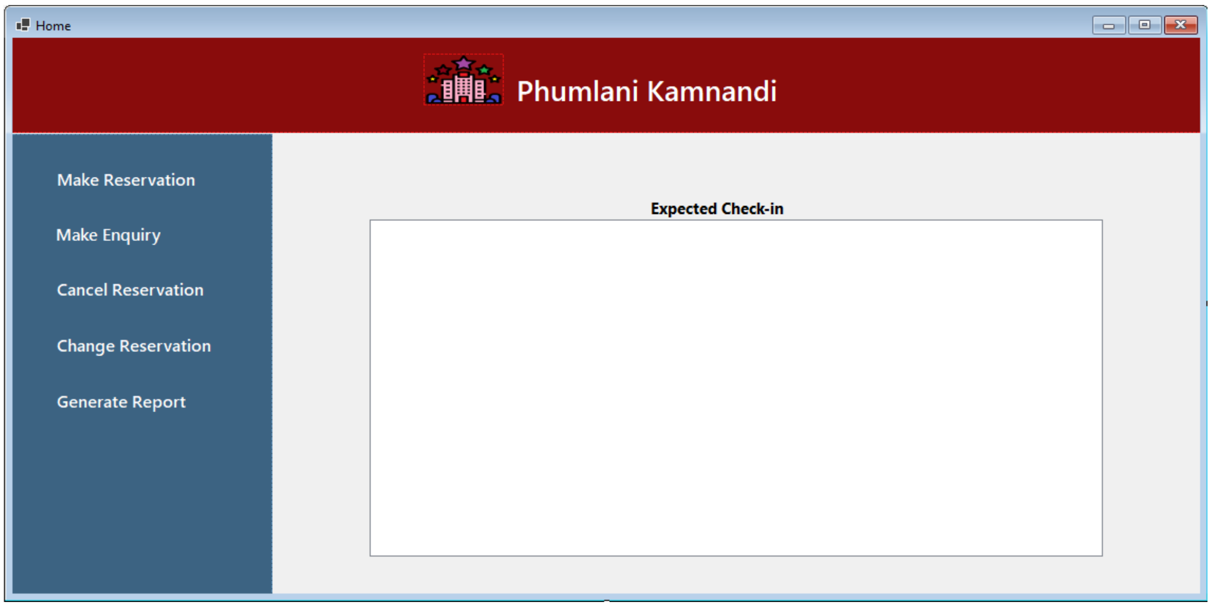
  - When users opt to create a booking, they are directed to the **Reservation Booking** interface, as illustrated below. This screen is designed for entering both guest and reservation details, enabling a seamless booking process.
  - **Personal Details:**
  - The user is prompted to fill in the guest's personal information:
- **First Name** and **Last Name**
- **ID Number** (used for identification purposes)
- **City**, **State**, **Zip Code**
- **Phone Number** and **Email Address**
- **Company Name** (if applicable)
  - Additionally, the user must specify whether the guest is a regular **Guest** or an **Agent**. For payment details, the system requires input for:
- **Credit Card Number**
- **Expiration Date** and **Security Code**
  - **Reservation Details:**
- **Room Preference**: Allows users to select the type of room desired.
- **Check-in and Check-out Dates**: Users must provide the expected check-in and check-out dates.
- **Number of Adults and Children**: These fields capture the number of people associated with the booking.
  - At the bottom right, the **Address 1** and **Address 2** fields provide additional space for the guest's contact information, if necessary.
- **Change reservation**
- **Add/Remove Room:** There are radio buttons at the top allowing the user to either add or remove a room from their reservation.
- **Room details:**
- A large text box is likely meant for input, but its purpose is not clear from the image.
- The "Number of Rooms" dropdown menu allows the user to select how many rooms they wish to add or remove.
- There's a "Room Type" dropdown menu labelled **Select**, which seems to be required but hasn't been filled out in the screenshot.
- A button labelled "Add Room" is provided to finalize the room addition.
- ***Dates:***
- "Check-in Date" and "Check-out Date" fields are displayed with dropdown menus to pick specific dates. In the current view, both dates are set to **Wednesday, 02 October 2024**.
- **User Information (right panel):**
- •A placeholder avatar at the top, labelled with generic placeholders like "[Name] [Surname]," "[Sex]," "[Age]," "[ID]," and a unique reference ID: **refID: 100256**.
  - This screen likely allows users to make changes to their booking, such as adding or removing rooms, updating dates, and seeing personal information about the reservation holder.
- **Cancel Reservation**
  - The interface consists of several key components arranged vertically in a minimalist style, each playing a specific role in the cancellation process:
- **Enter Reservation ID:** At the top of the interface, there is a field labelled "Enter Reservation ID." This input field is where the user is expected to enter a unique identifier, such as a booking or reservation number, which corresponds to their reservation. The presence of this field highlights the importance of having a unique reservation ID, which serves as a critical piece of information for accessing the reservation details.
  - The use of this ID ensures a secure way to search for and identify reservations, preventing unauthorized access. The system's reliance on this identifier indicates that the system is likely part of a structured database that efficiently stores and organizes reservations using unique keys.
- **Search Button:** Below the Reservation ID field, there is a large green **Search** button. This button initiates the process of searching the database for the corresponding reservation details based on the entered ID. The green colour, often associated with positivity and action, encourages the user to proceed with the search function. This step likely pulls up relevant details associated with the booking, such as the customer's name, room type, booking dates, and payment information.
  - The search function is a crucial step in the reservation management process, allowing users to verify the information before making any changes or cancellations. It also helps prevent accidental cancellations, as the system ensures that the reservation details are correct before proceeding with further actions.

- **Results Area:** The large blank box situated below the search button appears to be the space where the results of the search will be displayed. Once the reservation ID is entered and the search button is clicked, this section will likely populate with reservation details such as check-in and check-out dates, room information, and customer personal data.
    - This results area serves as a confirmation step, allowing the user to review all relevant information before taking any irreversible actions, like cancellation. In user interface design, it is important to offer this confirmation to minimize errors and ensure the user has full control over their decision-making process.
- **Home and Cancel Buttons*:*** At the bottom of the interface are two buttons:
- The **Home** button, represented by a small house icon, likely takes the user back to the main dashboard or home screen of the application. This button serves as an escape route if the user decides not to proceed with the cancellation, offering a way to return to a neutral starting point without performing any actions.
- The **Cancel Reservation** button is positioned to the right and is coloured in red, a colour commonly associated with warnings or actions that have significant consequences. Pressing this button would finalize the cancellation of the reservation, removing it from the system or marking it as cancelled. The red colour serves as a visual cue for the user to be cautious, emphasizing the serious and irreversible nature of the action*.*



