Hotel Reservation System

Database Systems Design
CSE333

May 2, 2025

—

Handed to:

Dr. Nivin Atef
Eng. Amira Fekry

—

Handed by:

<u>Team no. 16</u>

Moaz Mohamed Zakaria 22P0307
Patrick Hany 2200557
Ahmed Abbady 22p0308
Ahmed Wael 22p0221
Omar Mohamed Mostafa 22p0197

Table of Contents

Introduction

The Hotel Reservation System Database is designed to efficiently manage hotel operations, ensuring a seamless booking experience for guests while providing a structured approach to handling reservations, payments, rooms, staff, and additional services. This database system aims to improve data organization, enhance accuracy, and streamline hotel management processes.

Project Overview

The Hotel Reservation System is a desktop application developed in C# with a graphical user interface (GUI), designed to streamline the process of managing hotel bookings, customer data, and room availability. This system aims to automate the key functions of a hotel front desk, reducing manual effort, minimizing human error, and improving overall customer experience.

The application interfaces with a relational database (e.g., SQL Server or MySQL) to store and retrieve data efficiently. The database maintains structured records for rooms, guests, reservations, payments, and staff. The GUI is built using Windows Forms (WinForms) or WPF, offering an intuitive and user-friendly interface for hotel receptionists or administrators.

Database

Tables

This section defines the structure of the hotel reservation system by creating several related tables. Each table represents a key entity of the system.

Guest: Stores information about each hotel guest including email, first and last names.

Guest_PhoneNumber: Allows each guest to have multiple phone numbers.

Reservation: Records the details of each reservation, such as status, dates, guest, and guest count.

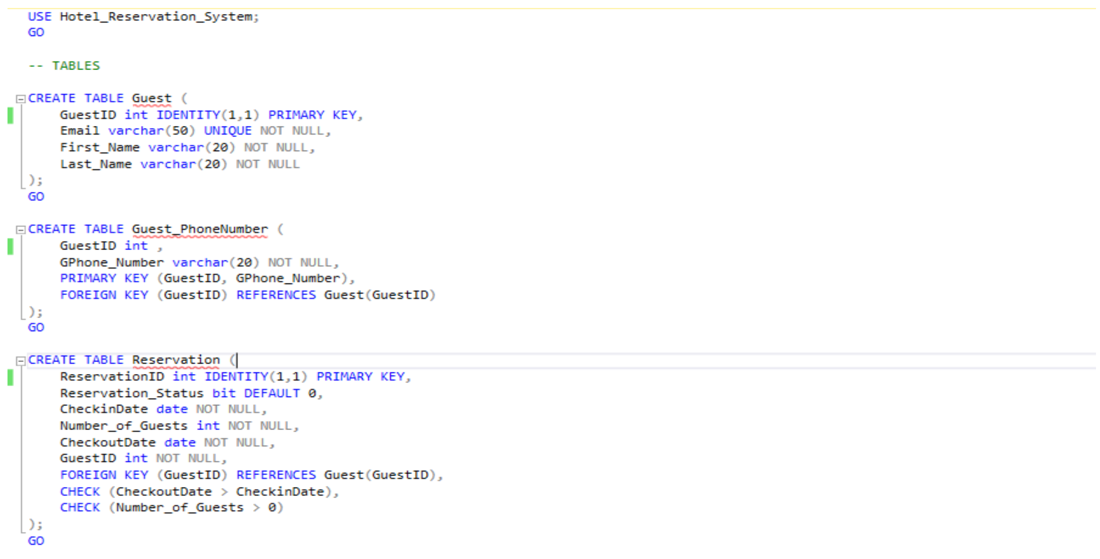
Payment: Tracks payments for reservations, including method, status, and amount paid.

Room: Contains details of each room, such as price, capacity, and status.

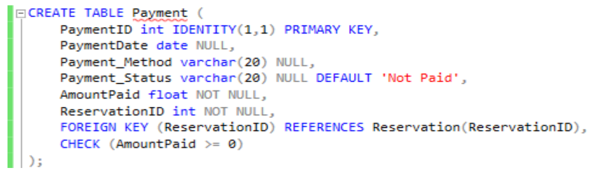
Service: Stores information about services offered (e.g., breakfast) and links to rooms.

Staff: Holds staff member details and their assigned roles.

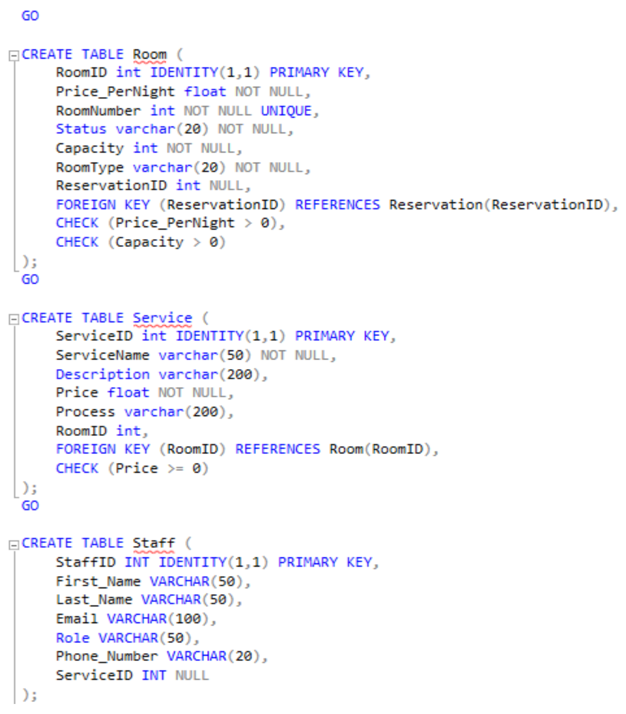
Handles: Manages the relationship between staff and reservations they handle.



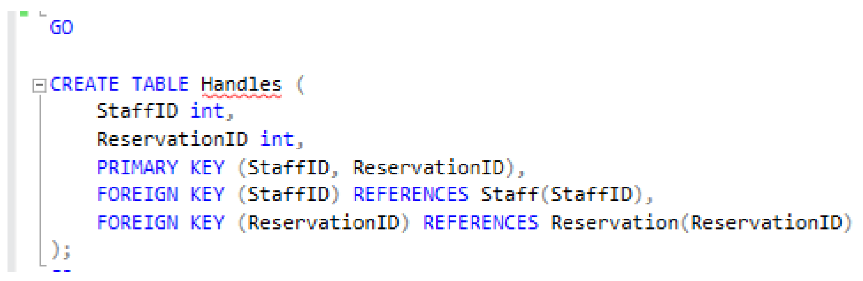
```sql
USE Hotel_Reservation_System;
GO

-- TABLES

CREATE TABLE Guest (
    GuestID int IDENTITY(1,1) PRIMARY KEY,
    Email varchar(50) UNIQUE NOT NULL,
    First_Name varchar(20) NOT NULL,
    Last_Name varchar(20) NOT NULL
);
GO

CREATE TABLE Guest_PhoneNumber (
    GuestID int ,
    GPhone_Number varchar(20) NOT NULL,
    PRIMARY KEY (GuestID, GPhone_Number),
    FOREIGN KEY (GuestID) REFERENCES Guest(GuestID)
);
GO

CREATE TABLE Reservation (
    ReservationID int IDENTITY(1,1) PRIMARY KEY,
    Reservation_Status bit DEFAULT 0,
    CheckinDate date NOT NULL,
    Number_of_Guests int NOT NULL,
    CheckoutDate date NOT NULL,
    GuestID int NOT NULL,
    FOREIGN KEY (GuestID) REFERENCES Guest(GuestID),
    CHECK (CheckoutDate > CheckinDate),
    CHECK (Number_of_Guests > 0)
);
GO
```



```sql
CREATE TABLE Payment (
    PaymentID int IDENTITY(1,1) PRIMARY KEY,
    PaymentDate date NULL,
    Payment_Method varchar(20) NULL,
    Payment_Status varchar(20) NULL DEFAULT 'Not Paid',
    AmountPaid float NOT NULL,
    ReservationID int NOT NULL,
    FOREIGN KEY (ReservationID) REFERENCES Reservation(ReservationID),
    CHECK (AmountPaid >= 0)
);
```



```sql
GO

CREATE TABLE Room (
    RoomID int IDENTITY(1,1) PRIMARY KEY,
    Price_PerNight float NOT NULL,
    RoomNumber int NOT NULL UNIQUE,
    Status varchar(20) NOT NULL,
    Capacity int NOT NULL,
    RoomType varchar(20) NOT NULL,
    ReservationID int NULL,
    FOREIGN KEY (ReservationID) REFERENCES Reservation(ReservationID),
    CHECK (Price_PerNight > 0),
    CHECK (Capacity > 0)
);
GO

CREATE TABLE Service (
    ServiceID int IDENTITY(1,1) PRIMARY KEY,
    ServiceName varchar(50) NOT NULL,
    Description varchar(200),
    Price float NOT NULL,
    Process varchar(200),
    RoomID int,
    FOREIGN KEY (RoomID) REFERENCES Room(RoomID),
    CHECK (Price >= 0)
);
GO

CREATE TABLE Staff (
    StaffID INT IDENTITY(1,1) PRIMARY KEY,
    First_Name VARCHAR(50),
    Last_Name VARCHAR(50),
    Email VARCHAR(100),
    Role VARCHAR(50),
    Phone_Number VARCHAR(20),
    ServiceID INT NULL
);
```



```sql
GO

CREATE TABLE Handles (
    StaffID int,
    ReservationID int,
    PRIMARY KEY (StaffID, ReservationID),
    FOREIGN KEY (StaffID) REFERENCES Staff(StaffID),
    FOREIGN KEY (ReservationID) REFERENCES Reservation(ReservationID)
);
--
```

Inserting Initial Data

This part inserts sample data into the tables to illustrate how the system would look with real values.

Room: Adds various rooms, each with a specific number, type, and capacity.

Guest: Inserts example guests with emails and names.

Guest_PhoneNumber: Associates phone numbers with guest IDs.

