DOCUMENTATION

USE CASE

Project: Smart Hotel Booking System

1.Introduction

This document outlines the Low-Level Design (LLD) for a Smart Hotel Booking System, which allows users to search, book, and manage hotel reservations. The platform supports hotel owners in listing their properties, defining room availability, and handling payments.
This design supports both Java (Spring Boot) and .NET (ASP.NET Core) frameworks for backend development.

2. Module Overview

2.1 User & Role Management

• Role-based access control (Admin, Hotel Manager, Guest).

• Secure user authentication and profile management.

2.2 Hotel & Room Management

• Hotel managers can list hotels and manage rooms.

• Set room types, pricing, amenities, and availability.

2.3 Booking & Payment Processing

• Users can search, book, and pay for rooms securely.

• Integration with payment gateways for transactions.

2.4 Reviews & Ratings

• Guests can rate hotels and leave reviews.

• Hotels can respond to feedback.

2.5 Loyalty & Rewards Program

• Guests earn reward points for bookings.

• Redeem points for discounts on future stays.

3. Architecture Overview

3.1 Architectural Style

• Frontend: Angular or React

• Backend: REST API-based architecture

• Database: Relational Database (MySQL/SQL Server)

3.2 Component Interaction

• Frontend communicates with the backend via REST APIs.

• Backend manages hotel data, booking operations, and payments.

4. Module-Wise Design

4.1 User & Role Management Module

4.1.1 Features • User authentication using JWT tokens.

• Role-based permissions: Admin, Hotel Manager, Guest.

4.1.2 Data Flow

1. Users register and log in.

2. Role-based permissions are assigned.

3. Admins manage hotel manager accounts.

4.1.3 Entities

• User (UserID, Name, Email, Password, Role, ContactNumber)

4.2 Hotel & Room Management Module

4.2.1 Features

• Hotel managers can list and manage hotels.

• Define room types, pricing, and amenities.

4.2.2 Data Flow

1. Hotel managers add hotel details and room inventory.

2. Users search for available rooms.

3. Availability updates when bookings are confirmed.

4.2.3 Entities

• Hotel (HotelID, Name, Location, ManagerID, Amenities, Rating)

• Room (RoomID, HotelID, Type, Price, Availability, Features)

4.3 Booking & Payment Processing Module

4.3.1 Features

• Users can search and book rooms.

• Secure payment processing and booking confirmation.

4.3.2 Data Flow

1. Users select a hotel and book a room.

2. Payments are processed securely.

3. Booking details are sent to users and hotel managers.

4.3.3 Entities

• Booking (BookingID, UserID, RoomID, CheckInDate, CheckOutDate, Status, PaymentID)

• Payment (PaymentID, UserID, BookingID, Amount, Status, PaymentMethod)

4.4 Reviews & Ratings Module

4.4.1 Features

• Guests can rate hotels and write reviews.

• Hotels can respond to reviews.

4.4.2 Data Flow

1. Users submit reviews after their stay.

2. The system moderates and publishes reviews.

3. Hotel managers can respond to feedback.

4.4.3 Entities

• Review (ReviewID, UserID, HotelID, Rating, Comment, Timestamp)

4.5 Loyalty & Rewards Program Module

4.5.1 Features

• Guests earn points for bookings.

• Points can be redeemed for discounts.

4.5.2 Data Flow

1. When a user books a hotel, reward points are added to their account.

2. Users can view their accumulated points.

3. At checkout, users can redeem points for discounts.

4.5.3 Entities

• LoyaltyAccount (LoyaltyID, UserID, PointsBalance, LastUpdated)

• Redemption (RedemptionID, UserID, BookingID, PointsUsed, DiscountAmount)

5. Deployment Strategy

5.1 Local Deployment

• Frontend Deployment: Angular/React dev server.

• Backend Deployment: Spring Boot/ASP.NET Core locally.

• Database: MySQL/PostgreSQL/SQL Server.

6. Database Design

6.1 Tables and Relationships

• User (UserID, Name, Email, Password, Role, ContactNumber)

• Hotel (HotelID, Name, Location, ManagerID, Amenities, Rating)

• Room (RoomID, HotelID, Type, Price, Availability, Features)

• Booking (BookingID, UserID, RoomID, CheckInDate, CheckOutDate, Status, PaymentID)

• Payment (PaymentID, UserID, BookingID, Amount, Status, PaymentMethod)

• Review (ReviewID, UserID, HotelID, Rating, Comment, Timestamp)

• LoyaltyAccount (LoyaltyID, UserID, PointsBalance, LastUpdated)

• Redemption (RedemptionID, UserID, BookingID, PointsUsed, DiscountAmount)

7. User Interface Design

7.1 Wireframes

• Guest Dashboard: Search and book rooms.

• Hotel Manager Dashboard: List and manage hotel properties.

• Admin Panel: Manage hotels, users, and reviews.

• Loyalty Program Page: View and redeem points.

8. Non-Functional Requirements

8.1 Performance

• Handles high-traffic room searches efficiently.

8.2 Scalability

• Supports multiple hotel chains and properties.

8.3 Security

• Secure JWT-based authentication.

• Encrypted payment transactions.

8.4 Usability

• Mobile-friendly interface for seamless booking experience.

9. Assumptions and Constraints

9.1 Assumptions

• Users can cancel bookings up to 24 hours before check-in.

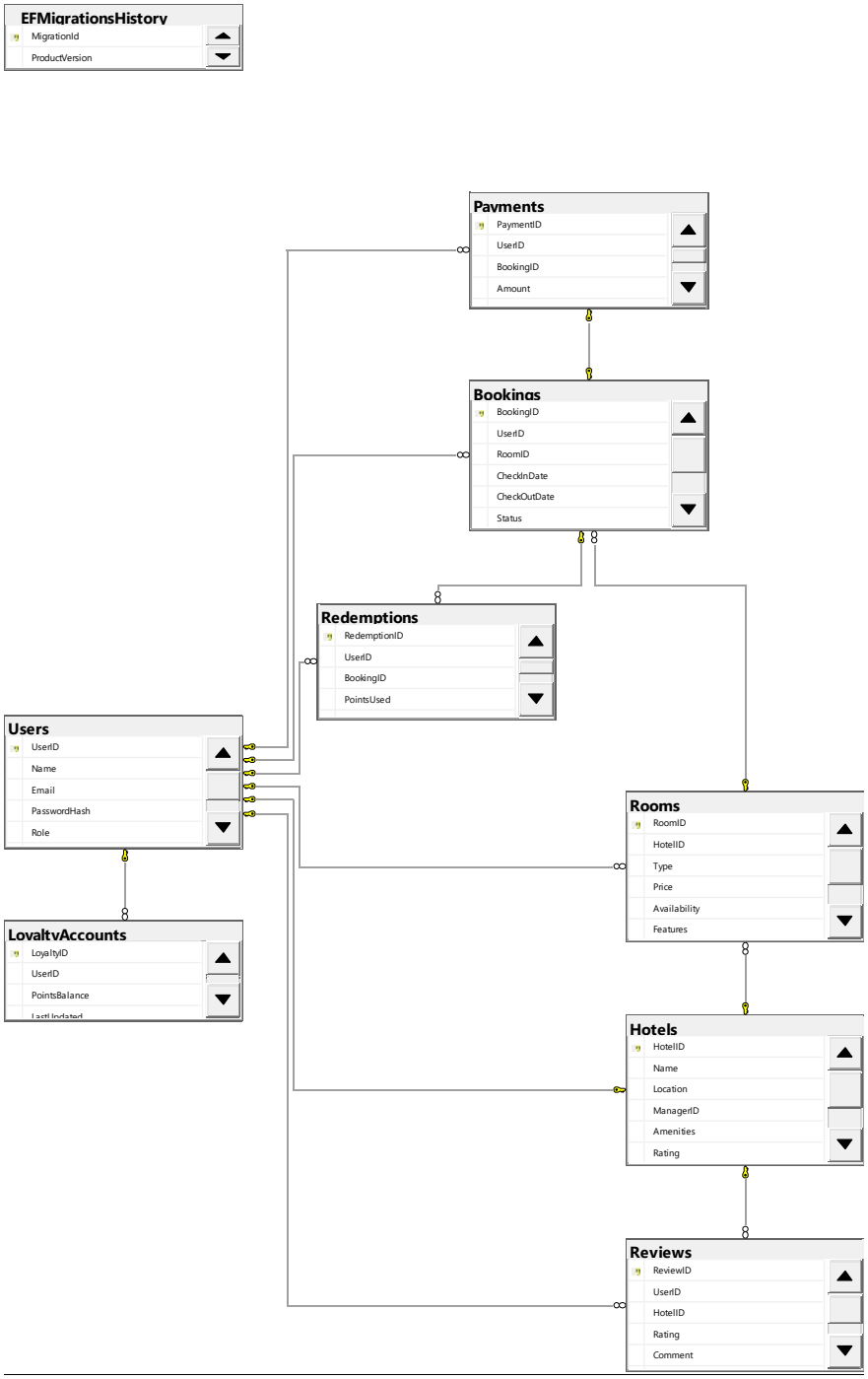
• Hotels must verify their identity before listing.

9.2 Constraints

• The system must support different currencies for payments.

• Hotels must comply with legal policies for cancellations and refunds

DATABASE



SCRIPT

-- Users Table

CREATE TABLE Users (

UserID INT PRIMARY KEY IDENTITY,

```
    Name NVARCHAR(100),
    Email NVARCHAR(100) UNIQUE,
    PasswordHash NVARCHAR(MAX),
    Role NVARCHAR(50),
    ContactNumber NVARCHAR(20)
);

-- Hotels Table
CREATE TABLE Hotels (
    HotelID INT PRIMARY KEY IDENTITY,
    Name NVARCHAR(150),
    Location NVARCHAR(200),
    ManagerID INT FOREIGN KEY REFERENCES Users(UserID),
    Amenities NVARCHAR(MAX),
    Rating DECIMAL(3, 2)
);

-- Rooms Table
CREATE TABLE Rooms (
    RoomID INT PRIMARY KEY IDENTITY,
    HotelID INT FOREIGN KEY REFERENCES Hotels(HotelID),
    Type NVARCHAR(50),
    Price DECIMAL(10, 2),
    Availability BIT,
    Features NVARCHAR(MAX)
);

-- Bookings Table
```

```
CREATE TABLE Bookings (
  BookingID INT PRIMARY KEY IDENTITY,
  UserID INT FOREIGN KEY REFERENCES Users(UserID),
  RoomID INT FOREIGN KEY REFERENCES Rooms(RoomID),
  CheckInDate DATE,
  CheckOutDate DATE,
  Status NVARCHAR(50),
  PaymentID INT
);

-- Payments Table
CREATE TABLE Payments (
  PaymentID INT PRIMARY KEY IDENTITY,
  UserID INT FOREIGN KEY REFERENCES Users(UserID),
  BookingID INT FOREIGN KEY REFERENCES Bookings(BookingID),
  Amount DECIMAL(10, 2),
  Status NVARCHAR(50),
  PaymentMethod NVARCHAR(50)
);

-- Reviews Table
CREATE TABLE Reviews (
  ReviewID INT PRIMARY KEY IDENTITY,
  UserID INT FOREIGN KEY REFERENCES Users(UserID),
  HotelID INT FOREIGN KEY REFERENCES Hotels(HotelID),
  Rating INT,
  Comment NVARCHAR(MAX),
  Timestamp DATETIME DEFAULT GETDATE()
```

```
);

-- Loyalty Accounts Table
CREATE TABLE LoyaltyAccounts (
  LoyaltyID INT PRIMARY KEY IDENTITY,
  UserID INT FOREIGN KEY REFERENCES Users(UserID),
  PointsBalance INT,
  LastUpdated DATETIME
);

-- Redemptions Table
CREATE TABLE Redemptions (
  RedemptionID INT PRIMARY KEY IDENTITY,
  UserID INT FOREIGN KEY REFERENCES Users(UserID),
  BookingID INT FOREIGN KEY REFERENCES Bookings(BookingID),
  PointsUsed INT,
  DiscountAmount DECIMAL(10, 2)
);
```

FrontEnd

```
SmartHotelBooking-Frontend/
├── src/
│   ├) for the SmartHotelBooking-Frontend project. This includes only── app/
│   │   ├── components/
│   │   │   ├── admin/
│   │   │   │   ├── the relevant parts visible from the code context.
```

```
SmartHotelBooking-Frontend/
├── src add-manager/
│   │   │   │   │   ├── add-manager/
│   └── app/
│       ├── app.config.ts
│   │   │   │   │   ├── add-manager.ts
│       ├── app.routes.ts
│       ├.html
│   │   │   │   ├── add-manager── components/
│           ├── admin/
│           .css
│   │       │   ├── admin-dashboard/
│   │       │   │   ├── admin-dashboard ├── add-manager/
│   │       │   │   ├── add-manager.ts
│   │       │   │   ├── add-manager.html
│   │       │   .ts
│   │       │   │   ├── admin-dashboard └── add-manager.css
│           .html
│   │       │   │   ├── admin-dashboard.css
│   │       │   ├── user-data/
│   │       ├ ├── admin-dashboard/
│   │       │   │   ├── admin-dashboard.ts
│       ── core/
│   │   │   │   ├── home/
│ │   │   │   ├── admin-dashboard│   │   │   │   │   ├── home.ts
│   │   │   │   ├── home.html
│   │   .html
│       │   │   │   └── admin-dashboard.css
│       │   │   │   ├── home.css
│   │       │   ├── navbar/
│   │       └── user-data/
│           │       └── user-data.ts (and related files)
│           ├── core/
│           │   ├── login/
│   │       │   ├── login.ts
│   │       ├ │   │   ├── home/
│           │   │   ├── home.ts
│           │   │   ├── home.html
│           │   │   └── home.css
│           │   ├── navbar/
│           │   │   └── navbar.html
│           │   └── (other core components)
│           ├── login.html
│   │       │   ├── login.css
│   │       ├── registration/
│   │   │   │   ├── registration.ts
```

```
│   │   │   │   ├── registration.html
│   │   │   │   ├── registration.css
│   │   ├── services/
│   │   │   ├── auth/
│   │   │   │   ├── auth.service.ts
│   ── login/
│   │   │   │   ├── login │   ├── app.config.ts
│   │   ├── app.routes.ts
│   ├── assets/
│   │   ├── Hotel1.webp
│   │   ├── Hotel2.webp
│   │   ├── Hotel3.webp
│   │   ├── Hotel4.webp
│   │   ├── Hotel_Icon.webp
│   ├── environments/
│   │   ├── environment.ts
│   │   ├── environment.prod.ts
│   ├── index.html
│   ├── main.ts
│   ├── styles.css
├── angular.json
├── package.json
├── README.md
```

BackEnd

SmartHotelBooking-Backend/

├── Controllers/

│ ├── AuthController.cs

│ ├── ManagersController.cs

│ ├── UserController.cs

│ ├── AdminController.cs

│ └── (Other controllers: BookingController.cs, HotelController.cs, etc.)

