HOTEL MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted BY:

KARTHIKHA SRE M	**230701143**
KAVIYA MADHIRAJU	**230701148**

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI-602105

NOVEMBER 2024

BONAFIDE CERTIFICATE

Certified that this project report **"HOTEL RESERVATION SYSTEM"** is the Bonafide work of **"KARTHIKHA SRE M-(230701143) & KAVIYA MADHIRAJU-(230701148)"** who carried out the project work under my supervision.

Submitted for the practical examination held on _____________

SIGNATURE
Mrs. K. MAHESMEENA,
Assistant Professor,
Computer Science and Engineering,
Rajalakshmi Engineering College,
(Autonomous),
Thandalam, Chennai - 602 105

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ABSTRACT:

The **Hotel Management System** is a comprehensive software solution designed to simplify and optimize the management of hotel bookings, aiming to enhance both operational efficiency and customer satisfaction. Using SQL as the backend, the system ensures reliable, secure, and scalable data handling to meet the dynamic needs of modern hotel operations.

This system integrates various functionalities to address critical aspects of hotel management, such as room availability tracking, guest information storage, booking management, and payment processing. It offers a user-friendly interface for customers to effortlessly browse available rooms, make reservations, and complete payments. Simultaneously, hotel administrators benefit from a streamlined backend interface to oversee bookings, monitor occupancy rates, and manage financial transactions.

The SQL-powered backend serves as the foundation of this system, providing robust data storage and retrieval capabilities. It ensures the integrity of critical information such as customer details, booking history, and payment records. Advanced query capabilities allow for real-time updates on room availability and the generation of detailed reports to analyze business trends and performance metrics. Additionally, the system prioritizes data security through features such as authentication protocols and data encryption, safeguarding sensitive customer information.

By combining an intuitive interface with the power of SQL-based data management, this project seeks to deliver a reliable and scalable solution tailored to hotels of varying sizes. It aims to improve resource management while offering customers a seamless and efficient reservation experience.

TABLE OF CONTENTS

INTRODUCTION

1.1 OVERVIEW

The **Hotel Management System** is a system that streamlines hotel booking and management processes, providing an efficient, user-friendly platform for both customers and administrators. The system offers features such as room availability checks, booking management, customer data storage, and payment handling. Its objective is to optimize hotel operations, minimize manual errors, and enhance customer satisfaction.

1.2 TECHNOLOGY STACK

- **JavaFX:** Creates an interactive and user-friendly interface.
- **MySQL:** Stores game data and player records.
- **JDBC (Java Database Connectivity):** Facilitates communication between the Java application and the database.

1.3 PROJECT SCOPE

The Hotel Reservation System simplifies hotel management and enhances user experience:

- **User-Friendly Interface:** Easy navigation for browsing rooms and making bookings.
- **Data Management:** Secure storage of customer and booking details using SQL.
- **Real-Time Operations:** Instant availability checks and automated booking confirmations.
- **Scalability:** Supports hotels of all sizes with multi-branch management and analytics.

1.4 FEATURES

- **Room Booking:** Customers can browse available rooms and make reservations in real-time.
- **Admin Dashboard:** Administrators can manage bookings and track room availability.
- **Payment Integration:** Secure processing of payments for bookings.
- **Booking History:** Customers can view their past reservations and details.

SYSTEM REQUIREMENTS

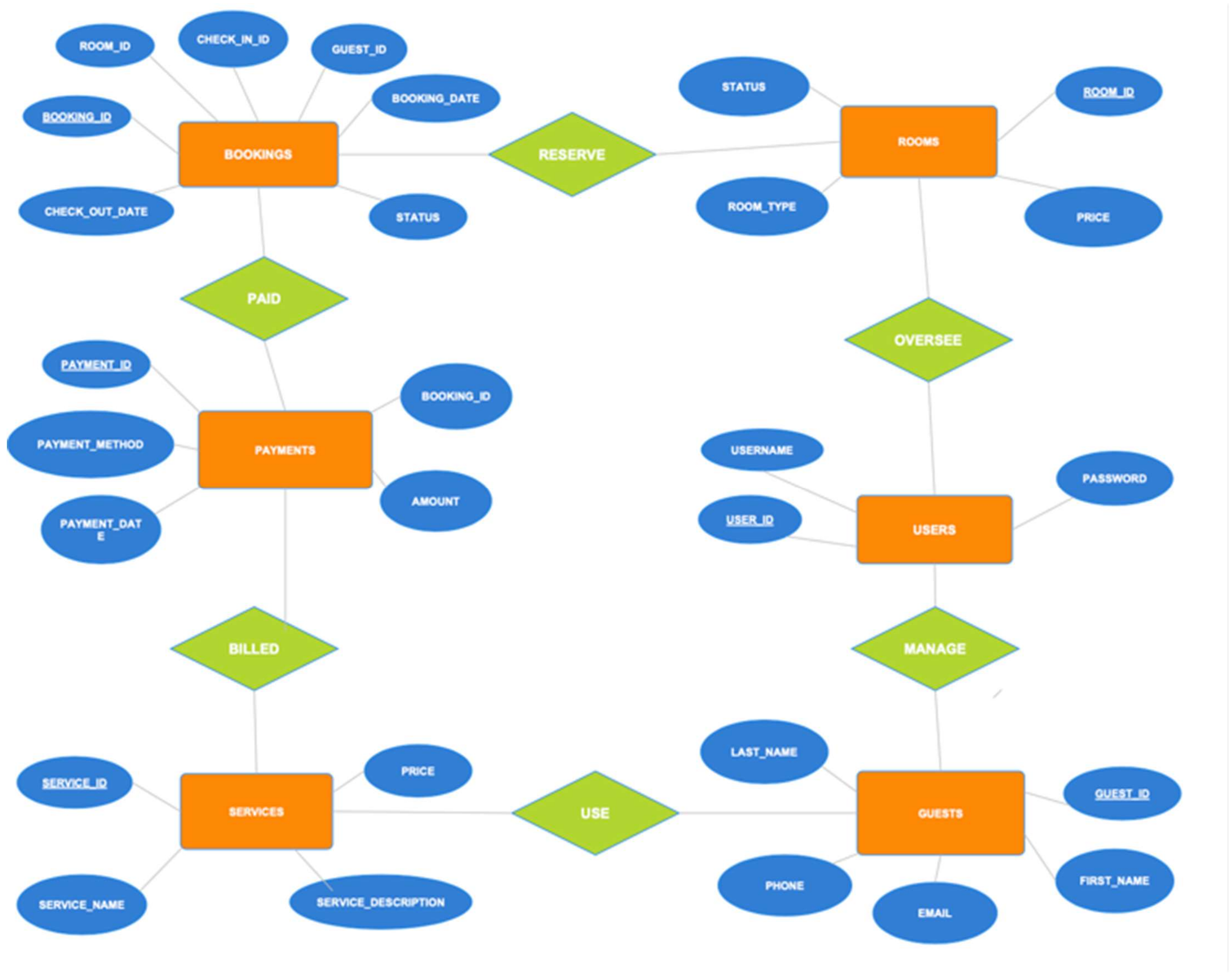
2.1 HARDWARE

- Processor: Intel i5 or equivalent
- Memory: Minimum 4GB RAM
- Hard Disk: 500 GB of free space
- Graphics Card: Capable of supporting JavaFX graphics

2.2 SOFTWARE

- Programming Language: Java
- GUI Library: JavaFX
- Database: MySQL
- Database Connectivity: JDBC (Java Database Connectivity)
- Operating System: Windows 10 or later
- IDE (Integrated Development Environment): Visual Studio Code