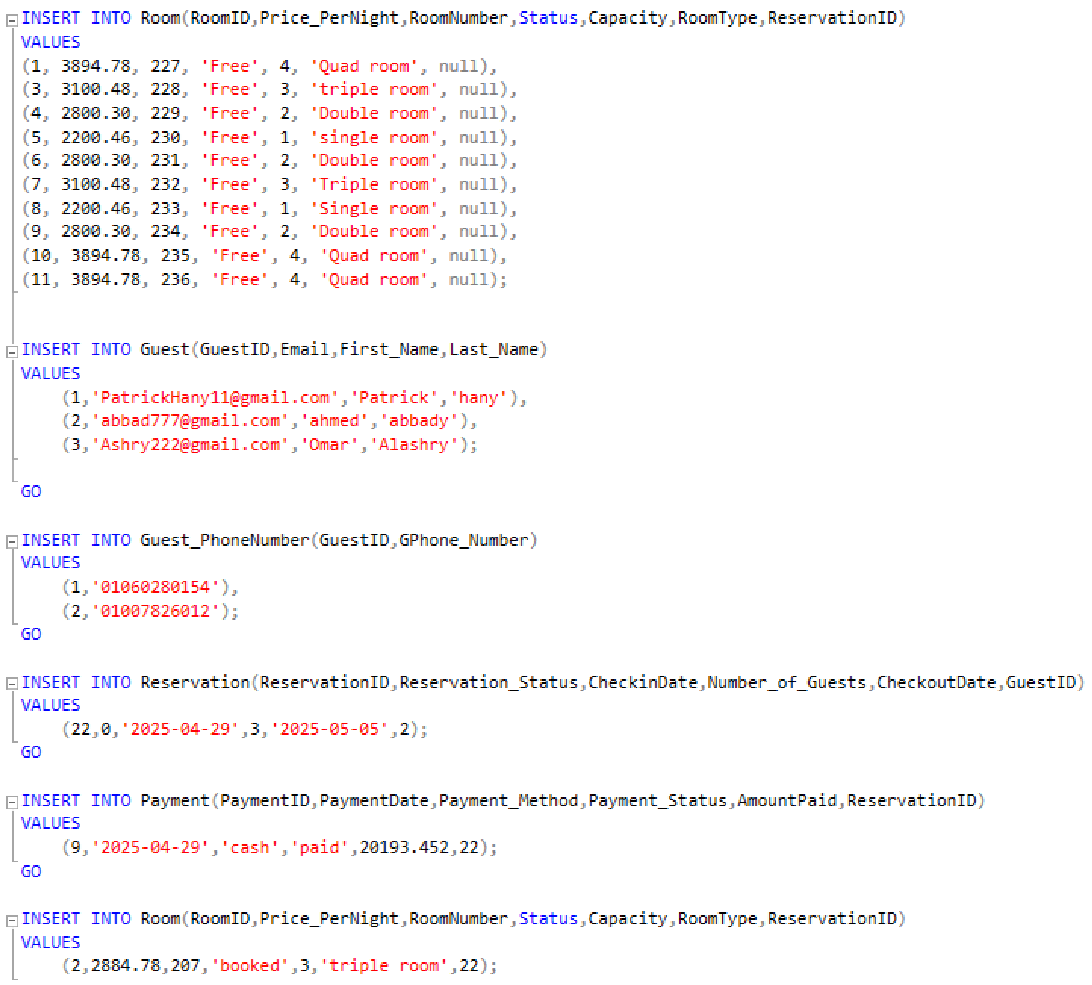
Reservation: Adds a sample reservation record.

Payment: Records a payment made for a reservation.

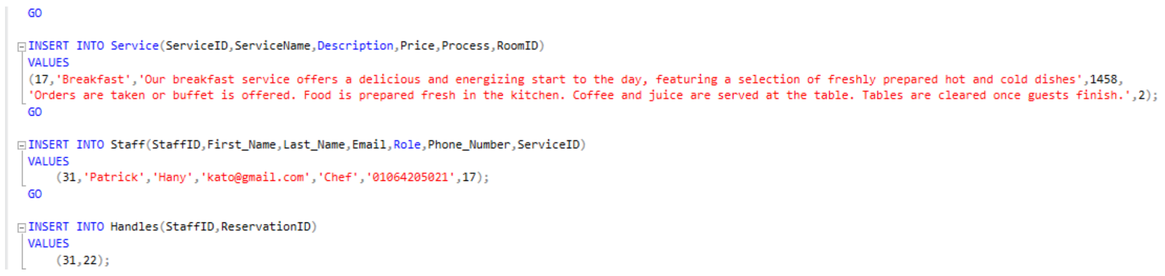
Service: Describes a service, including its price and process.

Staff: Adds a staff member and assigns them a role.

Handles: Links a staff member to a reservation they are responsible for.



```sql
INSERT INTO Room(RoomID,Price_PerNight,RoomNumber,Status,Capacity,RoomType,ReservationID)
VALUES
(1, 3894.78, 227, 'Free', 4, 'Quad room', null),
(3, 3100.48, 228, 'Free', 3, 'triple room', null),
(4, 2800.30, 229, 'Free', 2, 'Double room', null),
(5, 2200.46, 230, 'Free', 1, 'single room', null),
(6, 2800.30, 231, 'Free', 2, 'Double room', null),
(7, 3100.48, 232, 'Free', 3, 'Triple room', null),
(8, 2200.46, 233, 'Free', 1, 'Single room', null),
(9, 2800.30, 234, 'Free', 2, 'Double room', null),
(10, 3894.78, 235, 'Free', 4, 'Quad room', null),
(11, 3894.78, 236, 'Free', 4, 'Quad room', null);


INSERT INTO Guest(GuestID,Email,First_Name,Last_Name)
VALUES
    (1,'PatrickHany11@gmail.com','Patrick','hany'),
    (2,'abbad777@gmail.com','ahmed','abbady'),
    (3,'Ashry222@gmail.com','Omar','Alashry');

GO

INSERT INTO Guest_PhoneNumber(GuestID,GPhone_Number)
VALUES
    (1,'01060280154'),
    (2,'01007826012');
GO

INSERT INTO Reservation(ReservationID,Reservation_Status,CheckinDate,Number_of_Guests,CheckoutDate,GuestID)
VALUES
    (22,0,'2025-04-29',3,'2025-05-05',2);
GO

INSERT INTO Payment(PaymentID,PaymentDate,Payment_Method,Payment_Status,AmountPaid,ReservationID)
VALUES
    (9,'2025-04-29','cash','paid',20193.452,22);
GO

INSERT INTO Room(RoomID,Price_PerNight,RoomNumber,Status,Capacity,RoomType,ReservationID)
VALUES
    (2,2884.78,207,'booked',3,'triple room',22);
```



```sql
 GO

INSERT INTO Service(ServiceID,ServiceName,Description,Price,Process,RoomID)
VALUES
(17,'Breakfast','Our breakfast service offers a delicious and energizing start to the day, featuring a selection of freshly prepared hot and cold dishes',1458,
'Orders are taken or buffet is offered. Food is prepared fresh in the kitchen. Coffee and juice are served at the table. Tables are cleared once guests finish.',2);
GO

INSERT INTO Staff(StaffID,First_Name,Last_Name,Email,Role,Phone_Number,ServiceID)
VALUES
    (31,'Patrick','Hany','kato@gmail.com','Chef','01064205021',17);
GO

INSERT INTO Handles(StaffID,ReservationID)
VALUES
    (31,22);
```

Basic SELECT Queries and Table Joins

This section demonstrates how to retrieve data from the database and join related data across tables.

Basic SELECT * queries to view all contents of each table.

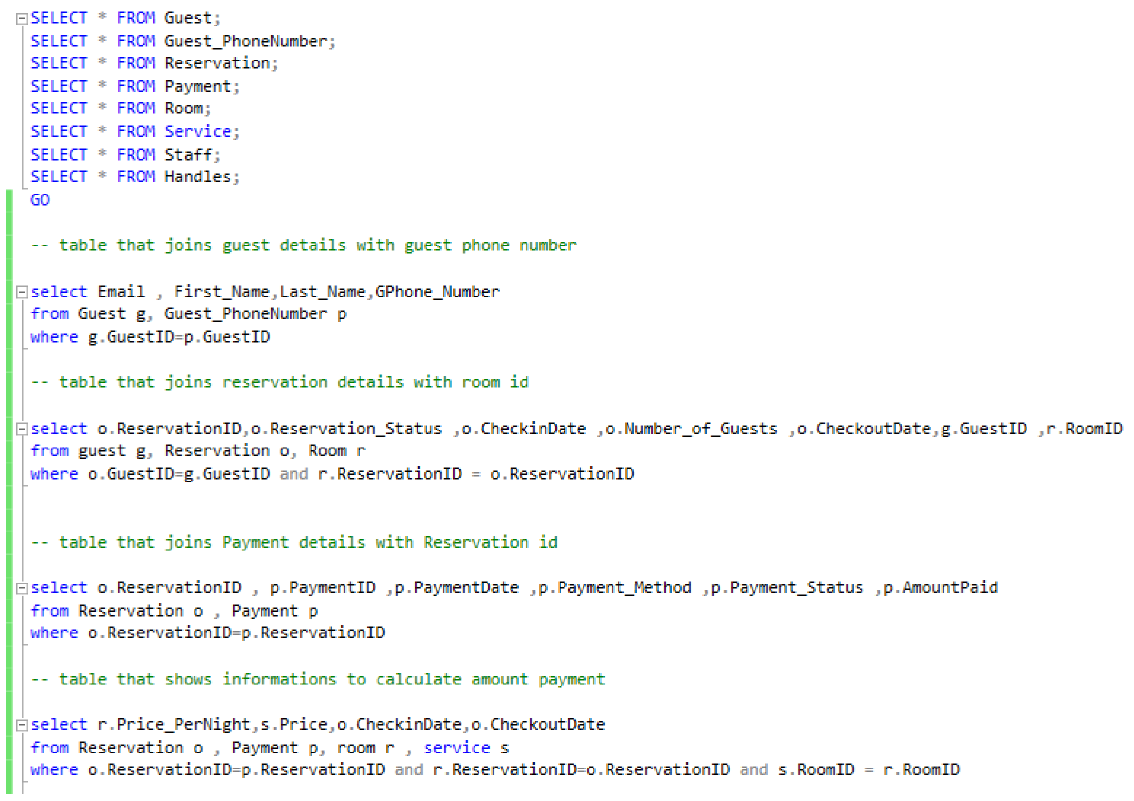
Example joins to:

Combine guest details with their phone numbers.

Show reservation details along with related room IDs.

Connect payment information to reservation records.