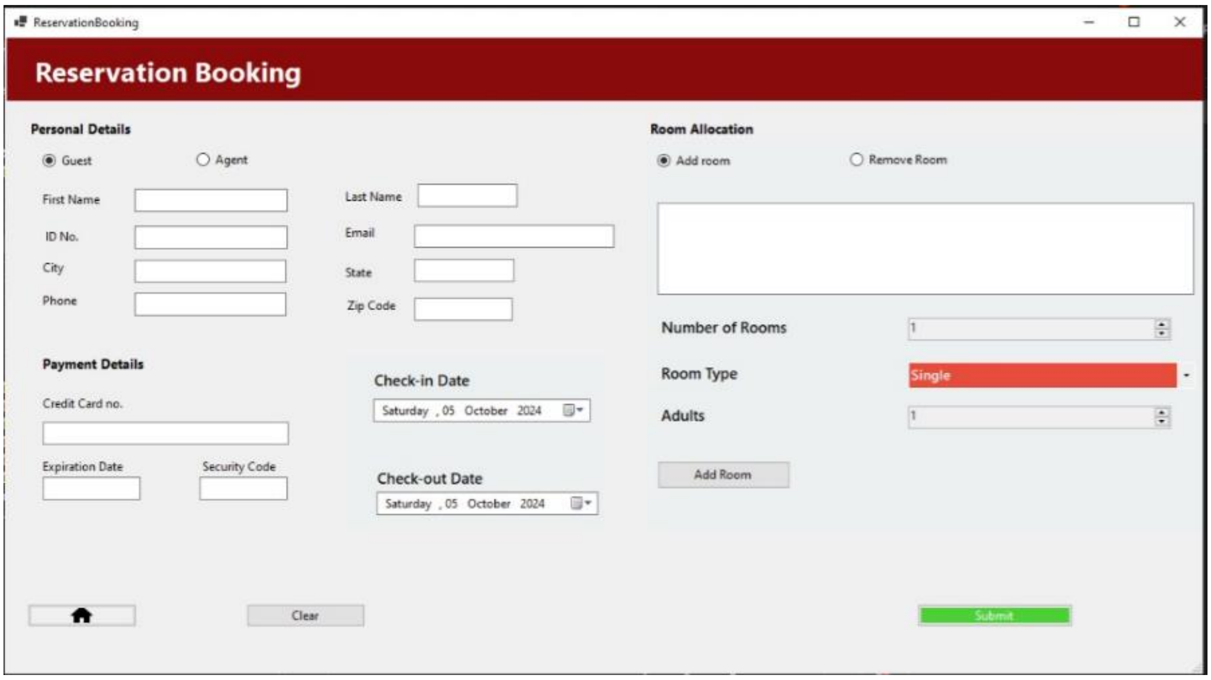
Login page

- The receptionist enters their username and password to log in and clicks the submit button, then the program returns a message box to show whether the login is successful or not . While the cancel button exits the app**.**



Home Page

- Then according to the Guests or Agents preference the options appear: Make Reservation, Make Enquiry, Cancel Reservation and Generating a report. In the middle there is information about the check ins that are expected in that week.



[✓ Guest] Agent

[✓ Add room] Remove Room

Check-in Date: Saturday , 05 October 2024

Check-out Date: Saturday , 05 October 2024

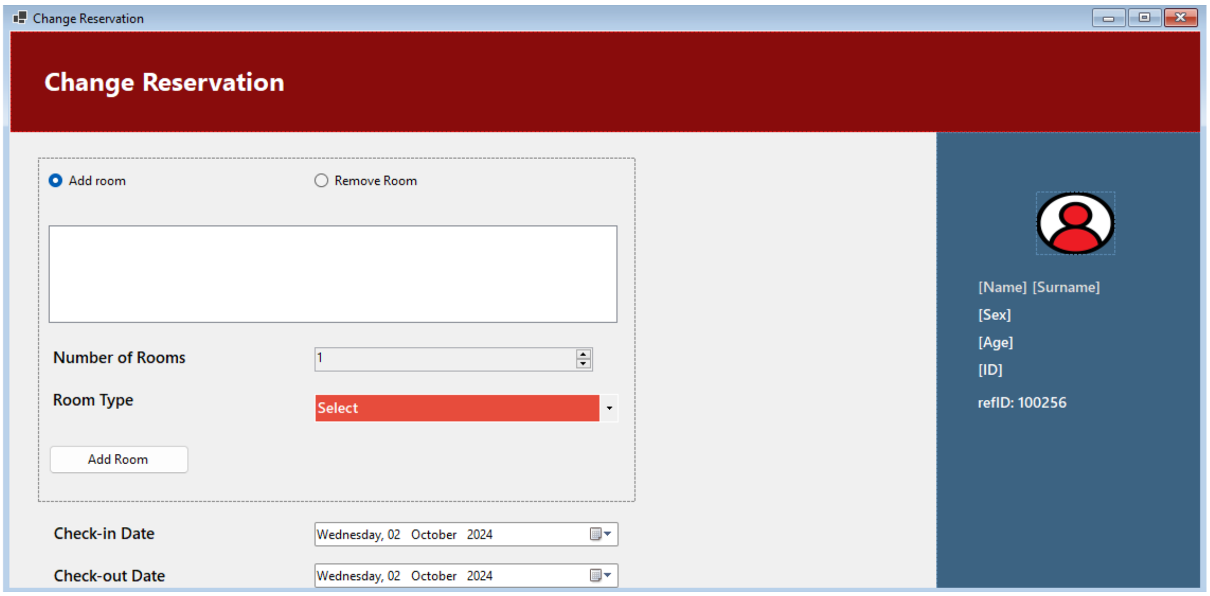
Number of Rooms: 1

Room Type: Single

Adults: 1

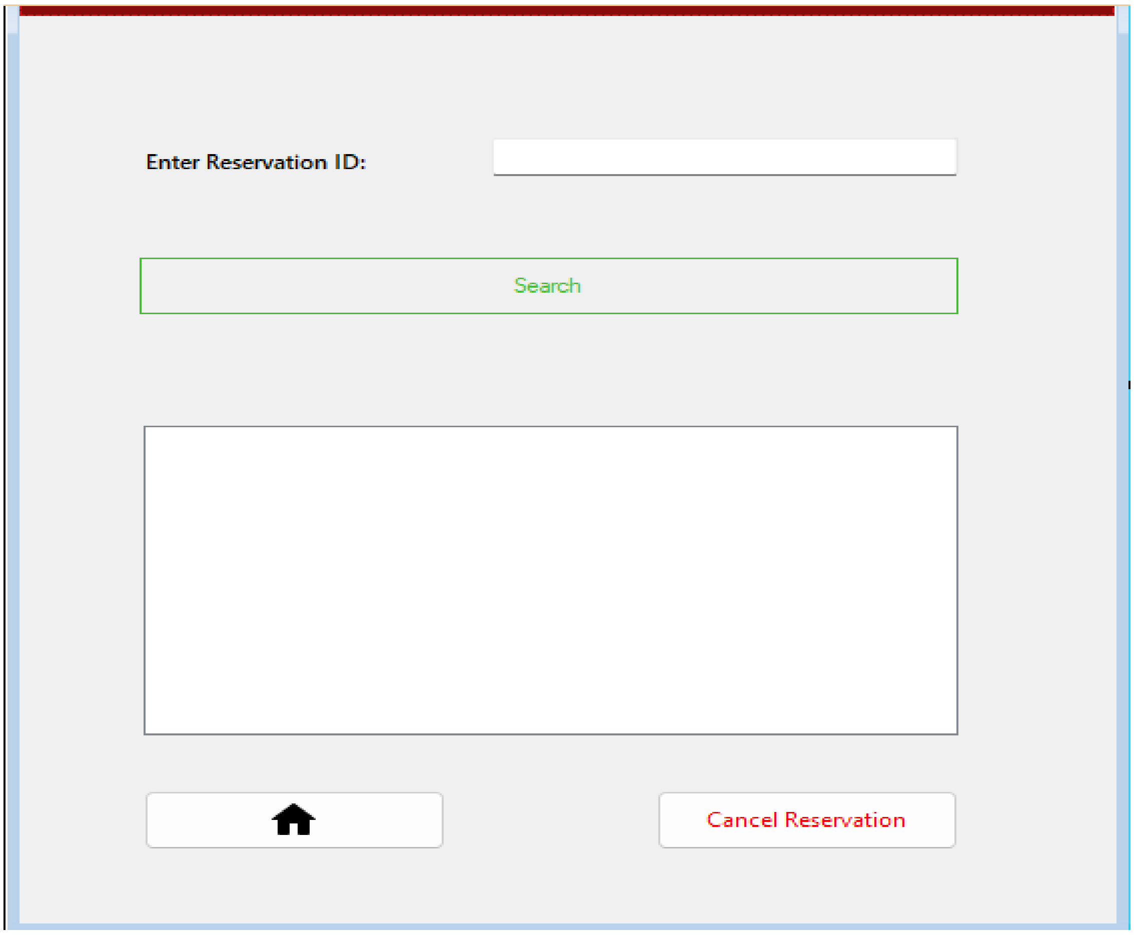
Reservation Booking Page.

- If they request to make a reservation, they are taken to the reservation booking page.



Change Reservation Page.

- Or if they choose to change a reservation the receptionist clicks the change reservation button then is taken to the change reservation page, and they enter the guests or agents' details.



Cancel Reservation Page.

Then if the guest or agent requests to cancel a reservation the receptionist is taken to the cancel reservation page.