ER DIAGRAM


ROOM_ID
CHECK_IN_ID
GUEST_ID
BOOKING_DATE
BOOKING_ID
BOOKINGS
CHECK_OUT_DATE
STATUS
RESERVE
STATUS
ROOMS
ROOM_ID
ROOM_TYPE
PRICE
PAID
OVERSEE
PAYMENT_ID
BOOKING_ID
PAYMENT_METHOD
PAYMENTS
AMOUNT
PAYMENT_DATE
USERNAME
USER_ID
USERS
PASSWORD
BILLED
MANAGE
SERVICE_ID
SERVICES
PRICE
SERVICE_NAME
SERVICE_DESCRIPTION
USE
LAST_NAME
GUESTS
GUEST_ID
PHONE
EMAIL
FIRST_NAME

SAMPLE CODE

4.1 CONTROLLER

AddRoomControleer.java:

```
package com.hotel.hotelmanagement;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import java.net.URL;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.ResourceBundle;
import static com.hotel.hotelmanagement.RoomController.roomList;
import static com.hotel.hotelmanagement.RoomController.rooms;
public class AddRoomController implements Initializable {
  @FXML
  private Button add;
  private TextField number;
  private TextField price;
  private TextField type;
  private Connection connection;
  private DBConnection dbConnection;
  private PreparedStatement pst;
  @Override
  public void initialize(URL url, ResourceBundle resourceBundle) {
    dbConnection = new DBConnection();
    connection = dbConnection.getConnection();
  }
  public void handleAddAction(javafx.event.ActionEvent actionEvent) {
    String query = "INSERT INTO rooms (roomNumber, roomType, price) VALUES (?,?,?)";
    try {
      pst = connection.prepareStatement(query);
      pst.setString(1, number.getText());
```

```
            pst.setString(2, type.getText());
            pst.setString(3, price.getText());
            roomList.add(new Room(Integer.parseInt(number.getText()), Integer.parseInt(price.getText()),
type.getText(), "Not Booked"));
            rooms.add(new Room(Integer.parseInt(number.getText()), Integer.parseInt(price.getText()), type.getText(),
"Not Booked"));
            pst.executeUpdate();
        } catch (SQLException e) {
            e.printStackTrace();        }    }}
```