Calculate payment-related information using multiple tables.



```sql
SELECT * FROM Guest;
SELECT * FROM Guest_PhoneNumber;
SELECT * FROM Reservation;
SELECT * FROM Payment;
SELECT * FROM Room;
SELECT * FROM Service;
SELECT * FROM Staff;
SELECT * FROM Handles;
GO

-- table that joins guest details with guest phone number

select Email , First_Name,Last_Name,GPhone_Number
from Guest g, Guest_PhoneNumber p
where g.GuestID=p.GuestID

-- table that joins reservation details with room id

select o.ReservationID,o.Reservation_Status ,o.CheckinDate ,o.Number_of_Guests ,o.CheckoutDate,g.GuestID ,r.RoomID
from guest g, Reservation o, Room r
where o.GuestID=g.GuestID and r.ReservationID = o.ReservationID


-- table that joins Payment details with Reservation id

select o.ReservationID , p.PaymentID ,p.PaymentDate ,p.Payment_Method ,p.Payment_Status ,p.AmountPaid
from Reservation o , Payment p
where o.ReservationID=p.ReservationID

-- table that shows informations to calculate amount payment

select r.Price_PerNight,s.Price,o.CheckinDate,o.CheckoutDate
from Reservation o , Payment p, room r , service s
where o.ReservationID=p.ReservationID and r.ReservationID=o.ReservationID and s.RoomID = r.RoomID
```

Stored Procedures

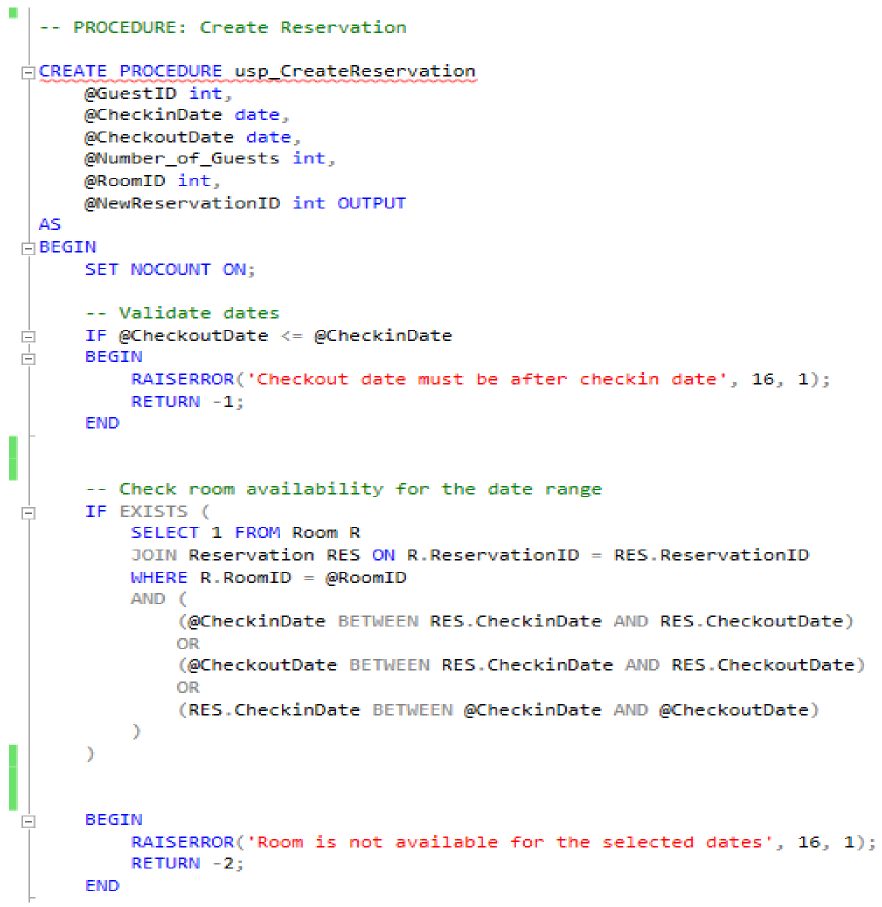
Stored procedures encapsulate common tasks and business logic within the database for reusability and consistency.

usp_CreateReservation: Validates dates and checks room availability before creating a reservation and associating it with a room.

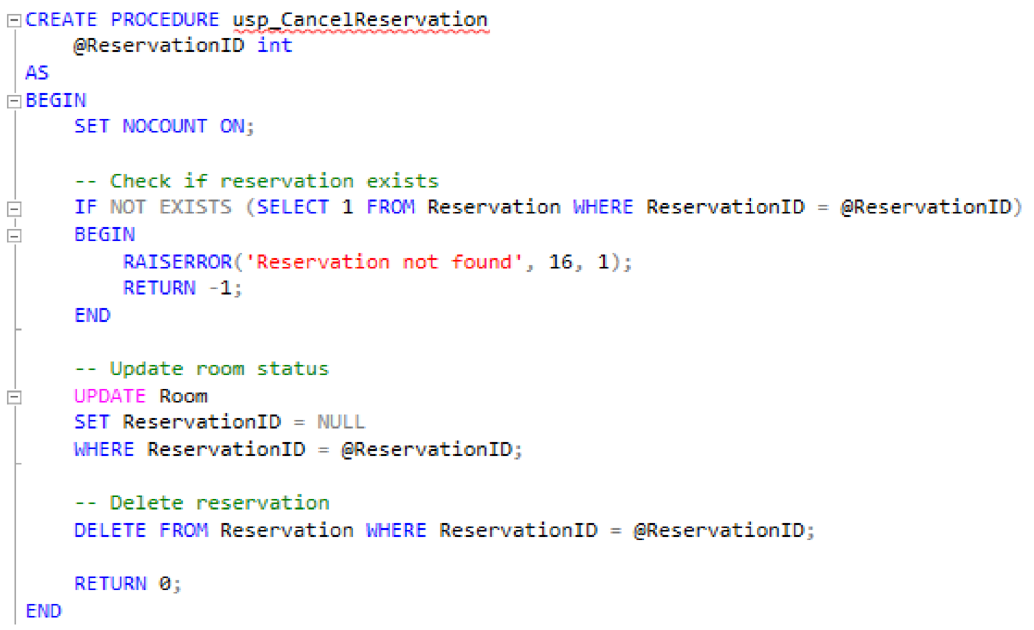
usp_CancelReservation: Cancels reservations after verifying their existence, updates room status, and deletes the reservation.

usp_RecordPayment: Records a payment, checks if the total due is paid, and updates the reservation status if completed.

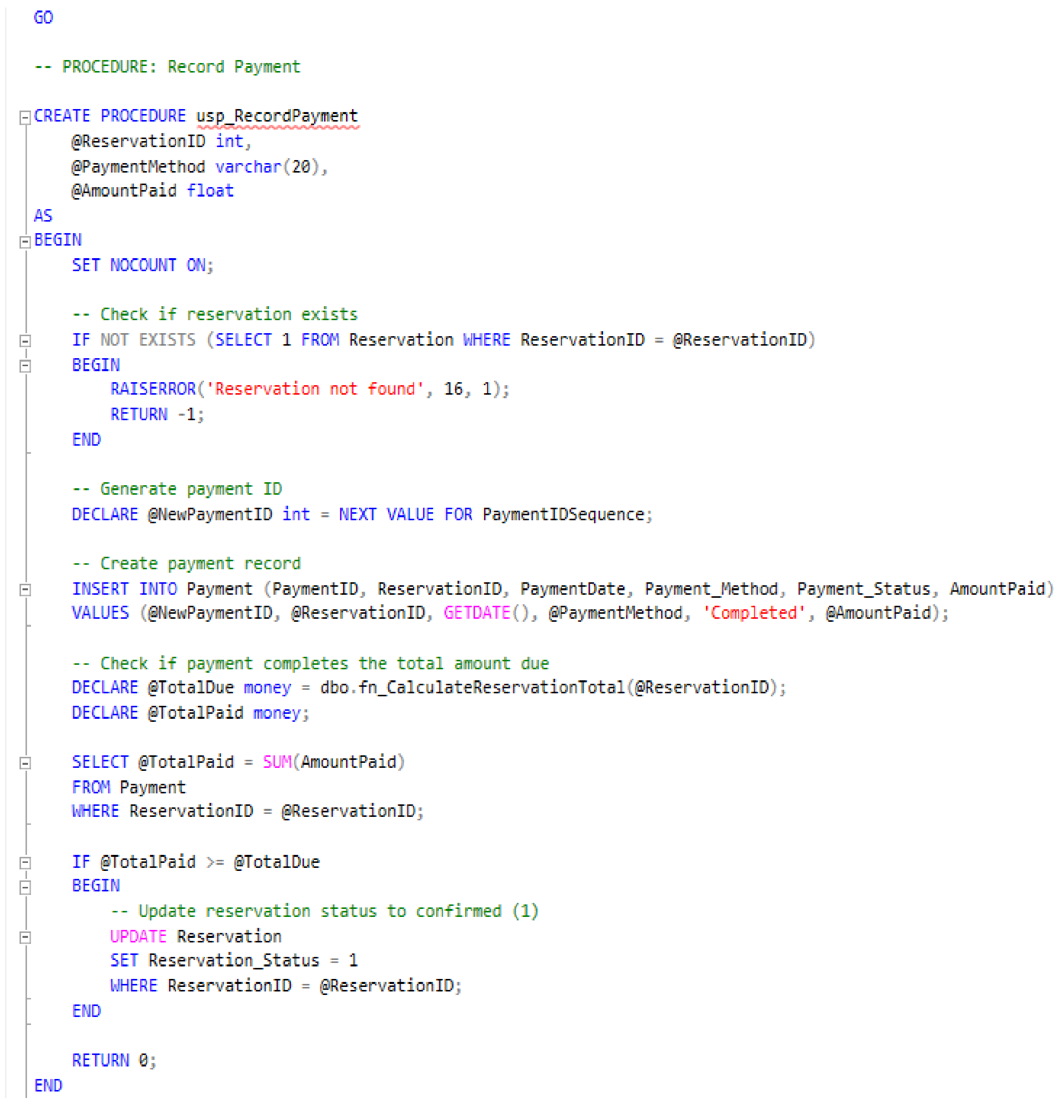
usp_AssignStaffToReservation: Assigns a staff member to handle a reservation, verifying the existence of both.



