# HOTEL MANAGEMENT SYSTEM

**A MINI-PROJECT BY:**

**Magesh Vasan.M 230701507**

**Deepak.k 230701066**

***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**

**NOVEMBER 2024**

# BONAFIDE CERTIFICATE

Certified that this project **“HOTEL MANAGEMENT SYSTEM”** is the bonafide work of **“Magesh Vasan.M,Deepak.k”** who carried out the project work under my supervision.

Submitted for the practical examination held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SIGNATURE SIGNATURE

**INTERNAL EXAMINER EXTERNAL EXAMINER**

# ABSTRACT

The Hotel Management System (HMS) project is a software solution designed to streamline and automate various operations within a hotel, enhancing efficiency and improving the guest experience. This system addresses key functional areas such as reservations, check-in and check-out processes, billing, inventory management, and reporting. By integrating all aspects of hotel operations into a centralized system, the HMS reduces manual effort, minimizes errors, and ensures a smooth flow of information across departments.

The project is developed with a user-friendly interface, enabling hotel staff to manage bookings, room assignments, guest profiles, and service requests with ease. It includes a reservation module where guests can book rooms directly or through third-party channels, allowing real-time availability updates and minimizing double bookings. The system also provides a detailed billing module that generates accurate invoices, supports multiple payment options, and allows for easy tracking of outstanding balances.

Additionally, the HMS features an inventory management module to track room supplies, food and beverage stock, and other hotel resources, ensuring timely restocking and effective cost management. A comprehensive reporting tool enables management to gain insights into occupancy rates, revenue trends, and customer preferences, helping in strategic planning and decision-making.

This project is implemented with modern database and security practices, ensuring data integrity and guest privacy. By providing an all-in-one solution for hotel operations, the HMS aims to elevate service quality, increase operational efficiency, and contribute to the overall success of hotel establishments.

**TABLE OF CONTENTS**

## 1. INTRODUCTION

1.1 INTRODUCTION

1.2 IMPLEMENTATION

1.3 SCOPE OF THE PROJECT

1.4 WEBSITE FEATURES

## 2. SYSTEM SPECIFICATION

2.1 HARDWARE SPECIFICATION

2.2 SOFTWARE SPECIFICATION

## 3. SAMPLE CODE

3.1 CUSTOMER

3.2 BOOKING DAO

3.3 BOOKING

3.4 CUSTOMER DAO

3.5 DATABASE CONNECTION

3.6 ROOM

3.7 HOTEL MANAGEMENT SYSTEM GUI

3.8 ROOM DAO

## 4. SNAPSHOTS

4.1 HOME PAGE

4.2 DATA ENTRY

4.3 CUSTOMER ADDITION

4.4 BOOKING

1. **CONCLUSION**

1. **REFERENCES**

# INTRODUCTION

## 1.1 INTRODUCTION

The project helps students by giving them an idea of potential research topics, where they will choose the interested topic and will be able to submit the finished dissertation on the website, where tutors will evaluate the project and results will be posted on the website.

## 1.1 IMPLEMENTATION

The **HOTEL MANAGEMENT SYSTEM** project discussed here is implemented using the concepts of **JAVA SWINGS** and **MYSQL**.

## 1.2 SCOPE OF THE PROJECT

**A hotel management system involves comprehensive user management with roles and secure authentication, detailed room management including categories, availability, and pricing, and a robust booking management system to handle reservations, confirmations, and check-ins/check-outs. It also includes integrated payment processing with billing, payment gateway, and invoice management, inventory management for supplies and vendors, and housekeeping task assignment and tracking. Additionally, it maintains guest profiles, collects feedback through a CRM, and generates financial, occupancy, and performance reports. Ensuring data security and compliance, the system integrates with third-party services like OTAs and restaurant management systems to streamline operations.**

## 1.3 WEBSITE FEATURES

* Registering and login page.

* Custom profile for each customer.

* Dashboard showing possible actions.

* Guide to choosing desired rooms.