BillController.java:

```
public class BillController implements Initializable {
    @FXML
    private TableColumn<Bill, String> Amount;
     private TableColumn<Bill, String> Date;
    private TableColumn<Bill, String> billID;
    private TableColumn<Bill, String> cusIDNumber;
    private TableColumn<Bill, String> customerName;
    private TableColumn<Bill, String> roomNumber;
    private TableView<Bill> billTable;
    private TextField search;
    private Connection connection;
    private DBConnection dbConnection;
    private PreparedStatement pst;
    public static final ObservableList<Bill> bills = FXCollections.observableArrayList();
    public static final List<Bill> billList = new ArrayList<>();
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        dbConnection = new DBConnection();
        connection = dbConnection.getConnection();
        roomNumber.setCellValueFactory(new PropertyValueFactory<>("roomNumber"));
        cusIDNumber.setCellValueFactory(new PropertyValueFactory<>("customerID"));
        billID.setCellValueFactory(new PropertyValueFactory<>("billID"));
        Amount.setCellValueFactory(new PropertyValueFactory<>("amount"));
        Date.setCellValueFactory(new PropertyValueFactory<>("date"));
        customerName.setCellValueFactory(new PropertyValueFactory<>("customerName"));
        try {
            initBillList();
        } catch (IOException e) {
            e.printStackTrace();
        }
        billTable.setItems(bills);    }
    public void initBillList() throws IOException {
        billList.clear();
        bills.clear();
        String query = "SELECT b.*, res.roomNumber, res.customerIDNumber, c.customerName FROM bills b\n" +
                "INNER JOIN reservations res ON b.reservationID = res.reservationID\n" +
                "INNER JOIN customers c ON res.customerIDNumber = c.customerIDNumber";
        try {
            pst = connection.prepareStatement(query);
            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                int room_number = Integer.parseInt(rs.getString("roomNumber"));
                int cus_number = Integer.parseInt(rs.getString("customerIDNumber"));
                int bill_id = Integer.parseInt(rs.getString("billID"));
                String date = rs.getString("billDate");
                String cus_name = rs.getString("customerName");
                int bill_amount = Integer.parseInt(rs.getString("billAmount"));
                billList.add(new Bill(bill_id, cus_name, cus_number, date, bill_amount, room_number));
                bills.add(new Bill(bill_id, cus_name, cus_number, date, bill_amount, room_number));
            }
        } catch (SQLException e) {
            e.printStackTrace();        }    }
    private void Search(ObservableList<Bill> bills, String s) {
        bills.clear();
        for (int i = 0; i < billList.size(); i++) {
            if (billList.get(i).getDate().indexOf(s) == 0) {
                bills.add(billList.get(i));            }        }    }
    public void handleSearchKey(KeyEvent event) {
        if (event.getEventType() == KeyEvent.KEY_RELEASED) {
            String s = search.getText();
            Search(bills, s);        }    }
    public void clickBill(MouseEvent event) throws IOException {
        if (event.getClickCount() == 2) {
            if (billTable.getSelectionModel().getSelectedItem() != null) {
                String path = "C:\\Users\\Mr.Cuong\\IdeaProjects\\HotelManagement\\res\\";
               Bill selectedBill = billTable.getSelectionModel().getSelectedItem();
                File file = new File(path + "bill" + selectedBill.getBillID() + ".pdf");
                if (file.toString().endsWith(".pdf"))
                    Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " + file);
                else {
                    Desktop desktop = Desktop.getDesktop();
                    desktop.open(file);                }            }        }    }}
import static com.hotel.hotelmanagement.RoomController.roomList;
import static com.hotel.hotelmanagement.RoomController.rooms;
public class BillInfoController implements Initializable {
    public static int selectedResID;
    public static Reservation selectedReservation;
    @FXML
    private TextField Amount;
    private Button print;
    private TextField customerIDNumber;
    private TextField customerName;
    private TextField roomNumber;
    private Connection connection;
    private DBConnection dbConnection;
    private PreparedStatement pst;
    public static void setSelectedReservationID(int selectedReservationID) {
        selectedResID = selectedReservationID;}
    public static void setSelectedReservation(Reservation reservation) {
        selectedReservation = reservation;
    } @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        dbConnection = new DBConnection();
        connection = dbConnection.getConnection();
        if (selectedResID != 0) {
            String query = "SELECT res.reservationID, res.roomNumber, c.customerIDNumber, c.customerName,
(r.price * DATEDIFF(res.checkOutDate, res.checkInDate)) AS totalPrice FROM customers c\n" +
                    "INNER JOIN reservations res ON c.customerIDNumber = res.customerIDNumber\n" +
                    "INNER JOIN rooms r ON r.roomNumber = res.roomNumber\n" +
                    "WHERE res.reservationID=?";
            try {
                pst = connection.prepareStatement(query);
                pst.setString(1, Integer.toString(selectedResID));
                ResultSet rs = pst.executeQuery();
                while (rs.next()) {
                    roomNumber.setText(rs.getString("roomNumber"));
                    customerIDNumber.setText(rs.getString("customerIDNumber"));
                    customerName.setText(rs.getString("customerName"));
                    Amount.setText(rs.getString("totalPrice"));
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
            roomNumber.setEditable(false);
            customerIDNumber.setEditable(false);
            customerName.setEditable(false);
            Amount.setEditable(false);        }    }
    public void handlePrintAction(javafx.event.ActionEvent actionEvent) throws IOException {
        String id = "";
        String insertBills = "INSERT INTO bills(reservationID, billDate, billAmount) VALUES (?, ?, ?)";
        String updateRoom = "UPDATE rooms SET status=\"Not Booked\" WHERE roomNumber=?";
        String updateReservation = "UPDATE reservations SET status=\"Checked Out\" WHERE reservationID=?";
        String selectBill = "SELECT billID FROM bills WHERE reservationID=?";
        if (!selectedReservation.getStatus().equals("Checked Out")) {
            try {
               pst = connection.prepareStatement(insertBills);
            } catch (SQLException e) {
                e.printStackTrace();
            }
            try {
                pst.setString(1, String.valueOf(selectedReservation.getResID()));
                pst.setString(2, selectedReservation.getCheckOutDate());
                pst.setString(3, String.valueOf(selectedReservation.getTotalPrice()));
                pst.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            try {
                pst = connection.prepareStatement(updateRoom);
                pst.setString(1, String.valueOf(selectedReservation.getRoomNumber()));
                pst.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace();
            }
            try {
                pst = connection.prepareStatement(updateReservation);
                pst.setString(1, String.valueOf(selectedReservation.getResID()));
                pst.executeUpdate();
            } catch (SQLException e) {
                e.printStackTrace();            }        }
        try {
            pst = connection.prepareStatement(selectBill);
            pst.setString(1, String.valueOf(selectedReservation.getResID()));
            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                id = rs.getString("billID");
            }
        } catch (SQLException e) {
            e.printStackTrace();        }
        System.out.println(id);
```

```
        createBill(id);    }
    private void createBill(String id) throws IOException {
        String billID = "";
        String customerName = "";
        String customerIDNumber = "";
        String customerPhoneNo = "";
        String roomNumber = "";
        String roomType = "";
        String priceRoom = "";
        String checkIn = "";
        String checkOut = "";
        String totalDay = "";
        String totalPrice = "";
        String path = "C:\\Users\\Mr.Cuong\\IdeaProjects\\HotelManagement\\res\\";
        String billQuery = "SELECT b.billID, c.customerIDNumber, c.customerName, c.customerPhoneNo,
r.roomNumber, r.roomType, r.price, res.checkInDate, res.checkOutDate, (r.price * DATEDIFF(res.checkOutDate,
res.checkInDate)) AS totalPrice, DATEDIFF(res.checkOutDate, res.checkInDate) AS totalDay FROM bills b\n" +
                "INNER JOIN reservations res ON b.reservationID = res.reservationID\n" +
                "INNER JOIN rooms r ON r.roomNumber = res.roomNumber\n" +
                "INNER JOIN customers c ON c.customerIDNumber = res.customerIDNumber\n" +
                "WHERE b.billID=?";
        try {
            pst = connection.prepareStatement(billQuery);
            pst.setString(1, id);
            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                billID = rs.getString("billID");
                customerName = rs.getString("customerName");
                customerIDNumber = rs.getString("customerIDNumber");
                customerPhoneNo = rs.getString("customerPhoneNo");
                roomNumber = rs.getString("roomNumber");
                roomType = rs.getString("roomType");
                priceRoom = rs.getString("price");
                checkIn = rs.getString("checkInDate");
                checkOut = rs.getString("checkOutDate");
                totalDay = rs.getString("totalDay");
                totalPrice = rs.getString("totalPrice");
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();        }
        Document doc = new Document();
        try {
            PdfWriter.getInstance(doc, new FileOutputStream(path + "bill" + id + ".pdf"));
            doc.open();
            Paragraph paragraph1 = new Paragraph("Bill ID: " + billID + "\nCustomer Details:\nName: " +
customerName + "\nID Number: " + customerIDNumber +
                    "\nMobile Number: " + customerPhoneNo + "\n");
            doc.add(paragraph1);
            Paragraph paragraph2 = new Paragraph("\nRoom Details:\nRoom Number: " + roomNumber + "\nRoom
Type: " + roomType +
                    "\nPrice Per Day " + priceRoom + "\n" + "\n");
            doc.add(paragraph2);
            PdfPTable table = new PdfPTable(4);
            table.addCell("Check In Date: " + checkIn);
            table.addCell("Check Out Date: " + checkOut);
            table.addCell("Number of Days Stay: " + totalDay);
            table.addCell("Total Amount Paid: " + totalPrice);
            doc.add(table);
        } catch (Exception e) {
            e.printStackTrace();        }
        doc.close();
        File file = new File(path + "bill" + id + ".pdf");
        if (file.toString().endsWith(".pdf"))
            Runtime.getRuntime().exec("rundll32 url.dll,FileProtocolHandler " + file);
        else {
            Desktop desktop = Desktop.getDesktop();
            desktop.open(file);        }    }}
public class CheckInController implements Initializable {
```

```
@FXML
private Label amount;
private Label days;
private Label price;
private TextField cEmail;
private TextField cGender;
private TextField cName;
private TextField cNationality;
private TextField cNumber;
private TextField cPhone;
private Button submit;
private DatePicker inDate;
private DatePicker outDate;
private ComboBox<String> rNo;
private ComboBox<String> rType;
private Connection connection;
private DBConnection dbConnection;
private PreparedStatement pst;
@Override
public void initialize(URL url, ResourceBundle resourceBundle) {
    dbConnection = new DBConnection();
    connection = dbConnection.getConnection();
    insertRoomType();
}
private void insertRoomType() {
    rType.getItems().removeAll(rType.getItems());
    String query = "SELECT DISTINCT roomType FROM rooms";
    try {
        pst = connection.prepareStatement(query);
        ResultSet rs = pst.executeQuery();
        while (rs.next()) {
            String room_type = rs.getString("roomType");
            rType.getItems().add(room_type);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
private void insertRoomNo() {
    rNo.getItems().removeAll(rNo.getItems());
    String type = rType.getSelectionModel().getSelectedItem();
    String query = "SELECT roomNumber FROM rooms WHERE roomType=? AND status='Not Booked'";
    try {
        pst = connection.prepareStatement(query);
        pst.setString(1, type);
        ResultSet rs = pst.executeQuery();
        while (rs.next()) {
            String room_no = rs.getString("roomNumber");
            rNo.getItems().add(room_no);
        }
    } catch (SQLException e) {
        e.printStackTrace();        }    }
public void handleSelectRoomType(javafx.event.ActionEvent actionEvent) {
    if (!rType.getSelectionModel().getSelectedItem().equals("")) {
        insertRoomNo();        }    }
public void handleSelectRoomNumber(javafx.event.ActionEvent actionEvent) {
    String priceVal = "Price: ";
    String no = rNo.getSelectionModel().getSelectedItem();
    String priceQuery = "SELECT price FROM rooms WHERE roomNumber=?";
    try {
        pst = connection.prepareStatement(priceQuery);
        pst.setString(1, no);
        ResultSet rs = pst.executeQuery();
        while (rs.next()) {
            priceVal = priceVal + rs.getString("price");
        }
    } catch (SQLException e) {
        e.printStackTrace();        }
        price.setText(priceVal);    }


    public void handleCheckInPick(javafx.event.ActionEvent actionEvent) {
        String date = inDate.getValue().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));    }
    public void handleCheckOutPick(javafx.event.ActionEvent actionEvent) {
        int x = outDate.getValue().compareTo(inDate.getValue());
        days.setText("Total days: " + x);
        int p = Integer.parseInt(price.getText().replace("Price: ", ""));
        amount.setText("Total Amount: " + (p * x));    }
    public void handleSubmitAction(javafx.event.ActionEvent actionEvent) {
        String name = cName.getText();
        String email = cEmail.getText();
        String gender = cGender.getText();
        String nationality = cNationality.getText();
        String number = cNumber.getText();