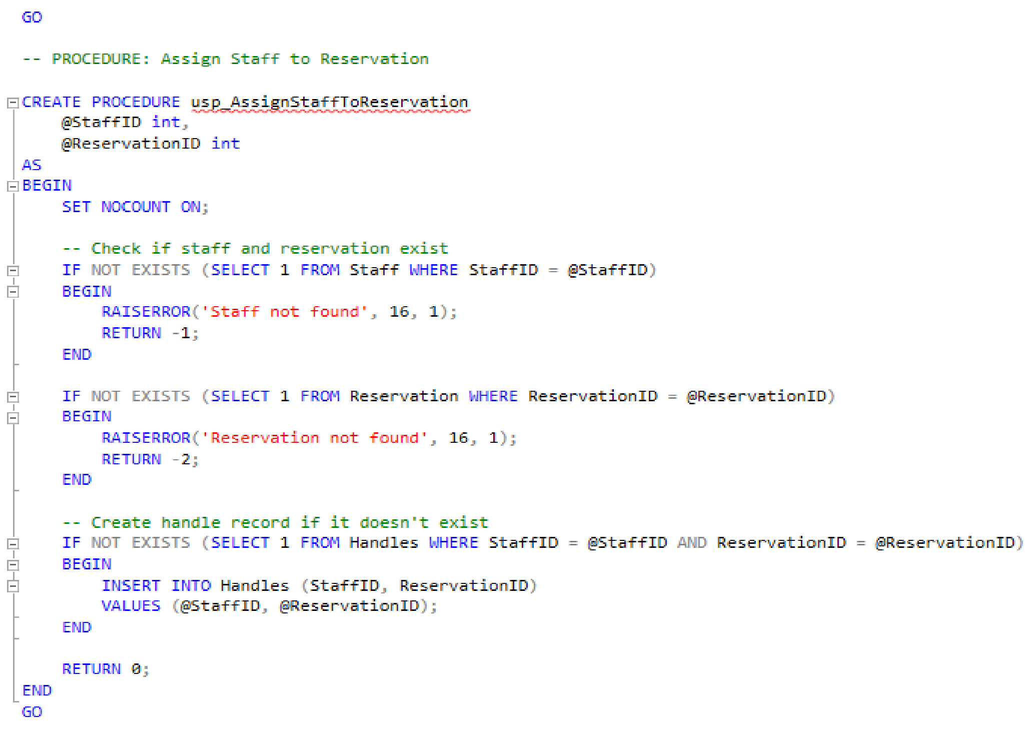
```sql
-- PROCEDURE: Create Reservation

CREATE PROCEDURE usp_CreateReservation
    @GuestID int,
    @CheckinDate date,
    @CheckoutDate date,
    @Number_of_Guests int,
    @RoomID int,
    @NewReservationID int OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    -- Validate dates
    IF @CheckoutDate <= @CheckinDate
    BEGIN
        RAISERROR('Checkout date must be after checkin date', 16, 1);
        RETURN -1;
    END


    -- Check room availability for the date range
    IF EXISTS (
        SELECT 1 FROM Room R
        JOIN Reservation RES ON R.ReservationID = RES.ReservationID
        WHERE R.RoomID = @RoomID
        AND (
            (@CheckinDate BETWEEN RES.CheckinDate AND RES.CheckoutDate)
            OR
            (@CheckoutDate BETWEEN RES.CheckinDate AND RES.CheckoutDate)
            OR
            (RES.CheckinDate BETWEEN @CheckinDate AND @CheckoutDate)
        )
    )


    BEGIN
        RAISERROR('Room is not available for the selected dates', 16, 1);
        RETURN -2;
    END
```



```sql
CREATE PROCEDURE usp_CancelReservation
    @ReservationID int
AS
BEGIN
    SET NOCOUNT ON;

    -- Check if reservation exists
    IF NOT EXISTS (SELECT 1 FROM Reservation WHERE ReservationID = @ReservationID)
    BEGIN
        RAISERROR('Reservation not found', 16, 1);
        RETURN -1;
    END

    -- Update room status
    UPDATE Room
    SET ReservationID = NULL
    WHERE ReservationID = @ReservationID;

    -- Delete reservation
    DELETE FROM Reservation WHERE ReservationID = @ReservationID;

    RETURN 0;
END
```



```sql
GO

-- PROCEDURE: Record Payment

CREATE PROCEDURE usp_RecordPayment
    @ReservationID int,
    @PaymentMethod varchar(20),
    @AmountPaid float
AS
BEGIN
    SET NOCOUNT ON;

    -- Check if reservation exists
    IF NOT EXISTS (SELECT 1 FROM Reservation WHERE ReservationID = @ReservationID)
    BEGIN
        RAISERROR('Reservation not found', 16, 1);
        RETURN -1;
    END

    -- Generate payment ID
    DECLARE @NewPaymentID int = NEXT VALUE FOR PaymentIDSequence;

    -- Create payment record
    INSERT INTO Payment (PaymentID, ReservationID, PaymentDate, Payment_Method, Payment_Status, AmountPaid)
    VALUES (@NewPaymentID, @ReservationID, GETDATE(), @PaymentMethod, 'Completed', @AmountPaid);

    -- Check if payment completes the total amount due
    DECLARE @TotalDue money = dbo.fn_CalculateReservationTotal(@ReservationID);
    DECLARE @TotalPaid money;

    SELECT @TotalPaid = SUM(AmountPaid)
    FROM Payment
    WHERE ReservationID = @ReservationID;

    IF @TotalPaid >= @TotalDue
    BEGIN
        -- Update reservation status to confirmed (1)
        UPDATE Reservation
        SET Reservation_Status = 1
        WHERE ReservationID = @ReservationID;
    END

    RETURN 0;
END
```



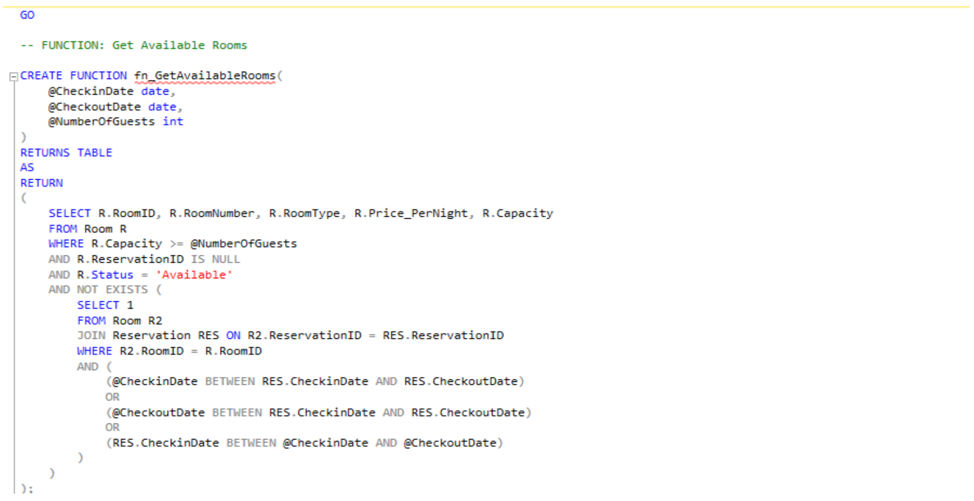
```sql
GO

-- PROCEDURE: Assign Staff to Reservation

CREATE PROCEDURE usp_AssignStaffToReservation
    @StaffID int,
    @ReservationID int
AS
BEGIN
    SET NOCOUNT ON;

    -- Check if staff and reservation exist
    IF NOT EXISTS (SELECT 1 FROM Staff WHERE StaffID = @StaffID)
    BEGIN
        RAISERROR('Staff not found', 16, 1);
        RETURN -1;
    END

    IF NOT EXISTS (SELECT 1 FROM Reservation WHERE ReservationID = @ReservationID)
    BEGIN
        RAISERROR('Reservation not found', 16, 1);
        RETURN -2;
    END

    -- Create handle record if it doesn't exist
    IF NOT EXISTS (SELECT 1 FROM Handles WHERE StaffID = @StaffID AND ReservationID = @ReservationID)
    BEGIN
        INSERT INTO Handles (StaffID, ReservationID)
        VALUES (@StaffID, @ReservationID);
    END

    RETURN 0;
END
GO
```

User-Defined Functions

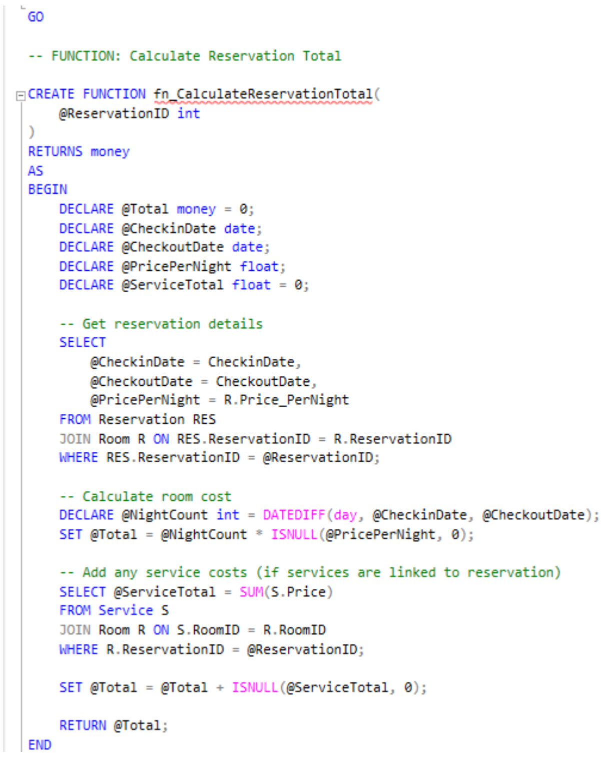
Functions add modularity to the code by encapsulating reusable calculations.

fn_GetAvailableRooms: Returns all available rooms for a given date range and guest count, considering overlapping reservations.

fn_CalculateReservationTotal: Calculates the total cost of a reservation, combining room charges and any associated service costs.



```sql
GO

-- FUNCTION: Get Available Rooms

CREATE FUNCTION fn_GetAvailableRooms(
    @CheckinDate date,
    @CheckoutDate date,
    @NumberOfGuests int
)
RETURNS TABLE
AS
RETURN
(
    SELECT R.RoomID, R.RoomNumber, R.RoomType, R.Price_PerNight, R.Capacity
    FROM Room R
    WHERE R.Capacity >= @NumberOfGuests
    AND R.ReservationID IS NULL
    AND R.Status = 'Available'
    AND NOT EXISTS (
        SELECT 1
        FROM Room R2
        JOIN Reservation RES ON R2.ReservationID = RES.ReservationID
        WHERE R2.RoomID = R.RoomID
        AND (
            (@CheckinDate BETWEEN RES.CheckinDate AND RES.CheckoutDate)
            OR
            (@CheckoutDate BETWEEN RES.CheckinDate AND RES.CheckoutDate)
            OR
            (RES.CheckinDate BETWEEN @CheckinDate AND @CheckoutDate)
        )
    )
);
```

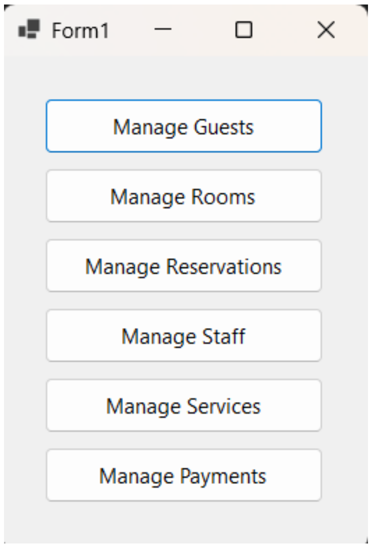


```sql
GO

-- FUNCTION: Calculate Reservation Total

CREATE FUNCTION fn_CalculateReservationTotal(
    @ReservationID int
)
RETURNS money
AS
BEGIN
    DECLARE @Total money = 0;
    DECLARE @CheckinDate date;
    DECLARE @CheckoutDate date;
    DECLARE @PricePerNight float;
    DECLARE @ServiceTotal float = 0;

    -- Get reservation details
    SELECT
        @CheckinDate = CheckinDate,
        @CheckoutDate = CheckoutDate,
        @PricePerNight = R.Price_PerNight
    FROM Reservation RES
    JOIN Room R ON RES.ReservationID = R.ReservationID
    WHERE RES.ReservationID = @ReservationID;

    -- Calculate room cost
    DECLARE @NightCount int = DATEDIFF(day, @CheckinDate, @CheckoutDate);
    SET @Total = @NightCount * ISNULL(@PricePerNight, 0);

    -- Add any service costs (if services are linked to reservation)
    SELECT @ServiceTotal = SUM(S.Price)
    FROM Service S
    JOIN Room R ON S.RoomID = R.RoomID
    WHERE R.ReservationID = @ReservationID;

    SET @Total = @Total + ISNULL(@ServiceTotal, 0);

    RETURN @Total;
END
```