3. DESIGN SEQUENCE DIAGRAMS

3.1. MAKE A RESERVATION BOOKING DETAILED DESIGN SEQUENCE DIAGRAM

Participants:

- **Receptionist**: The person handling the phone reservation request.
- **homePage**: Represents the user interface from where the receptionist initiates the booking process.
- **BookingHandler**: The controller managing the booking logic.
- **guest: Guest**: The guest object holding guest details.
- **r: Reservation**: The reservation object created during the booking.
- **room: Room**: The room object representing the room being booked.
- **DBHandler**: The controller managing communication with the database.
- **BookingDB**: The data access object that interacts with the database.

Key Steps in the Process:

Entering Guest ID:

- The receptionist begins by entering the guest's ID into the system.
- The system (homePage) invokes getGuestDetails() to retrieve the guest's information.
- The BookingHandler controller checks if the guest exists in the database by calling VerifyGuest() on BookingDB.

Guest Verification (ALT Block):

- The diagram handles two cases:
  Customer does not exist:
- If the guest is not found in the database, a new customer must be created.
- The receptionist enters the guest's details (name, address, credit number), and the system creates a new guest by invoking CreateNewCustomer().
- The new guest is written to the database (**WriteToDB()),** and a confirmation message is returned.

Existing Guest:

- If the guest already exists in the database, a confirmation message is sent back to the receptionist.

Entering Booking Details:

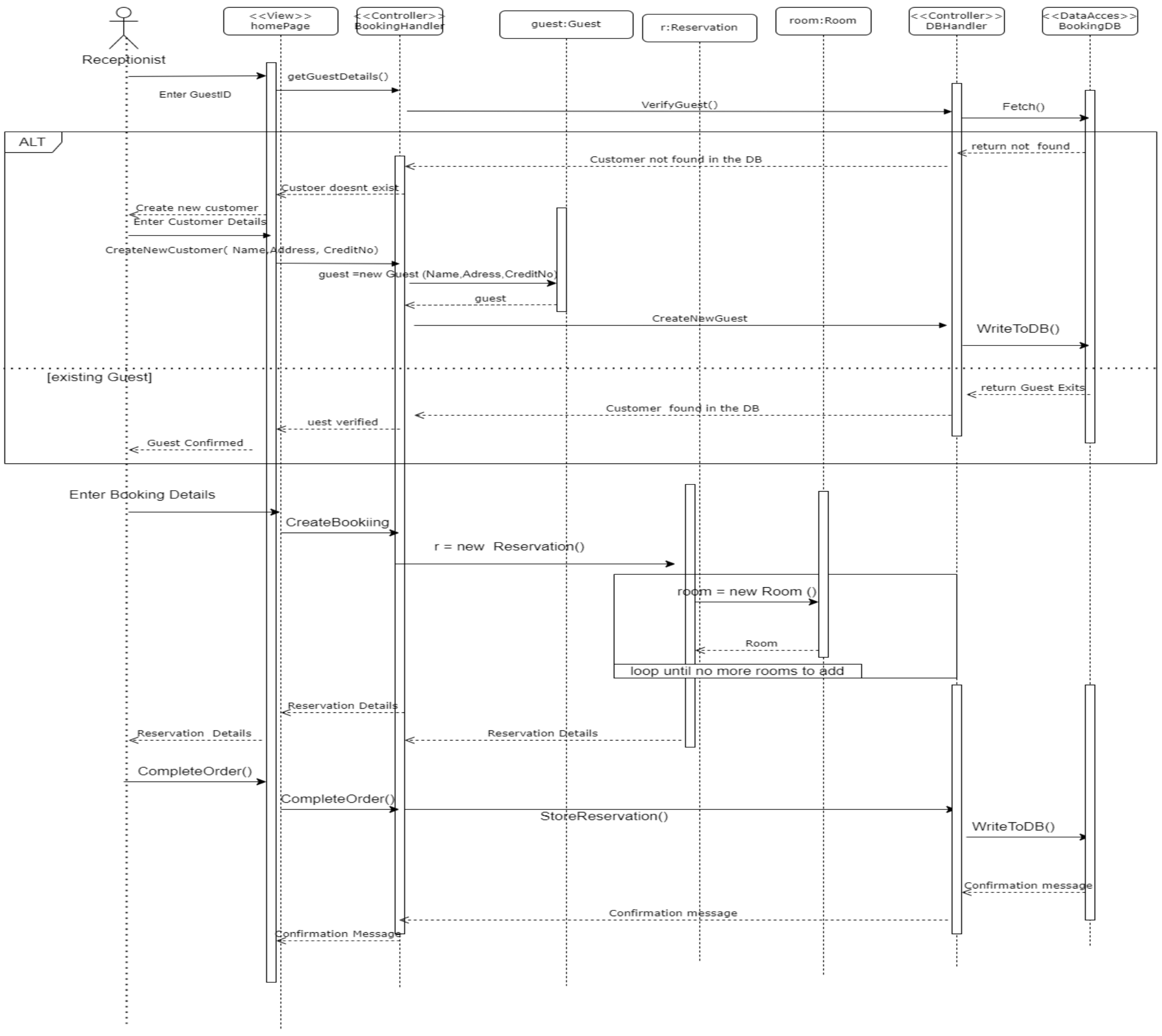
- Once the guest is verified, the receptionist enters the booking details.
- The system creates a new reservation (**CreateBooking()),** which involves the instantiation of the reservation object (**r = new Reservation()).**

Room Assignment:

- The system assigns a room by creating a new room object (room = new Room()).
- A loop checks for available rooms, and the system adds rooms until no more are available.

Completion of the Booking:

- After the room is assigned and all details are verified, the reservation is stored in the database (**StoreReservation()).**
- The database handler writes the reservation to the database (**WriteToDB()),** and a confirmation message is sent back to the receptionist



Make Reservation Detailed Sequence Diagram

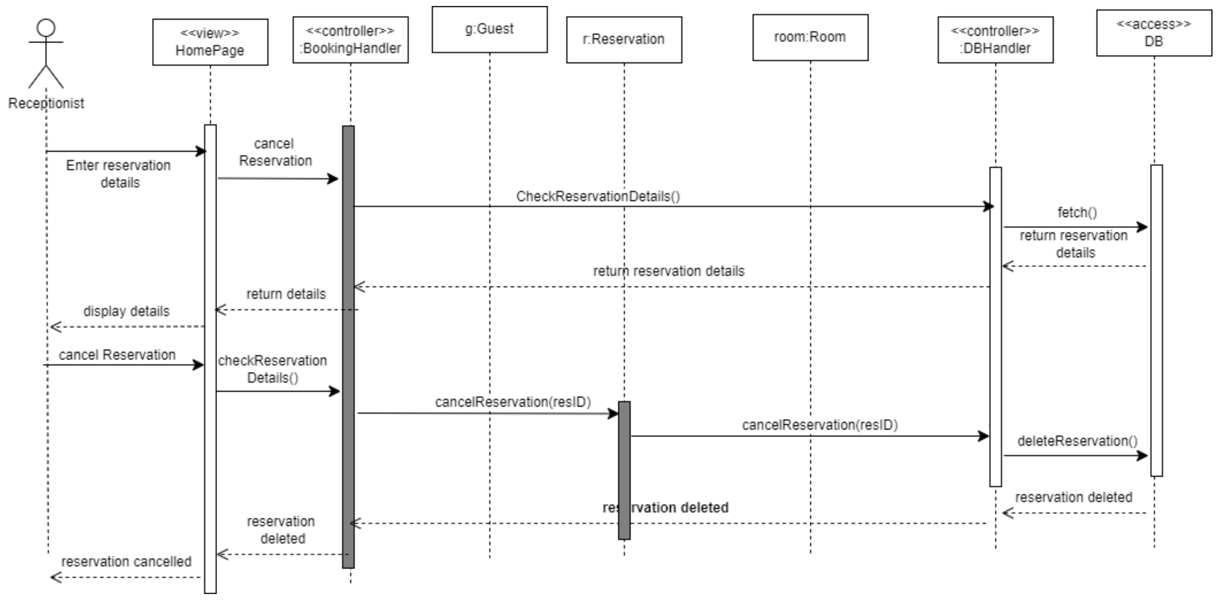
3.2. CANCEL RESERVATION DETAILED SEQUENCE DIAGRAM

Participants Involved:

- **Receptionist**: Initiates the reservation cancellation process.
- **HomePage**: The user interface from where the receptionist works.
- **BookingHandler**: The controller responsible for managing the booking and cancellation logic.
- **Guest (g)**: Represents the guest details stored in the system.
- **Reservation (r)**: The reservation object that is being cancelled.
- **Room (room)**: The room object associated with the reservation.
- **DBHandler**: The controller managing communication between the application and the database.
- **Database (DB)**: The actual database where all reservation data is stored.