├── DTOs/

│ ├── RegisterDto.cs

│ ├── LoginDto.cs

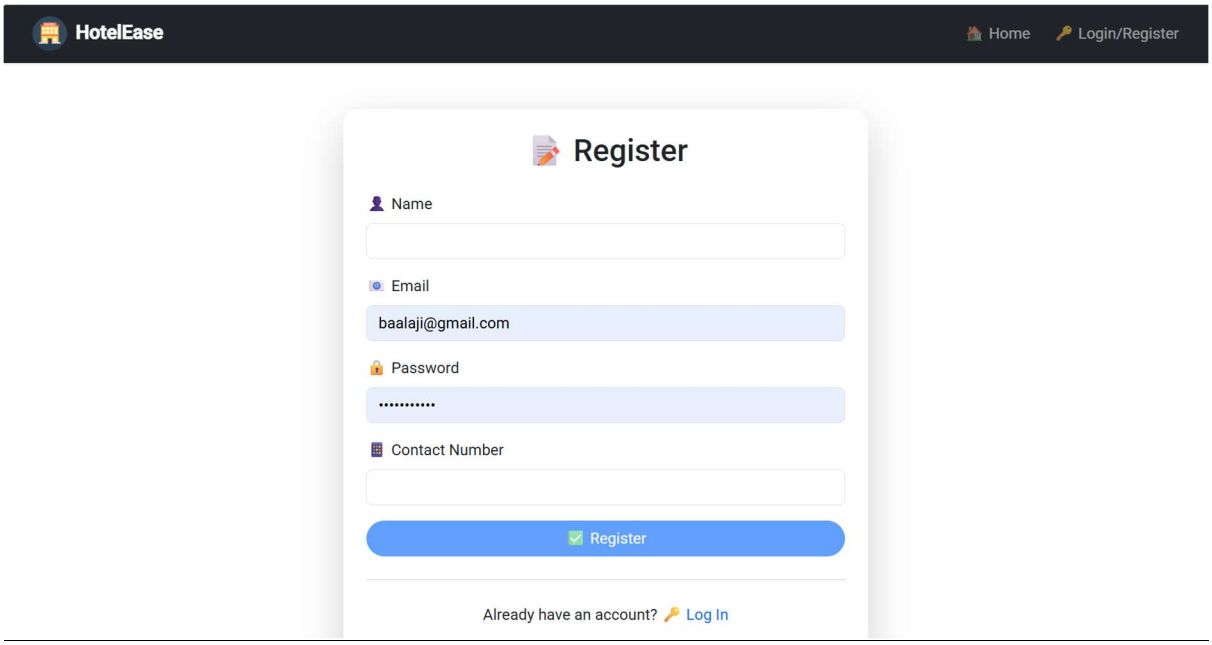
│ ├── AuthResponseDto.cs

│ ├── UserDTO.cs

```
│   └── (Other DTOs)
├── Models/
│   ├── User.cs
│   ├── Booking.cs
│   ├── Hotel.cs
│   └── (Other models)
├── Services/
│   ├── Interfaces/
│   │   ├── IAuthService.cs
│   │   ├── IUserService.cs
│   │   └── (Other interfaces)
│   └── Implementations/
│       ├── AuthService.cs
│       ├── UserService.cs
│       └── (Other services)
├── Repositories/
│   ├── Interfaces/
│   │   └── IUserRepository.cs
│   │   └── (Other interfaces)
│   └── Implementations/
│       └── UserRepository.cs
│       └── (Other implementations)
├── Helpers/
│   └── JwtService.cs
│   └── (Other helpers)
├── Data/
│   └── AppDbContext.cs
```

```
├── Mappings/
│   └── AutoMapperProfile.cs
├── appsettings.json
├── Program.cs
├── Startup.cs (if .NET 5/3.1) or in Program.cs (if .NET 6+)
└── (Other configuration and root files)
```

REGISTRATION



Registrartion.ts

```
import { Component, OnInit } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { RouterModule, Router } from '@angular/router';
import { AuthService } from '../../services/auth/auth.service';


@Component({
  selector: 'app-registration',
  standalone: true,
  templateUrl: './registration.html',
  styleUrls: ['./registration.css'],
  imports: [FormsModule, CommonModule, RouterModule],
})
export class Registration implements OnInit {
  user = {
```

```
  name: '',
  email: '',
  password: '',
  contactNumber: '',
  role: 'User' // ✅ Default role set here
 };

 constructor(private authService: AuthService, private router: Router) {}

 ngOnInit(): void {
  // Component initialized
 }

 onSubmit() {
  // Send the form data with role=User
  this.authService.register(this.user).subscribe({
    next: (res: any) => {
     alert('✅ Registration successful!');
     console.log(res);
     this.router.navigate(['/login']); // ✅ Redirect to Login
    },
    error: (err: any) => {
     alert('❌ Registration failed!');
     console.error(err);
    }
  });
 }
}
```

Registration.html

```
<div class="container mt-5 mb-5">
  <div class="row justify-content-center">
    <div class="col-md-7">
      <div class="card shadow-lg rounded-4 border-0">
        <div class="card-body p-4">
          <h2 class="text-center mb-4">📝 Register</h2>
          <form #form="ngForm" (ngSubmit)="onSubmit()">
            <!-- Name -->
            <div class="mb-3">
              <label for="name" class="form-label">👤 Name</label>
              <input
                type="text"
                id="name"
                name="name"
                class="form-control"
                [(ngModel)]="user.name"
                #nameRef="ngModel"
                required
              />
              <small class="text-danger" *ngIf="nameRef.invalid && nameRef.touched">
                <span *ngIf="nameRef.errors?.['required']">Name is required.</span>
              </small>
            </div>

            <!-- Email -->
            <div class="mb-3">
```

```
  <label for="email" class="form-label">📧 Email</label>
  <input
    type="email"
    id="email"
    name="email"
    class="form-control"
    [(ngModel)]="user.email"
    #emailRef="ngModel"
    required
    email
  />
  <small class="text-danger" *ngIf="emailRef.invalid && emailRef.touched">
    <span *ngIf="emailRef.errors?.['required']">Email is required.</span>
    <span *ngIf="emailRef.errors?.['email']">Enter a valid email address.</span>
  </small>
</div>

<!-- Password -->
<div class="mb-3">
  <label for="password" class="form-label">🔒 Password</label>
  <input
    type="password"
    id="password"
    name="password"
    class="form-control"
    [(ngModel)]="user.password"
    #passwordRef="ngModel"
    required
```

```
          minlength="8"
        />
        <small class="text-danger" *ngIf="passwordRef.invalid &&
passwordRef.touched">
          <span *ngIf="passwordRef.errors?.['required']">Password is required.</span>
          <span *ngIf="passwordRef.errors?.['minlength']">Minimum 8 characters
required.</span>
        </small>
      </div>

      <!-- Contact Number -->
      <div class="mb-3">
        <label for="contactNumber" class="form-label">📱 Contact Number</label>
        <input
          type="text"
          id="contactNumber"
          name="contactNumber"
          class="form-control"
          [(ngModel)]="user.contactNumber"
          #contactRef="ngModel"
          pattern="^[0-9]{10}$"
        />
        <small class="text-danger" *ngIf="contactRef.invalid && contactRef.touched">
          <span *ngIf="contactRef.errors?.['pattern']">Enter a valid 10-digit
number.</span>
        </small>
      </div>

      <!-- Submit Button -->
```

```
        <div class="d-grid">
         <button
          type="submit"
          class="btn btn-primary rounded-pill"
          [disabled]="form.invalid"
         >
          ✅ Register
         </button>
        </div>
       </form>

       <hr class="my-4" />

       <p class="text-center">
        Already have an account?
        <a routerLink="/login" class="text-decoration-none">🔑 Log In</a>
       </p>
      </div>
     </div>
    </div>
   </div>
</div>
```

Registration.css

```
/* src/app/components/registration/registration.css */

/* Container and card styling */
.container {
  margin-top: 40px;
```

```
  margin-bottom: 40px;
}

.card {
  border-radius: 1.5rem;
  box-shadow: 0 4px 24px rgba(44, 62, 80, 0.08);
  border: none;
}

/* Form titles */
h2.text-center {
  font-weight: 700;
  color: #007bff;
  margin-bottom: 1.5rem;
}

/* Form labels and inputs */
.form-label {
  font-weight: 500;
}

.form-control {
  border-radius: 0.75rem;
  padding: 0.6rem 1rem;
  font-size: 1.08rem;
}

.form-control:focus {
```

```
  box-shadow: 0 0 0 0.2rem rgba(0,123,255,0.08);
  border-color: #0d6efd;
}

/* Buttons */
.btn-primary.rounded-pill,
.btn-success.rounded-pill {
  padding: 0.6rem 0;
  font-size: 1.1rem;
  border-radius: 2rem !important;
  font-weight: 600;
}

.d-grid {
  margin-top: 1.4rem;
}

/* Validation error messages */
.text-danger {
  font-size: 0.98rem;
  font-weight: 400;
}

/* Responsive adjustments */
@media (max-width: 768px) {
  .container {
    margin-top: 20px;
    margin-bottom: 20px;
```

```
  }
  .card {
    margin: 0 0.3rem;
  }
  h2.text-center {
    font-size: 1.5rem;
  }
}

/* Extra spacing for the login link */
hr.my-4 {
  margin-top: 2rem;
  margin-bottom: 2rem;
}

p.text-center {
  margin-top: 1.2rem;
  font-size: 1.04rem;
}

a.text-decoration-none {
  color: #007bff;
  font-weight: 500;
}

a.text-decoration-none:hover {
  text-decoration: underline;
}
```

User.cs

```
using System;
using System.Collections.Generic;

namespace SmartHotelBooking.Models;

public partial class User
{
    public int UserId { get; set; }
    public string? Name { get; set; }
    public string? Email { get; set; }
    public string? PasswordHash { get; set; }
    public string? Role { get; set; }
    public string? ContactNumber { get; set; }

    public virtual ICollection<Booking> Bookings { get; set; } = new List<Booking>();
    public virtual ICollection<Hotel> Hotels { get; set; } = new List<Hotel>();
    public virtual ICollection<LoyaltyAccount> LoyaltyAccounts { get; set; } = new
List<LoyaltyAccount>();
    public virtual ICollection<Payment> Payments { get; set; } = new List<Payment>();
    public virtual ICollection<Redemption> Redemptions { get; set; } = new
List<Redemption>();
    public virtual ICollection<Review> Reviews { get; set; } = new List<Review>();
}
```

registerDto.cs

```
using System.ComponentModel.DataAnnotations;

namespace SmartHotelBooking.DTOs
```

```
{
   public class RegisterDto
   {
      [Required(ErrorMessage = "Name is required.")]
      [StringLength(50, MinimumLength = 2, ErrorMessage = "Name must be between 2 and 50 characters.")]
      public string? Name { get; set; }

      [Required(ErrorMessage = "Email is required.")]
      [EmailAddress(ErrorMessage = "Invalid email format.")]
      public string? Email { get; set; }

      [Required(ErrorMessage = "Password is required.")]
      [StringLength(100, MinimumLength = 8, ErrorMessage = "Password must be at least 8 characters.")]
      public string? Password { get; set; }

      [Required(ErrorMessage = "Role is required.")]
      [RegularExpression("^(Admin|User|Manager)$", ErrorMessage = "Role must be Admin, User, or Manager.")]
      public string? Role { get; set; }

      [Required(ErrorMessage = "Contact Number is required.")]
      [Phone(ErrorMessage = "Invalid contact number format.")]
      public string? ContactNumber { get; set; }
   }
}
```

IUserRepository.cs

```
using SmartHotelBooking.Models;
```

```
using System.Threading.Tasks;

namespace SmartHotelBooking.Repositories.Interfaces
{
  public interface IUserRepository
  {
    Task<User> AddAsync(User user);
    Task<User?> GetByEmailAsync(string email);
    // Other CRUD methods as needed
  }
}
```

IAuthService.cs

```
using SmartHotelBooking.DTOs;
using System.Threading.Tasks;

namespace SmartHotelBooking.Services.Interfaces
{
  public interface IAuthService
  {
    Task<AuthResponseDto> RegisterAsync(RegisterDto registerDto);
    Task<AuthResponseDto> LoginAsync(LoginDto loginDto);
  }
}
```

AuthService.cs

```
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;
using SmartHotelBooking.Services.Interfaces;
```

```
using System.Threading.Tasks;

public class AuthService : IAuthService
{
    private readonly IUserRepository _userRepository;
    private readonly IJwtService _jwtService;

    public AuthService(IUserRepository userRepository, IJwtService jwtService)
    {
        _userRepository = userRepository;
        _jwtService = jwtService;
    }

    public async Task<AuthResponseDto> RegisterAsync(RegisterDto registerDto)
    {
        var hashedPassword = BCrypt.Net.BCrypt.HashPassword(registerDto.Password);

        var newUser = new User
        {
            Name = registerDto.Name,
            Email = registerDto.Email,
            PasswordHash = hashedPassword,
            Role = registerDto.Role,
            ContactNumber = registerDto.ContactNumber
        };

        var createdUser = await _userRepository.AddAsync(newUser);
```

```
        var token = _jwtService.GenerateToken(createdUser);

        return new AuthResponseDto
        {
            Message = "Registration SuccessFull",
            Name = createdUser.Name,
            Email = createdUser.Email,
            Role = createdUser.Role,
            UserId = createdUser.UserId,
            // Token = token
        };
    }

    // Other methods like LoginAsync...
}
```

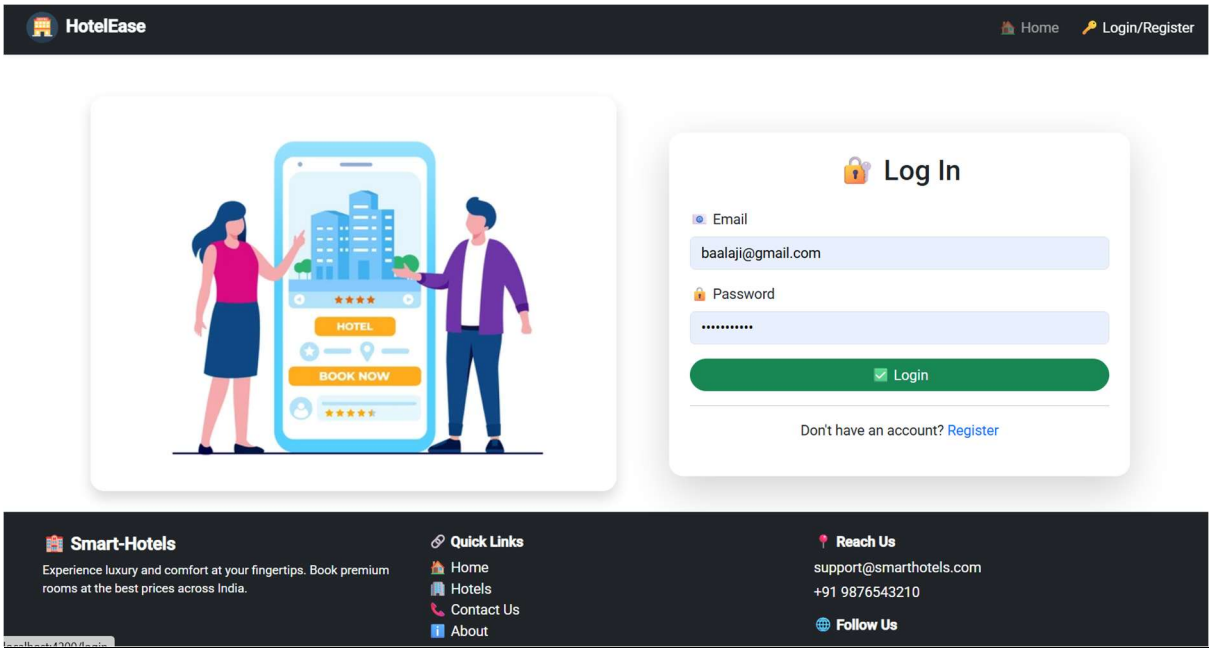
UserService.cs

```
using AutoMapper;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;
using SmartHotelBooking.Services.Interfaces;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace SmartHotelBooking.Services.Implementations
{
    public class UserService : IUserService
    {
```

```
private readonly IUserRepository _userRepository;
private readonly IMapper _mapper;

public UserService(IUserRepository userRepository, IMapper mapper)
{
   _userRepository = userRepository;
   _mapper = mapper;
}

public async Task<IEnumerable<UserDTO>> GetAllUsersAsync()
{
   var users = await _userRepository.GetAllAsync();
   return _mapper.Map<IEnumerable<UserDTO>>(users);
}

public async Task<UserDTO> GetUserByIdAsync(int id)
{
   var user = await _userRepository.GetByIdAsync(id);
   return _mapper.Map<UserDTO>(user);
}

public async Task<bool> UpdateUserAsync(int id, UpdateUserDto dto)
{
   var user = await _userRepository.GetByIdAsync(id);
   if (user == null)
      return false;

   user.Name = dto.Name;
```

```
        user.Email = dto.Email;
        user.ContactNumber = dto.ContactNumber;
        user.Role = dto.Role;

        await _userRepository.UpdateAsync(user);
        return true;
      }

      public async Task<bool> DeleteUserAsync(int id)
      {
        var user = await _userRepository.GetByIdAsync(id);
        if (user == null)
          return false;

        await _userRepository.DeleteAsync(user);
        return true;
      }
    }
}
```