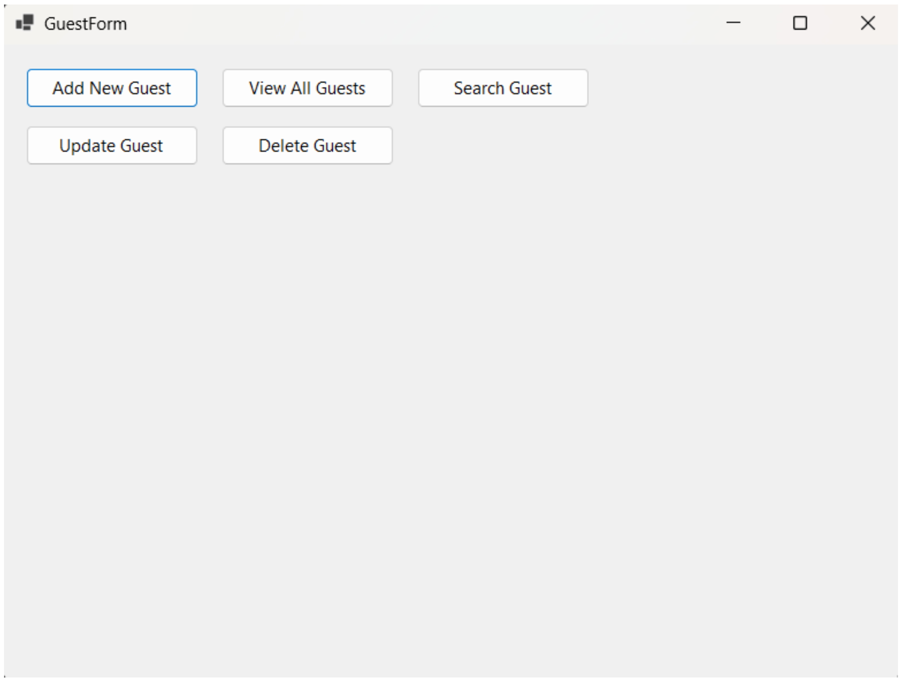
GUI

Screens

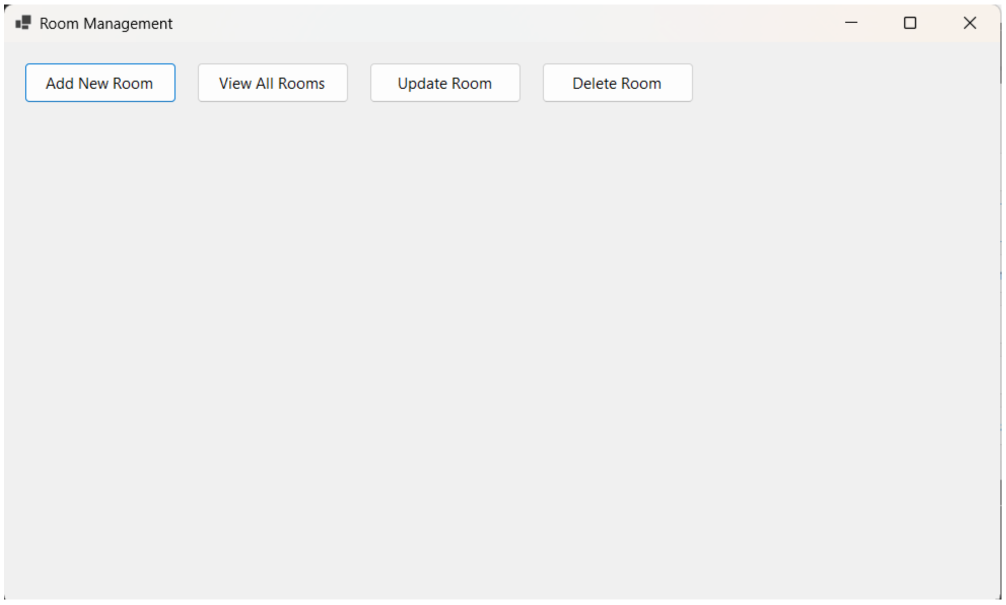
Main Screen



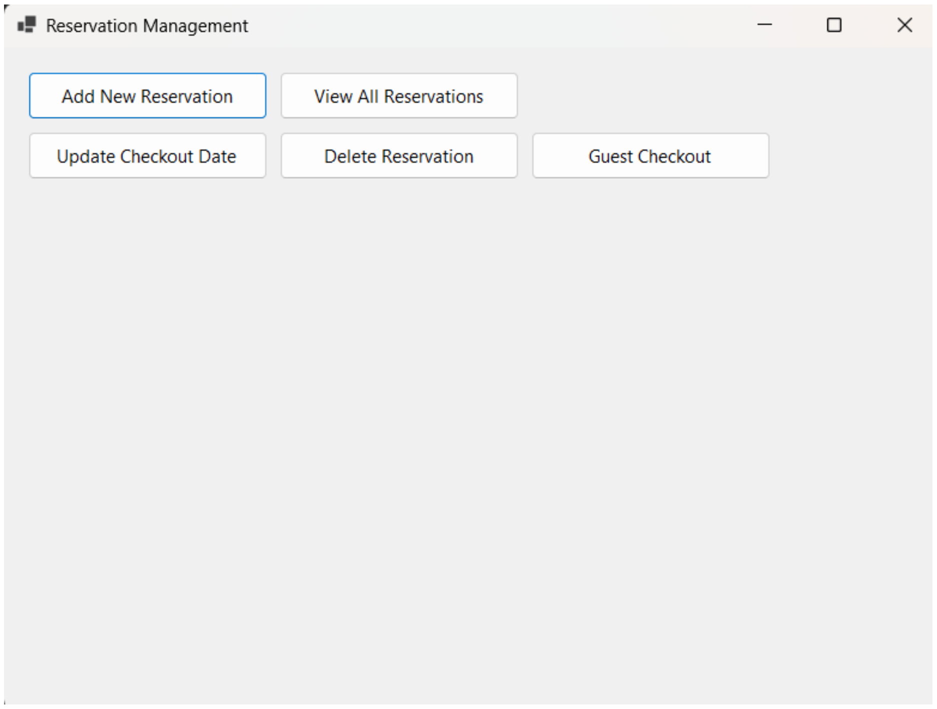
Manage Guests



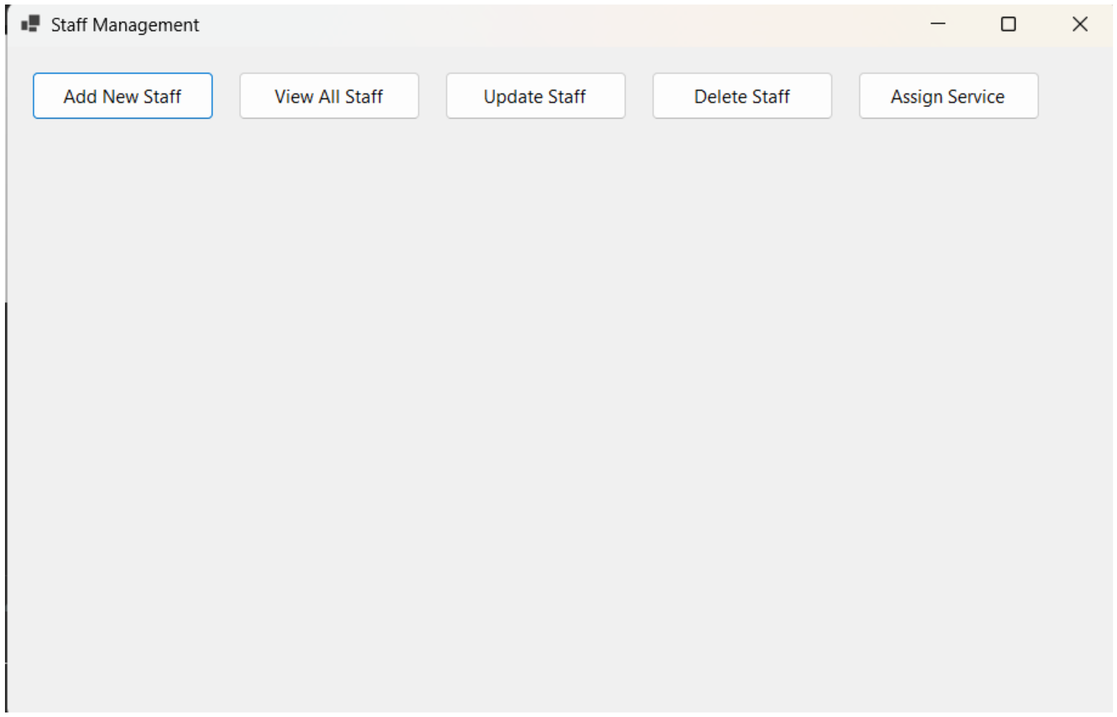
Manage Rooms



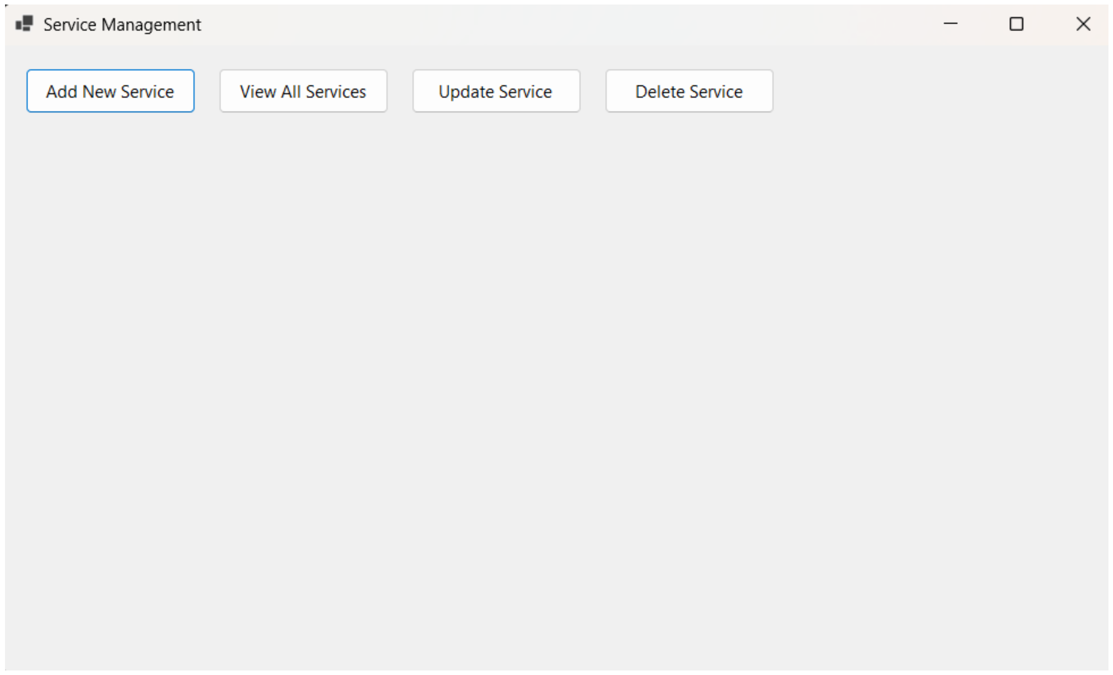
Manage Reservations



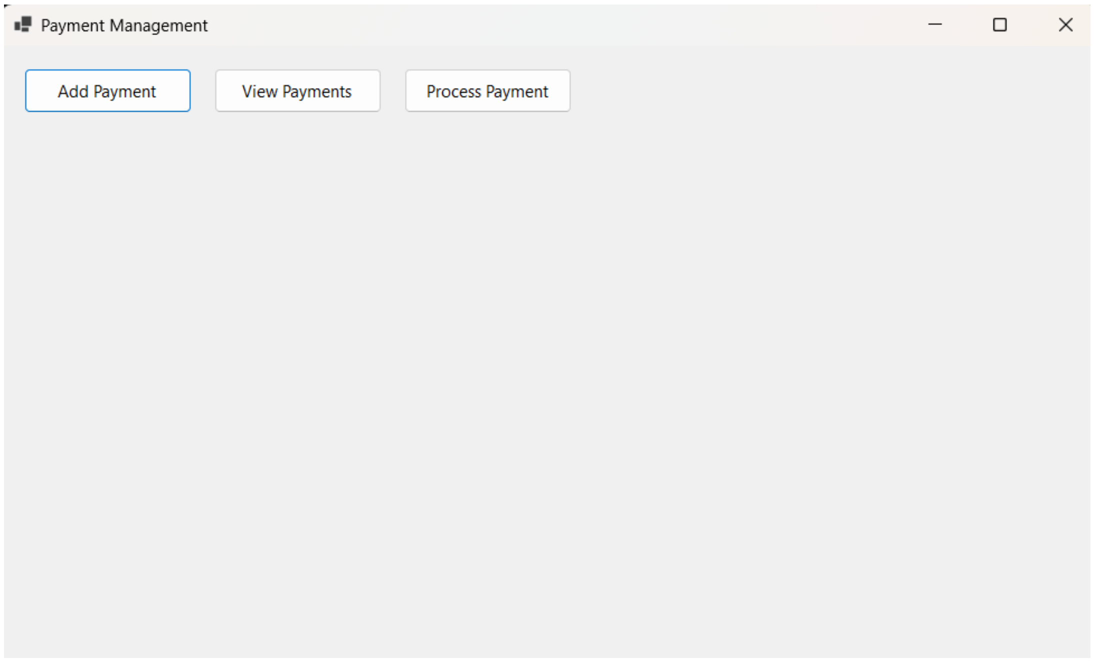
Manage Staff



Manage Services

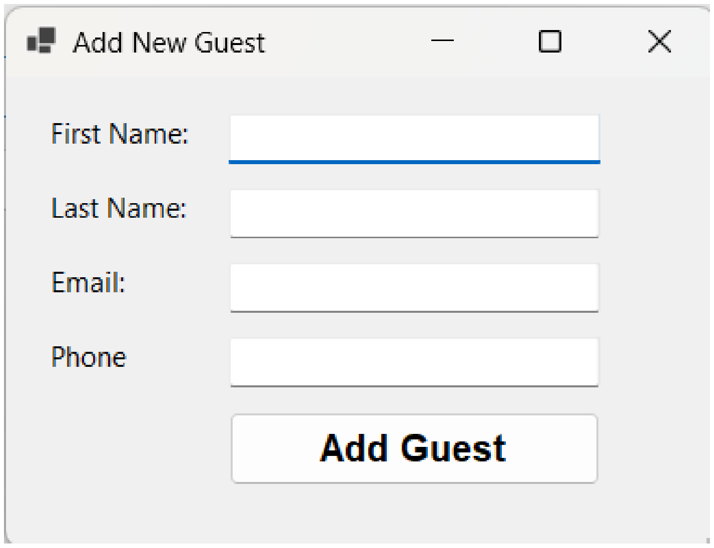


Manage Payments

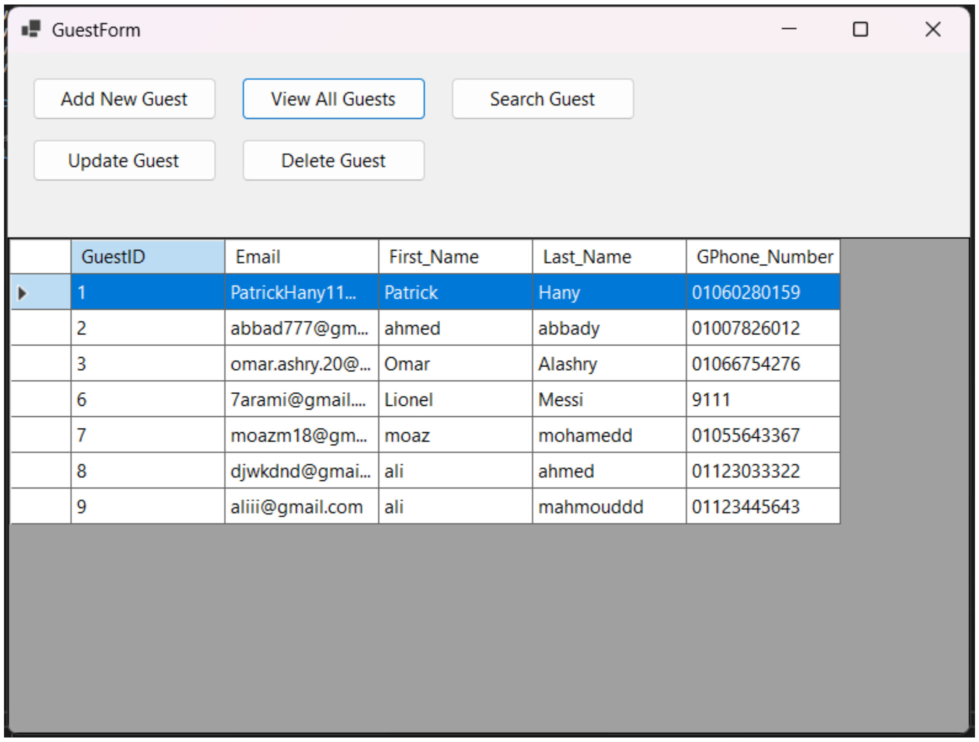