        String phone = cPhone.getText();
        String roomNo = rNo.getSelectionModel().getSelectedItem();
        String checkIn = inDate.getValue().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));
        String checkOut = outDate.getValue().format(DateTimeFormatter.ofPattern("yyyy-MM-dd"));
        if (name.equals("") || email.equals("") || gender.equals("") || nationality.equals("")
                || number.equals("") || phone.equals("") || roomNo.equals("") || checkIn.equals("") || checkOut.equals("")) {
            OptionPane("Every Field is required", "Error Message");
        } else {
            String insertCustomer = "INSERT INTO customers(customerIDNumber, customerName,
customerNationality, customerGender, customerPhoneNo, customerEmail)"
                    + "VALUES (?, ?, ?, ?, ?, ?)";
            String insertReservation = "INSERT INTO reservations(customerIDNumber, roomNumber, checkInDate,
checkOutDate) VALUES (?, ?, ?, ?)";
            String updateRoom = "UPDATE rooms SET status=\"Booked\" WHERE roomNumber=?";
            try {
                pst = connection.prepareStatement(insertCustomer);
            } catch (SQLException e) {
                e.printStackTrace();
            }
```

```
try {
    pst.setString(1, number);
    pst.setString(2, name);
    pst.setString(3, nationality);
    pst.setString(4, gender);
    pst.setString(5, phone);
    pst.setString(6, email);
    pst.executeUpdate();
} catch (SQLException e) {
    e.printStackTrace();
}
try {
    pst = connection.prepareStatement(insertReservation);
} catch (SQLException e) {
    e.printStackTrace();
}
try {
    pst.setString(1, number);
    pst.setString(2, roomNo);
    pst.setString(3, checkIn);
    pst.setString(4, checkOut);
    pst.executeUpdate();
} catch (SQLException e) {
    e.printStackTrace();            }
try {
    pst = connection.prepareStatement(updateRoom);
} catch (SQLException e) {
    e.printStackTrace();            }
try {
    pst.setString(1, roomNo);
    pst.executeUpdate();
} catch (SQLException e) {
    e.printStackTrace();
}
```

```
            OptionPane("Check In Successful", "Message");
        }
    }

    private void OptionPane(String message, String title) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.initStyle(StageStyle.UTILITY);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}

public class CheckOutController implements Initializable {
    @FXML
    private TableColumn<Reservation, String> checkIn;
    private TableColumn<Reservation, String> checkOut;
    private TableColumn<Reservation, String> customerName;
    private TableColumn<Reservation, String> roomNumber;
    private TableView<Reservation> roomTable;
    private TextField search;
     private Button today;
    private TableColumn<Reservation, String> totalDays;
    private TableColumn<Reservation, String> totalPrice;
    private TableColumn<?, ?> status;
    private Button unspecified;
    private ComboBox<String> sort;
    private Connection connection;
    private DBConnection dbConnection;
    private PreparedStatement pst;
    public static final ObservableList<Reservation> reservations  = FXCollections.observableArrayList();
    public static final List<Reservation> reservationList = new ArrayList<>();
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        dbConnection = new DBConnection();
        connection = dbConnection.getConnection();
        sort.getItems().removeAll(sort.getItems());
        sort.getItems().addAll("Today", "Checked In", "Checked Out");
        roomNumber.setCellValueFactory(new PropertyValueFactory<>("roomNumber"));
        customerName.setCellValueFactory(new PropertyValueFactory<>("customerName"));
        checkIn.setCellValueFactory(new PropertyValueFactory<>("checkInDate"));
        checkOut.setCellValueFactory(new PropertyValueFactory<>("checkOutDate"));
        totalDays.setCellValueFactory(new PropertyValueFactory<>("totalDays"));
        totalPrice.setCellValueFactory(new PropertyValueFactory<>("totalPrice"));
        status.setCellValueFactory(new PropertyValueFactory<>("status"));
        try {
            initReservationList();
        } catch (IOException e) {
            e.printStackTrace();
        }
        roomTable.setItems(reservations);
    }
    public void initReservationList() throws IOException {
        reservationList.clear();
        reservations.clear();
        String query = "SELECT res.status, res.reservationID, res.roomNumber, c.customerName, res.checkInDate,
res.checkOutDate, DATEDIFF(res.checkOutDate, res.checkInDate) AS totalDays, (r.price *
DATEDIFF(res.checkOutDate, res.checkInDate)) AS totalPrice FROM customers c\n" +
                "INNER JOIN reservations res ON c.customerIDNumber = res.customerIDNumber\n" +
                "INNER JOIN rooms r ON r.roomNumber = res.roomNumber\n";
        try {
            pst = connection.prepareStatement(query);
            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                int res_id = Integer.parseInt(rs.getString("reservationID"));
                int room_no = Integer.parseInt(rs.getString("roomNumber"));
                String cus_name = rs.getString("customerName");
                String check_in = rs.getString("checkInDate");
                String check_out = rs.getString("checkOutDate");
                int total_price = Integer.parseInt(rs.getString("totalPrice"));
                int total_days = Integer.parseInt(rs.getString("totalDays"));
                String res_status = rs.getString("status");
                reservationList.add(new Reservation(res_id, room_no, cus_name, check_in, check_out, total_days,
total_price, res_status));
                reservations.add(new Reservation(res_id, room_no, cus_name, check_in, check_out, total_days,
total_price, res_status));
            }


        } catch (SQLException e) {
            e.printStackTrace();
        }
    }


    private void searchByRoomNumber(ObservableList<Reservation> res, String s) {
        res.clear();
        for (int i = 0; i < reservationList.size(); i++) {
            if (Integer.toString(reservationList.get(i).getRoomNumber()).indexOf(s) == 0) {
                res.add(reservationList.get(i));
            }
        }
    }
    public void handleSearchKey(KeyEvent event) {
        if (event.getEventType() == KeyEvent.KEY_RELEASED) {
            String s = search.getText();
            searchByRoomNumber(reservations, s);
        }
    }
    public void handleCheckoutButton(javafx.event.ActionEvent actionEvent) {
    }
    public void updateTable(Reservation x) {
        for (int i = 0; i < reservations.size(); i++) {
```

```
        if (reservations.get(i).equals(x)) {
            reservations.get(i).setStatus("Checked Out");
        }
    }
    roomTable.setItems(reservations);
}
public void clickItem(MouseEvent event) throws IOException {
    if (event.getClickCount() == 2) {
        if (roomTable.getSelectionModel().getSelectedItem() != null) {
            Reservation selectedReservation = roomTable.getSelectionModel().getSelectedItem();
            BillInfoController.setSelectedReservationID(selectedReservation.getResID());
            BillInfoController.setSelectedReservation(selectedReservation);
            Stage add = new Stage();
            Parent root = FXMLLoader.load(getClass().getResource("billinfo.fxml"));
            Scene scene = new Scene(root);
            add.setScene(scene);
            add.show();
        }
    }
}

public void handleComboboxSelection(javafx.event.ActionEvent actionEvent) {
    if (sort.getSelectionModel().getSelectedItem().equals("Today")) {
        reservations.clear();
        for (int i = 0; i < reservationList.size(); i++) {
            if (reservationList.get(i).getCheckOutDate().equals(java.time.LocalDate.now().toString()) &&
            reservationList.get(i).getStatus().equals("Checked In")) {
                reservations.add(reservationList.get(i));
            }
        }
    } else if (sort.getSelectionModel().getSelectedItem().equals("Checked In")) {
        reservations.clear();
        for (int i = 0; i < reservationList.size(); i++) {
            if (reservationList.get(i).getStatus().equals("Checked In")) {
                reservations.add(reservationList.get(i));
            }
        }
    } else if (sort.getSelectionModel().getSelectedItem().equals("Checked Out")) {
        reservations.clear();
        for (int i = 0; i < reservationList.size(); i++) {
            if (reservationList.get(i).getStatus().equals("Checked Out")) {
                reservations.add(reservationList.get(i));
            }
        }
    }
  }
}
public class CustomerController implements Initializable {

  public static int selectedRoomNumber;

  @FXML
  private TextField email;
  private TextField gender;
  private TextField inDate;
  private TextField nationality;
  private TextField outDate;
  private TextField phone;
  private TextField price;
  private Connection connection;
  private DBConnection dbConnection;
  private PreparedStatement pst;
  public static void setSelectedRoomNumber(int selectedRoomNumber) {
    CustomerController.selectedRoomNumber = selectedRoomNumber;
  }
  @Override
  public void initialize(URL url, ResourceBundle resourceBundle) {
    dbConnection = new DBConnection();
```

```
        connection = dbConnection.getConnection();
        if (selectedRoomNumber != 0) {
            String query = "SELECT c.*, res.checkInDate, res.checkOutDate, (r.price * DATEDIFF(res.checkOutDate,
res.checkInDate)) AS Total FROM customers c \n" +
                    "INNER JOIN reservations res ON c.customerIDNumber = res.customerIDNumber\n" +
                    "INNER JOIN rooms r ON r.roomNumber = res.roomNumber\n" +
                    "WHERE r.roomNumber=?";
            try {
                pst = connection.prepareStatement(query);
                pst.setString(1, Integer.toString(selectedRoomNumber));
                ResultSet rs = pst.executeQuery();
                while (rs.next()) {
                    price.setText(rs.getString("Total"));
                    name.setText(rs.getString("customerName"));
                    email.setText(rs.getString("customerEmail"));
                    phone.setText(rs.getString("customerPhoneNo"));
                    gender.setText(rs.getString("customerGender"));
                    nationality.setText(rs.getString("customerNationality"));
                    inDate.setText(rs.getString("checkInDate"));
                    outDate.setText(rs.getString("checkOutDate"));
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
            price.setEditable(false);
            name.setEditable(false);
            email.setEditable(false);
            phone.setEditable(false);
            gender.setEditable(false);
            nationality.setEditable(false);
            inDate.setEditable(false);
            outDate.setEditable(false);
        }
    }
```

```
}
public class DashBoardController implements Initializable {
    @FXML
    private Label avaRoom;
    private Label earning;
    private Label pending;
    private Label totalRoom;
    private Connection connection;
    private DBConnection dbConnection;
    private PreparedStatement pst;
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        dbConnection = new DBConnection();
        connection = dbConnection.getConnection();
        String query = "SELECT COUNT(roomNumber) AS totalRooms, a.totalNotBooked, booked.totalBooked
FROM rooms, " +
                "(SELECT COUNT(roomNumber) AS totalBooked FROM rooms WHERE status = 'Booked') AS booked,
" +
                "(SELECT COUNT(roomNumber) AS totalNotBooked FROM rooms WHERE status = 'Not Booked') AS
a";
        String query2 = "SELECT SUM(b.billAmount) AS totalEarnings, (SELECT SUM((r.price *
DATEDIFF(res.checkOutDate, res.checkInDate))) AS Pending FROM reservations res \n" +
                "INNER JOIN rooms r ON r.roomNumber = res.roomNumber \n" +
                "WHERE res.status = 'Checked In') AS totalPendings FROM bills b \n" +
                "INNER JOIN reservations res ON res.reservationID = b.reservationID;";
        try {
            pst = connection.prepareStatement(query);
            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                totalRoom.setText(rs.getString("totalRooms"));
                bookedRoom.setText(rs.getString("totalBooked"));
                avaRoom.setText(rs.getString("totalNotBooked"));
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
        try {
            pst = connection.prepareStatement(query2);
            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                earning.setText(rs.getString("totalEarnings"));
                pending.setText(rs.getString("totalPendings"));
            }
        } catch (SQLException throwables) {
            throwables.printStackTrace();
        }
    }
}
public class ForgotController implements Initializable {
    @FXML
    private TextField answer;
    private Button login;
    private TextField password;


    private TextField question;
    private Button save;
    private Button search;
    private TextField username;
   private Connection connection;
    private DBConnection dbConnection;
    private PreparedStatement pst;
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        dbConnection = new DBConnection();
        connection = dbConnection.getConnection();
    }

    @FXML
    public void handleLoginButton(javafx.event.ActionEvent actionEvent) throws IOException {
```

```
    login.getScene().getWindow().hide();
    Stage signup = new Stage();
    Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));
    Scene scene = new Scene(root);
    signup.setScene(scene);
    signup.show();
}

@FXML
public void handleSaveAction(javafx.event.ActionEvent actionEvent) {
    int check = 0;
    String username_text = username.getText();
    String password_text = password.getText();
    String question_text = question.getText();
    String answer_text = answer.getText();
    if (username_text.equals("") || password_text.equals("") || question_text.equals("") || answer_text.equals("")) {
        OptionPane("Every Field is required", "Error Message");
    } else {
        String query = "SELECT * FROM users WHERE username=? AND securityQuestion=? AND answer=?";
        try {
            pst = connection.prepareStatement(query);
            pst.setString(1, username_text);
            pst.setString(2, question_text);
            pst.setString(3, answer_text);
            ResultSet rs = pst.executeQuery();
            if (rs.next()) {
                check = 1;
                String update = "UPDATE users set password=? WHERE username=?";
                pst = connection.prepareStatement(update);