Login



Login.ts

```
import { Component } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from '../../services/auth/auth.service';


@Component({
  selector: 'app-login',
  standalone: true,
  templateUrl: './login.html',
  styleUrls: ['./login.css']
})
export class Login {
  credentials = {
    email: '',
    password: ''
  };


  constructor(private authService: AuthService, private router: Router) {}
```

```
  onSubmit() {
   this.authService.login(this.credentials).subscribe({
     next: (res: any) => {
      this.authService.saveUserData(res);
      alert('✅ Login successful!');
      this.router.navigate(['/home']);
     },
     error: (err: any) => {
      alert('❌ Login failed! Please check your credentials.');
      console.error(err);
     }
    });
  }
}
```

Login.html

```
<div class="container mt-5 mb-5">
  <div class="row justify-content-center">
    <div class="col-md-7">
      <div class="card shadow-lg rounded-4 border-0">
        <div class="card-body p-4">
          <h2 class="text-center mb-4">🔐 Log In</h2>
          <form #form="ngForm" (ngSubmit)="onSubmit()">
            <!-- Email -->
            <div class="mb-3">
              <label for="email" class="form-label">📧 Email</label>
              <input
                type="email"
```

```
    id="email"
    name="email"
    class="form-control"
    [(ngModel)]="credentials.email"
    #emailRef="ngModel"
    required
    email
   />
   <small class="text-danger" *ngIf="emailRef.invalid && emailRef.touched">
    <span *ngIf="emailRef.errors?.['required']">Email is required.</span>
    <span *ngIf="emailRef.errors?.['email']">Enter a valid email address.</span>
   </small>
  </div>

  <!-- Password -->
  <div class="mb-3">
   <label for="password" class="form-label">🔒 Password</label>
   <input
    type="password"
    id="password"
    name="password"
    class="form-control"
    [(ngModel)]="credentials.password"
    #passwordRef="ngModel"
    required
    minlength="8"
   />
```

```
          <small class="text-danger" *ngIf="passwordRef.invalid &&
passwordRef.touched">
            <span *ngIf="passwordRef.errors?.['required']">Password is required.</span>
            <span *ngIf="passwordRef.errors?.['minlength']">Minimum 8 characters
required.</span>
          </small>
        </div>

        <!-- Submit -->
        <div class="d-grid">
          <button
            type="submit"
            class="btn btn-success rounded-pill"
            [disabled]="form.invalid"
          >
            ✅ Login
          </button>
        </div>
      </form>

      <hr />
      <p class="text-center mt-3">
        Don't have an account?
        <a routerLink="/register" class="text-decoration-none">Register</a>
      </p>
    </div>
  </div>
</div>
```

```
  </div>
</div>
```

Login.css

```
/* src/app/components/login/login.css */

/* Container and card styling */
.container {
  margin-top: 40px;
  margin-bottom: 40px;
}

.card {
  border-radius: 1.5rem;
  box-shadow: 0 4px 24px rgba(44, 62, 80, 0.08);
  border: none;
}

/* Form titles */
h2.text-center {
  font-weight: 700;
  color: #28a745;
  margin-bottom: 1.5rem;
}

/* Form labels and inputs */
.form-label {
  font-weight: 500;
}
```

```
.form-control {
  border-radius: 0.75rem;
  padding: 0.6rem 1rem;
  font-size: 1.08rem;
}

.form-control:focus {
  box-shadow: 0 0 0 0.2rem rgba(40,167,69,0.08);
  border-color: #28a745;
}

/* Buttons */
.btn-success.rounded-pill {
  padding: 0.6rem 0;
  font-size: 1.1rem;
  border-radius: 2rem !important;
  font-weight: 600;
}

.d-grid {
  margin-top: 1.4rem;
}

/* Validation error messages */
.text-danger {
  font-size: 0.98rem;
  font-weight: 400;
```

```
}

/* Responsive adjustments */
@media (max-width: 768px) {
  .container {
    margin-top: 20px;
    margin-bottom: 20px;
  }
  .card {
    margin: 0 0.3rem;
  }
  h2.text-center {
    font-size: 1.5rem;
  }
}

hr {
  margin-top: 2rem;
  margin-bottom: 2rem;
}

p.text-center {
  margin-top: 1.2rem;
  font-size: 1.04rem;
}

a.text-decoration-none {
  color: #28a745;
```

```
  font-weight: 500;
}

a.text-decoration-none:hover {
  text-decoration: underline;
}
```

Auth.service.cs

```
// AuthService for handling user authentication

import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { BehaviorSubject, Observable, catchError, tap } from 'rxjs';
import { environment } from '../../../environments/environment';
import { CookieService } from 'ngx-cookie-service';

@Injectable({
  providedIn: 'root'
})

export class AuthService {
  private loggedIn: BehaviorSubject<boolean>;
  private role: BehaviorSubject<string>;

  constructor(private http: HttpClient, private cookieService: CookieService) {
    this.loggedIn = new BehaviorSubject<boolean>(this.cookieService.check('token'));
    this.role = new BehaviorSubject<string>(this.cookieService.get('role') || '');
  }
```

```
get isLoggedIn$(): Observable<boolean> {
  return this.loggedIn.asObservable();
}

get role$(): Observable<string> {
  return this.role.asObservable();
}

// Register method
register(userData: any): Observable<any> {
  return this.http.post(` ${environment.apiBaseUrl}/Auth/register` , userData).pipe(
    tap(response => console.log('API Response:', response)),
    catchError(error => {
      console.error('API Error:', error);
      throw error;
    })
  );
}

// Login method
login(credentials: { email: string; password: string }): Observable<any> {
  return this.http.post<any>(` ${environment.apiBaseUrl}/Auth/login` , credentials);
}

// Save user data and update state
saveUserData(data: any): void {
  this.cookieService.set('token', data.token);
  this.cookieService.set('role', data.role);
```

```
    this.cookieService.set('name', data.name);
    this.cookieService.set('email', data.email);
    this.cookieService.set('userId', data.userId);

    this.loggedIn.next(true);
    this.role.next(data.role);
  }

  // ... (other helper methods)
}
```

LoginDto.cs

```
using System.ComponentModel.DataAnnotations;

namespace SmartHotelBooking.DTOs
{
    public class LoginDto
    {
        [Required(ErrorMessage = "Email is required.")]
        [EmailAddress(ErrorMessage = "Invalid email format.")]
        public string Email { get; set; } = string.Empty;

        [Required(ErrorMessage = "Password is required.")]
        [StringLength(100, MinimumLength = 8, ErrorMessage = "Password must be at least 8 characters.")]
        public string Password { get; set; } = string.Empty;
    }
}
```

IUserRepository.cs

```
using SmartHotelBooking.Models;
using System.Threading.Tasks;

namespace SmartHotelBooking.Repositories.Interfaces
{
  public interface IUserRepository
  {
    Task<User> AddAsync(User user);
    Task<User?> GetByEmailAsync(string email);
    // Other CRUD methods as needed
  }
}
```

IAuthService.cs

```
using SmartHotelBooking.DTOs;
using System.Threading.Tasks;

namespace SmartHotelBooking.Services.Interfaces
{
  public interface IAuthService
  {
    Task<AuthResponseDto> RegisterAsync(RegisterDto registerDto);
    Task<AuthResponseDto> LoginAsync(LoginDto loginDto);
  }
}
```

AuthService.cs

```
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
```

```
using SmartHotelBooking.Repositories.Interfaces;
using SmartHotelBooking.Services.Interfaces;
using System.Threading.Tasks;

public class AuthService : IAuthService
{
  private readonly IUserRepository _userRepository;
  private readonly IJwtService _jwtService;

  public AuthService(IUserRepository userRepository, IJwtService jwtService)
  {
    _userRepository = userRepository;
    _jwtService = jwtService;
  }

  public async Task<AuthResponseDto> LoginAsync(LoginDto loginDto)
  {
    var user = await _userRepository.GetByEmailAsync(loginDto.Email);

    if (user == null)
      throw new UnauthorizedAccessException("Invalid email or password.");

    if (!BCrypt.Net.BCrypt.Verify(loginDto.Password, user.PasswordHash))
      throw new UnauthorizedAccessException("Invalid email or password! Please
Register.");

    var token = _jwtService.GenerateToken(user);
```

```
        return new AuthResponseDto
        {
            Message = "Login SuccessFull",
            Name = user.Name,
            Email = user.Email,
            Role = user.Role,
            Token = token,
            UserId = user.UserId
        };
    }

    // ...RegisterAsync and other methods
}
```

UserService.cs

```
using AutoMapper;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;
using SmartHotelBooking.Services.Interfaces;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace SmartHotelBooking.Services.Implementations
{
    public class UserService : IUserService
    {
        private readonly IUserRepository _userRepository;
        private readonly IMapper _mapper;
```

```
public UserService(IUserRepository userRepository, IMapper mapper)
{
   _userRepository = userRepository;
   _mapper = mapper;
}

public async Task<IEnumerable<UserDTO>> GetAllUsersAsync()
{
   var users = await _userRepository.GetAllAsync();
   return _mapper.Map<IEnumerable<UserDTO>>(users);
}

public async Task<UserDTO> GetUserByIdAsync(int id)
{
   var user = await _userRepository.GetByIdAsync(id);
   return _mapper.Map<UserDTO>(user);
}

public async Task<bool> UpdateUserAsync(int id, UpdateUserDto dto)
{
   var user = await _userRepository.GetByIdAsync(id);
   if (user == null)
      return false;

   user.Name = dto.Name;
   user.Email = dto.Email;
   user.ContactNumber = dto.ContactNumber;
```

```
        user.Role = dto.Role;

        await _userRepository.UpdateAsync(user);
        return true;
      }

      public async Task<bool> DeleteUserAsync(int id)
      {
        var user = await _userRepository.GetByIdAsync(id);
        if (user == null)
          return false;

        await _userRepository.DeleteAsync(user);
        return true;
      }
    }
}
```

AuthResponseDto.cs

```
namespace SmartHotelBooking.DTOs
{
  public class AuthResponseDto
  {
    public string Message { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public string Role { get; set; }
    public int UserId { get; set; }
    public string Token { get; set; }
```

```
    }
}
```

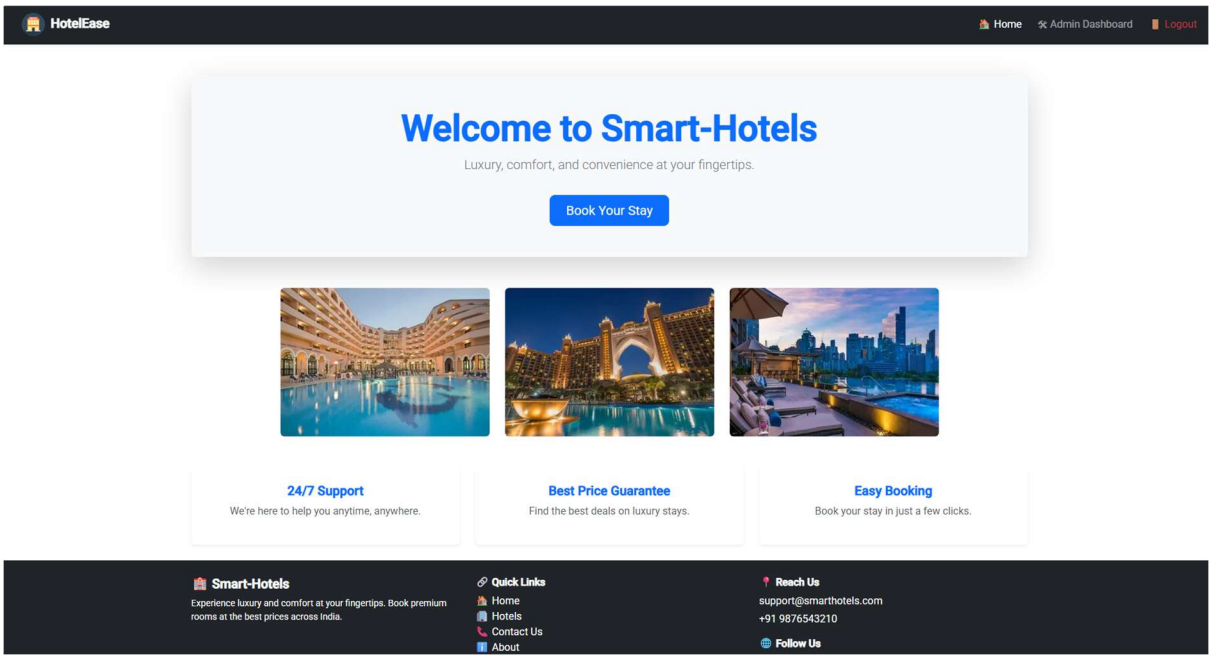
AuthController.cs

```
using Microsoft.AspNetCore.Mvc;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Services.Interfaces;
using System.Threading.Tasks;

namespace SmartHotelBooking.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class AuthController : ControllerBase
    {
        private readonly IAuthService _authService;
        public AuthController(IAuthService authService)
        {
            _authService = authService;
        }

        [HttpPost("login")]
        [AllowAnonymous]
        public async Task<IActionResult> Login([FromBody] LoginDto dto)
        {
            try
            {
                var response = await _authService.LoginAsync(dto);
                return Ok(response);
```

```
        }
        catch (UnauthorizedAccessException ex)
        {
           return Unauthorized(new { message = ex.Message });
        }
      }
   }
}
```

ADMIN ROLE

Home



Home.ts

```
import { Component } from '@angular/core';
import { AuthService } from '../../../services/auth/auth.service';
import { Router } from '@angular/router';


@Component({
```

```
  selector: 'app-home',
  imports: [],
  templateUrl: './home.html',
  styleUrl: './home.css',
})

export class Home {
  userRole: string = '';
  constructor(private router: Router, private authService: AuthService) {
    this.userRole = this.authService.getRole() || ''; // 🔥 Get role from cookie
  }

  handleBooking() {
    if (this.authService.isLoggedIn()) {
      this.router.navigate(['/hotel']);
    } else {
      this.router.navigate(['/login']);
    }
  }
}
```