Key Steps in the Cancel Reservation Process:

- **Enter Reservation Details**:
    - The receptionist enters the reservation details (like reservation ID or guest details) on the HomePage interface.
    - This step triggers a request to cancel the reservation in the system.
- **Cancel Reservation Request**:
    - The system invokes the **checkReservationDetails()** method through the BookingHandler, which checks for the reservation details linked to the entered **ID**.
    - The system requests the reservation information from the Reservation and DBHandler controllers.
- **Fetching Reservation Details**:
    - The DBHandler interacts with the DB (database) and calls the fetch() function to retrieve the reservation information.
    - The fetched reservation details are returned from the database to the DBHandler, and subsequently to the BookingHandler.
    - The reservation details are then returned and displayed to the receptionist on the HomePage interface.
- **Confirm Cancellation**:
    - After verifying the reservation, the receptionist proceeds to confirm the cancellation by invoking **cancelReservation(resID).**
- **Cancel Reservation Action**:
    - The BookingHandler calls the **cancelReservation()** method in the Reservation object, which communicates with the DBHandler to proceed with the deletion.
    - The DBHandler interacts with the database to delete the reservation using the **deleteReservation(resID)** function.
- **Reservation Deleted**:
    - The reservation is successfully removed from the database, and a confirmation message is returned from the DB to the DBHandler, then to the BookingHandler, and finally back to the HomePage.
    - The receptionist receives a confirmation that the reservation has been successfully cancelled.



Cancel Reservation Detailed Sequence Diagram

4. DESIGN CLASS DIAGRAMS

Class diagram showcases the relationships and interactions between different entities within the hotel reservation system. Below is a brief explanation of the key classes:

1.Person (Super Class)

- **Attributes**:
  - PersonID, FirstName, LastName, Contact Details, EmailAddress.
- **Methods**:
  - GetFullName(): Returns the full name of the person.
  - UpdateContactDetails(): Updates the contact information of a person.
  - UpdateAddress(): Updates the address of a person.
- **Purpose**: Acts as a base class for entities like Guest, Booking Agent, and Staff, providing shared attributes and methods.

2. Guest (Inherits from Person)

- **Attributes**:
  - GuestID, CreditCardNo, ExpirationDate, CVV.
- **Methods**:
  - UpdateEmail(), UpdatePaymentDetails(), GetGuestDetails().
- **Purpose**: Represents a customer booking the rooms, with attributes specific to guests, such as payment details.

3. Booking Agent (Inherits from Person)

- **Attributes:**
    - AgentID, CompanyName, NoOfGuests.
- **Methods:**
    - GetBookingAgentDetails(), UpdateCompanyName().
- **Purpose:** Represents an agent booking reservations for guests, typically on behalf of a company.

4. **Reservations**

- **Attributes**:
    - ReservationID, GuestID, AgentID, StartDate, EndDate, CheckInDate, CheckOutDate, RoomNo.
- **Methods**:
    - GetStayDuration(), CalculateTotalStayCost(), GetReservationDetails(), UpdateDates(), AddRoom(), RemoveRoom().
- **Purpose**: Manages all details related to a reservation, such as check-in/out dates, room assignment, and the cost of the stay.

5. Rooms

- **Attributes:**
    - RoomID, RoomType, OccupancyStatus, BedType, RatePerNight.
- **Methods:**
    - AssignRoom(), GetRoomDetails().
- **Purpose**: Manages room details and their availability status. Each reservation is linked to one or more rooms.

6. Accounts

- **Attributes**:
    - AccountID, GuestID, AgentID, TotalAmountDue, Charges, PaymentsReserved, AccountTotal.
- **Purpose**: Tracks the financial details of a guest or agent, including total amount due, charges, and payments.

7. **RoomController**

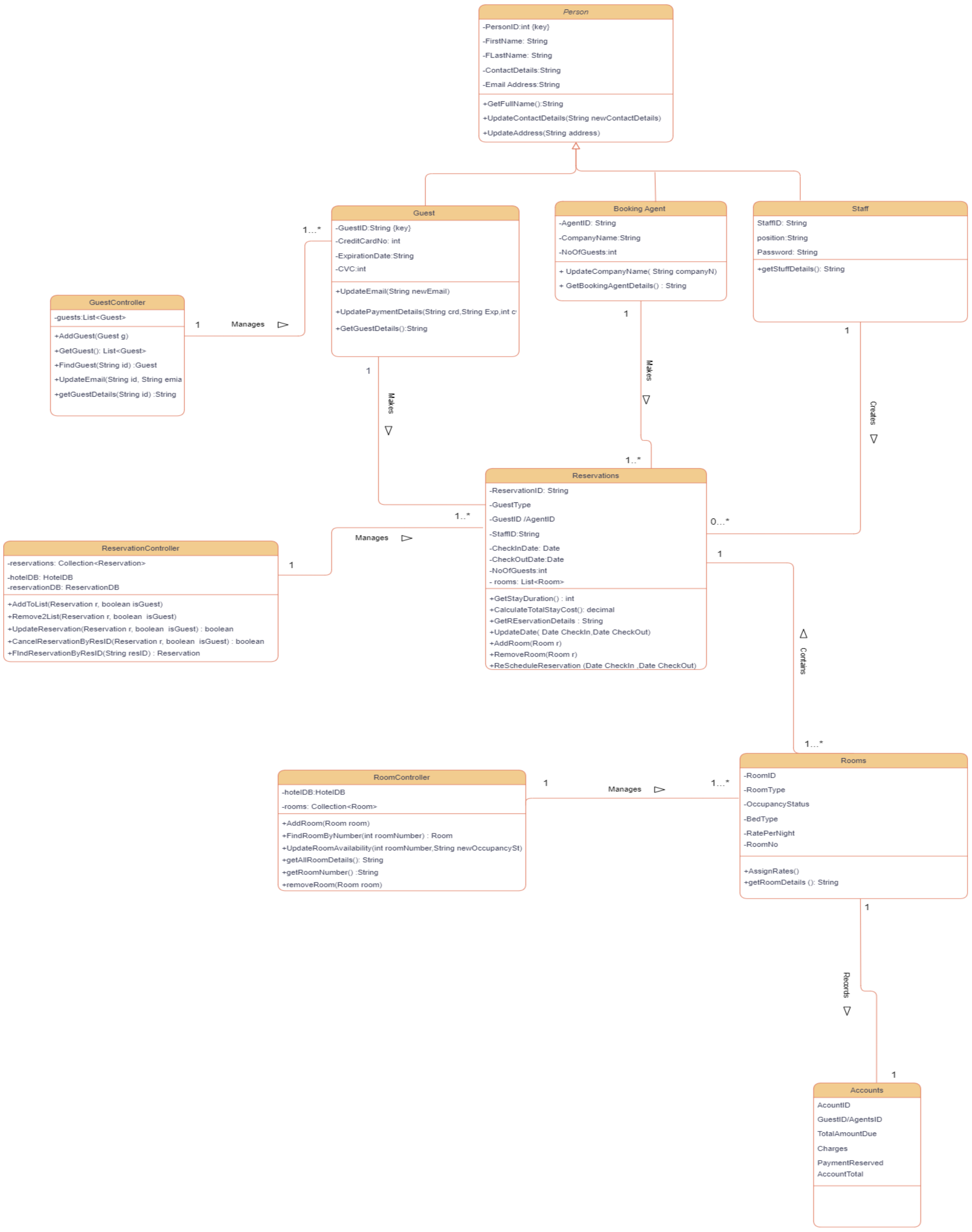
- **Attributes**:
    - rooms: A collection of rooms.
- **Methods**:
    - AddRoom(), FindRoom(), UpdateRoomDetails(), RemoveRoom().
- **Purpose**: Manages operations related to room inventory, including adding, removing, and updating room details.