Functionalities

Add a new Guest

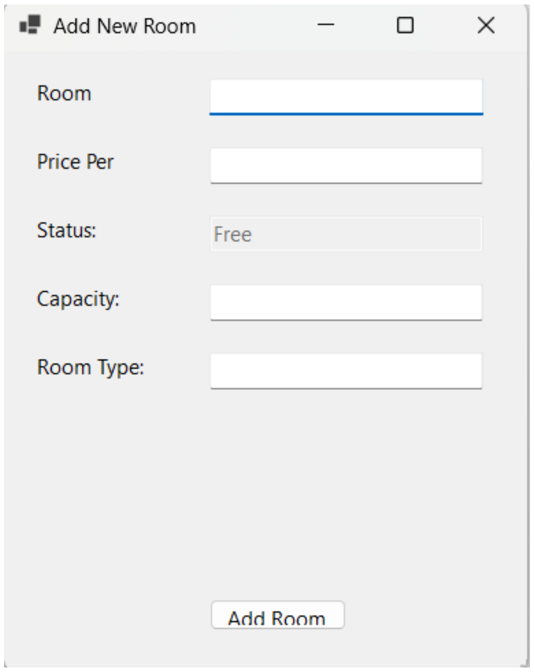


View all Guests

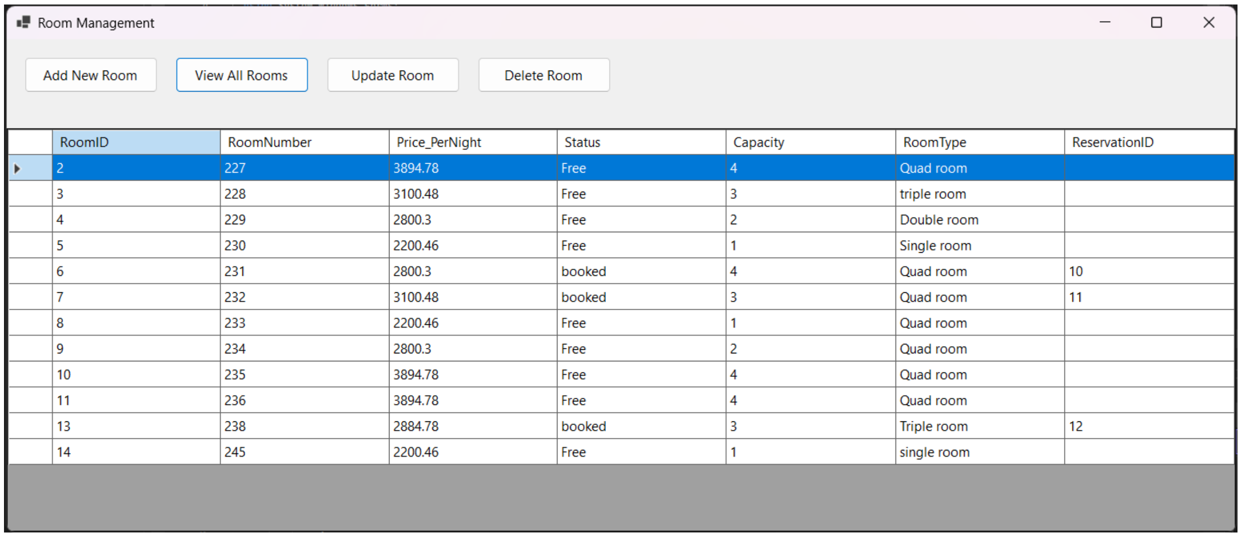


GuestID	Email	First_Name	Last_Name	GPhone_Number
1	PatrickHany11...	Patrick	Hany	01060280159
2	abbad777@gm...	ahmed	abbady	01007826012
3	omar.ashry.20@...	Omar	Alashry	01066754276
6	7arami@gmail....	Lionel	Messi	9111
7	moazm18@gm...	moaz	mohamedd	01055643367
8	djwkdnd@gmai...	ali	ahmed	01123033322
9	aliii@gmail.com	ali	mahmouddd	01123445643