# SYSTEM SPECIFICATIONS

**2.1 HARDWARE SPECIFICATIONS:**

PROCESSOR : Intel i5

MEMORY SIZE : 4GB(Minimum)

HARD DISK : 500 GB of free space

**2.2 SOFTWARE SPECIFICATIONS:**

PROGRAMMING LANGUAGE : Java, MySQL

FRONT-END : Java

BACK-END : MySQL

OPERATING SYSTEM : Windows 10 **SAMPLE CODE**

## 3.1 CUSTOMER.JAVA

public class Customer {

private int customerId;

private String name;

private long phoneNo;

private String gender;

private String idNumber;

private String country;

// Constructor

public Customer(String name, long phoneNo, String gender, String idNumber, String country) {

this.name = name;

this.phoneNo = phoneNo;

this.gender = gender;

this.idNumber = idNumber;

this.country = country;

}

// Getters and Setters

public int getCustomerId() { return customerId; }

public void setCustomerId(int customerId) { this.customerId = customerId; }

public String getName() { return name; }

public long getPhoneNo() { return phoneNo; }

public String getGender() { return gender; }

public String getIdNumber() { return idNumber; }

public String getCountry() { return country; }

}

## 3.2 BOOKING DAO:

**import java.sql.\*;**

**import java.util.ArrayList;**

**import java.util.List;**

**public class BookingDAO {**

**public void addBooking(Booking booking) {**

**String query = "INSERT INTO bookings (customer\_id, room\_no, check\_in\_date, check\_out\_date, status) VALUES (?, ?, ?, ?, ?)";**

**try (Connection connection = DatabaseConnection.getConnection();**

**PreparedStatement statement = connection.prepareStatement(query)) {**

**statement.setInt(1, booking.getCustomerId());**

**statement.setInt(2, booking.getRoomNo());**

**statement.setDate(3, booking.getCheckInDate());**

**statement.setDate(4, booking.getCheckOutDate());**

**statement.setString(5, booking.getStatus());**

**statement.executeUpdate();**

**System.out.println("Booking added successfully.");**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**public List<Booking> getAllBookings() {**

**List<Booking> bookings = new ArrayList<>();**

**String query = "SELECT \* FROM bookings";**

**try (Connection connection = DatabaseConnection.getConnection();**

**PreparedStatement statement = connection.prepareStatement(query);**

**ResultSet resultSet = statement.executeQuery()) {**

**while (resultSet.next()) {**

**Booking booking = new Booking(**

**resultSet.getInt("customer\_id"),**

**resultSet.getInt("room\_no"),**

**resultSet.getDate("check\_in\_date"),**

**resultSet.getDate("check\_out\_date"),**

**resultSet.getString("status")**

**);**

**booking.setBookingId(resultSet.getInt("booking\_id"));**

**bookings.add(booking);**

**}**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**return bookings;**

**}**

**public void updateBookingStatus(int bookingId, String status) {**

**String query = "UPDATE bookings SET status = ? WHERE booking\_id = ?";**

**try (Connection connection = DatabaseConnection.getConnection();**

**PreparedStatement statement = connection.prepareStatement(query)) {**

**statement.setString(1, status);**

**statement.setInt(2, bookingId);**

**statement.executeUpdate();**

**System.out.println("Booking status updated successfully.");**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**public void deleteBooking(int bookingId) {**

**String query = "DELETE FROM bookings WHERE booking\_id = ?";**

**try (Connection connection = DatabaseConnection.getConnection();**

**PreparedStatement statement = connection.prepareStatement(query)) {**

**statement.setInt(1, bookingId);**

**statement.executeUpdate();**

**System.out.println("Booking deleted successfully.");**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**}**

**3.3 BOOKING:**

import java.sql.Date;

public class Booking {

private int bookingId; // Field for booking ID

private int customerId;

private int roomNo;

private Date checkInDate;

private Date checkOutDate;

private String status;

// Constructor without bookingId (for new bookings)

public Booking(int customerId, int roomNo, Date checkInDate, Date checkOutDate, String status) {

this.customerId = customerId;

this.roomNo = roomNo;

this.checkInDate = checkInDate;

this.checkOutDate = checkOutDate;

this.status = status;

}

// Constructor with bookingId (for existing bookings)

public Booking(int bookingId, int customerId, int roomNo, Date checkInDate, Date checkOutDate, String status) {

this.bookingId = bookingId;

this.customerId = customerId;

this.roomNo = roomNo;

this.checkInDate = checkInDate;

this.checkOutDate = checkOutDate;

this.status = status;

}

// Getters and Setters

public int getBookingId() {

return bookingId;

}

public void setBookingId(int bookingId) {

this.bookingId = bookingId;

}

public int getCustomerId() {

return customerId;

}

public int getRoomNo() {

return roomNo;

}

public Date getCheckInDate() {

return checkInDate;

}

public Date getCheckOutDate() {

return checkOutDate;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

}

## 3.4 CUSTOMER DAO

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class CustomerDAO {

public void addCustomer(Customer customer) {

String query = "INSERT INTO customers (name, phone\_no, gender, id\_number, country) VALUES (?, ?, ?, ?, ?)";

try (Connection connection = DatabaseConnection.getConnection();