Home.html

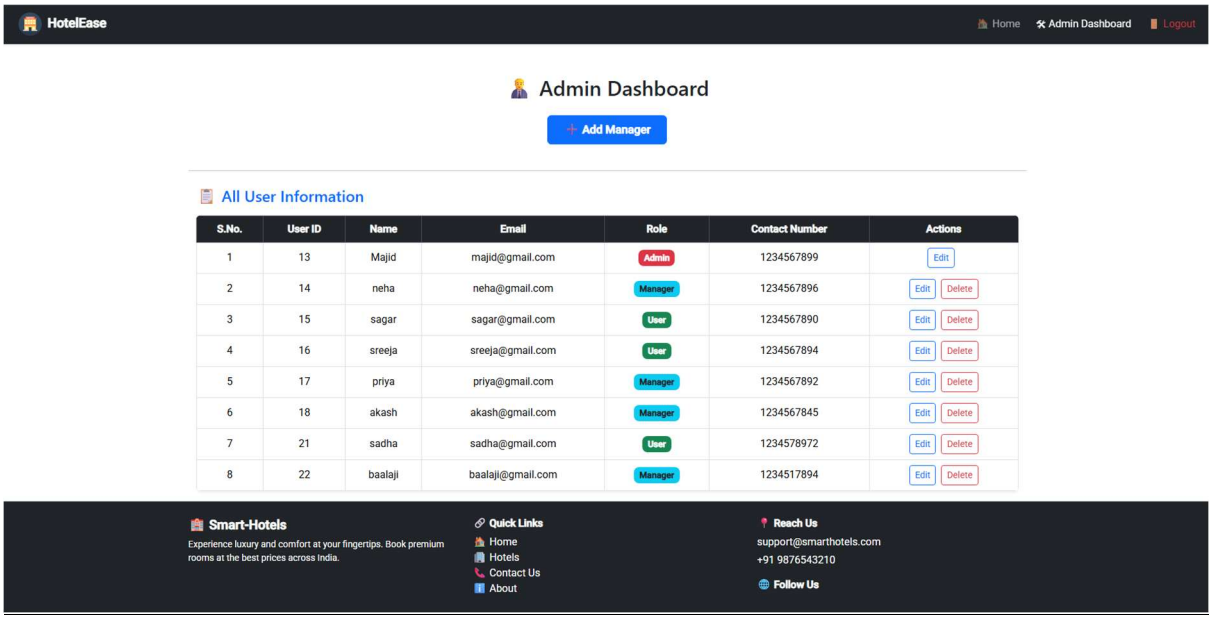
```
<div class="container mt-5">

  <!-- 🔥 Hero Section -->
  <div class="text-center bg-light p-5 rounded shadow-lg mb-5">
    <h1 class="display-4 fw-bold text-primary">Welcome to Smart-Hotels</h1>
    <p class="lead text-secondary">Luxury, comfort, and convenience at your fingertips.</p>
```

```
  <button class="btn btn-primary btn-lg mt-3 px-4" [disabled]="userRole === 'Admin'" (click)="handleBooking()"  >Book Your Stay</button>
 </div>

 <!-- 🏨 Hotel Image Carousel -->
 <div id="hotelCarousel" class="carousel slide" data-bs-ride="carousel">
  <div class="carousel-inner">

   <!-- Slide 1 -->
   <div class="carousel-item active">
    <div class="d-flex justify-content-center gap-4">
     <img src="assets/Hotel1.webp" class="rounded shadow-sm w-25" alt="Hotel 1">
     <img src="assets/Hotel2.webp" class="rounded shadow-sm w-25" alt="Hotel 2">
     <img src="assets/Hotel3.webp" class="rounded shadow-sm w-25" alt="Hotel 3">
    </div>
   </div>

   <!-- Slide 2 -->
   <div class="carousel-item">
    <div class="d-flex justify-content-center gap-4">
     <img src="assets/Hotel2.webp" class="rounded shadow-sm w-25" alt="Hotel 2">
     <img src="assets/Hotel3.webp" class="rounded shadow-sm w-25" alt="Hotel 3">
     <img src="assets/Hotel4.webp" class="rounded shadow-sm w-25" alt="Hotel 4">
    </div>
   </div>

   <!-- Slide 3 -->
   <div class="carousel-item">
```

```
      <div class="d-flex justify-content-center gap-4">
        <img src="assets/Hotel3.webp" class="rounded shadow-sm w-25" alt="Hotel 3">
        <img src="assets/Hotel4.webp" class="rounded shadow-sm w-25" alt="Hotel 4">
        <img src="assets/Hotel1.webp" class="rounded shadow-sm w-25" alt="Hotel 1">
      </div>
    </div>


  </div>


  <!-- Carousel Controls -->
  <button class="carousel-control-prev" type="button" data-bs-
target="#hotelCarousel" data-bs-slide="prev">
    <span class="carousel-control-prev-icon"></span>
  </button>
  <button class="carousel-control-next" type="button" data-bs-
target="#hotelCarousel" data-bs-slide="next">
    <span class="carousel-control-next-icon"></span>
  </button>
 </div>


 <!-- 💼 Services Section -->
 <div class="row text-center mt-5">
  <div class="col-md-4 mb-4">
   <div class="p-4 shadow-sm rounded bg-white h-100">
    <h5 class="fw-bold text-primary">24/7 Support</h5>
    <p class="text-muted">We're here to help you anytime, anywhere.</p>
   </div>
  </div>
```

```
<div class="col-md-4 mb-4">
  <div class="p-4 shadow-sm rounded bg-white h-100">
    <h5 class="fw-bold text-primary">Best Price Guarantee</h5>
```

Admin Dashboard



Admin-Dashboard.ts

```
import { Component } from '@angular/core';
import { RouterModule, Router } from '@angular/router';
import { CommonModule } from '@angular/common';
import { UserData } from '../user-data/user-data';


@Component({
  selector: 'app-admin-dashboard',
  standalone: true,
  imports: [CommonModule, RouterModule, UserData],
  templateUrl: './admin-dashboard.html',
  styleUrls: ['./admin-dashboard.css']
})
```

```
export class AdminDashboard {
  constructor(private router: Router) {}

  goToAddManager() {
    this.router.navigate(['/add-manager']);
  }
}
```

Admin-dashboard.html

```
<div class="container mt-5">
  <div class="text-center">
    <h2 class="mb-4">👨‍💼 Admin Dashboard</h2>
    <button class="btn btn-primary mb-4 px-4 py-2 fw-bold" (click)="goToAddManager()">
      ➕ Add Manager
    </button>
  </div>

  <hr>

  <app-user-data></app-user-data>

</div>
```

Admin-dashboard.css

```
h2 {
  font-family: 'Segoe UI', sans-serif;
}

button {
```

```
  font-size: 1.1rem;
  transition: all 0.3s;
}

button:hover {
  background-color: #0d6efd;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.2);
}
```

Admin.service.ts

```
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;
using SmartHotelBooking.Services.Interfaces;
using System.Threading.Tasks;

namespace SmartHotelBooking.Services.Implementations
{
    public class AuthService : IAuthService
    {
        private readonly IUserRepository _userRepository;
        private readonly IJwtService _jwtService;

        public AuthService(IUserRepository userRepository, IJwtService jwtService)
        {
            _userRepository = userRepository;
            _jwtService = jwtService;
        }
```

```
public async Task<AuthResponseDto> RegisterAsync(RegisterDto registerDto)
{
   var hashedPassword = BCrypt.Net.BCrypt.HashPassword(registerDto.Password);

   var newUser = new User
   {
      Name = registerDto.Name,
      Email = registerDto.Email,
      PasswordHash = hashedPassword,
      Role = registerDto.Role,
      ContactNumber = registerDto.ContactNumber
   };

   var createdUser = await _userRepository.AddAsync(newUser);

   var token = _jwtService.GenerateToken(createdUser);

   return new AuthResponseDto
   {
      Message = "Registration SuccessFull",
      Name = createdUser.Name,
      Email = createdUser.Email,
      Role = createdUser.Role,
      UserId = createdUser.UserId,
      Token = token
   };
}
```

```
    // ...LoginAsync and other methods
  }
}
```

User.cs

```
using System.Collections.Generic;

namespace SmartHotelBooking.Models
{
  public class User
  {
    public int UserId { get; set; }
    public string? Name { get; set; }
    public string? Email { get; set; }
    public string? PasswordHash { get; set; }
    public string? Role { get; set; }
    public string? ContactNumber { get; set; }

    // Navigation properties (optional for related data)
    public virtual ICollection<Booking> Bookings { get; set; } = new List<Booking>();
    public virtual ICollection<Hotel> Hotels { get; set; } = new List<Hotel>();
    // ... other navigation properties as needed
  }
}
```

UserDto.cs

```
namespace SmartHotelBooking.DTOs
{
  public class UserDTO
  {
```

```
        public int UserId { get; set; }
        public string? Name { get; set; }
        public string? Email { get; set; }
        public string? Role { get; set; }
        public string? ContactNumber { get; set; }
    }
}
```

IUserRepository.cs

```
namespace SmartHotelBooking.DTOs
{
    public class UserDTO
    {
        public int UserId { get; set; }
        public string? Name { get; set; }
        public string? Email { get; set; }
        public string? Role { get; set; }
        public string? ContactNumber { get; set; }
    }
}
```

UserRepository.cs

```
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;
using Microsoft.EntityFrameworkCore;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace SmartHotelBooking.Repositories.Implementations
{
```

```
    public class UserRepository : IUserRepository
    {
        private readonly AppDbContext _context;
        public UserRepository(AppDbContext context)
        {
            _context = context;
        }

        public async Task<IEnumerable<User>> GetAllAsync()
        {
            return await _context.Users.ToListAsync();
        }

        public async Task<User?> GetByIdAsync(int id)
        {
            return await _context.Users.FindAsync(id);
        }

        // ... other methods (AddAsync, DeleteAsync, etc.)
    }
}
```

IUserService.cs

```
using SmartHotelBooking.DTOs;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace SmartHotelBooking.Services.Interfaces
{
```

```
    public interface IUserService
    {
        Task<IEnumerable<UserDTO>> GetAllUsersAsync();
        Task<UserDTO?> GetUserByIdAsync(int id);
        // ... other methods as needed
    }
}
```

UserService.cs

```
using AutoMapper;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;
using SmartHotelBooking.Services.Interfaces;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace SmartHotelBooking.Services.Implementations
{
    public class UserService : IUserService
    {
        private readonly IUserRepository _userRepository;
        private readonly IMapper _mapper;

        public UserService(IUserRepository userRepository, IMapper mapper)
        {
            _userRepository = userRepository;
            _mapper = mapper;
        }
```

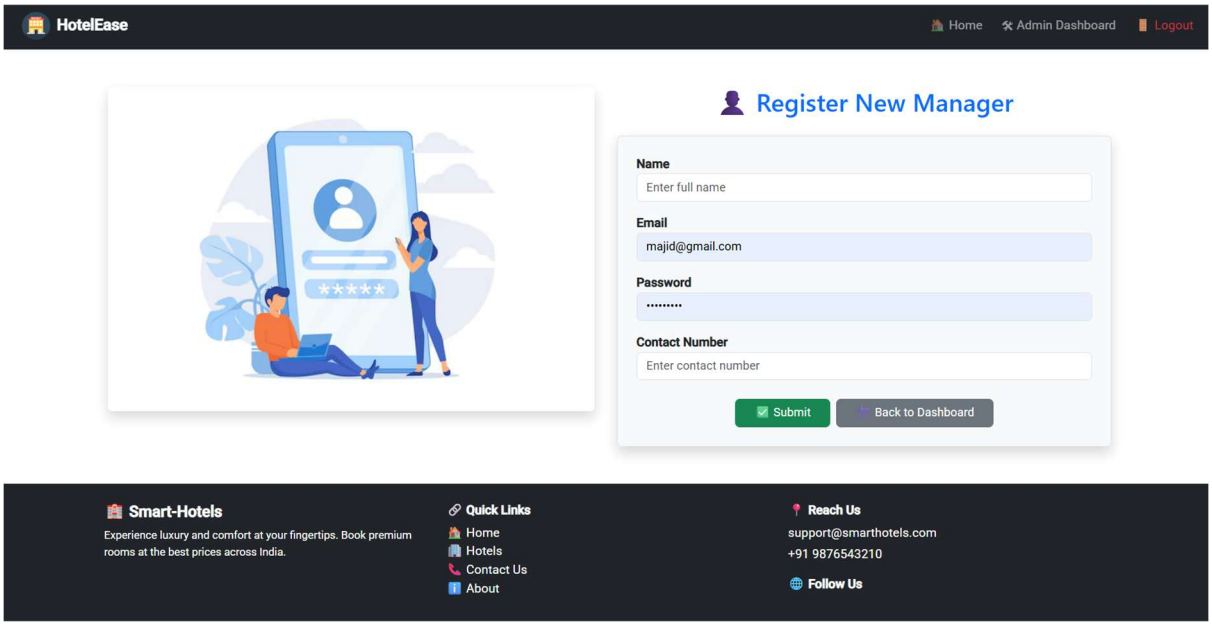
```
    public async Task<IEnumerable<UserDTO>> GetAllUsersAsync()
    {
      var users = await _userRepository.GetAllAsync();
      return _mapper.Map<IEnumerable<UserDTO>>(users);
    }

    public async Task<UserDTO?> GetUserByIdAsync(int id)
    {
      var user = await _userRepository.GetByIdAsync(id);
      return user == null ? null : _mapper.Map<UserDTO>(user);
    }
  }
}
```

UserController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Services.Interfaces;
using System.Threading.Tasks;

namespace SmartHotelBooking.Controllers
{
  [Route("api/[controller]")]
  [ApiController]
  [Authorize(Roles = "Admin,Manager")]
  public class UserController : ControllerBase
  {
```

```
        private readonly IUserService _userService;

        public UserController(IUserService userService)
        {
            _userService = userService;
        }

        // GET: api/user
        [HttpGet]
        public async Task<IActionResult> GetAll()
        {
            var users = await _userService.GetAllUsersAsync();
            return Ok(users);
        }

        // GET: api/user/{id}
        [HttpGet("{id}")]
        public async Task<IActionResult> Get(int id)
        {
            var user = await _userService.GetUserByIdAsync(id);
            if (user == null)
                return NotFound();
            return Ok(user);
        }
    }
}
```

Register New Manager



Add-manager.ts

```
import { Component } from '@angular/core';

import { Router } from '@angular/router';

import { FormBuilder, FormGroup, ReactiveFormsModule, Validators } from '@angular/forms';

import { HttpClient } from '@angular/common/http';

import { CommonModule } from '@angular/common';


@Component({
  selector: 'app-add-manager',
  standalone: true,
  imports: [CommonModule, ReactiveFormsModule],
  templateUrl: './add-manager.html',
  styleUrls: ['./add-manager.css']
})
export class AddManager {
  managerForm: FormGroup;
```

```
constructor(
  private fb: FormBuilder,
  private http: HttpClient,
  private router: Router
) {
  this.managerForm = this.fb.group({
    name: ['', Validators.required],
    email: ['', [Validators.required, Validators.email]],
    password: ['', Validators.required],
    contactNumber: ['', Validators.required]
  });
}

onSubmit() {
  if (this.managerForm.valid) {
    const data = {
      ...this.managerForm.value,
      role: 'Manager'
    };

    this.http.post('http://localhost:5281/api/Managers/register', data).subscribe({
      next: (res: any) => {
        alert(res.message);
        console.log('Manager Registered:', res);
        this.router.navigate(['/admin-dashboard']);
      },
      error: (err) => {
```

```
        console.error('Registration Failed:', err);
        alert('Something went wrong');
      }
    });
  }
 }

 goBack() {
  this.router.navigate(['/admin-dashboard']);
 }
}
```

Add-manager.html

```
<!-- src/app/components/admin/add-manager/add-manager.html -->
<div class="container mt-5 mb-5">
 <div class="row justify-content-center align-items-start">

  <!-- 🖼️ Enlarged Image Column -->
  <div class="col-md-6 mb-4 text-center d-flex justify-content-center align-items-center">
    <img
     src="https://img.freepik.com/premium-vector/user-verification-unauthorized-access-prevention-private-account-authentication-cyber-security_566886-2817.jpg"
     alt="Add Manager"
     class="img-fluid rounded shadow"
     style="height: 100%; max-height: 460px; object-fit: cover; border-radius: 20px;"
    />
  </div>
```

```
  <!-- 📋 Form Column -->
  <div class="col-md-6">
    <h2 class="text-center text-primary mb-4">👤 Register New Manager</h2>