Add new Room

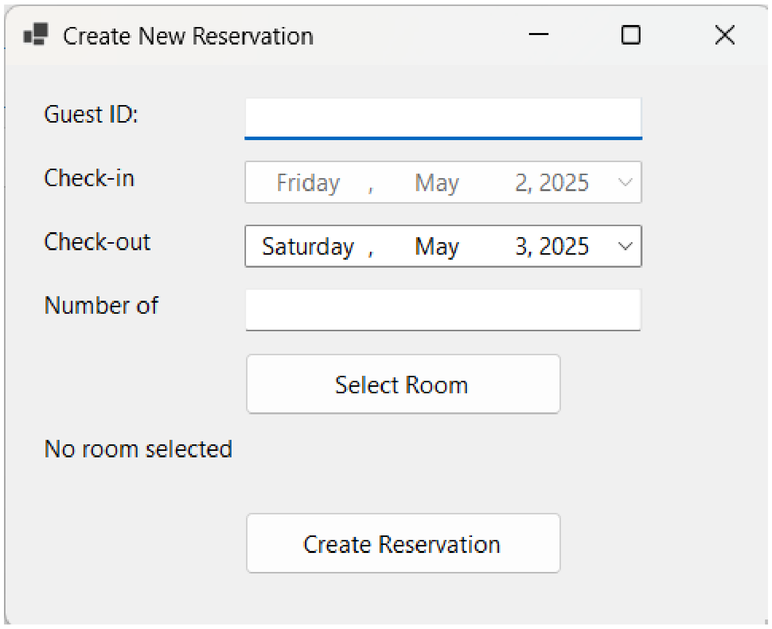


View all Rooms



RoomID	RoomNumber	Price_PerNight	Status	Capacity	RoomType	ReservationID
2	227	3894.78	Free	4	Quad room	
3	228	3100.48	Free	3	triple room	
4	229	2800.3	Free	2	Double room	
5	230	2200.46	Free	1	Single room	
6	231	2800.3	booked	4	Quad room	10
7	232	3100.48	booked	3	Quad room	11
8	233	2200.46	Free	1	Quad room	
9	234	2800.3	Free	2	Quad room	
10	235	3894.78	Free	4	Quad room	
11	236	3894.78	Free	4	Quad room	
13	238	2884.78	booked	3	Triple room	12
14	245	2200.46	Free	1	single room	

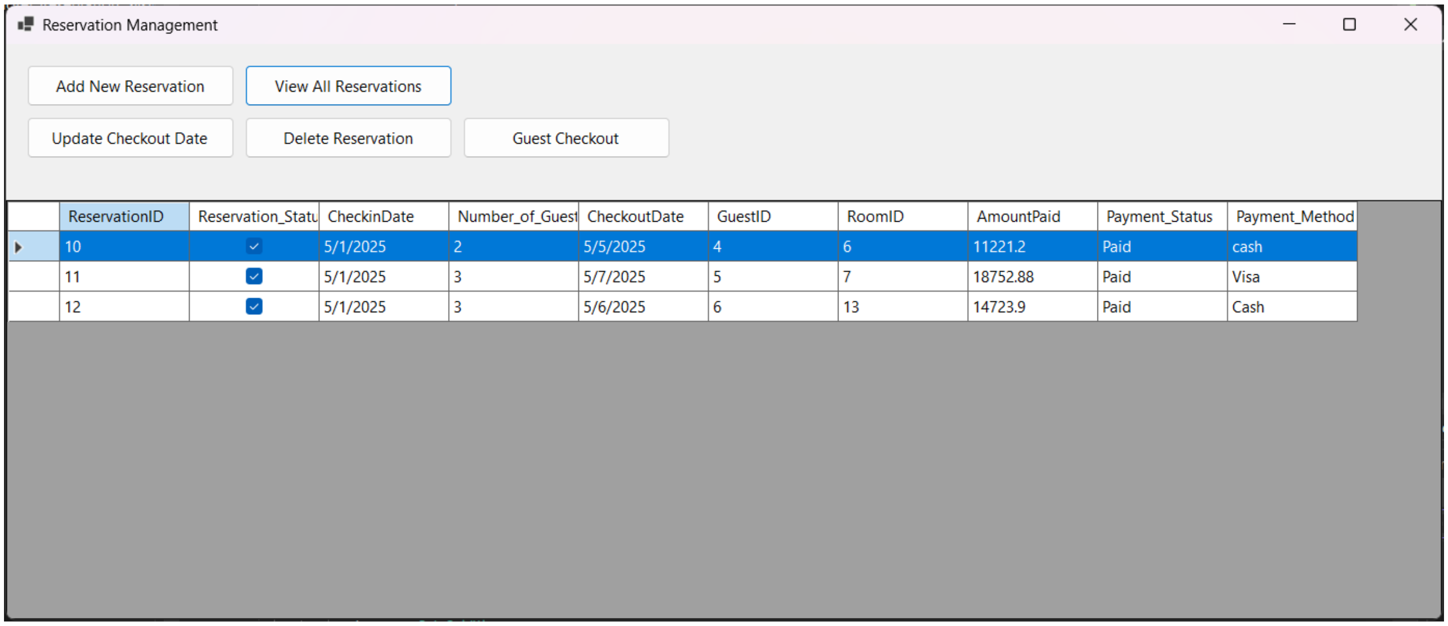
Create a new Reservation



Guest ID:	
Check-in	Friday , May 2, 2025
Check-out	Saturday , May 3, 2025
Number of	

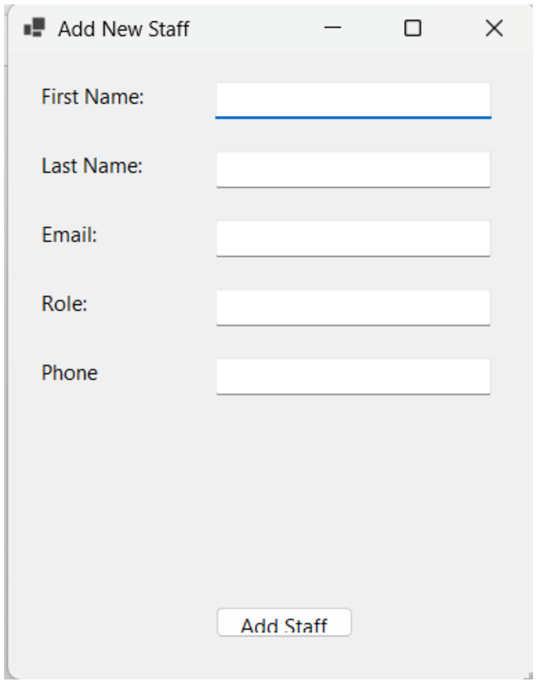
No room selected

View all Reservations

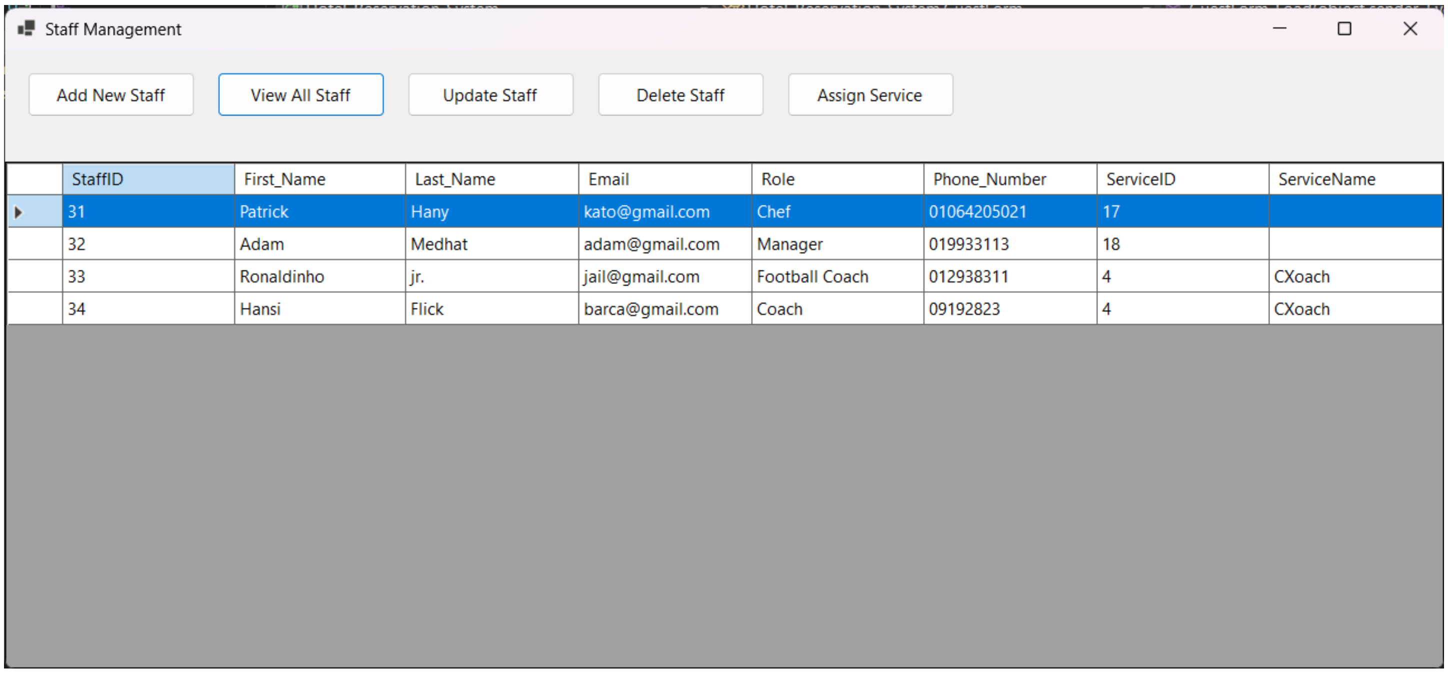


ReservationID	Reservation_Statu	CheckinDate	Number_of_Guest	CheckoutDate	GuestID	RoomID	AmountPaid	Payment_Status	Payment_Method
10	✓	5/1/2025	2	5/5/2025	4	6	11221.2	Paid	cash
11	✓	5/1/2025	3	5/7/2025	5	7	18752.88	Paid	Visa
12	✓	5/1/2025	3	5/6/2025	6	13	14723.9	Paid	Cash