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setString(1, customer.getName());

statement.setLong(2, customer.getPhoneNo());

statement.setString(3, customer.getGender());

statement.setString(4, customer.getIdNumber());

statement.setString(5, customer.getCountry());

statement.executeUpdate();

System.out.println("Customer added successfully.");

} catch (SQLException e) {

e.printStackTrace();

}

}

public List<Customer> getAllCustomers() {

List<Customer> customers = new ArrayList<>();

String query = "SELECT \* FROM customers";

try (Connection connection = DatabaseConnection.getConnection();

PreparedStatement statement = connection.prepareStatement(query);

ResultSet resultSet = statement.executeQuery()) {

while (resultSet.next()) {

Customer customer = new Customer(

resultSet.getString("name"),

resultSet.getLong("phone\_no"),

resultSet.getString("gender"),

resultSet.getString("id\_number"),

resultSet.getString("country")

);

customer.setCustomerId(resultSet.getInt("customer\_id"));

customers.add(customer);

}

} catch (SQLException e) {

e.printStackTrace();

}

return customers;

}

public void updateCustomer(Customer customer) {

String query = "UPDATE customers SET name = ?, phone\_no = ?, gender = ?, id\_number = ?, country = ? WHERE customer\_id = ?";

try (Connection connection = DatabaseConnection.getConnection();

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setString(1, customer.getName());

statement.setLong(2, customer.getPhoneNo());

statement.setString(3, customer.getGender());

statement.setString(4, customer.getIdNumber());

statement.setString(5, customer.getCountry());

statement.setInt(6, customer.getCustomerId());

statement.executeUpdate();

System.out.println("Customer updated successfully.");

} catch (SQLException e) {

e.printStackTrace();

}

}

public void deleteCustomer(int customerId) {

String query = "DELETE FROM customers WHERE customer\_id = ?";

try (Connection connection = DatabaseConnection.getConnection();

PreparedStatement statement = connection.prepareStatement(query)) {

statement.setInt(1, customerId);

statement.executeUpdate();

System.out.println("Customer deleted successfully.");

} catch (SQLException e) {

e.printStackTrace();

}

}

}

## 3.5 DATABASE CONNECTION

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

private static final String url = "jdbc:mysql://localhost:3306/hotel\_management";

private static final String user = "root";

private static final String password = "DEE#050606#pak";

public static Connection getConnection() throws SQLException {

return DriverManager.getConnection(url, user, password);

}

}

## 3.6 ROOM

public class Room {

private int roomNo;

private String roomType;

private double rate;

private String status;

// Constructor

public Room(int roomNo, String roomType, double rate, String status) {

this.roomNo = roomNo;

this.roomType = roomType;

this.rate = rate;

this.status = status;

}

// Getters and Setters

public int getRoomNo() { return roomNo; }

public String getRoomType() { return roomType; }

public double getRate() { return rate; }

public String getStatus() { return status; }

}

## 3.7 HOTEL MANAGEMENT SYSTEM GUI

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.Date;

public class HotelManagementSystemGUI {

private JFrame frame;

private JTextField nameField, phoneField, genderField, idField, countryField;

private JButton addCustomerButton, viewRoomsButton, makeBookingButton;

private CustomerDAO customerDAO;

private RoomDAO roomDAO;

private BookingDAO bookingDAO;

public HotelManagementSystemGUI() {

customerDAO = new CustomerDAO();

roomDAO = new RoomDAO();

bookingDAO = new BookingDAO();

frame = new JFrame("Hotel Management System");

frame.setSize(600, 400);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setLayout(new BorderLayout());

// Create the input panel for customer information

JPanel inputPanel = new JPanel();

inputPanel.setLayout(new GridLayout(6, 2, 10, 10));

inputPanel.setBorder(BorderFactory.createEmptyBorder(20, 20, 20, 20));

// Labels and input fields

inputPanel.add(new JLabel("Name:"));

nameField = new JTextField();

inputPanel.add(nameField);

inputPanel.add(new JLabel("Phone:"));

phoneField = new JTextField();

inputPanel.add(phoneField);

inputPanel.add(new JLabel("Gender:"));

genderField = new JTextField();

inputPanel.add(genderField);

inputPanel.add(new JLabel("ID Number:"));

idField = new JTextField();

inputPanel.add(idField);

inputPanel.add(new JLabel("Country:"));

countryField = new JTextField();

inputPanel.add(countryField);

frame.add(inputPanel, BorderLayout.CENTER);

// Create a button panel at the bottom

JPanel buttonPanel = new JPanel();

buttonPanel.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 10));