    <form [formGroup]="managerForm" (ngSubmit)="onSubmit()" class="border p-4 shadow rounded bg-light">
      <div class="mb-3">
        <label>Name</label>
        <input type="text" class="form-control" formControlName="name" placeholder="Enter full name">
      </div>

      <div class="mb-3">
        <label>Email</label>
        <input type="email" class="form-control" formControlName="email" placeholder="Enter email address">
      </div>

      <div class="mb-3">
        <label>Password</label>
        <input type="password" class="form-control" formControlName="password" placeholder="Choose a password">
      </div>

      <div class="mb-3">
        <label>Contact Number</label>
        <input type="text" class="form-control" formControlName="contactNumber" placeholder="Enter contact number">
      </div>
```

```
      <div class="text-center mt-4">
        <button class="btn btn-success me-2 px-4" type="submit">✅ Submit</button>
        <button class="btn btn-secondary px-4" type="button" (click)="goBack()">🔙 Back to Dashboard</button>
      </div>
    </form>
   </div>
  </div>
</div>
```

Add-manager.css

```
h2.text-center {
  font-family: 'Segoe UI', sans-serif;
  font-weight: 700;
}

form {
  background-color: #f9f9f9;
  border-radius: 1rem;
}

label {
  font-weight: 500;
}

input.form-control {
  border-radius: 0.7rem;
  font-size: 1.07rem;
```

```
}

button.btn-success {
  font-size: 1.1rem;
  border-radius: 2rem;
  font-weight: 600;
}

button.btn-secondary {
  border-radius: 2rem;
}

button[type='submit']:hover {
  background-color: #198754;
}
```

ManagersController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Services.Interfaces;
using System.Threading.Tasks;

namespace SmartHotelBooking.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class ManagersController : ControllerBase
    {
```

```
    private readonly IAuthService _authService;
    public ManagersController(IAuthService authService)
    {
      _authService = authService;
    }

    // Only Admins can register Managers
    [HttpPost("register")]
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> Register([FromBody] RegisterDto dto)
    {
      // Make sure the role is always "Manager" for this endpoint
      dto.Role = "Manager";
      var response = await _authService.RegisterAsync(dto);
      return Ok(response);
    }
  }
}
```

RgisterDto.cs

```
using System.ComponentModel.DataAnnotations;

namespace SmartHotelBooking.DTOs
{
  public class RegisterDto
  {
    [Required(ErrorMessage = "Name is required.")]
    [StringLength(50, MinimumLength = 2, ErrorMessage = "Name must be between 2 and 50 characters.")]
```

```
        public string? Name { get; set; }

        [Required(ErrorMessage = "Email is required.")]
        [EmailAddress(ErrorMessage = "Invalid email format.")]
        public string? Email { get; set; }

        [Required(ErrorMessage = "Password is required.")]
        [StringLength(100, MinimumLength = 8, ErrorMessage = "Password must be at least 8 characters.")]
        public string? Password { get; set; }

        [Required(ErrorMessage = "Role is required.")]
        [RegularExpression("^(Admin|User|Manager)$", ErrorMessage = "Role must be Admin, User, or Manager.")]
        public string? Role { get; set; }

        [Required(ErrorMessage = "Contact Number is required.")]
        [Phone(ErrorMessage = "Invalid contact number format.")]
        public string? ContactNumber { get; set; }
    }
}
```

User.cs

```
using System.Collections.Generic;

namespace SmartHotelBooking.Models
{
    public class User
    {
        public int UserId { get; set; }
```

```
        public string? Name { get; set; }
        public string? Email { get; set; }
        public string? PasswordHash { get; set; }
        public string? Role { get; set; }
        public string? ContactNumber { get; set; }

        // Navigation properties (if needed)
        public virtual ICollection<Booking> Bookings { get; set; } = new List<Booking>();
        public virtual ICollection<Hotel> Hotels { get; set; } = new List<Hotel>();
        // ...other navigation properties
    }
}
```

IUserRepository.cs

```
using SmartHotelBooking.Models;
using System.Threading.Tasks;

namespace SmartHotelBooking.Repositories.Interfaces
{
    public interface IUserRepository
    {
        Task<User> AddAsync(User user);
        Task<User?> GetByEmailAsync(string email);
        // ...other methods as needed
    }
}
```

IAuthService.cs

```
using SmartHotelBooking.DTOs;
using System.Threading.Tasks;
```

```
namespace SmartHotelBooking.Services.Interfaces
{
  public interface IAuthService
  {
    Task<AuthResponseDto> RegisterAsync(RegisterDto registerDto);
    Task<AuthResponseDto> LoginAsync(LoginDto loginDto);
  }
}
```

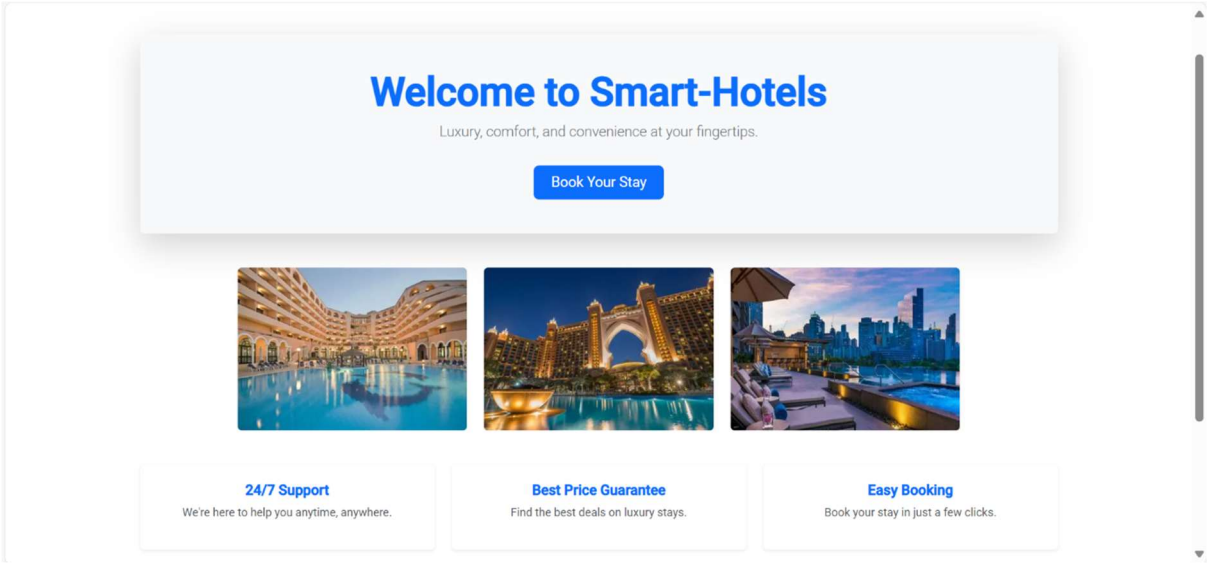
AuthService.cs

```
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;
using SmartHotelBooking.Services.Interfaces;
using System.Threading.Tasks;

namespace SmartHotelBooking.Services.Implementations
{
  public class AuthService : IAuthService
  {
    private readonly IUserRepository _userRepository;
    private readonly IJwtService _jwtService;

    public AuthService(IUserRepository userRepository, IJwtService jwtService)
    {
      _userRepository = userRepository;
      _jwtService = jwtService;
    }
```

```
public async Task<AuthResponseDto> RegisterAsync(RegisterDto registerDto)
{
   var hashedPassword = BCrypt.Net.BCrypt.HashPassword(registerDto.Password);

   var newUser = new User
   {
      Name = registerDto.Name,
      Email = registerDto.Email,
      PasswordHash = hashedPassword,
      Role = registerDto.Role,
      ContactNumber = registerDto.ContactNumber
   };

   var createdUser = await _userRepository.AddAsync(newUser);

   var token = _jwtService.GenerateToken(createdUser);

   return new AuthResponseDto
   {
      Message = "Registration SuccessFull",
      Name = createdUser.Name,
      Email = createdUser.Email,
      Role = createdUser.Role,
      UserId = createdUser.UserId,
      Token = token
   };
}
```

```
    // ...LoginAsync and other methods
  }
}
```

HOME PAGE - UI



Home.Html

```
<div class="container mt-5">
  <!-- 🔥 Hero Section -->
  <div class="text-center bg-light p-5 rounded shadow-lg mb-5">
    <h1 class="display-4 fw-bold text-primary">Welcome to Smart-Hotels</h1>
    <p class="lead text-secondary">Luxury, comfort, and convenience at your fingertips.</p>
    <button class="btn btn-primary btn-lg mt-3 px-4" [disabled]="userRole === 'Admin'" (click)="handleBooking()"  >Book Your Stay</button>
  </div>
  <!-- 🏨 Hotel Image Carousel -->
  <div id="hotelCarousel" class="carousel slide" data-bs-ride="carousel">
```

```
    <div class="carousel-inner">
      <!-- Slide 1 -->
      <div class="carousel-item active">
        <div class="d-flex justify-content-center gap-4">
          <img src="assets/Hotel1.webp" class="rounded shadow-sm w-25" alt="Hotel 1">
          <img src="assets/Hotel2.webp" class="rounded shadow-sm w-25" alt="Hotel 2">
          <img src="assets/Hotel3.webp" class="rounded shadow-sm w-25" alt="Hotel 3">
        </div>
      </div>
      <!-- Slide 2 -->
      <div class="carousel-item">
        <div class="d-flex justify-content-center gap-4">
          <img src="assets/Hotel2.webp" class="rounded shadow-sm w-25" alt="Hotel 2">
          <img src="assets/Hotel3.webp" class="rounded shadow-sm w-25" alt="Hotel 3">
          <img src="assets/Hotel4.webp" class="rounded shadow-sm w-25" alt="Hotel 4">
        </div>
      </div>
      <!-- Slide 3 -->
      <div class="carousel-item">
        <div class="d-flex justify-content-center gap-4">
          <img src="assets/Hotel3.webp" class="rounded shadow-sm w-25" alt="Hotel 3">
          <img src="assets/Hotel4.webp" class="rounded shadow-sm w-25" alt="Hotel 4">
          <img src="assets/Hotel1.webp" class="rounded shadow-sm w-25" alt="Hotel 1">
        </div>
      </div>
    </div>
    <!-- Carousel Controls -->
    <button class="carousel-control-prev" type="button" data-bs-target="#hotelCarousel"
data-bs-slide="prev">
```

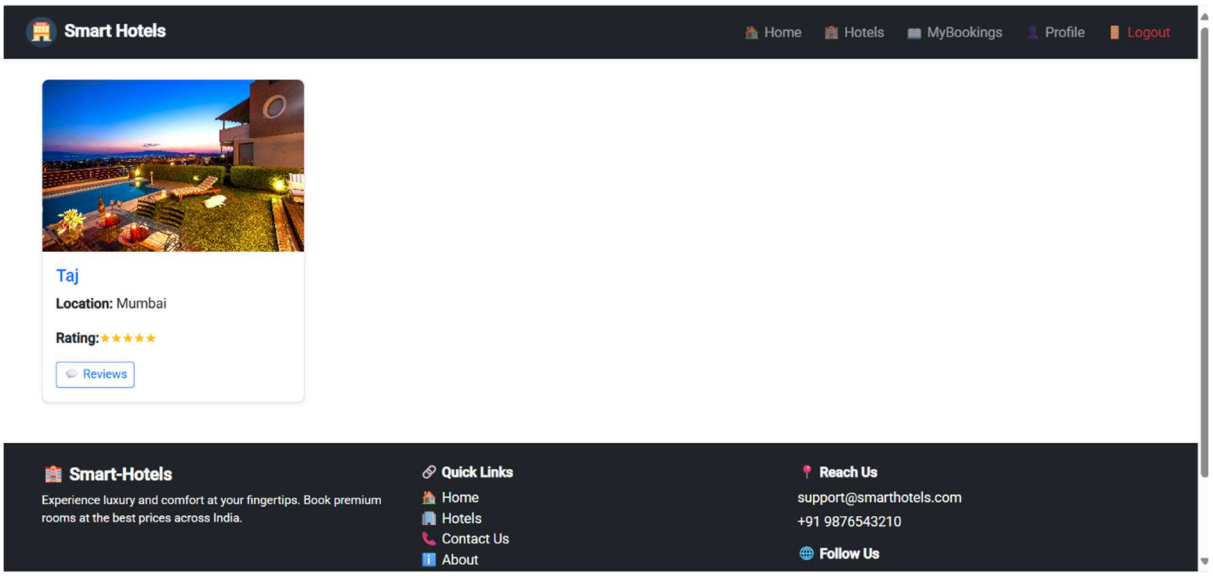
```
      <span class="carousel-control-prev-icon"></span>
    </button>
    <button class="carousel-control-next" type="button" data-bs-target="#hotelCarousel" data-bs-slide="next">
      <span class="carousel-control-next-icon"></span>
    </button>
  </div>
  <!-- 💼 Services Section -->
  <div class="row text-center mt-5">
    <div class="col-md-4 mb-4">
      <div class="p-4 shadow-sm rounded bg-white h-100">
        <h5 class="fw-bold text-primary">24/7 Support</h5>
        <p class="text-muted">We're here to help you anytime, anywhere.</p>
      </div>
    </div>
    <div class="col-md-4 mb-4">
      <div class="p-4 shadow-sm rounded bg-white h-100">
        <h5 class="fw-bold text-primary">Best Price Guarantee</h5>
        <p class="text-muted">Find the best deals on luxury stays.</p>
      </div>
    </div>
    <div class="col-md-4 mb-4">
      <div class="p-4 shadow-sm rounded bg-white h-100">
        <h5 class="fw-bold text-primary">Easy Booking</h5>
        <p class="text-muted">Book your stay in just a few clicks.</p>
      </div>
    </div>
  </div>
</div>
```

Hotel.ts

```
// src/app/components/core/homr/home.ts
import { Component } from '@angular/core';
import { AuthService } from '../../../services/auth/auth.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-home',
  imports: [],
  templateUrl: './home.html',
  styleUrl: './home.css',
})

export class Home {
  userRole: string = '';
  constructor(private router: Router, private authService: AuthService) {
    this.userRole = this.authService.getRole() || ''; // 🔥 Get role from cookie
  }

  handleBooking() {
    if (this.authService.isLoggedIn()) {
      this.router.navigate(['/hotel']);
    } else {
      this.router.navigate(['/login']);
    }
  }
```

}

HOTELS - UI



Hotel.services.ts

```
// src/app/services/hotel.service.ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { environment } from '../../../environments/environment';

export interface Hotel {
  hotelID: number;
  name: string;
```

```
  location: string;
  managerID: number;
  amenities: string;
  rating: number;
  imageUrl: string;
}

@Injectable({
  providedIn: 'root',
})
export class HotelService {
  constructor(private http: HttpClient) { }

  // Method to get all hotels
  getAllHotels(): Observable<Hotel[]> {
    return this.http.get<Hotel[]>(`${environment.apiBaseUrl}/Hotels`, { withCredentials: true
});
  }

  // Method to add Hotel by Manager Only
  addHotel(formData: FormData, managerID: number): Observable<any> {
    return this.http.post(`${environment.apiBaseUrl}/Hotels/manager/${managerID}`,
formData, {
      withCredentials: true,
    });
  }

  // Method to get all hotels created by the LoggedIn Manager Only.
  getHotelsByManagerId(managerID: number): Observable<Hotel[]> {
```

```
    return
this.http.get<Hotel[]>(`${environment.apiBaseUrl}/Hotels/manager/${managerID}`, {
      withCredentials: true,
    });
  }

  // Method to delete a hotel by ID and Manager ID
  deleteHotel(hotelID: number, managerID: number): Observable<any> {
    return
this.http.delete(`${environment.apiBaseUrl}/Hotels/${hotelID}/manager/${managerID}`, {
      withCredentials: true,
    });
  }

  // Method to update a hotel by ID and Manager ID
  updateHotel(hotelID: number, managerID: number, hotelData: any): Observable<any> {
    return this.http.put(
      `${environment.apiBaseUrl}/Hotels/${hotelID}/manager/${managerID}`,
      hotelData,
      {
        withCredentials: true,
        headers: { 'Content-Type': 'application/json' }
      }
    );
  }

  // Method to get a Hotel by it's ID.
  getHotelById(hotelID: number): Observable<Hotel> {
    return this.http.get<Hotel>(`${environment.apiBaseUrl}/Hotels/${hotelID}`, {
      withCredentials: true
```

```
    });
  }
}
```