Add new Staff

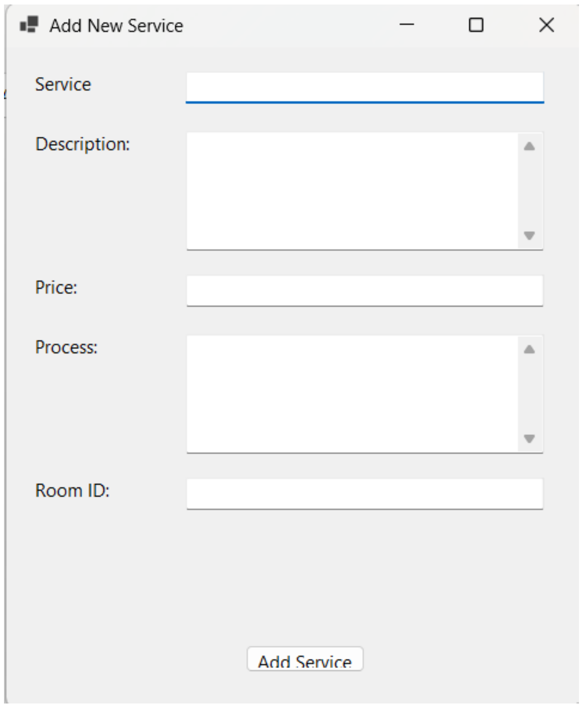


View all Staff

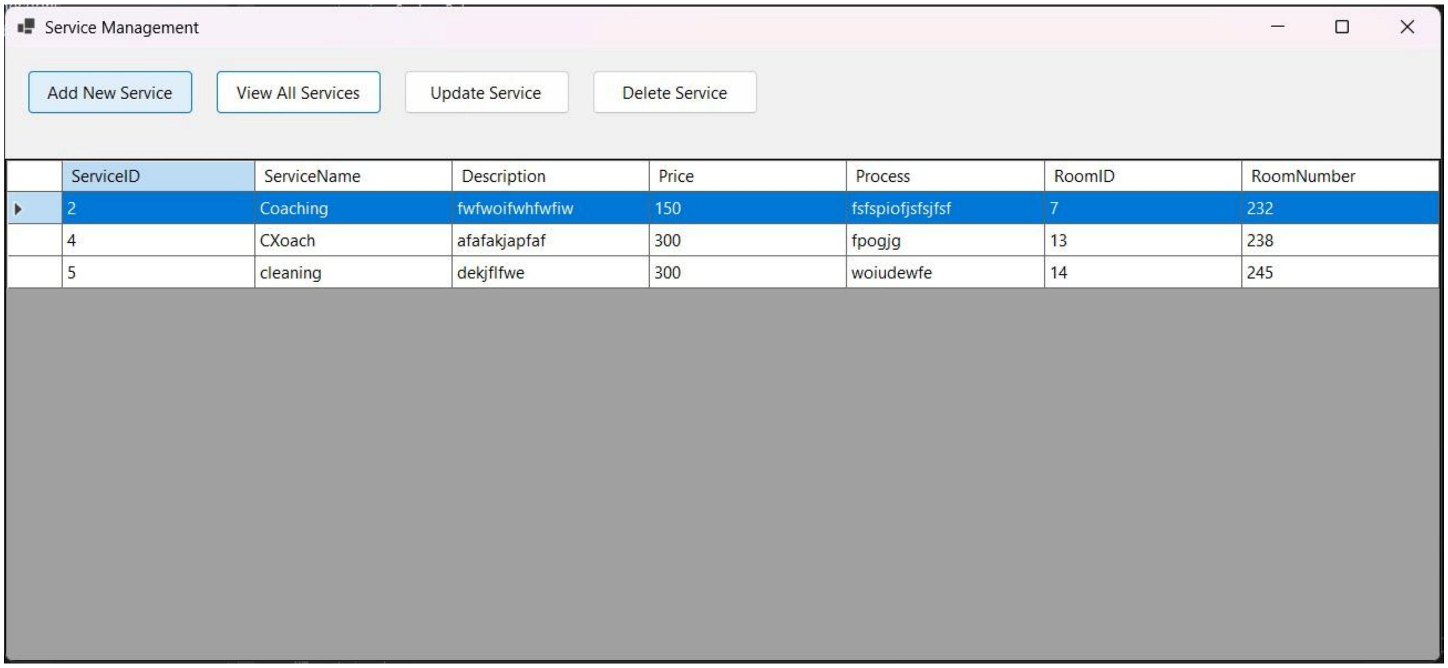


StaffID	First_Name	Last_Name	Email	Role	Phone_Number	ServiceID	ServiceName
31	Patrick	Hany	kato@gmail.com	Chef	01064205021	17	
32	Adam	Medhat	adam@gmail.com	Manager	019933113	18	
33	Ronaldinho	jr.	jail@gmail.com	Football Coach	012938311	4	CXoach
34	Hansi	Flick	barca@gmail.com	Coach	09192823	4	CXoach

Add new Service

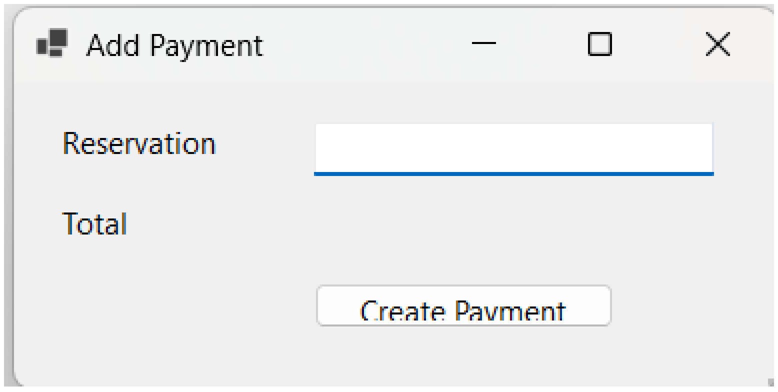


View all Services

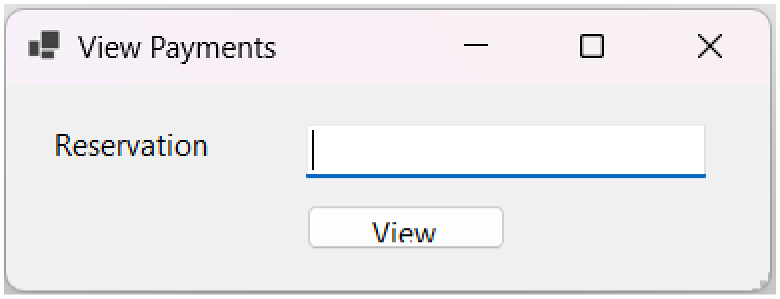


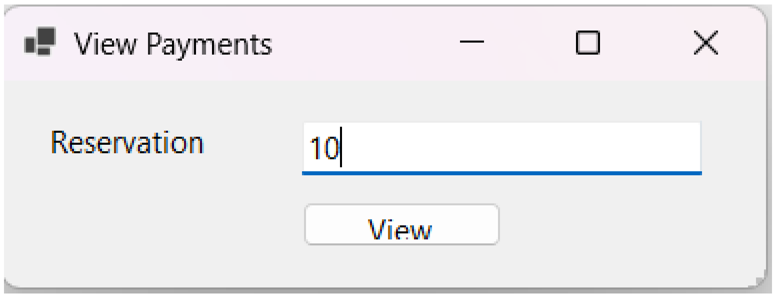
ServiceID	ServiceName	Description	Price	Process	RoomID	RoomNumber
2	Coaching	fwfwoifwhfwfiw	150	fsfspiofjsfsjfsf	7	232
4	CXoach	afafakjapfaf	300	fpogjg	13	238
5	cleaning	dekjflfwe	300	woiudewfe	14	245

Add Payment

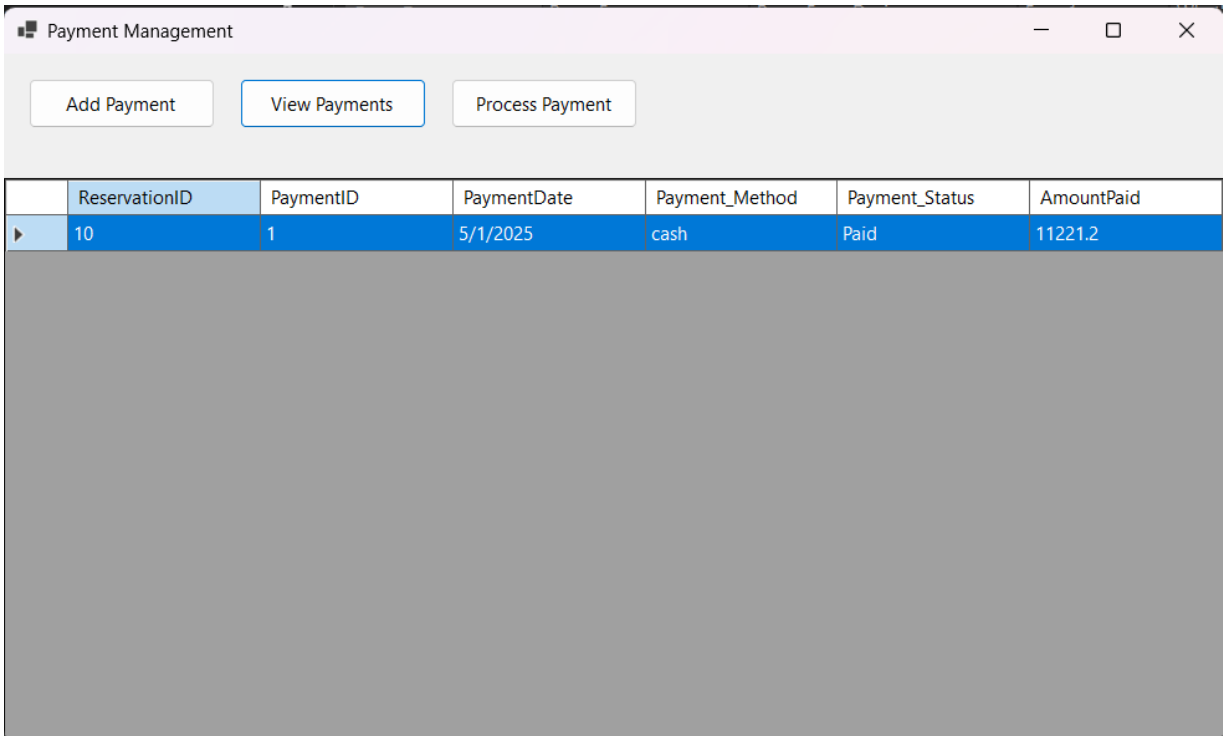


View Payments





Reservation: 10



ReservationID	PaymentID	PaymentDate	Payment_Method	Payment_Status	AmountPaid
10	1	5/1/2025	cash	Paid	11221.2