// Buttons

addCustomerButton = new JButton("Add Customer");

viewRoomsButton = new JButton("View Rooms");

makeBookingButton = new JButton("Make Booking");

addCustomerButton.addActionListener(new AddCustomerActionListener());

viewRoomsButton.addActionListener(new ViewRoomsActionListener());

makeBookingButton.addActionListener(new MakeBookingActionListener());

buttonPanel.add(addCustomerButton);

buttonPanel.add(viewRoomsButton);

buttonPanel.add(makeBookingButton);

frame.add(buttonPanel, BorderLayout.SOUTH);

frame.setVisible(true);

}

// Action listener to add a customer

private class AddCustomerActionListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

String name = nameField.getText();

long phone = Long.parseLong(phoneField.getText());

String gender = genderField.getText();

String idNumber = idField.getText();

String country = countryField.getText();

Customer customer = new Customer(name, phone, gender, idNumber, country);

customerDAO.addCustomer(customer);

JOptionPane.showMessageDialog(frame, "Customer added successfully!");

}

}

// Action listener to view rooms

private class ViewRoomsActionListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

java.util.List<Room> rooms = roomDAO.getAllRooms();

StringBuilder roomsInfo = new StringBuilder("Room List:\n");

for (Room room : rooms) {

roomsInfo.append("Room No: ").append(room.getRoomNo())

.append(", Type: ").append(room.getRoomType())

.append(", Rate: ").append(room.getRate())

.append(", Status: ").append(room.getStatus())

.append("\n");

}

JOptionPane.showMessageDialog(frame, roomsInfo.toString());

}

}

// Action listener to make a booking

private class MakeBookingActionListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

try {

int customerId = Integer.parseInt(JOptionPane.showInputDialog("Enter Customer ID:"));

int roomNo = Integer.parseInt(JOptionPane.showInputDialog("Enter Room No:"));

Date checkInDate = Date.valueOf(JOptionPane.showInputDialog("Enter Check-in Date (YYYY-MM-DD):"));

Date checkOutDate = Date.valueOf(JOptionPane.showInputDialog("Enter Check-out Date (YYYY-MM-DD):"));

Booking booking = new Booking(customerId, roomNo, checkInDate, checkOutDate, "Reserved");

bookingDAO.addBooking(booking);

JOptionPane.showMessageDialog(frame, "Booking added successfully!");

} catch (Exception ex) {

JOptionPane.showMessageDialog(frame, "Error in booking: " + ex.getMessage());

}

}

}

public static void main(String[] args) {

new HotelManagementSystemGUI();

}

}

## 3.8 ROOM DAO

**import java.sql.\*;**

**import java.util.ArrayList;**

**import java.util.List;**

**public class RoomDAO {**

**// Method to add a room**

**public void addRoom(Room room) {**

**String query = "INSERT INTO rooms (room\_no, room\_type, rate, status) VALUES (?, ?, ?, ?)";**

**try (Connection connection = DatabaseConnection.getConnection();**

**PreparedStatement statement = connection.prepareStatement(query)) {**

**statement.setInt(1, room.getRoomNo());**

**statement.setString(2, room.getRoomType());**

**statement.setDouble(3, room.getRate());**

**statement.setString(4, room.getStatus());**

**statement.executeUpdate();**

**System.out.println("Room added successfully.");**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**// Method to get all rooms**

**public List<Room> getAllRooms() {**

**List<Room> rooms = new ArrayList<>();**

**String query = "SELECT \* FROM rooms";**

**try (Connection connection = DatabaseConnection.getConnection();**

**PreparedStatement statement = connection.prepareStatement(query);**

**ResultSet resultSet = statement.executeQuery()) {**

**while (resultSet.next()) {**

**Room room = new Room(**

**resultSet.getInt("room\_no"),**

**resultSet.getString("room\_type"),**

**resultSet.getDouble("rate"),**

**resultSet.getString("status")**

**);**

**rooms.add(room);**

**}**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**return rooms;**

**}**

**// Method to get a specific room by room number**

**public Room getRoom(int roomNo) {**

**Room room = null;**

**String query = "SELECT \* FROM rooms WHERE room\_no = ?";**

**try (Connection connection = DatabaseConnection.getConnection();**

**PreparedStatement statement = connection.prepareStatement(query)) {**

**statement.setInt(1, roomNo);**

**try (ResultSet resultSet = statement.executeQuery()) {**

**if (resultSet.next()) {**

**room = new Room(**

**resultSet.getInt("room\_no"),**

**resultSet.getString("room\_type"),**

**resultSet.getDouble("rate"),**

**resultSet.getString("status")**

**);**

**}**

**}**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**return room;**

**}**

**// Method to update room status**

**public void updateRoomStatus(int roomNo, String status) {**

**String query = "UPDATE rooms SET status = ? WHERE room\_no = ?";**

**try (Connection connection = DatabaseConnection.getConnection();**

**PreparedStatement statement = connection.prepareStatement(query)) {**

**statement.setString(1, status);**

**statement.setInt(2, roomNo);**

**statement.executeUpdate();**

**System.out.println("Room status updated successfully.");**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**// Method to update room rate**