8. ReservationController

- **Attributes**:
  - reservations: A collection of reservations.
- **Methods**:
  - AddReservation(), RemoveReservation(), UpdateReservation(), FindReservation().
- **Purpose**: Handles the management of reservations, including adding, updating, and deleting reservations from the system.

9. GuestController

- **Attributes**:
  - guests: A list of guests.
- **Methods**:
  - AddGuest(), FindGuest(), GetGuestDetails().
- **Purpose**: Manages the guest list, including adding new guests and retrieving their details.


Person
-PersonID:int {key}
-FirstName: String
-FLastName: String
-ContactDetails:String
-Email Address:String
+GetFullName():String
+UpdateContactDetails(String newContactDetails)
+UpdateAddress(String address)
Guest
-GuestID:String {key}
-CreditCardNo: int
-ExpirationDate:String
-CVC:int
+UpdateEmail(String newEmail)
+UpdatePaymentDetails(String crd,String Exp,int c
+GetGuestDetails():String
Booking Agent
-AgentID: String
-CompanyName:String
-NoOfGuests:int
+ UpdateCompanyName(String companyN)
+ GetBookingAgentDetails() : String
Staff
StaffID: String
position:String
Password: String
+getStuffDetails(): String
GuestController
-guests:List<Guest>
+AddGuest(Guest g)
+GetGuest(): List<Guest>
+FindGuest(String id) :Guest
+UpdateEmail(String id, String emia
+getGuestDetails(String id) :String
1…*
1
Manages
1
Makes
1
Makes
1..*
1
Creates
Reservations
-ReservationID: String
-GuestType
-GuestID /AgentID
-StaffID:String
-CheckInDate: Date
-CheckOutDate:Date
-NoOfGuests:int
- rooms: List<Room>
+GetStayDuration() : int
+CalculateTotalStayCost(): decimal
+GetREservationDetails : String
+UpdateDate(Date CheckIn,Date CheckOut)
+AddRoom(Room r)
+RemoveRoom(Room r)
+ReScheduleReservation (Date CheckIn ,Date CheckOut)
1..*
0…*
1
ReservationController
-reservations: Collection<Reservation>
-hotelDB: HotelDB
-reservationDB: ReservationDB
+AddToList(Reservation r, boolean isGuest)
+Remove2List(Reservation r, boolean isGuest)
+UpdateReservation(Reservation r, boolean isGuest) : boolean
+CancelReservationByResID(Reservation r, boolean isGuest) : boolean
+FIndReservationByResID(String resID) : Reservation
1
Manages
Contains
1…*
RoomController
-hotelDB:HotelDB
-rooms: Collection<Room>
+AddRoom(Room room)
+FindRoomByNumber(int roomNumber) : Room
+UpdateRoomAvailability(int roomNumber,String newOccupancySt)
+getAllRoomDetails(): String
+getRoomNumber() :String
+removeRoom(Room room)
1
Manages
1…*
Rooms
-RoomID
-RoomType
-OccupancyStatus
-BedType
-RatePerNight
-RoomNo
+AssignRates()
+getRoomDetails (): String
1
Records
1
Accounts
AcountID
GuestID/AgentsID
TotalAmountDue
Charges
PaymentReserved
AccountTotal

Detailed Class Diagram

5. ENTITY RELATIONSHIP DIAGRAM

The entity diagram includes tables contained within the **Reservations** section. We aimed to structure all our tables in **Third Normal Form (3NF)**. Below is a description of each table and its attributes:

1. Guest

- **Attributes**:
  - GuestID (Primary Key), Name, Email, Address, Phone.
- **Relationships**:
  - A guest makes multiple reservations (linked through GuestReservation).

2. Agent

- **Attributes**:
  - AgentID (Primary Key), CompanyName, AgentName, Email, Phone.
- **Relationships**:
  - Agents can make multiple reservations on behalf of guests (linked through AgentReservation).

3. Room

- **Attributes**:
  - RoomID (Primary Key), RatePerNight, RoomType.
- **Relationships**:
  - Rooms are assigned to reservations via the GuestReservationRooms or AgentReservationRooms tables.

4.Staff

- **Attributes**:
  - StaffID (Primary Key), Name, Email, Password, Position.
- **Relationships**:
  - Staff members manage guest and agent reservations (linked via GuestReservation and AgentReservation).

5. GuestReservation

- **Attributes**:
  - ReservationID (Primary Key), GuestID (Foreign Key), CheckInDate, CheckOutDate, TotalAmount, NoOfGuests, StaffID (Foreign Key).
- **Relationships**:
  - Links guests to their reservations and connects to GuestReservationRooms for assigning rooms.

6. AgentReservation

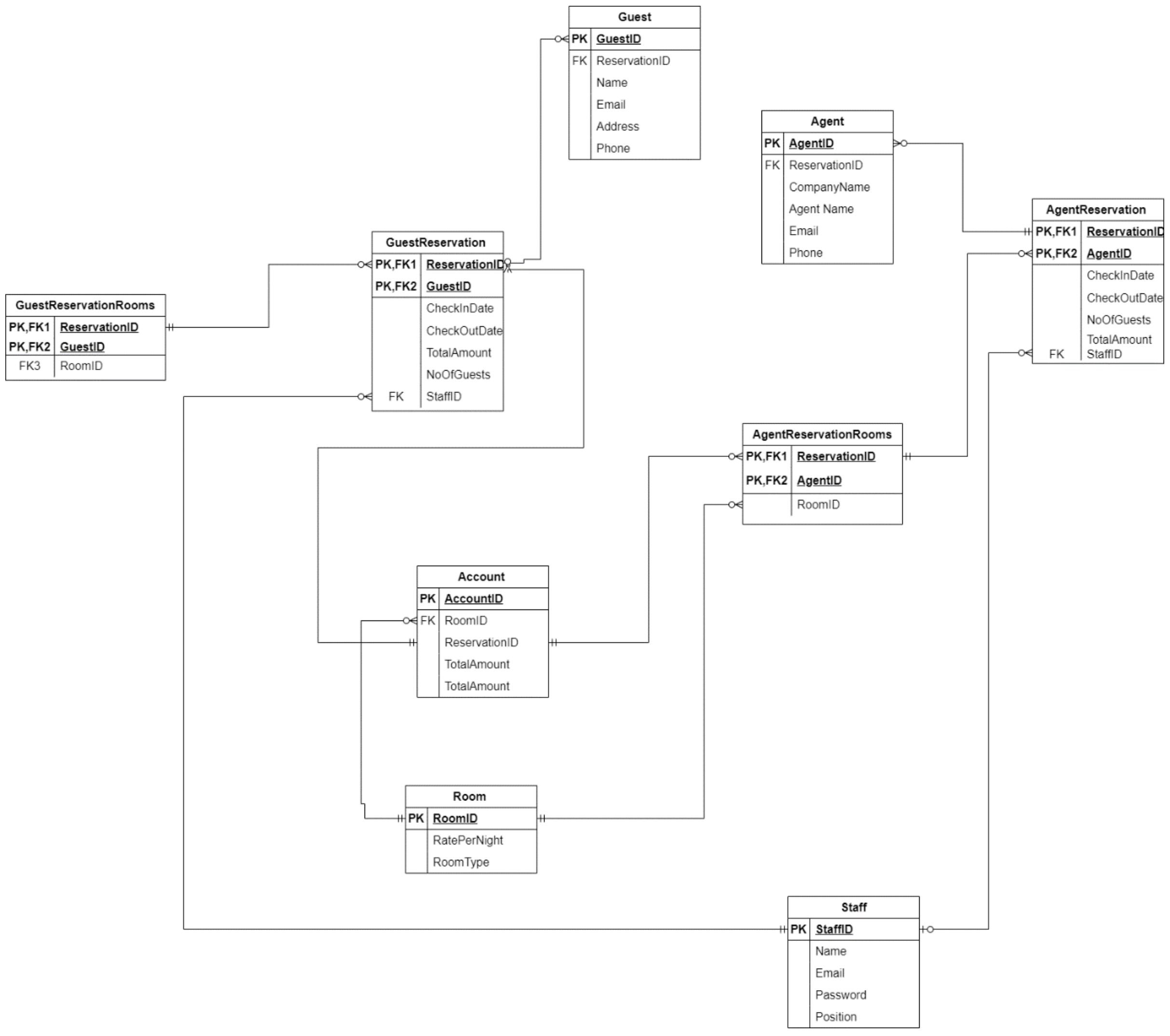
- **Attributes**:
    - ReservationID (Primary Key), AgentID (Foreign Key), CheckInDate, CheckOutDate, NoOfGuests, TotalAmount, StaffID (Foreign Key).
- **Relationships**:
    - Similar to GuestReservation, but for reservations made by agents, linked to AgentReservationRooms for room assignments.