                pst.setString(2, username_text);
                pst.setString(1, password_text);
                OptionPane("Password Set Successfully", "Message");
                pst.executeUpdate();
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    if (check == 0) {
        OptionPane("Wrong Answer", "Error Message");
    }
}

@FXML
public void handleSearchAction(javafx.event.ActionEvent actionEvent) {
    int check = 0;
    String query = "SELECT * FROM users WHERE username=?";
    try {
        pst = connection.prepareStatement(query);
        pst.setString(1, username.getText());
        ResultSet rs = pst.executeQuery();
        if (rs.next()) {
            check = 1;
            question.setEditable(false);
            question.setText(rs.getString(6));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    if (check == 0) {
        OptionPane("Incorrect Username", "Error Message");
    }
}

private void OptionPane(String message, String title) {
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.initStyle(StageStyle.UTILITY);
    alert.setTitle(title);
```

```
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}

public class HomePageController implements Initializable {

    @FXML
    private Label adminName;
    private Button bill;
    private Button dash;
    private Button checkin;
    private Button checkout;
    private AnchorPane holdPane;
    private Button room;
    private AnchorPane Pane;
    public static String name;
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        adminName.setText(name);
    }

    private void setNode(Node node) {
        holdPane.getChildren().clear();
        holdPane.getChildren().add((Node) node);
        FadeTransition ft = new FadeTransition(Duration.millis(1000));
        ft.setNode(node);
        ft.setFromValue(0.1);
        ft.setToValue(1);
        ft.setCycleCount(1);
        ft.setAutoReverse(false);
        ft.play();
```

```
}
public void createRoom(javafx.event.ActionEvent actionEvent) {
    try {
        checkin.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        checkout.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        bill.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        dash.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        Pane = FXMLLoader.load(getClass().getResource("room.fxml"));
        setNode(Pane);
        room.setStyle("-fx-background-color:  #2D3347; -fx-text-fill: #ffffff");
    } catch (IOException e) {
        e.printStackTrace();
    }
}
public void createCheckIn(javafx.event.ActionEvent actionEvent) {
    try {
        room.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        checkout.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        bill.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        Pane = FXMLLoader.load(getClass().getResource("checkin.fxml"));
        dash.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        setNode(Pane);
        checkin.setStyle("-fx-background-color:  #2D3347; -fx-text-fill: #ffffff");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void createCheckOut(javafx.event.ActionEvent actionEvent) {
    try {
        room.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        checkin.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        bill.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        dash.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        Pane = FXMLLoader.load(getClass().getResource("checkout.fxml"));
        setNode(Pane);
        checkout.setStyle("-fx-background-color:  #2D3347; -fx-text-fill: #ffffff");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void createCustomerBill(javafx.event.ActionEvent actionEvent) {
    try {
        room.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        checkin.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        checkout.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        dash.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        Pane = FXMLLoader.load(getClass().getResource("bill.fxml"));
        setNode(Pane);
        bill.setStyle("-fx-background-color:  #2D3347; -fx-text-fill: #ffffff");
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public void createDash(javafx.event.ActionEvent actionEvent) {
    try {
        room.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        checkin.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        checkout.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        bill.setStyle("-fx-background-color:  #ffffff; -fx-text-fill: #000000");
        Pane = FXMLLoader.load(getClass().getResource("dashboard.fxml"));
        setNode(Pane);
        dash.setStyle("-fx-background-color:  #2D3347; -fx-text-fill: #ffffff");
    } catch (IOException e) {
        e.printStackTrace();
    }
    }

    public void handleLogout(MouseEvent event) throws IOException {
        bill.getScene().getWindow().hide();
        Stage login = new Stage();
        Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));
        Scene scene = new Scene(root);
        login.setScene(scene);
        login.show();
    }
}
public class LoginController implements Initializable {

    @FXML
    private Button forgotpassword;
    private Button login;
    private PasswordField password;
    private TextField username;
    private Connection connection;
    private DBConnection dbConnection;
    private PreparedStatement pst;
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        dbConnection = new DBConnection();
    }

    @FXML
    public void handleSignupButton(javafx.event.ActionEvent actionEvent) throws IOException {
        login.getScene().getWindow().hide();
        Stage signup = new Stage();
        Parent root = FXMLLoader.load(getClass().getResource("signup.fxml"));
        Scene scene = new Scene(root);
        signup.setScene(scene);
        signup.show();
```

```
}

@FXML
public void handleLoginAction(javafx.event.ActionEvent actionEvent) throws IOException {
    connection = dbConnection.getConnection();
    String query = "SELECT * FROM users WHERE username=? AND password=?";
    try {
        pst = connection.prepareStatement(query);
        pst.setString(1, username.getText());
        pst.setString(2, password.getText());
        ResultSet rs = pst.executeQuery();
        int count = 0;
        while (rs.next()) {
            HomePageController.name = rs.getString("name");
            count = 1;
        }
        if (count == 1) {
            login.getScene().getWindow().hide();
            Stage signup = new Stage();
            Parent root = FXMLLoader.load(getClass().getResource("homepage.fxml"));
            Scene scene = new Scene(root);
            signup.setScene(scene);
            signup.show();
        } else {
            OptionPane("Username or Password is not Correct", "Error Message");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

@FXML
public void handleForgotAction(javafx.event.ActionEvent actionEvent) throws IOException {
    login.getScene().getWindow().hide();
```

```
        Stage signup = new Stage();
        Parent root = FXMLLoader.load(getClass().getResource("forgotpassword.fxml"));
        Scene scene = new Scene(root);
        signup.setScene(scene);
        signup.show();
    }

    private void OptionPane(String message, String title) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.initStyle(StageStyle.UTILITY);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
        alert.showAndWait();
    }
}
public class RoomController implements Initializable {

    @FXML
    private TableColumn<Room, String> price;
    private TableColumn<Room, String> roomNumber;
    private TableView<Room> roomTable;
    private TableColumn<Room, String> roomType;
    private TextField search;
    private TableColumn<Room, String> status;
    private Connection connection;

    private DBConnection dbConnection;
    private PreparedStatement pst;
    public static final ObservableList<Room> rooms = FXCollections.observableArrayList();
    public static final List<Room> roomList = new ArrayList<>();
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        dbConnection = new DBConnection();
        connection = dbConnection.getConnection();
        roomNumber.setCellValueFactory(new PropertyValueFactory<>("number"));
        roomType.setCellValueFactory(new PropertyValueFactory<>("type"));
        price.setCellValueFactory(new PropertyValueFactory<>("price"));
        status.setCellValueFactory(new PropertyValueFactory<Room, String>("status"));
        try {
            initRoomList();
        } catch (IOException e) {
            e.printStackTrace();
        }
        roomTable.setItems(rooms);
    }

    public void handleAddAction(javafx.event.ActionEvent actionEvent) throws IOException {
        Stage add = new Stage();
        Parent root = FXMLLoader.load(getClass().getResource("addroom.fxml"));
        Scene scene = new Scene(root);
        add.setScene(scene);
        add.show();
    }

    public void handleViewAction(javafx.event.ActionEvent actionEvent) throws IOException {


    }
    public void clickItem(MouseEvent event) throws IOException {
        if (event.getClickCount() == 2) {
            if (roomTable.getSelectionModel().getSelectedItem() != null) {
                if (roomTable.getSelectionModel().getSelectedItem().getStatus().equals("Booked")) {
                    CustomerController.setSelectedRoomNumber(roomTable.getSelectionModel().getSelectedItem().getNu
mber());
                    Stage add = new Stage();
                    Parent root = FXMLLoader.load(getClass().getResource("customerinfo.fxml"));
                    Scene scene = new Scene(root);
                    add.setScene(scene);
                    add.show();
                }
            }
        }
    }
    public void initRoomList() throws IOException {
        roomList.clear();
        rooms.clear();
        String query = "SELECT * FROM rooms";
        try {
            pst = connection.prepareStatement(query);
            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                int room_price = Integer.parseInt(rs.getString("price"));
                String room_type = rs.getString("roomType");
                String room_status = rs.getString("status");
                int room_num = Integer.parseInt(rs.getString("roomNumber"));
                roomList.add(new Room(room_num, room_price, room_type, room_status));
                rooms.add(new Room(room_num, room_price, room_type, room_status));
            }

        } catch (SQLException e) {
            e.printStackTrace();
        }   }
    private void Search(ObservableList<Room> rooms, String s) {
        rooms.clear();
        for (int i = 0; i < roomList.size(); i++) {
            if (Integer.toString(roomList.get(i).getNumber()).indexOf(s) == 0) {
                rooms.add(roomList.get(i));
            }       }    }
    public void handleSearchKey(KeyEvent event) {
        if (event.getEventType() == KeyEvent.KEY_RELEASED) {
            String s = search.getText();
            Search(rooms, s);
        }    }}
public class RoomController implements Initializable {
    @FXML
    private TableColumn<Room, String> price;
    private TableColumn<Room, String> roomNumber;
    private TableView<Room> roomTable;
    private TableColumn<Room, String> roomType;
    private TextField search;
   private TableColumn<Room, String> status;
    private Connection connection;
    private DBConnection dbConnection;
    private PreparedStatement pst;
    public static final ObservableList<Room> rooms = FXCollections.observableArrayList();
    public static final List<Room> roomList = new ArrayList<>();
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        dbConnection = new DBConnection();
        connection = dbConnection.getConnection();
        roomNumber.setCellValueFactory(new PropertyValueFactory<>("number"));
        roomType.setCellValueFactory(new PropertyValueFactory<>("type"));
        price.setCellValueFactory(new PropertyValueFactory<>("price"));
        status.setCellValueFactory(new PropertyValueFactory<Room, String>("status"));
        try {
            initRoomList();
        } catch (IOException e) {
            e.printStackTrace();        }
        roomTable.setItems(rooms);    }
    public void handleAddAction(javafx.event.ActionEvent actionEvent) throws IOException {
        Stage add = new Stage();
        Parent root = FXMLLoader.load(getClass().getResource("addroom.fxml"));
        Scene scene = new Scene(root);
        add.setScene(scene);
        add.show();
    }
    public void handleViewAction(javafx.event.ActionEvent actionEvent) throws IOException {    }
    public void clickItem(MouseEvent event) throws IOException {
        if (event.getClickCount() == 2) {
            if (roomTable.getSelectionModel().getSelectedItem() != null) {
                if (roomTable.getSelectionModel().getSelectedItem().getStatus().equals("Booked")) {
                    CustomerController.setSelectedRoomNumber(roomTable.getSelectionModel().getSelectedItem().getNu
mber());
                    Stage add = new Stage();
                    Parent root = FXMLLoader.load(getClass().getResource("customerinfo.fxml"));
                    Scene scene = new Scene(root);
                    add.setScene(scene);
                    add.show();
                }           }       }    }
    public void initRoomList() throws IOException {
        roomList.clear();
        rooms.clear();
        String query = "SELECT * FROM rooms";
        try {
            pst = connection.prepareStatement(query);
            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                int room_price = Integer.parseInt(rs.getString("price"));
                String room_type = rs.getString("roomType");
                String room_status = rs.getString("status");
                int room_num = Integer.parseInt(rs.getString("roomNumber"));
                roomList.add(new Room(room_num, room_price, room_type, room_status));
                rooms.add(new Room(room_num, room_price, room_type, room_status));
            }

        } catch (SQLException e) {
            e.printStackTrace();       }    }

    private void Search(ObservableList<Room> rooms, String s) {
        rooms.clear();
        for (int i = 0; i < roomList.size(); i++) {
            if (Integer.toString(roomList.get(i).getNumber()).indexOf(s) == 0) {
                rooms.add(roomList.get(i));
            }       }    }
    public void handleSearchKey(KeyEvent event) {
        if (event.getEventType() == KeyEvent.KEY_RELEASED) {
            String s = search.getText();
            Search(rooms, s);
        }    }}
```