**public void updateRoomRate(int roomNo, double rate) {**

**String query = "UPDATE rooms SET rate = ? WHERE room\_no = ?";**

**try (Connection connection = DatabaseConnection.getConnection();**

**PreparedStatement statement = connection.prepareStatement(query)) {**

**statement.setDouble(1, rate);**

**statement.setInt(2, roomNo);**

**statement.executeUpdate();**

**System.out.println("Room rate updated successfully.");**

**} catch (SQLException e) {**

**e.printStackTrace();**

**}**

**}**

**// Method to delete a room by room number**

**public void deleteRoom(int roomNo) {**

**String query = "DELETE FROM rooms WHERE room\_no = ?";**

**try (Connection connection = DatabaseConnection.getConnection();**

**PreparedStatement statement = connection.prepareStatement(query)) {**

**statement.setInt(1, roomNo);**

**statement.executeUpdate();**

**System.out.println("Room deleted successfully.");**

**} catch (SQLException e) {**

**e.printStackTrace();**

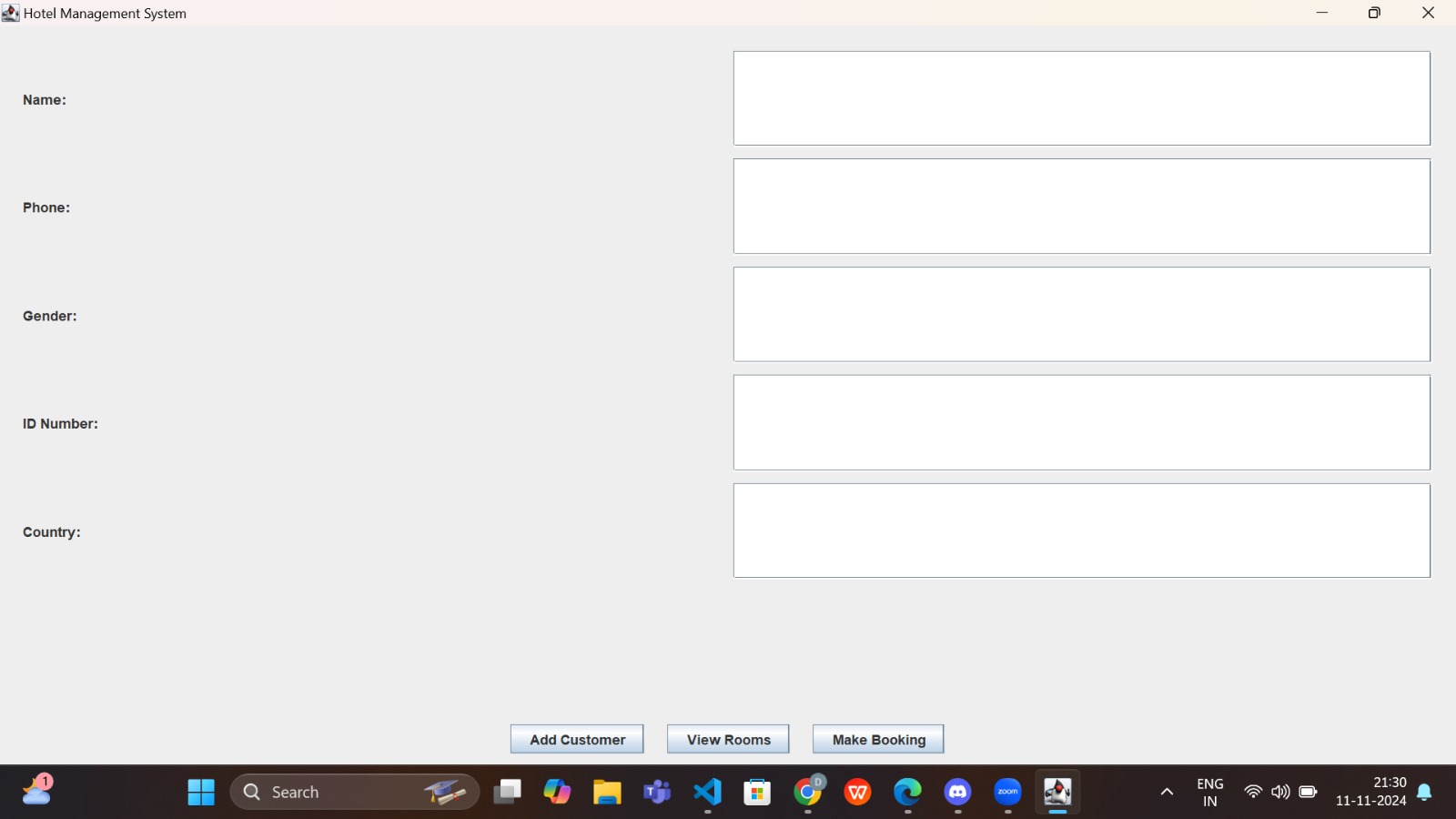
**}**

**}**

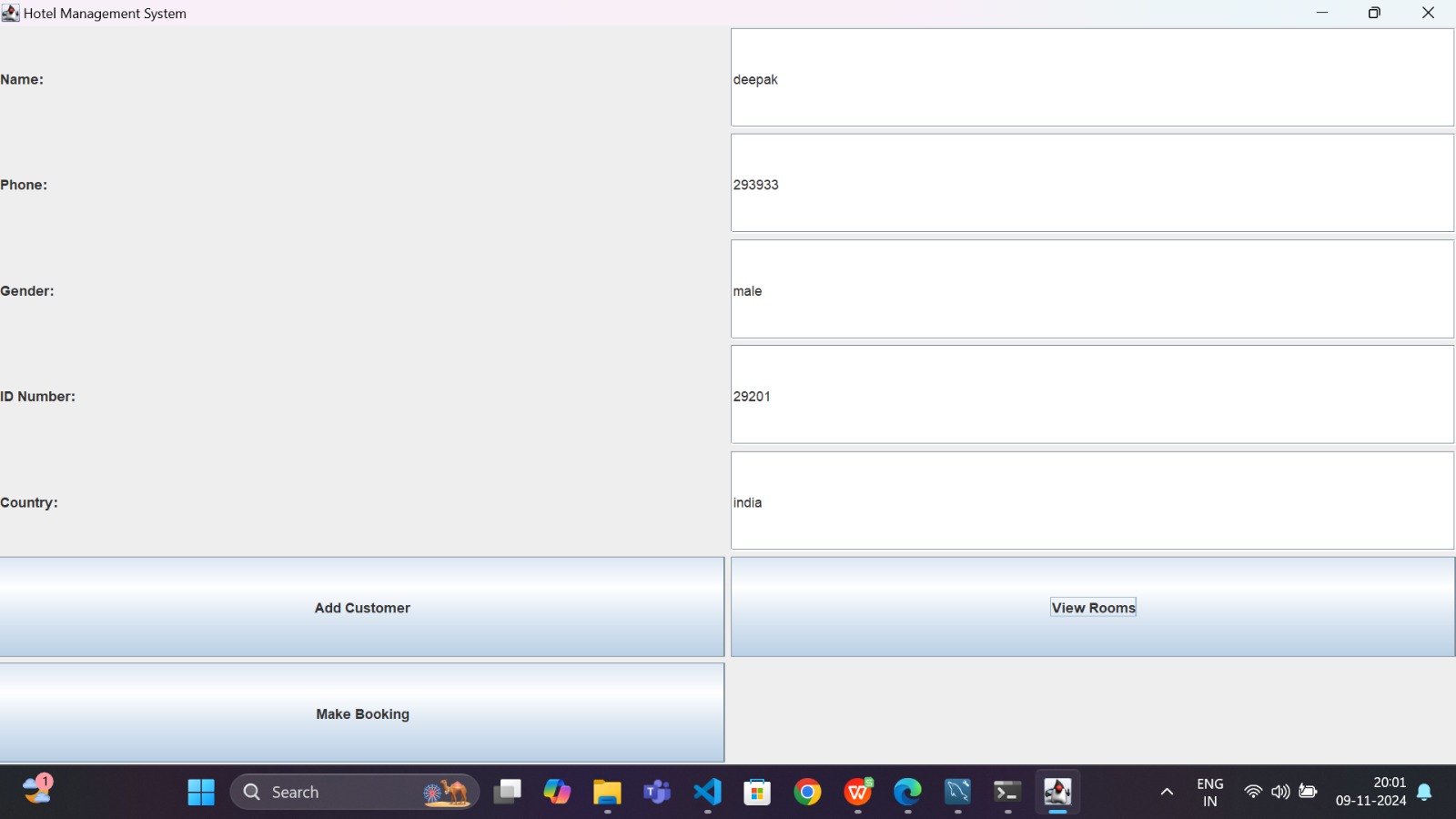
**}**

**SNAPSHOTS**

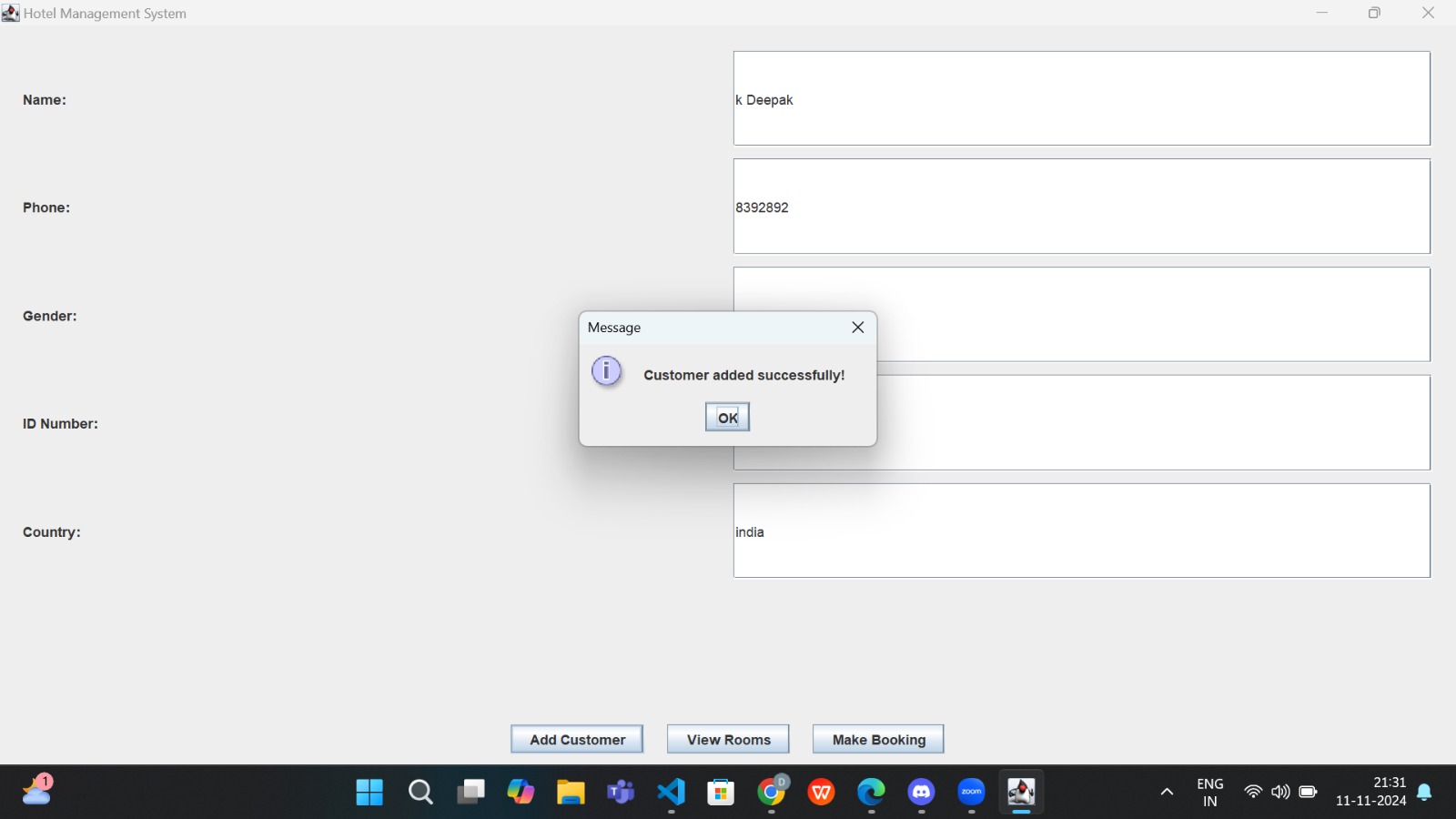
## 4.1 HOME PAGE

****

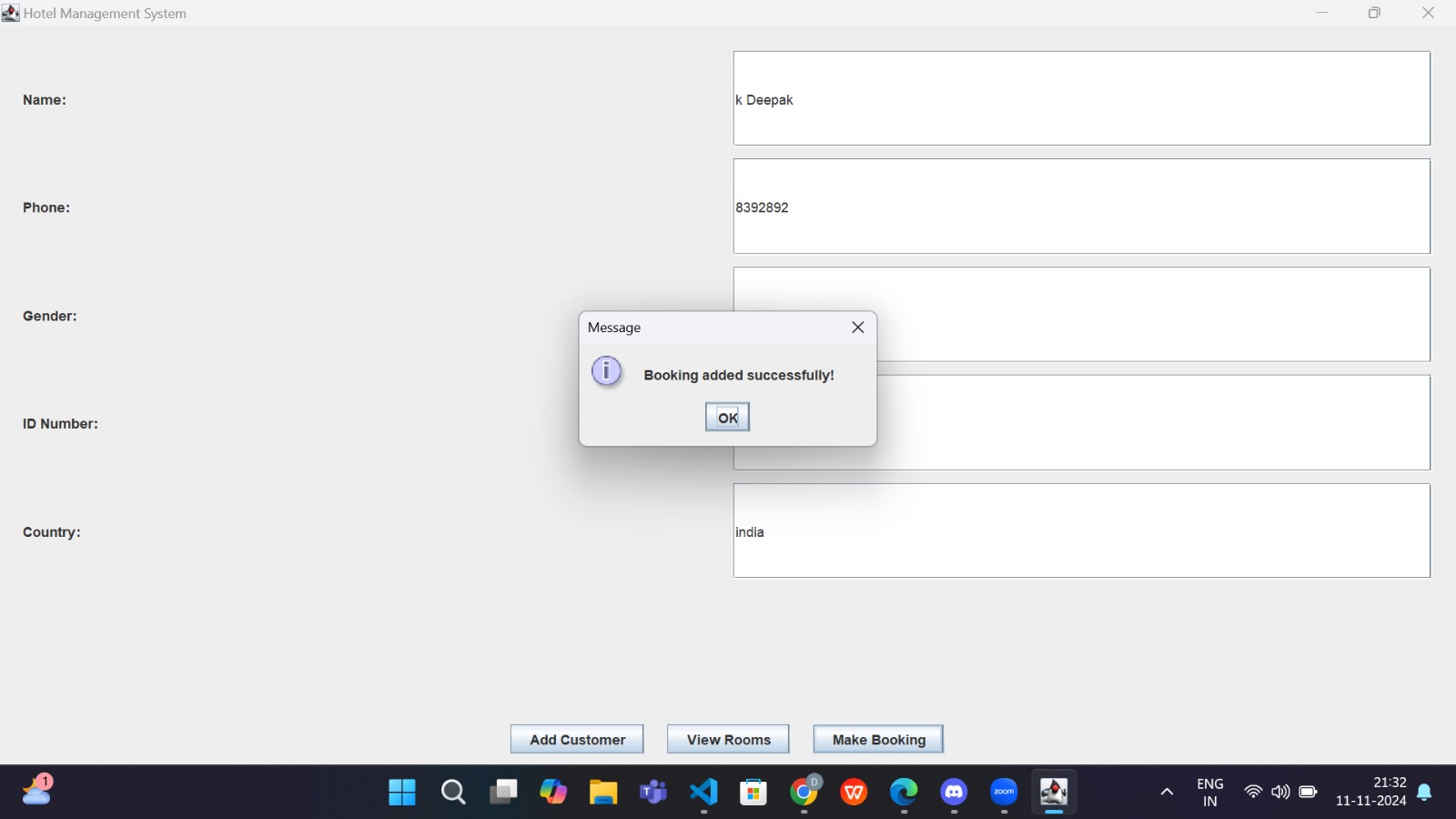
## 4.2 DETAILS ENTRY

****

## 4.3 CUSTOMER ADDITION

****

## 4.4 BOOKING

****

# CONCLUSION

**A hotel management system serves as an essential tool for modern hospitality businesses, enhancing efficiency, guest satisfaction, and revenue management. By automating tasks such as room bookings, check-ins, billing, inventory control, and customer relationship management, the system minimizes human error and reduces manual labor, allowing staff to focus on providing excellent guest experiences. Additionally, such a system can provide valuable insights through data analytics, aiding in strategic decision-making for hotel operations and marketing.**

**In conclusion, a well-implemented hotel management system streamlines operations, ensures better resource utilization, and ultimately contributes to improved profitability and customer loyalty, positioning the hotel to remain competitive and responsive to changing industry demands.**

**REFERENCES**

1. [**https://www.javatpoint.com/java-tutorial**](https://www.javatpoint.com/java-tutorial)

1. <https://www.wikipedia.org/>

1. <https://www.w3schools.com/sql/>

1. [SQL | Codecademy](https://www.codecademy.com/resources/docs/sql)