7. GuestReservationRooms & AgentReservationRooms

- **Attributes**:
    - ReservationID (Foreign Key), GuestID or AgentID (Foreign Key), RoomID (Foreign Key).
- **Purpose**: These tables link reservations to specific rooms, showing which room(s) are allocated to each guest or agent reservation.

8. Account

- **Attributes**:
    - AccountID (Primary Key), RoomID (Foreign Key), ReservationID (Foreign Key), TotalAmount.
- **Purpose**: Manages billing details for each reservation, linking reservations to rooms and tracking the total cost of the stay



Entity Diagram

6. REPORT DESIGN

6.1. OCCUPANCY LEVELS REPORT

6.1.1. Detailed Output Requirements

1. **Output Type & ID**

- **Report Type**: Room Occupancy Report
- **Report ID**: Occupancy_Levels001
- **Purpose:** To display the list of rooms currently occupied in the hotel, along with the corresponding guests and reservations.

2. **Report Objectives**

The primary goal is to show which guests are occupying specific rooms, alongside their reservation details, allowing the hotel staff to monitor room occupancy in real time.

3. **Audience**

- **Receptionists:** To manage occupied rooms and assist guests currently staying in the hotel.
- **Reservation Staff**: To ensure room bookings match actual room occupancy.
- **Stakeholders:** To assess room utilization and make strategic decisions regarding room management.

4. Content

The report will contain the following columns for each occupied room:

- Room ID: Identifies the specific room being occupied.
- Guest ID: A unique identifier for the guest staying in the room.
- Reservation ID: The reservation associated with the guest and room.

5. Layout

- The report will be displayed in columns without graphs.
- The main columns: Room ID, Guest ID, Reservation ID.

6. Selection

- The data is selected based on current room occupancy, showing only rooms that are currently occupied.
- The report can be filtered to show active reservations only, excluding rooms not yet occupied or already vacated.

7. Sequence

- The report will be sorted in ascending order by Room ID (from smallest to largest), which also reflects the room type tier, allowing the hotel to see which room types are occupied.
- This report allows comparison of room occupancy by date, helping the hotel understand which room types are preferred on specific days or periods.
- It can also be used to track occupancy trends, allowing management to optimize room pricing and availability based on demand.

8. Grouping/Summarization

- The report provides a summary of all occupied rooms, returning the Room ID, Guest ID, and Reservation ID.
- Rooms can be grouped by Room Type, allowing the hotel to identify the most and least frequently occupied room types.

9. Media

- The report can be electronically generated and viewed through the hotel's internal system.
- It can also be printed for use by the staff, particularly for daily occupancy meetings or audits.

10. Frequency, Timing, Delivery

- The report can be generated daily or on demand by hotel staff to check real-time room statuses.

- It can also be scheduled to run at specific times, such as at the start or end of each shift, to provide updated information on room occupancy.

12. Distribution

- **Electronic Distribution**: The report can be accessed via the hotel's reservation management system by the staff responsible for managing rooms and reservations.
- **Email Notifications:** Stakeholders or managers can receive automated email reports for daily occupancy updates.

13. Privacy, Security & Integrity Requirements

- **Restricted Access**: The report will only be accessible to receptionists, reservation staff, and authorized stakeholders.
- **Data Anonymization**: Guest details are protected by using unique Guest IDs instead of personal data, ensuring privacy compliance.
- **Data Integrity**: The report is generated from the live reservation system to provide up-to-date and accurate occupancy information***.***

6.1.2. Room Occupancy Report Layout

RoomID	GuestID	Reservation
100	Gus18	Res23
101	Gus27	Res67
210	Gus45	Res89
312	Gus67	Res54
253	Gus82	Res34
356	Gus64	Res12
456	Gus78	Res56

6.2. PENDING RESERVATIONS REPORT

6.2.1. Detailed Output Requirements

1. **Output Type & ID**

- **Report Type**: Pending Reservations Report

- **Report ID**: Pending_Resv001
- **Purpose:** To display a list of reservations that are upcoming but have not yet been confirmed, assisting the receptionists in following up with guests for confirmation.

2. **Report Objectives**

- The primary goal is to track and manage pending reservations, allowing receptionists to contact guests who haven't confirmed their bookings yet. By freeing up rooms that have not been confirmed, it helps optimize room availability.

3. **Audience**

- **Receptionists**: To follow up with guests and confirm pending reservations.
- **Reservation Staff**: To manage room bookings and ensure accuracy in room availability.
- **Stakeholders:** To understand room booking flow and improve guest management strategies.

4. **Content**

The report will contain the following columns for each pending reservation:

- **Guest Name**: The name of the guest with the pending reservation.
- **Contact Information**: Phone number or email to reach out to the guest.
- **Room Number:** The room that has been reserved but is awaiting confirmation.
- **Reservation Date**: The date when the reservation was made or is scheduled for.
- **Status**: The status of the reservation (**Pending/Confirmed/Cancelled**).

5. **Layout**

- The report will be displayed in columns without graphs. The main columns: **Guest Name**, **Contact Information**, **Room Number**, **Reservation Date, Status**.

6. **Selection**

- The data is selected based on upcoming reservations that have not yet been confirmed. Filters can be applied to show active reservations only or reservations for a specific room or time.

7. **Sequence**

- The report will be sorted in ascending order by Reservation Date, allowing the receptionist to prioritize the most urgent pending reservations. Rooms can also be filtered or sorted by Room Number to show reservations per room type or tier.

8. **Comparison**

- This report can be compared to the total number of available rooms for the selected date, to identify potential overbooking or underutilization. Furthermore, the report can help track which room types are more frequently reserved but often left pending or frequently cancelled , providing insights into guest preferences or room availability issues.

9. **Grouping/Summarization**

- The report can be summarized by Room Type (e.g**., Single, Double, Suite**), showing how many reservations are pending for each room category.
- **Status Summary**: The report can group rooms by Pending, Confirmed, or Cancelled statuses to provide a high-level overview of the hotel's booking status.

10. **Media**

- The report can be electronically generated and viewed through the hotel's internal reservation system on the home page.

11. **Frequency, Timing, Delivery**

- The report can be generated daily or on demand as needed. Receptionists can run the report multiple times during the day to keep track of changes in reservation statuses.

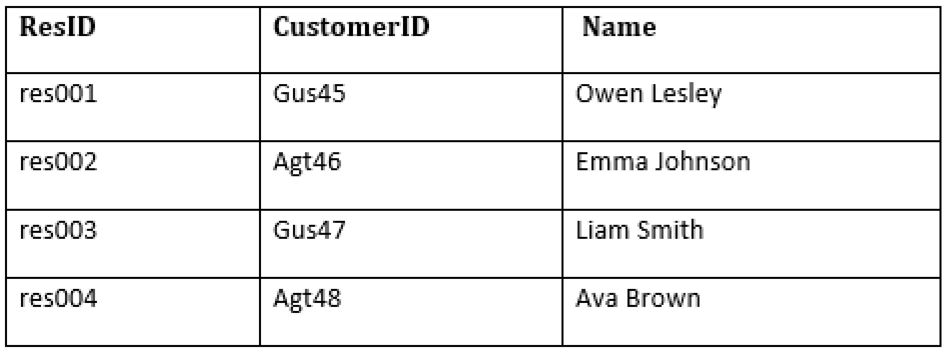
12. **Distribution**

- Electronic Distribution: Via the hotel's reservation system, the report can be accessed by receptionist in charge of managing reservations.