4.2 RESERVATION

```
package com.hotel.hotelmanagement;
public class Reservation {
    private int resID;
   private int roomNumber;
    private String customerName;
    private String checkInDate;
    private String checkOutDate;
    private int totalDays;
    private int totalPrice;
    private String status;
    public Reservation(int resID, int roomNumber, String customerName, String checkInDate, String checkOutDate,
int totalDays, int totalPrice, String status) {
        this.roomNumber = roomNumber;
        this.customerName = customerName;
        this.checkInDate = checkInDate;
        this.checkOutDate = checkOutDate;
        this.totalDays = totalDays;
        this.totalPrice = totalPrice;
        this.resID = resID;
        this.status = status;
    }

    public int getRoomNumber() {
        return roomNumber;    }
    public String getCustomerName() {
        return customerName;    }
    public String getStatus() {
        return status;    }
    public int getResID() {
       return resID;    }
    public String getCheckOutDate() {
        return checkOutDate;    }
    public String getCheckInDate() {
        return checkInDate;    }
    public int getTotalDays() {
        return totalDays;    }
    public int getTotalPrice() {
        return totalPrice;    }
    public void setRoomNumber(int roomNumber) {
        this.roomNumber = roomNumber;    }
    public void setCustomerName(String customerName) {
        this.customerName = customerName;    }
    public void setCheckOutDate(String checkOutDate) {
        this.checkOutDate = checkOutDate;    }
    public void setCheckInDate(String checkInDate) {
        this.checkInDate = checkInDate;    }
    public void setTotalDays(int totalDays) {
        this.totalDays = totalDays;    }
    public void setTotalPrice(int totalPrice) {
        this.totalPrice = totalPrice;    }
    public void setStatus(String status) {
        this.status = status;    }
    public void setResID(int resID) {
        this.resID = resID;    }}
```

