TITLE :

"HOTEL BOOKING SYSTEM"

NAME :

P. BALA NARASIMHULU

REG NO :

192372019

Under the guidance of :

Dr.GEETHA

in partial fulfillment for the completion of Course

CSA1087-SOFTWARE ENGINEERING

SIMATS ENGINEERING

THANDALAM

DECEMBER 2024

Hotel Booking System

Project Description:

Design and implement a full-featured Hotel Booking System where users can search for hotels, check room availability, make reservations, and process payments. The system will also include an admin dashboard for hotel managers to handle bookings, manage rooms, and analyse reports.

Core Features:

For Guests (Users):

1. Hotel Search

   . Search hotels by location, date, price, and amenities.

2. Room Booking

   . Real-time room availability, room type selection, and booking.

3. User Accounts

   . Allow users to create profiles to manage their bookings.

4. Payment Integration

   . Secure online payment gateway (e.g., Stripe or PayPal).

5. Review System

   . Leave reviews and ratings for hotels after a stay.

For Admins (Hotel Managers):

1. Room Management

   . Add, edit, or remove room details (e.g., pricing, availability, photos).

2. Booking Management

   . View and manage guest bookings.

3. Analytics Dashboard

   . View metrics like occupancy rates, revenue, and booking trends.

4. Notifications

   . Receive alerts for overbooked or underutilized rooms.

Technologies to Use:

1. Frontend
   . HTML, CSS, JavaScript (React.js, Angular, or Vue.js).
   . Mobile app: Flutter or React Native (optional).
2. Backend
   . Node.js, Django, or Spring Boot for server-side logic.
3. Database
   . Relational: MySQL or PostgreSQL.
   . NoSQL (for analytics): MongoDB.
4. Payment Gateway
   . Stripe, Razorpay, or PayPal APIs.
5. APIs and Services
   . Google Maps API for hotel location.
   . Email/SMS APIs for notifications.
6. Hosting and Deployment
   . Cloud services: AWS, Azure, or Google Cloud.
   . Docker for containerization.

Optional Advanced Features:

1. Dynamic Pricing
   . Use demand-based pricing algorithms to adjust room rates dynamically.
2. AI-Powered Recommendations
   . Suggest hotels to users based on past bookings or preferences.
3. Multilingual Support
   . Localize the app for different languages.
4. Integration with Travel Platforms
   . Sync with platforms like Expedia or Booking.com.

1. **Back-end**

```
// server.js
const express = require('express');
const app = express();
const bodyParser = require('body-parser');
const mongoose = require('mongoose');

// Middleware
app.use(bodyParser.json());

// Database connection
mongoose.connect('mongodb://localhost:27017/hotel_booking', {
  useNewUrlParser: true,
  useUnifiedTopology: true,
}).then(() => console.log('Database connected')).catch(err => console.log(err));

// Room schema
const roomSchema = new mongoose.Schema({
  name: String,
  price: Number,
  isBooked: { type: Boolean, default: false },
  checkInDate: Date,
  checkOutDate: Date,
});

const Room = mongoose.model('Room', roomSchema);
```

```
// API Endpoints

// Get all rooms
app.get('/rooms', async (req, res) => {
  const rooms = await Room.find();
  res.json(rooms);
});

// Book a room
app.post('/book', async (req, res) => {
  const { roomId, checkInDate, checkOutDate } = req.body;
  const room = await Room.findById(roomId);

  if (!room || room.isBooked) {
    return res.status(400).json({ error: 'Room not available' });
  }

  room.isBooked = true;
  room.checkInDate = new Date(checkInDate);
  room.checkOutDate = new Date(checkOutDate);
  await room.save();

  res.json({ message: 'Room booked successfully', room });
});

// Add a room (admin functionality)
app.post('/rooms', async (req, res) => {
  const { name, price } = req.body;
```

```
  const newRoom = new Room({ name, price });
  await newRoom.save();
  res.json({ message: 'Room added', newRoom });
});

// Start the server
app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

2. front-end

```
// App.js
import React, { useEffect, useState } from 'react';

function App() {
  const [rooms, setRooms] = useState([]);
  const [booking, setBooking] = useState({ roomId: '', checkInDate: '',
checkOutDate: '' });

  useEffect(() => {
    fetch('/rooms')
      .then((response) => response.json())
      .then((data) => setRooms(data));
  }, []);

  const handleBooking = async () => {
    const response = await fetch('/book', {
      method: 'POST',
```

```
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(booking),
    });

    const result = await response.json();
    if (response.ok) {
      alert(result.message);
      setRooms(rooms.map(room => room._id === booking.roomId ? { ...room, isBooked: true } : room));
    } else {
      alert(result.error);
    }
  };

  return (
    <div>
      <h1>Hotel Booking System</h1>
      <h2>Available Rooms</h2>
      <ul>
        {rooms.map(room => (
          <li key={room._id}>
            {room.name} - ${room.price} - {room.isBooked ? 'Booked' : 'Available'}
            {!room.isBooked && (
              <button onClick={() => setBooking({ ...booking, roomId: room._id })}>
                Book Now
              </button>
            )}
```

```
            </li>
          ))}
        </ul>

        {booking.roomId && (
          <div>
            <h3>Booking Details</h3>
            <label>
              Check-In Date:
              <input
                type="date"
                onChange={(e) => setBooking({ ...booking, checkInDate:
e.target.value })}
              />
            </label>
            <label>
              Check-Out Date:
              <input
                type="date"
                onChange={(e) => setBooking({ ...booking, checkOutDate:
e.target.value })}
              />
            </label>
            <button onClick={handleBooking}>Confirm Booking</button>
          </div>
        )}
      </div>
    );
```

}

3. database schema (mango DB)

```
db.rooms.insertMany([
  { name: "Deluxe Room", price: 120, isBooked: false },
  { name: "Standard Room", price: 80, isBooked: false },
  { name: "Suite", price: 200, isBooked: false }
]);
```

Learning Outcomes:

. Experience in full-stack development.

. Understanding of real-world database design and optimization.

. API development and third-party service integration.

. Exposure to deployment and cloud services.

. Implementation of secure payment systems.