13. **Privacy, Security & Integrity Requirements**

- **Restricted Access**: Only receptionists, reservation staff, and authorized stakeholders will have access to generate and view this report.
- **Data Anonymization**: For privacy, instead of displaying the guest's full details, a unique Guest ID will be assigned to each reservation, ensuring that personal data is protected.
- **Data Integrity**: To maintain the accuracy of the report, it will be generated from the live reservation system, pulling the most up-to-date information on pending bookings from the database.

6.2.2 PENDING RESERVATIONS REPORT



ResID	CustomerID	Name
res001	Gus45	Owen Lesley
res002	Agt46	Emma Johnson
res003	Gus47	Liam Smith
res004	Agt48	Ava Brown

Pending Reservations Report layout

7. INPUT-OUTPUT STANDARDS & CONTROLS

7.1. FORMALISED OUTPUTS:

The formalized system outputs included in the system are designed to meet specific display requirements and minimize redundancy. Key points regarding formalized outputs are:

- Upon completing a booking, the system generates a confirmation letter, which is displayed on the screen. This letter includes the guest's details, room information, stay duration, and booking reference number.

- Guests or travel agents can request to view this confirmation letter on-screen, but no printed, email, or fax options are required. At the time of guest check-out, the system provides a detailed invoice summarizing all charges (room, additional services, and any extras). This invoice is displayed on the screen for the guest to review.
- A list of rooms assigned to arriving guests and a housekeeping priority list are displayed on the management and housekeeping consoles.
- For hotel managers, the system displays real-time financial reports on the screen, showing daily income from room bookings, bar and restaurant services, and other hotel facilities.

7.2. BUILT-IN VALIDATION TO ENSURE REQUIREMENTS ARE MET

Date Validation

- **Check-In and Check-Out Dates**: The system will validate that the check-in date is earlier than the check-out date and that the selected dates are within a valid range. This ensures guests do not accidentally book invalid or impossible date ranges.
- **Room Availability**: The system will check real-time room availability before finalizing a booking to prevent overbooking or assigning unavailable rooms.

Credit Card Validation

- **Credit Card Format and Expiry Check**: When guests provide their credit card information for booking or payment, the system will validate the card number, ensure the card is not expired, and flag any invalid entries before processing payments.
- **Deposit Verification**: If a deposit is required, the system will verify that a valid payment method is attached and that the amount is appropriately calculated and authorized before confirming the booking.

Guest Capacity Validation

- **Room Occupancy Limits**: The system will enforce room capacity rules, such as a maximum of four people per room. If a user tries to exceed this limit during booking, the system will flag an error and prevent overbooking.
- **Child Discounts**: The system will automatically apply the correct discount based on the number of children and their ages, ensuring accurate room charges are applied for families.

Service Charge Validation

- **Additional Service Validation**: For any additional services (e.g., room service, laundry), the system will validate that the guest's room account is active, ensuring services are only billed to checked-in guests. If the guest is checking out, the system prevents further charges from being added to their account.

7.3. INPUT INTEGRITY CONTROLS

To maintain the accuracy and validity of the data entered the system, several input integrity controls are implemented. These controls ensure that the data captured is correct, complete, and appropriate for the given fields. Below are a few key input integrity controls:

Field Type and Format Validation

- **Data Type Validation**: Each input field in the system is designed to accept only specific data types. For example, dates must be entered in the correct format (e.g., DD/MM/YYYY), numeric fields such as room rates or payment amounts accept only numerical values, and text fields are restricted to appropriate characters.

- **Email and Contact Number Validation**: For fields requiring email addresses or phone numbers, the system checks that they are in the correct format (e.g., an "@" symbol and domain for emails, and proper length for phone numbers), preventing invalid or incomplete entries.

Required Field Checks

- **Mandatory Fields**: The system enforces mandatory completion of critical fields (e.g., guest name, check-in/check-out dates, payment details). It prevents the submission of any form with missing required fields, ensuring that important information is never omitted.
- **Dynamic Field Requirements**: Depending on the user's selections, certain fields may dynamically become required. For instance, if a guest chooses to pay by credit card, the system ensures that the credit card number, expiry date, and CVV are entered.

Duplicate Entry Prevention

- **Unique Record Checks**: The system prevents duplicate entries by enforcing uniqueness in key fields, such as guest ID, reservation numbers, and invoice numbers. For example, if a reservation number already exists, the system flags an error, ensuring that no duplicate reservations are created for the same guest or room.
- **Credit Card Duplication Check**: If a guest attempts to use the same credit card for multiple bookings, the system checks for possible fraud or accidental reuse and prompts the user for confirmation or clarification.

7.4. OUTPUT INTEGRITY CONTROLS

Error Handling Mechanisms

- **Error Messages:** Implement clear and informative error messages to indicate when outputs fail validation checks.
- **Fallback Outputs:** Provide default outputs or error logs if the primary output fails integrity checks.

Access Controls

- **User Permissions:** Limit access to output data based on user roles to prevent unauthorized alterations or access.
- **Approval Processes:** Require approval for sensitive outputs before they are finalized and distributed.

8. IMPLEMENTATION PLAN

The plan outlines the steps to implement a hotel management system covering four main entities: Agents, Guests, Rooms, and Staff. It involves the following key phases:

1.**Initial Setup**: Establish the development environment and set up the database schema for Agents, Guests, Rooms, Agent Reservation, Guest Reservation, AgentReseravtionRoom, GuestReservationRoom, AgentAccount, GuestAccount and Staff.

2.**Database Design**: Design normalized tables with relationships between entities, ensuring data integrity for operations across all entities. On our database design we used multiple tables and included tables to handle individual reservations based on whether it's a guest or agent.

3.**CRUD Operations**: Implement Create, Read, Update, and Delete (CRUD) functions for each entity, ensuring smooth data management.

4.**UI Integration**: We then integrated our forms to be user friendly and so that it is easier to use by the receptionist and the state holders if updating customer information and extracting required information.

5.**Error Handling & Logging**: on the code we wrote on our classes we used try and catch statements to try and avoid unaccounted errors. For example, when the user is logging in we try and evaluate whether we have the user in our system and if they make mistakes we return message boxes showing them whether the error is and requesting them to fix it.

6.**Testing:** we will be conducting unit and integration testing, ensuring all CRUD operations work smoothly and that the system (system testing) is running smoothly. We use these CRUD operations to check if our application or project is working in a correct manner, like inserting or populating the room table and querying through the rooms that are occupied or those that are not occupied. We also created new reservations in the Agent Reservation table using the CRUD operation using the application and checked if the information is stored/updated in the Agent Reservation table.

9. TEST PLAN

9.1. TEST ENVIRONMENT

- Unit Testing Framework:
- NUnit: A popular unit testing framework for .NET applications. It provides a robust set of tools for running automated tests.
- UI Testing Tool:
- Coded UI Test (CUIT): A Visual Studio-based tool for automated UI testing of Windows Forms and other .NET applications (Note: Coded UI has been deprecated but might still be in use in some legacy projects).
- GitHub Action: Offers built-in workflows to run tests and perform builds on different platforms, including Windows.

9.2. TEST ITEMS

Test the Login Functionality

- **Description:** Verity's that the login process works as expected for different types of users (e.g., customers, admins).
- **Test Cases:**
  - Valid credentials allow successful login.
  - Invalid credentials (incorrect username/password) result in appropriate error messages.

Test Modifying/Changing the Reservation Details

- **Description:** Ensures that users can modify existing reservation details (e.g., dates, number of guests, room type).
- **Test Cases:**
  - Validates that reservations can be updated with valid information.
  - Checks if changes are correctly saved and reflected in the system.
  - Ensures the system provides feedback on successful modification.
  - Tests error handling for invalid inputs (e.g., changing to unavailable dates).

Test Cancelling a Reservation

- **Description:** Verify the process of cancelling an existing reservation, including user prompts and system updates.
- **Test Cases:**
    - Confirm that users can successfully cancel reservations.
    - Ensure that the system updates availability after a cancellation.
    - Check if confirmation messages is displayed on the screen after performing the use case.

Test Creating a New Reservation

- **Description:** Validate the process for creating a new reservation, from selecting dates and room types to final confirmation.
- **Test Cases:**
    - Confirm that all fields required for a new reservation can be entered (e.g., dates, room type, number of guests).
    - Validate pricing calculations based on selected options.
    - Verify that reservation creation triggers confirmation messages.