Hotels.html

```
<!-- src/app/components/core/hotel/hotel.html -->
<div class="container my-4">
 <div class="row">
  <ng-container *ngIf="hotels$ | async as hotels; else loading">
   <div *ngIf="hotels.length > 0; else noHotels">
    <div class="row">
     <div *ngFor="let hotel of hotels" class="col-12 col-sm-6 col-md-4 col-lg-3 mb-4">
      <div class="card shadow-sm h-100 hotel-card"
(click)="viewRooms(hotel.hotelID)" style="cursor: pointer;">
       <img [src]="getImageUrl(hotel.imageUrl)" class="card-img-top" alt="{{
hotel.name }}"
        style="height: 200px; object-fit: cover;">
       <div class="card-body">
        <h5 class="card-title text-primary">{{ hotel.name }}</h5>
        <p class="card-text"><strong>Location:</strong> {{ hotel.location }}</p>
        <p class="card-text">
         <strong>Rating:</strong>
         <span class="text-warning">
          <ng-container *ngFor="let star of [].constructor(hotel.rating)">★</ng-
container>
         </span>
       </p>
```

```
            <!-- ⭐ Add this below rating -->
            <div class="mt-2">
              <button class="btn btn-outline-primary btn-sm"
(click)="viewReviews(hotel.hotelID)">
                💬 Reviews
              </button>
            </div>
          </div>
        </div>
      </div>
    </div>
  </ng-container>
  <ng-template #loading>
    <p>Loading hotels...</p>
  </ng-template>
  <ng-template #noHotels>
    <p>No hotels available.</p>
  </ng-template>
 </div>
</div>
```

Hotel.ts

```
// src/app/components/hotel/hotel.ts
import {
  ChangeDetectionStrategy,
  Component,
  OnInit
} from '@angular/core';
```

```
import { HotelService, Hotel } from '../../../services/hotel/hotel.service';
import { Router } from '@angular/router';
import { CommonModule } from '@angular/common';
import { Observable } from 'rxjs';
import { AuthService } from '../../../services/auth/auth.service';

@Component({
  selector: 'app-hotel',
  templateUrl: './hotel.html',
  styleUrls: ['./hotel.css'],
  changeDetection: ChangeDetectionStrategy.OnPush,
  standalone: true,
  imports: [CommonModule],
})
export class HotelComponent implements OnInit {
  hotels$!: Observable<Hotel[]>;
  userRole: string = '';

  constructor(
    private hotelService: HotelService,
    private router: Router,
    private authService: AuthService
  ) { }

  ngOnInit(): void {
    this.userRole = this.authService.getRole() ?? '';
    this.hotels$ = this.hotelService.getAllHotels();

    // Debugging API response
```

```
  this.hotels$.subscribe({
    next: (hotels) => console.log('Hotels fetched:', hotels),
    error: (err) => console.error('Error fetching hotels:', err)
  });
}

viewRooms(hotelID: number): void {
  this.router.navigate(['/rooms', hotelID]);
}

addHotel(): void {
  this.router.navigate(['/add-hotel']);
}

// Method to construct the image URL
getImageUrl(imagePath: string): string {
  const baseUrl = 'http://localhost:5281'; // Replace with your actual base URL
  return `${baseUrl}${imagePath}`;
}

// ✅ Fix this in hotel.ts
addRoom(hotelID: number): void {
  const managerID = this.authService.getUserId();
  console.log('Redirecting to add-room with:', { hotelID, managerID });

  // ✅ Use lowercase keys to match add-room.ts
  this.router.navigate(['/add-room'], {
    queryParams: {
      hotelId: hotelID,      // lowercase
```

```
      managerId: managerID   // lowercase
    }
   });
  }

  viewReviews(hotelID: number): void {
   this.router.navigate(['/hotel-reviews', hotelID]);
  }
}
```

Hotel.css

```
.hotel-container {
  padding: 20px;
}
.add-hotel-button {
  margin-bottom: 20px;
}
.add-hotel-button button {
  padding: 10px 20px;
  font-size: 16px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}
.hotel-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 20px;
```

```
}

.hotel-card {
  border: 1px solid #ddd;
  border-radius: 8px;
  overflow: hidden;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  cursor: pointer;
  transition: transform 0.2s ease;
}
.hotel-card:hover {
  transform: scale(1.02);
}
.hotel-card img {
  width: 100%;
  height: 200px;
  object-fit: cover;
}
.hotel-info {
  padding: 15px;
}
.hotel-info h5 {
  margin: 0 0 10px;
}
.stars {
  color: #ffc107;
}
```

Backend-code

HotelsControllers.cs:

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [EnableCors("AllowFrontend")]
    public class HotelsController : ControllerBase
    {
        private readonly IHotelService _hotelService;

        public HotelsController(IHotelService hotelService)
        {
            _hotelService = hotelService;
        }
        [HttpGet("{hotelId}")]
        [Authorize(Roles = "User,Manager")]
        public async Task<IActionResult> GetHotelById(int hotelId)
        {
            var hotel = await _hotelService.GetHotelByIdAsync(hotelId);

            if (hotel == null)
```

```
            return NotFound(new { Message = "Hotel not found" });


        return Ok(hotel);
    }
    [HttpGet]
    [Authorize(Roles = "User,Manager")]
    public async Task<IActionResult> GetAll() => Ok(await
_hotelService.GetAllHotelsAsync());


    [HttpGet("manager/{managerId}")]
    public async Task<IActionResult> GetHotelsByManager(int managerId)
    {
        var hotels = await _hotelService.GetHotelsByManagerAsync(managerId);
        return Ok(hotels);
    }


    [HttpPost("manager/{managerId}")]
    [Authorize(Roles = "Manager")]
    public async Task<IActionResult> CreateHotel(int managerId, [FromForm]
CreateHotelDto dto)
    {
        if (!ModelState.IsValid)
            return BadRequest(ModelState);


        try
        {
            var created = await _hotelService.CreateHotelAsync(managerId, dto);
            return Ok(new { Message = "Hotel created successfully", Hotel = created });
        }
        catch (InvalidOperationException ex)
```

```
            {
                return Conflict(new { Message = ex.Message });
            }
        }
        [HttpPut("{hotelId}/manager/{managerId}")]
        [Authorize(Roles = "Manager")]
        public async Task<IActionResult> UpdateHotel(int hotelId, int managerId,
UpdateHotelDto dto)
        {
            var result = await _hotelService.UpdateHotelAsync(hotelId, managerId, dto);
            if (!result)
                return NotFound(new { Message = "Hotel not found or unauthorized" });

            return Ok(new { Message = "Hotel updated successfully" });
        }

        [HttpDelete("{hotelId}/manager/{managerId}")]
        [Authorize(Roles = "Manager")]
        public async Task<IActionResult> DeleteHotel(int hotelId, int managerId)
        {

            var result = await _hotelService.DeleteHotelAsync(hotelId, managerId);
            if (!result)
                return NotFound(new { Message = "Hotel not found or unauthorized" });

            return Ok(new { Message = "Hotel deleted successfully" });
        }
    }
}
```

HotelServices.cs

```
using AutoMapper;
using Microsoft.EntityFrameworkCore;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Services.Implementations
{
    public class HotelService : IHotelService
    {
        private readonly HotelBookingContext _context;
        private readonly IMapper _mapper;



        public HotelService(HotelBookingContext context, IMapper mapper)
        {
            _context = context;
            _mapper = mapper;

        }

        public async Task<IEnumerable<HotelDTO>> GetAllHotelsAsync()
        {
            var hotels = await _context.Hotels.ToListAsync();
            return _mapper.Map<IEnumerable<HotelDTO>>(hotels);
        }
```

```
    public async Task<HotelDTO> GetHotelByIdAsync(int hotelId)
    {
        var hotel = await _context.Hotels.FindAsync(hotelId);
        return _mapper.Map<HotelDTO>(hotel);

    }

    public async Task<List<HotelDTO>> GetHotelsByManagerAsync(int managerId)
    {
        var hotels = await _context.Hotels
            .Where(h => h.ManagerId == managerId)
            .ToListAsync();

        return _mapper.Map<List<HotelDTO>>(hotels);
    }

    public async Task<HotelDTO> CreateHotelAsync(int managerId, CreateHotelDto
hoteldto)
    {
        var existingHotel = await _context.Hotels.ToListAsync();

        if (existingHotel.Any(h => h.Name == hoteldto.Name && h.Location ==
hoteldto.Location && h.ManagerId == managerId))
            throw new InvalidOperationException("Hotel already exists for this manager.");

        var hotel = _mapper.Map<Hotel>(hoteldto);
        hotel.ManagerId = managerId;

        // ✅ Save image
```

```
        if (hoteldto.Image != null && hoteldto.Image.Length > 0)
        {
            var uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot/images/hotels");
            Directory.CreateDirectory(uploadsFolder); // Ensure folder exists

            var fileName = $"{Guid.NewGuid()}_{hoteldto.Image.FileName}";
            var filePath = Path.Combine(uploadsFolder, fileName);

            using (var stream = new FileStream(filePath, FileMode.Create))
            {
                await hoteldto.Image.CopyToAsync(stream);
            }

            hotel.ImageUrl = $"/images/hotels/{fileName}";
        }

        await _context.Hotels.AddAsync(hotel);
        await _context.SaveChangesAsync();

        return _mapper.Map<HotelDTO>(hotel);
    }

    public async Task<bool> UpdateHotelAsync(int hotelId, int managerId,
UpdateHotelDto dto)
    {

        // Check if manager exists
        var managerExists = await _context.Users.AnyAsync(u => u.UserId == managerId);
        if (!managerExists)
```

```
        return false;



    var hotel = await _context.Hotels
        .FirstOrDefaultAsync(h => h.HotelId == hotelId && h.ManagerId == managerId);

    if (hotel == null)
        return false;

    _mapper.Map(dto, hotel);

    try
    {
        await _context.SaveChangesAsync();
        return true;
    }
    catch (DbUpdateException)
    {
        // Log or handle exception
        return false;
    }

}

public async Task<bool> DeleteHotelAsync(int hotelId, int managerId)
{
    var hotel = await _context.Hotels
        .Include(h=>h.Rooms)
        .FirstOrDefaultAsync(h => h.HotelId == hotelId && h.ManagerId == managerId);
```

```
            if (hotel == null)
                return false;



            if (hotel.Rooms.Any())
                return false; // Or return a custom message like "Hotel has rooms and cannot be deleted."

            _context.Hotels.Remove(hotel);

            try
            {
                await _context.SaveChangesAsync();
                return true;
            }
            catch (DbUpdateException)
            {
                // Log or handle exception
                return false;
            }

        }
    }
}
```

HotelRepository.cs

```
using Microsoft.EntityFrameworkCore;
```

```
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;

namespace SmartHotelBooking.Repositories.Implementations
{
    public class HotelRepository : IHotelRepository
    {
        private readonly HotelBookingContext _context;

        public HotelRepository(HotelBookingContext context)
        {
            _context = context;
        }

        public async Task<Hotel> GetByIdAsync(int hotelId) =>
            await _context.Hotels.FindAsync(hotelId);

        public async Task<IEnumerable<Hotel>> GetAllAsync() =>
            await _context.Hotels.ToListAsync();

        public async Task AddAsync(Hotel hotel) =>
            await _context.Hotels.AddAsync(hotel);

        public void Update(Hotel hotel) =>
            _context.Hotels.Update(hotel);
        public void Delete(Hotel hotel) =>
            _context.Hotels.Remove(hotel);
        public async Task SaveChangesAsync() =>
            await _context.SaveChangesAsync();
```

```
    }
}
```

Hotel.cs (model)

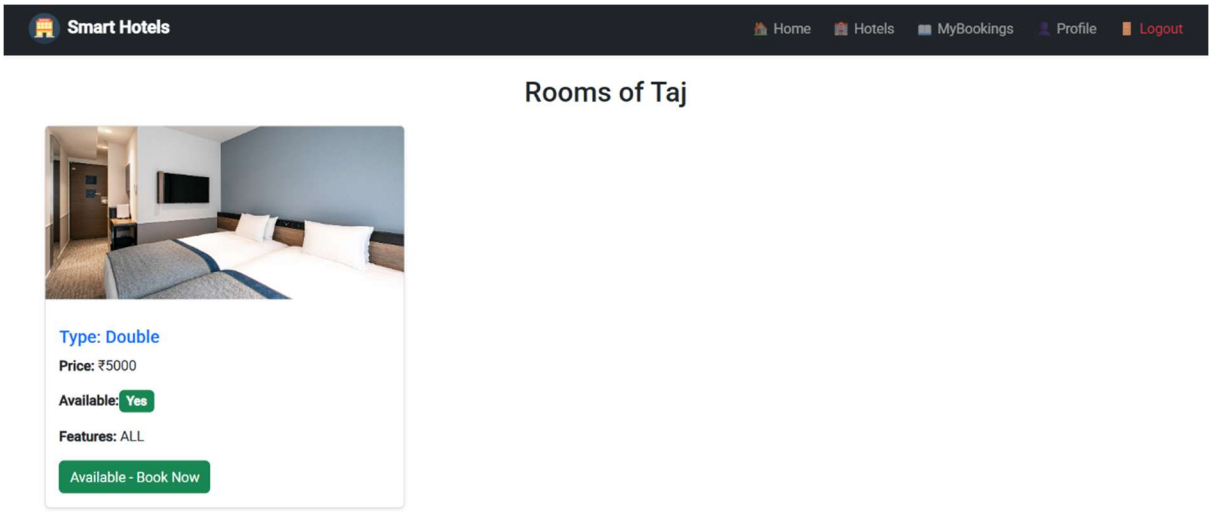
```
using System;
using System.Collections.Generic;

namespace SmartHotelBooking.Models;

public partial class Hotel
{
    public int HotelId { get; set; }

    public string? Name { get; set; }

    public string? Location { get; set; }

    public int? ManagerId { get; set; }

    public string? Amenities { get; set; }

    public decimal? Rating { get; set; }

    public virtual User? Manager { get; set; }

    public string? ImageUrl { get; set; }

    public virtual ICollection<Review> Reviews { get; set; } = new List<Review>();
```

```
    public virtual ICollection<Room> Rooms { get; set; } = new List<Room>();
}
```

HotelDto.cs

```
namespace SmartHotelBooking.DTOs
{
    public class HotelDTO
    {
        public int HotelID { get; set; }
        public string? Name { get; set; }
        public string? Location { get; set; }
        public int? ManagerID { get; set; }
        //public string ManagerName as Name {get;set;}
        public string? Amenities { get; set; }
        public decimal? Rating { get; set; }

        public string? ImageUrl { get; set; }
    }
}
```

ROOMS -UI



Room.html

```
<!-- src/app/components/core/room/room.html -->
<h2 class="text-center my-4">Rooms of {{hotelName}} </h2>


<div class="container">
  <div class="row">
    <div
      *ngFor="let room of rooms; trackBy: trackByRoomId"
      class="col-md-6 col-lg-4 mb-4 room-card"
    >
      <div class="card shadow-sm h-100">
        <img
          [src]="getRoomImage(room.type)"
          class="card-img-top mb-3"
          alt="{{ room.type }}"
          loading="lazy"
          style="height: 200px; object-fit: cover"
        />
        <div class="card-body">
          <h5 class="card-title text-primary">Type: {{ room.type }}</h5>
```