4.3 ROOM

```
package com.hotel.hotelmanagement;
public class Room {
    private int number;
    private int price;
    private String type;
    private String status;
    public Room(int roomNumber, int price, String roomType, String status) {
        this.number = roomNumber;
        this.price = price;
        this.type = roomType;
        this.status = status;    }
    public int getNumber() {
        return number;    }
    public int getPrice() {
        return price;    }
    public String getStatus() {
        return status;    }
    public String getType() {
        return type;    }
    public void setNumber(int number) {
        this.number = number;    }
    public void setPrice(int price) {
        this.price = price;    }
    public void setStatus(String status) {
        this.status = status;    }
    public void setType(String type) {
        this.type = type;    }}
```

4.4 MAIN

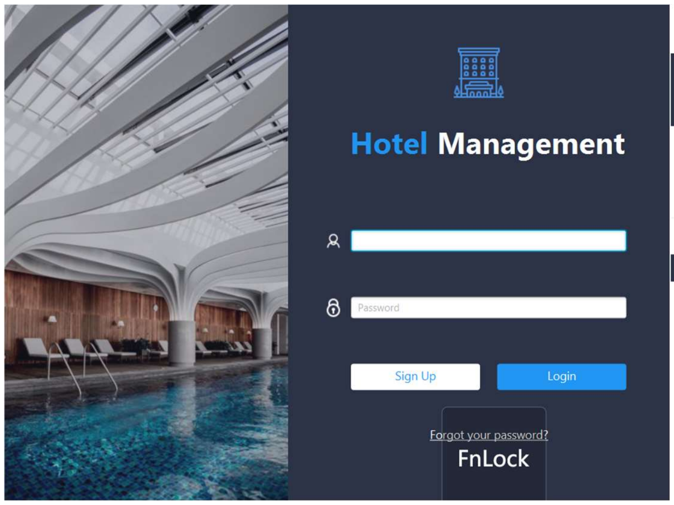
```
package com.hotel.hotelmanagement;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
public class Main extends Application {
    @Override
    public void start(Stage stage) {
        try {
            Parent root = FXMLLoader.load(getClass().getResource("login.fxml"));
            Scene scene = new Scene(root);
            stage.setScene(scene);
            stage.show();
        } catch (Exception exception) {
            exception.printStackTrace();        }    }
    public static void main(String[] args) {
        launch();    }}
```

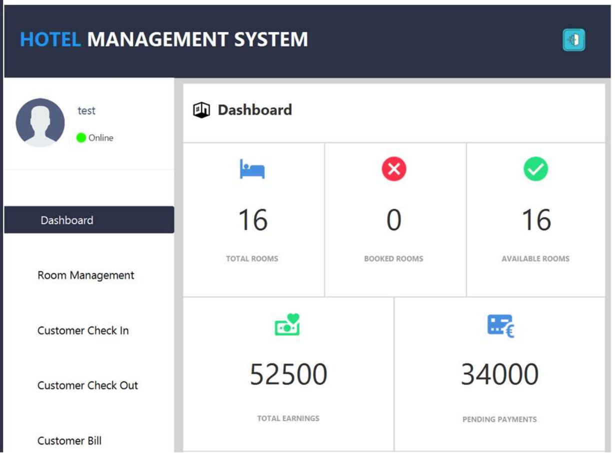
4.5 DATABASE CONNECTIVITY

```
package com.hotel.hotelmanagement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DBConnection {
    Connection connection;
    String url = "jdbc:mysql://localhost:3306/hotel_management";
    String username = "root";
    String password = "Karthikha*1011";
    public Connection getConnection() {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
        } catch (Exception e) {
            e.printStackTrace();
        }
        try {
            connection = DriverManager.getConnection(url, username, password);
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return connection;    }}
```