Test Generating a Report

- **Description:** Ensure that reports can be generated, displaying accurate and up-to-date information. This includes validating data and checking report formats.

Pending Reservations Report

- **Description:** This report should display all pending reservations that have not been confirmed or processed.
- **Test Cases:**
    - Confirms that all pending reservations are accurately displayed in the report.
    - Ensures that the report includes relevant details (e.g., customer names and dates etc).
    - Tests the report generation for different date ranges and filter criteria.

Occupancy Levels Report

- **Description:** This report provides a summary of the current occupancy levels of the facility, showing booked and available rooms.
- **Test Cases:**
    - Verify that the occupancy levels match the actual data in the system.
    - Ensure that the report shows accurate room statuses (e.g., booked, available).
    - Test the ability to filter by date range, room type, or other criteria.
    - Confirm that data is updated dynamically as reservations are created, modified, or cancelled.

9.3. TEST APPROACHES

Unit Testing:

- **Scope:** We tested individual classes, ensuring that they perform their intended functions without errors.
    - **Tested Class:**
        - **Login Class:** Ensured that user authentication works correctly with valid and invalid credentials.
        - **Reservation Class:** Verified that methods for creating, modifying, and cancelling reservations function as expected.
        - **Report Generation Classes:** Tested that reports like Pending Reservations and Occupancy Levels are generated accurately.

Integration Testing:

- **Scope:**
    - **Integration Between Business Classes and the UI:** We tested that user inputs from the UI correctly interact with the business logic, ensuring smooth data flow and functionality.
        - **Tested Feature:** Verified that making changes through the UI (such as modifying or cancelling a reservation) correctly triggers business logic and updates the system state.
    - **Integration Between Business Classes and Database Classes:** Ensured that the business logic successfully interacts with the database, verifying that data retrieval, updates, and inserts are working properly.
        - **Tested Feature:** Ensured that when a reservation is modified in the business logic, the database is updated accordingly and reflected in the UI.
    - **Database Classes and UI Variables:** Tested the functionality of database classes to ensure that they retrieve the correct data and display it properly in the UI.
        - **Tested Feature:** Verified that customer and reservation data retrieved from the database matches what is shown on the UI.

System Testing:

- **Scope:** We tested the main features of the system to ensure they function correctly when evaluated as a whole.
    - **Tested Features:**
        1. **Login Functionality:**
            - Verified that the login process works correctly, ensuring that valid users can log in while incorrect credentials result in appropriate error messages.
        2. **Modifying/Changing Reservation Details:**

- Tested that users can modify existing reservations, and those changes are reflected throughout the system, including updates to room availability and the database.

3. **Cancelling a Reservation:**
   - Confirmed that users can successfully cancel reservations, and the system updates both availability and any necessary cancellation policies.
4. **Creating a New Reservation:**
   - Ensured that new reservations can be created with correct availability checks, and that the system properly stores and reflects the new bookings.
5. **Generating Reports:**
   - **Pending Reservations Report:** Tested that the system accurately generates a report of all pending reservations, filtering the data correctly.
   - **Occupancy Levels Report:** Ensured that the report accurately shows current occupancy levels, based on bookings and room availability.

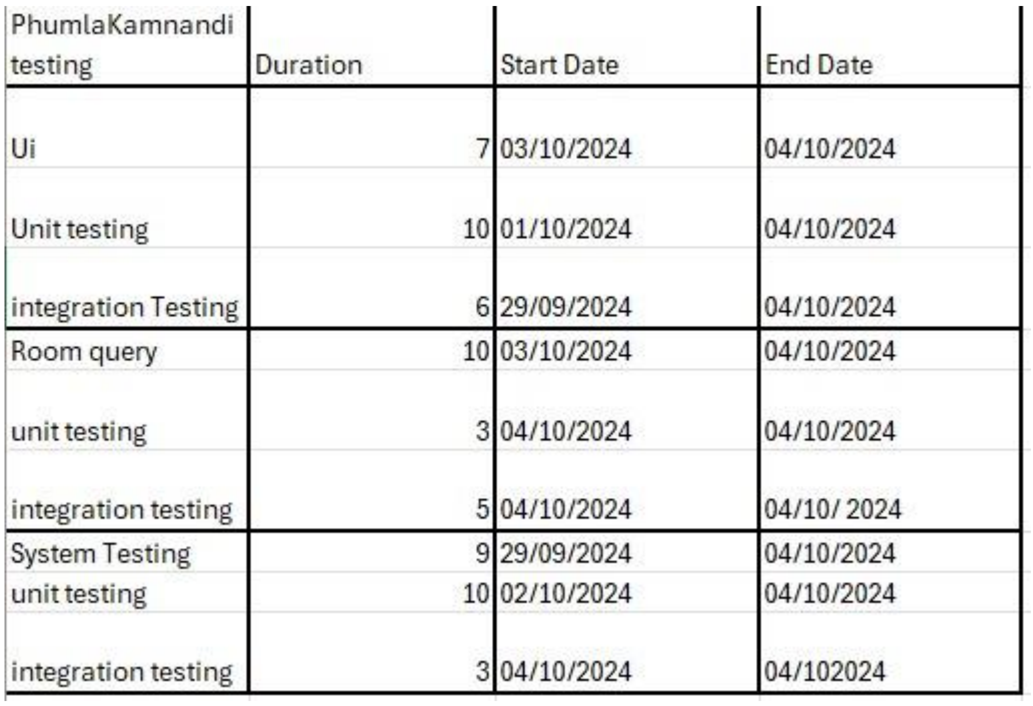
9.4. PROBLEM TRACKING (TEST CASES)

Test Case ID	**Test Scenario**	**Steps to Perform / User action.**	**Test Data**	**Expected Results: System behaviour or state.**	**Result - Comments**
TC01	Verify that the login process works for valid a	User successfully logs in with valid credentials; error message with invalid credentials	Username: Siyabonga Password: ZNGSIY012	User logs into the system successfully	pass
TC02	Verify that the login process works for valid and invalid credentials	User tries logs into the system with the wrong credentials	Username: James Password: James001	User is declined since they might not be authorized	*Pass if the user is not a receptionist or stakeholder*
TC03	Check that reservation details can be modified and the system updates accordingly	The user tries updating the reservation details that are already in the system	Guest Name: Given ResId: 001 Updated to: Tshepo	Reservation is updated and reflected in the system and database	Pass if updates are correct, fail otherwise
TC04	Ensure that users	The receptionist cancels a	resID: 001	Reservation is cancelled,	Pass if cancellation

	can cancel reservations, and the system frees up availability	reservation based on the guest or agents request	searched through the database and deleted the status of the room is changed to available.	availability is updated, and confirmation message is displayed on the screen	occurs and availability is updated, fail otherwise
TC05	Test that new reservations can be made and stored in the database	A new guest makes a reservation, and the receptionist takes in the data	The receptionist takes in the information and clicks the submit button and then reloads the home page to check if the reservation appears on the expected sign ins and updates the room status to occupied	New reservation is created, and room availability is updated accordingly	Pass if reservation is created and stored, fail otherwise
TC06	Verify that the occupancy level report is generated and include accurate data	The user queries through the rooms table and checks for the updated status	If the status is occupied, we use the count operation on our query which returns the number of occupied rooms	Reports are generated with accurate and up-to-date data	Pass if status is occupied
TC07	Verify that the Pending reservations is generated and include accurate data	The expected reservations or active reservations are displayed on the home page	The receptionist checks with the database.	Reports are generated with accurate and up-to-date data	Pass if the data is up to date

9.5. Test Schedule

Below is our testing schedule.



PhumlaKamnandi testing	Duration	Start Date	End Date
Ui	7	03/10/2024	04/10/2024
Unit testing	10	01/10/2024	04/10/2024
integration Testing	6	29/09/2024	04/10/2024
Room query	10	03/10/2024	04/10/2024
unit testing	3	04/10/2024	04/10/2024
integration testing	5	04/10/2024	04/10/ 2024
System Testing	9	29/09/2024	04/10/2024
unit testing	10	02/10/2024	04/10/2024
integration testing	3	04/10/2024	04/102024