```
        <p class="card-text"><strong>Price:</strong> ₹{{ room.price }}</p>
        <p class="card-text">
          <strong>Available:</strong>
          <span
            class="badge"
            [ngClass]="room.availability ? 'bg-success' : 'bg-danger'"
          >
            {{ room.availability ? 'Yes' : 'No' }}
          </span>
        </p>
        <p class="card-text">
          <strong>Features:</strong> {{ room.features }}
        </p>
        <button
          [disabled]="!room.availability"
          [ngClass]="room.availability ? 'btn btn-success' : 'btn btn-danger'"
          (click)="bookRoom(room.roomID)"
        >
          {{ room.availability ? 'Available - Book Now' : 'Not Available' }}
        </button>
      </div>
    </div>
  </div>
</div>
```

Room.css

```
/* src/app/components/room/room.css */
.room-card .card {
  transition: transform 0.3s ease, box-shadow 0.3s ease;
```

```
}
.room-card .card:hover {
  transform: translateY(-5px);
  box-shadow: 0 8px 20px rgba(0, 0, 0, 0.15);
}
.card-title {
  color: #007bff;
}
.badge {
  font-size: 0.9em;
  padding: 0.4em 0.6em;
}
```

Room.ts

```
// src/app/components/room/room.component.ts
import {
  Component,
  OnInit,
  OnDestroy,
  ChangeDetectionStrategy,
  ChangeDetectorRef,
} from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { RoomService, Room } from '../../../services/room/room.service';
import { Subscription } from 'rxjs';
import { CommonModule } from '@angular/common';
import { HotelService, Hotel } from '../../../services/hotel/hotel.service';
import { AuthService } from '../../../services/auth/auth.service';

@Component({
```

```
  selector: 'app-room',
  templateUrl: './room.html',
  styleUrls: ['./room.css'],
  changeDetection: ChangeDetectionStrategy.OnPush,
  standalone: true,
  imports: [CommonModule],
})
export class RoomComponent implements OnInit, OnDestroy {
  rooms: Room[] = [];
  hotelName: string = ''; // Added : To store hotel name
  hotelID!: number;
  private roomSub!: Subscription;

  constructor(
    private route: ActivatedRoute,
    private roomService: RoomService,
    private hotelService: HotelService, // ✅ Add this
    private cdr: ChangeDetectorRef,
    private router: Router
  ) {}

  ngOnInit(): void {
    this.hotelID = +this.route.snapshot.paramMap.get('hotelID')!;

    // ✅ Fetch hotel name
    this.hotelService.getHotelById(this.hotelID).subscribe({
      next: (hotel) => {
        this.hotelName = hotel.name;
        this.cdr.markForCheck();
```

```
    },
    error: (err) => console.error('Hotel fetch error:', err),
  });
  console.log('Room Component, Hotel ID:', this.hotelID); // Debugging log

  // Fetch rooms by hotel ID
  this.roomSub = this.roomService
    .getRoomsByHotelId(this.hotelID)
    .subscribe((data) => {
      this.rooms = data;
      console.log('Rooms fetched successfully:', this.rooms);
      this.cdr.markForCheck(); // Ensure change detection runs
    });
}

ngOnDestroy(): void {
  if (this.roomSub) {
    this.roomSub.unsubscribe();
  }
}

trackByRoomId(index: number, room: Room): number {
  return room.roomID;
}

bookRoom(roomID: number): void {
  this.router.navigate(['/bookings', roomID]);
}
```

```
  // Helper Method to add ROOM Image
  getRoomImage(type: string): string {
    switch (type.toLowerCase()) {
      case 'single':
        return '/assets/RoomsImage/SingleRoom.jpeg';
      case 'double':
        return '/assets/RoomsImage/DoubleRoom.jpg';
      case 'family':
        return '/assets/RoomsImage/FamilyRoom.jpeg';
      case 'deluxe':
        return '/assets/RoomsImage/DeluxeRoom.jpeg';
      default:
        return '/assets/RoomsImage/default.jpeg'; // Optional: Add a fallback image
    }
  }
}
```

Room.service.ts

```
// src/app/services/room.service.ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { environment } from '../../../environments/environment';

export interface Room {
  roomID: number;
  hotelID: number;
  managerID: number;
  type: string;
```

```
  price: number;
  availability: boolean;
  features: string;
  imageUrl?: string;
}

export interface CreateRoom {
  hotelID: number;
  managerID: number;
  type: string;
  price: number;
  availability: boolean;
  features: string;
}

@Injectable({
  providedIn: 'root'
})
export class RoomService {

  constructor(private http: HttpClient) {}

  getRoomsByHotelId(hotelID: number): Observable<Room[]> {
    return this.http.get<Room[]>(`${environment.apiBaseUrl}/Rooms/hotel/${hotelID}`);
  }

  getRoomById(roomID: number): Observable<Room> {
    return this.http.get<Room>(`${environment.apiBaseUrl}/Rooms/${roomID}`);
  }
```

```
  addRoom(room: CreateRoom): Observable<Room> {
    return this.http.post<Room>(`${environment.apiBaseUrl}/Rooms`, room);
  }
}
```

Backend-code

RoomsController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [EnableCors("AllowFrontend")]
    public class RoomsController : ControllerBase
    {
        private readonly IRoomService _roomService;

        public RoomsController(IRoomService roomService)
        {
            _roomService = roomService;
        }

        [HttpGet("hotel/{hotelId}")]
```

```
    [Authorize(Roles = "User,Manager")]
    public async Task<IActionResult> GetRoomsByHotel(int hotelId) => Ok(await
_roomService.GetRoomsByHotelIdAsync(hotelId));

    [HttpPost]
    [Authorize(Roles = "Manager")]
    public async Task<IActionResult> Create(CreateRoomDto dto) => Ok(await
_roomService.CreateRoomAsync(dto));

    [HttpPut("{roomId}")]
    [Authorize(Roles = "Manager")]
    public async Task<IActionResult> Update(int roomId, UpdateRoomDto dto) =>
Ok(await _roomService.UpdateRoomAsync(roomId, dto));

    [HttpDelete("{roomId}")]
    [Authorize(Roles = "Manager")]
    public async Task<IActionResult> Delete(int roomId) => Ok(await
_roomService.DeleteRoomAsync(roomId));


    [HttpGet("{roomId}")]
    [Authorize(Roles = "User,Manager")]
    public async Task<IActionResult> GetRoomById(int roomId)
    {
        var room = await _roomService.GetRoomByIdAsync(roomId);
        if (room == null)
            return NotFound();
        return Ok(room);
    }
}
```

```
}
```

RoomService.cs

```
using AutoMapper;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Services.Implementations
{
    public class RoomService : IRoomService
    {
        private readonly IRoomRepository _roomRepository;
        private readonly IMapper _mapper;

        public RoomService(IRoomRepository roomRepository, IMapper mapper)
        {
            _roomRepository = roomRepository;
            _mapper = mapper;
        }

        public async Task<RoomDTO> GetRoomByIdAsync(int roomId)
        {
            var room = await _roomRepository.GetByIdAsync(roomId);
            return _mapper.Map<RoomDTO>(room);
        }

        public async Task<IEnumerable<RoomDTO>> GetAllRoomsAsync()
        {
```

```
    var rooms = await _roomRepository.GetAllAsync();
    return _mapper.Map<IEnumerable<RoomDTO>>(rooms);
}

public async Task<IEnumerable<RoomDTO>> GetRoomsByHotelIdAsync(int hotelId)
{
    var rooms = await _roomRepository.GetRoomsByHotelIdAsync(hotelId);
    return _mapper.Map<IEnumerable<RoomDTO>>(rooms);
}

public async Task<RoomDTO> CreateRoomAsync(CreateRoomDto roomDto)
{
    var room = _mapper.Map<Room>(roomDto);
    room.ManagerId = roomDto.ManagerID; // ✅ Assign Manager ID

    await _roomRepository.AddAsync(room);
    await _roomRepository.SaveChangesAsync();
    return _mapper.Map<RoomDTO>(room);
}

public async Task<bool> UpdateRoomAsync(int roomId, UpdateRoomDto roomDto)
{
    var existingRoom = await _roomRepository.GetByIdAsync(roomId);
    if (existingRoom == null) return false;

    _mapper.Map(roomDto, existingRoom);
    _roomRepository.Update(existingRoom);
    await _roomRepository.SaveChangesAsync();
    return true;
```

```
        }

        public async Task<bool> DeleteRoomAsync(int roomId)
        {
            var room = await _roomRepository.GetByIdAsync(roomId);
            if (room == null) return false;
            _roomRepository.Delete(room);
            await _roomRepository.SaveChangesAsync();
            return true;
        }
    }
}
```

RoomsRepository.cs

```
using Microsoft.EntityFrameworkCore;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;

namespace SmartHotelBooking.Repositories.Implementations
{
    public class RoomRepository : IRoomRepository
    {
        private readonly HotelBookingContext _context;

        public RoomRepository(HotelBookingContext context)
        {
            _context = context;
        }
```

```
        public async Task<Room> GetByIdAsync(int roomId) =>
            await _context.Rooms.FindAsync(roomId);

        public async Task<IEnumerable<Room>> GetAllAsync() =>
            await _context.Rooms.ToListAsync();

        public async Task<IEnumerable<Room>> GetRoomsByHotelIdAsync(int hotelId) =>
            await _context.Rooms.Where(r => r.HotelId == hotelId).ToListAsync();

        public async Task AddAsync(Room room) =>
            await _context.Rooms.AddAsync(room);

        public void Update(Room room) =>
            _context.Rooms.Update(room);

        public void Delete(Room room) =>
            _context.Rooms.Remove(room);

        public async Task SaveChangesAsync() =>
            await _context.SaveChangesAsync();
    }
}
```

Rooms.cs (model)

```
using System;
using System.Collections.Generic;

namespace SmartHotelBooking.Models;
```

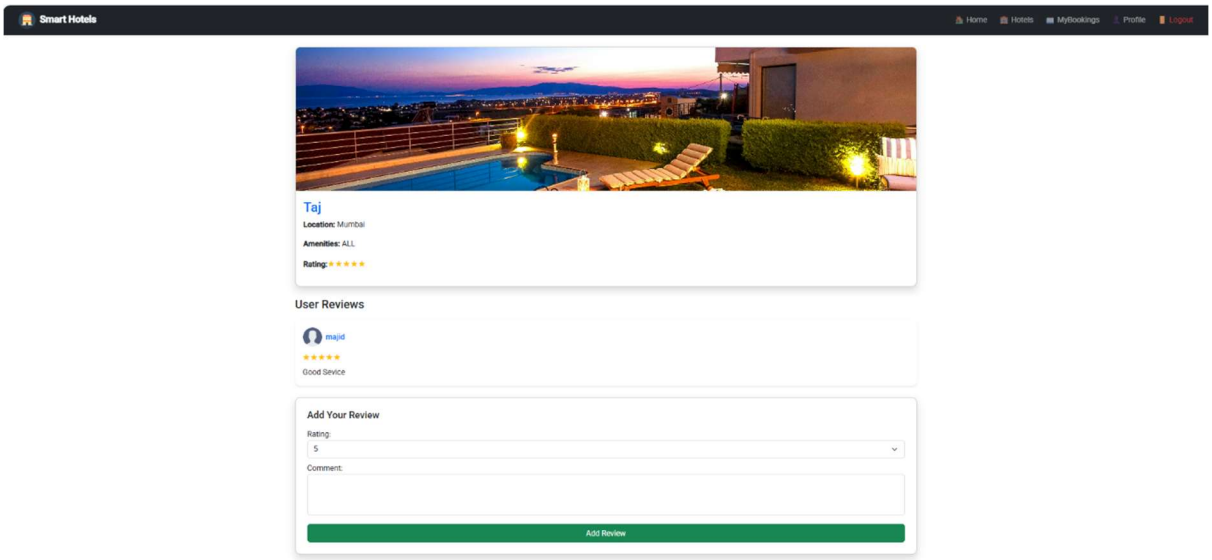
```
public partial class Room
{
    public int RoomId { get; set; }
    public int? HotelId { get; set; }
    public int? ManagerId { get; set; } // ✅ NEW

    public string? Type { get; set; }
    public decimal? Price { get; set; }
    public bool? Availability { get; set; } = true;
    public string? Features { get; set; }

    public virtual Hotel? Hotel { get; set; }
    public virtual User? Manager { get; set; } // ✅ NEW
    public virtual ICollection<Booking> Bookings { get; set; } = new List<Booking>();
}
```

RoomDTO.cs

```
namespace SmartHotelBooking.DTOs
{
    public class RoomDTO
    {
        public int RoomID { get; set; }
        public int HotelID { get; set; }
        public int ManagerID { get; set; } // ✅ Add if you want to expose in response
        public string? Type { get; set; }
        public decimal Price { get; set; }
        public bool Availability { get; set; }
        public string? Features { get; set; }
```

```
  }
}
```

USER-Reviews -UI



Review.html

```
<!-- src/app/components/core/review/review.html -->
<div *ngIf="hotel">
  <div class="container my-4">
    <!-- Hotel Card -->
    <div class="card shadow mb-4">
```

```
    <img [src]="hotel.imageUrl" (error)="hotel.imageUrl = 'assets/default-hotel.jpg'" alt="{{
hotel.name }} Image"
      class="card-img-top" style="height: 300px; width: 100%; object-fit: cover" />
    <div class="card-body">
      <h3 class="card-title text-primary">{{ hotel.name }}</h3>
      <p><strong>Location:</strong> {{ hotel.location }}</p>
      <p><strong>Amenities:</strong> {{ hotel.amenities }}</p>
      <p>
        <strong>Rating:</strong>
        <span class="text-warning">
          <ng-container *ngFor="let star of [].constructor(hotel.rating)">★</ng-container>
        </span>
      </p>
    </div>
  </div>

  <!-- Reviews -->
  <div class="mb-4">
    <h4>User Reviews</h4>
    <div *ngIf="reviews.length === 0" class="text-muted">No reviews yet. Be the
first!</div>

    <div *ngFor="let review of reviews" class="card mb-3 shadow-sm border-0">
      <div class="card-body">
        <div class="d-flex align-items-center mb-2">
          <img src="https://cdn-icons-png.flaticon.com/512/149/149071.png" alt="User Icon"
class="rounded-circle me-2"
            width="40" height="40" />
          <h6 class="mb-0 text-primary">{{ review.userName || 'Guest' }}</h6>
        </div>
```

```
        <div class="mb-1 text-warning">
          <ng-container *ngFor="let star of [].constructor(review.rating)">★</ng-container>
        </div>
        <p class="mb-0">{{ review.comment }}</p>
      </div>
    </div>
  </div>

  <!-- Add Review -->
  <div class="card shadow p-4">
    <h5>Add Your Review</h5>
    <div class="mb-2">
      <label>Rating:</label>
      <select class="form-select" [(ngModel)]="newRating">
        <option *ngFor="let r of [1,2,3,4,5]" [value]="r">{{ r }}</option>
      </select>
    </div>
    <div class="mb-2">
      <label>Comment:</label>
      <textarea class="form-control" rows="3" [(ngModel)]="newComment"></textarea>
    </div>
    <button class="btn btn-success" (click)="addReview()">Add Review</button>
  </div>
 </div>
</div>
```

Review.css

```
.card {
  border-radius: 10px;
```

```
  overflow: hidden;
 }
 .card-img-top {
  height: 300px;
  object-fit: cover;
  border-bottom: 1px solid #ddd;
 }

 .card-body {
  padding: 1rem;
 }

 .text-warning {
  font-size: 1.2rem;
 }
 textarea.form-control {
  resize: none;
 }
 .btn-success {
  margin-top: 10px;
 }
 h4,
 h5 {
  margin-bottom: 1rem;
 }
 .me-2 {
  margin-right: 0.5rem;
 }
```