5 VISUAL REPRESENTATION:

5.1 HOME PAGE





16 TOTAL ROOMS

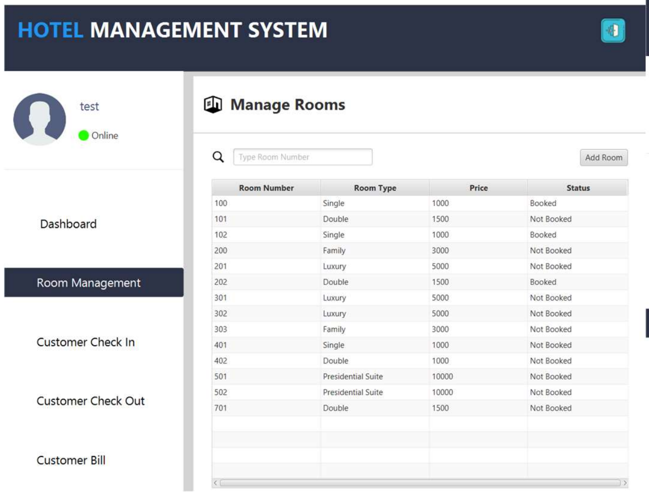
0 BOOKED ROOMS

16 AVAILABLE ROOMS

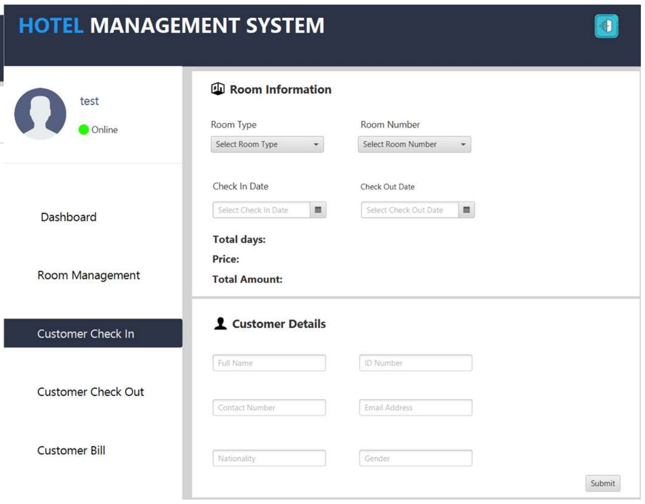
52500 TOTAL EARNINGS

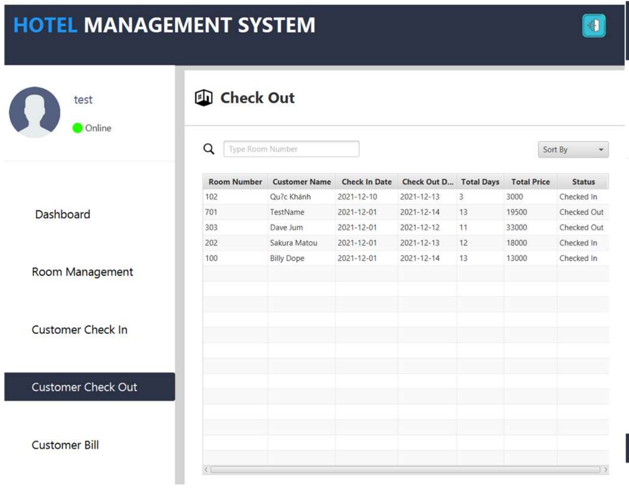
34000 PENDING PAYMENTS

5.2 DASHBOARD

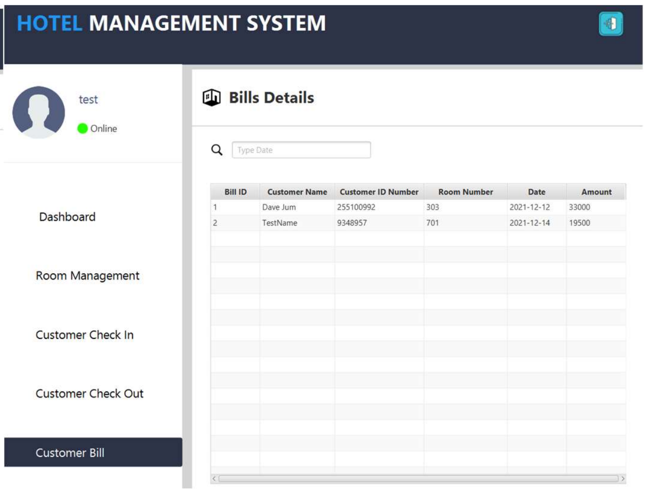


Room Number	Room Type	Price	Status
100	Single	1000	Booked
101	Double	1500	Not Booked
102	Single	1000	Booked
200	Family	3000	Not Booked
201	Luxury	5000	Not Booked
202	Double	1500	Booked
301	Luxury	5000	Not Booked
302	Luxury	5000	Not Booked
303	Family	3000	Not Booked
401	Single	1000	Not Booked
402	Double	1000	Not Booked
501	Presidential Suite	10000	Not Booked
502	Presidential Suite	10000	Not Booked
701	Double	1500	Not Booked



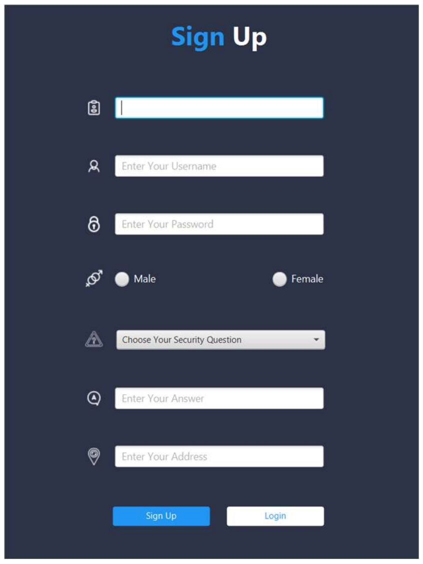


Room Number	Customer Name	Check In Date	Check Out D...	Total Days	Total Price	Status
102	Qu?c Khánh	2021-12-10	2021-12-13	3	3000	Checked In
701	TestName	2021-12-01	2021-12-14	13	19500	Checked Out
303	Dave Jum	2021-12-01	2021-12-12	11	33000	Checked Out
202	Sakura Matou	2021-12-01	2021-12-13	12	18000	Checked In
100	Billy Dope	2021-12-01	2021-12-14	13	13000	Checked In



Bill ID	Customer Name	Customer ID Number	Room Number	Date	Amount
1	Dave Jum	255100992	303	2021-12-12	33000
2	TestName	9348957	701	2021-12-14	19500

5.3 SIGNUP PAGE



CONCLUSIONS

The Hotel Management System developed using JavaFX and MySQL successfully streamlines hotel operations by automating tasks such as room reservations, customer management, and service tracking. The use of JavaFX ensures a user-friendly interface, enhancing accessibility and usability for staff, while MySQL provides a secure, centralized database for managing hotel data efficiently. This system reduces manual errors, improves operational efficiency, and offers real-time insights through statistical reports, making it a reliable and scalable solution for modern hotel management needs.

REFERENCES

1. JavaFX Official Documentation - https://openjfx.io/
2. MySQL Documentation - https://dev.mysql.com/doc/
3. Java JDBC Tutorial - https://www.javatpoint.com/java-jdbc
4. JavaFX Tutorial - https://www.tutorialspoint.com/javafx/index.htm