Review.ts

```
// src/app/components/hotel-reviews/reviews.ts
import {
  ChangeDetectionStrategy,
  ChangeDetectorRef,
  Component,
  OnInit
} from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { HotelService, Hotel } from '../../../services/hotel/hotel.service';
import { ReviewService } from '../../../services/review/review.services';
import { AuthService } from '../../../services/auth/auth.service';
import { FormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';
import { environment } from '../../../../environments/environment';

type Review = {
  reviewID: number;
  userID: number;
  hotelID: number;
  rating: number;
  comment: string;
  timestamp: string;
  userName?: string;
};

@Component({
  selector: 'app-reviews',
  templateUrl: './review.html',
  styleUrls: ['./review.css'],
```

```
  changeDetection: ChangeDetectionStrategy.OnPush,
  standalone: true,
  imports: [FormsModule, CommonModule],
})
export class Reviews implements OnInit {
  hotel!: Hotel;
  reviews: Review[] = [];
  hotelID!: number;

  newRating: number = 5;
  newComment: string = '';

  constructor(
    private route: ActivatedRoute,
    private hotelService: HotelService,
    private reviewService: ReviewService,
    private authService: AuthService,
    private cdr: ChangeDetectorRef
  ) {}

  ngOnInit(): void {
    this.hotelID = Number(this.route.snapshot.paramMap.get('hotelID'));

    // Hotel Detail Section
    this.hotelService.getHotelById(this.hotelID).subscribe({
      next: (data) => {
        data.imageUrl = data.imageUrl?.startsWith('http')
          ? data.imageUrl
          : `${environment.fileBaseUrl}${data.imageUrl}`; // 👈 Change
```

```
      this.hotel = data;
      console.log('Full hotel object after image processing:', this.hotel);
      this.cdr.markForCheck();
    },
    error: (err) => console.error('Hotel fetch error', err)
  });


  // Hotel Review Section
  this.reviewService.getReviewsByHotelId(this.hotelID).subscribe({
    next: (data) => {
      this.reviews = data;
      this.cdr.markForCheck();
    },
    error: (err) => console.error('Review fetch error', err)
  });
}

addReview(): void {
  const review: Review = {
    reviewID: 0,
    userID: this.authService.getUserId(),
    hotelID: this.hotelID,
    rating: this.newRating,
    comment: this.newComment,
    timestamp: new Date().toISOString(),
    userName: this.authService.getName()
  };
```

```
    this.reviewService.addReview(review).subscribe({
      next: (data) => {
        this.reviews = [...this.reviews, data];
        this.newComment = '';
        this.newRating = 5;
        this.cdr.markForCheck();
      },
      error: (err) => console.error('Add review error', err)
    });
  }
}
```

Review.sevices.ts

```
// src/app/services/review/review.services.ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { environment } from '../../../environments/environment';

export interface Review {
  reviewID: number;
  userID: number;
  hotelID: number;
  rating: number;
  comment: string;
  timestamp: string;
  userName?: string;
}
```

```
@Injectable({
  providedIn: 'root'
})
export class ReviewService {
  constructor(private http: HttpClient) {}

  getReviewsByHotelId(hotelId: number): Observable<Review[]> {
    return this.http.get<Review[]>(`${environment.apiBaseUrl}/Reviews/hotel/${hotelId}`, {
      withCredentials: true
    });
  }

  addReview(review: Review): Observable<Review> {
    return this.http.post<Review>(`${environment.apiBaseUrl}/Reviews`, review, {
      withCredentials: true
    });
  }
}
```

Backend-code

ReviewController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Services.Interfaces;
```

```
namespace SmartHotelBooking.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   [EnableCors("AllowFrontend")]
   public class ReviewsController : ControllerBase
   {
      private readonly IReviewService _reviewService;

      public ReviewsController(IReviewService reviewService)
      {
         _reviewService = reviewService;
  }
      [HttpPost]
      [Authorize(Roles = "User")]
      public async Task<IActionResult> AddReview(CreateReviewDto dto) => Ok(await
_reviewService.CreateReviewAsync(dto));

      [HttpGet("hotel/{hotelId}")]
      [Authorize(Roles = "User,Manager")]
      public async Task<IActionResult> GetReviews(int hotelId) => Ok(await
_reviewService.GetReviewsByHotelIdAsync(hotelId));
   }
}
```

Review.service.cs

```
using AutoMapper;

using SmartHotelBooking.DTOs;

using SmartHotelBooking.Models;
```

```
using SmartHotelBooking.Repositories.Interfaces;

using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Services.Implementations

{

    public class ReviewService : IReviewService

    {

        private readonly IReviewRepository _reviewRepository;

        private readonly IMapper _mapper;

        public ReviewService(IReviewRepository reviewRepository, IMapper mapper)

        {

            _reviewRepository = reviewRepository;

            _mapper = mapper;

        }

        public async Task<ReviewDTO> GetReviewByIdAsync(int reviewId)
```

```
{
    var review = await _reviewRepository.GetByIdAsync(reviewId);
    if (review == null) return null;
    return new ReviewDTO
    {
        ReviewID = review.ReviewId,
        UserID = review.UserId ?? 0,
        HotelID = review.HotelId ?? 0,
        Rating = review.Rating ?? 0,
        Comment = review.Comment,
        Timestamp = review.Timestamp ?? DateTime.MinValue,
        UserName = review.User?.Name
    };
}
public async Task<IEnumerable<ReviewDTO>> GetAllReviewsAsync()
```

```
    {
        var reviews = await _reviewRepository.GetAllAsync();
        return reviews.Select(r => new ReviewDTO
        {
            ReviewID = r.ReviewId,
            UserID = r.UserId ?? 0,
            HotelID = r.HotelId ?? 0,
            Rating = r.Rating ?? 0,
            Comment = r.Comment,
            Timestamp = r.Timestamp ?? DateTime.MinValue,
            UserName = r.User?.Name
        });
    }
    public async Task<IEnumerable<ReviewDTO>> GetReviewsByHotelIdAsync(int
hotelId)
```

```
{
    var reviews = await _reviewRepository.GetReviewsByHotelIdAsync(hotelId);
    return reviews.Select(r => new ReviewDTO

    {
        ReviewID = r.ReviewId,

        UserID = r.UserId ?? 0,

        HotelID = r.HotelId ?? 0,

        Rating = r.Rating ?? 0,

        Comment = r.Comment,
        Timestamp = r.Timestamp ?? DateTime.MinValue,
        UserName = r.User?.Name

    });
}
public async Task<ReviewDTO> CreateReviewAsync(CreateReviewDto reviewDto)

{

    var review = _mapper.Map<Review>(reviewDto);
    await _reviewRepository.AddAsync(review);
    await _reviewRepository.SaveChangesAsync();
    var savedReview = await _reviewRepository.GetByIdAsync(review.ReviewId);
    return new ReviewDTO
```

```
        {

            ReviewID = savedReview.ReviewId,
            UserID = savedReview.UserId ?? 0,
            HotelID = savedReview.HotelId ?? 0,
            Rating = savedReview.Rating ?? 0,
            Comment = savedReview.Comment,
            Timestamp = savedReview.Timestamp ?? DateTime.MinValue,
            UserName = savedReview.User?.Name

        };

    }

    public async Task<bool> UpdateReviewAsync(int reviewId, UpdateReviewDto
reviewDto)

    {

        var existingReview = await _reviewRepository.GetByIdAsync(reviewId);

        if (existingReview == null) return false;

        _mapper.Map(reviewDto, existingReview);

        _reviewRepository.Update(existingReview);

        await _reviewRepository.SaveChangesAsync();
```

```
            return true;

        }

        public async Task<bool> DeleteReviewAsync(int reviewId)

        {

            var review = await _reviewRepository.GetByIdAsync(reviewId);

            if (review == null) return false;

            _reviewRepository.Delete(review);

            await _reviewRepository.SaveChangesAsync();

            return true;
        }
    }
}
```

ReviewRepository.cs

```
using Microsoft.EntityFrameworkCore;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;

namespace SmartHotelBooking.Repositories.Implementations
{
    public class RoomRepository : IRoomRepository
    {
```

```
        private readonly HotelBookingContext _context;

        public RoomRepository(HotelBookingContext context)
        {
            _context = context;
        }

        public async Task<Room> GetByIdAsync(int roomId) =>
            await _context.Rooms.FindAsync(roomId);

        public async Task<IEnumerable<Room>> GetAllAsync() =>
            await _context.Rooms.ToListAsync();

        public async Task<IEnumerable<Room>> GetRoomsByHotelIdAsync(int hotelId) =>
            await _context.Rooms.Where(r => r.HotelId == hotelId).ToListAsync();

        public async Task AddAsync(Room room) =>
            await _context.Rooms.AddAsync(room);

        public void Update(Room room) =>
            _context.Rooms.Update(room);

        public void Delete(Room room) =>
            _context.Rooms.Remove(room);

        public async Task SaveChangesAsync() =>
            await _context.SaveChangesAsync();
    }
}
```

Review.cs (Model)

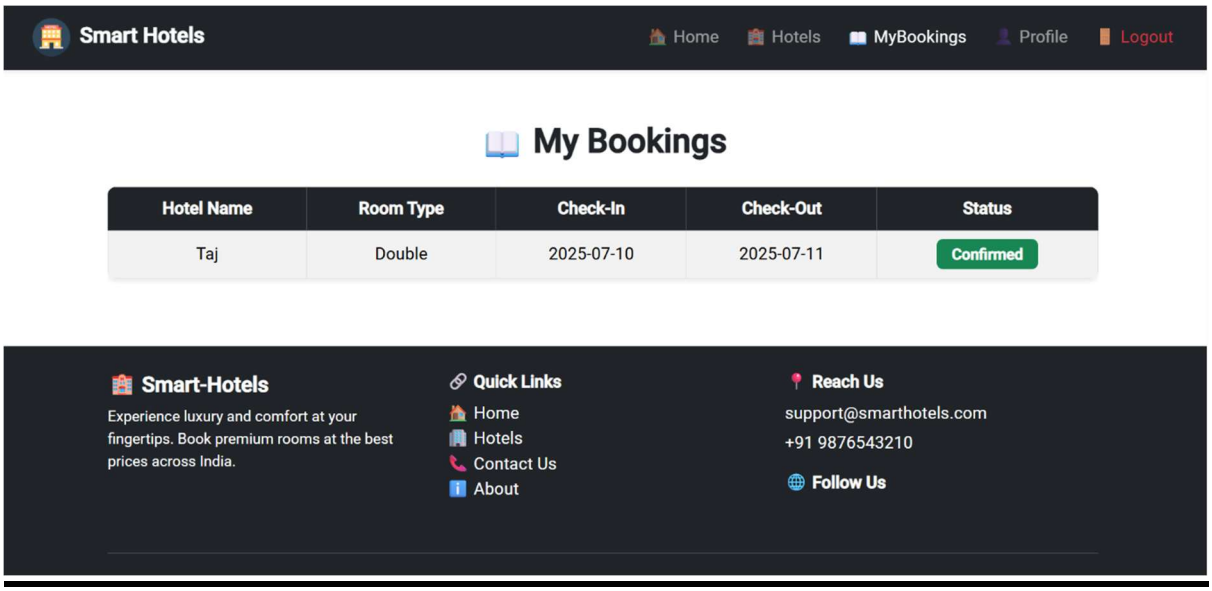
```
using System;
using System.Collections.Generic;
namespace SmartHotelBooking.Models;

public partial class Review
{
    public int ReviewId { get; set; }

    public int? UserId { get; set; }

    public int? HotelId { get; set; }

    public int? Rating { get; set; }

    public string? Comment { get; set; }

    public DateTime? Timestamp { get; set; }
    public virtual Hotel? Hotel { get; set; }

    public virtual User? User { get; set; }
}
```

ReviewDto.cs

```
namespace SmartHotelBooking.DTOs
{
    public class ReviewDTO
    {
```

```
        public int ReviewID { get; set; }
        public int UserID { get; set; }
        public int HotelID { get; set; }
        public int Rating { get; set; }
        public string? Comment { get; set; }
        public DateTime Timestamp { get; set; }
        // ✅ Add this line
        public string? UserName { get; set; }
        //public string? ImageUrl { get; set; }

    }
}
```

My.Booking – USER



Booking.html

<!-- src/app/components/user/bookings/bookings.html -->

```
<div class="container my-5" *ngIf="room">
  <div class="card mx-auto" style="max-width: 600px">
    <!-- <img
      [src]="room.imageUrl"
      class="card-img-top"
      alt="{{ room.type }}"
      style="height: 250px; object-fit: cover"
    /> -->
    <img [src]="getRoomImage(room.type)" class="card-img-top mb-3" alt="{{ room.type }}" loading="lazy"
                style="height: 250px; object-fit: cover;">
    <div class="card-body">
      <h5 class="card-title">{{ room.type }}</h5>
      <p class="card-text"><strong>Price:</strong> ₹{{ room.price }}</p>
      <p class="card-text"><strong>Features:</strong> {{ room.features }}</p>
      <button class="btn btn-primary" [routerLink]="['/add-booking-details']" [queryParams]="{ roomID: room.roomID, price: room.price }">
        Book Your Room
      </button>

    </div>
  </div>
</div>

<!-- Optional loading indicator -->
<div *ngIf="!room" class="text-center my-5">
  <p>Loading room details...</p>
</div>
```

```
<!-- <div class="container my-5" *ngIf="room">
   <div class="card mx-auto" style="max-width: 600px">
     <img
       [src]="room.imageUrl"
       class="card-img-top"
       alt="{{ room.type }}"
       style="height: 250px; object-fit: cover"
     />
     <div class="card-body">
       <h5 class="card-title">{{ room.type }}</h5>
       <p class="card-text"><strong>Price:</strong> ₹{{ room.price }}</p>
       <p class="card-text"><strong>Features:</strong> {{ room.features }}</p>
       <button class="btn btn-primary" (click)="bookNow()">Book Your Room</button>
     </div>
   </div>
 </div>

 Optional loading indicator
 <div *ngIf="!room" class="text-center my-5">
   <p>Loading room details...</p>
 </div> -->
```

Booking.ts

```
// src/app/components/bookings/bookings.ts
import { routes } from './../../../app.routes';
import { ChangeDetectorRef, Component, OnInit } from '@angular/core';
import { ActivatedRoute, RouterModule } from '@angular/router';
import { RoomService, Room } from '../../../services/room/room.service';
import { CommonModule } from '@angular/common';
import { Router } from '@angular/router';

@Component({
  selector: 'app-bookings',
  templateUrl: './bookings.html',
  styleUrls: ['./bookings.css'],
  imports: [CommonModule, RouterModule],
})
export class BookingsComponent implements OnInit {
  room!: Room;

  constructor(
    private route: ActivatedRoute,
    private roomService: RoomService,
    private cdr: ChangeDetectorRef,
    private router: Router
  ) {}

  ngOnInit(): void {
    const roomID = +this.route.snapshot.paramMap.get('roomID')!;
    this.roomService.getRoomById(roomID).subscribe((data) => {
      this.room = data;
```

```
    console.log('Room details fetched successfully:', this.room);
    this.cdr.markForCheck(); // Ensure change detection runs
  });
}

bookNow(): void {
  this.router.navigate(['/add-booking-details'], {
    queryParams: {
      roomID: this.room.roomID,
      price: this.room.price,
    },
  });
}

// Helper Method to add ROOM Image
getRoomImage(type: string): string {
  switch (type.toLowerCase()) {
    case 'single':
      return '/assets/RoomsImage/SingleRoom.jpeg';
    case 'double':
      return '/assets/RoomsImage/DoubleRoom.jpg';
    case 'family':
      return '/assets/RoomsImage/FamilyRoom.jpeg';
    case 'deluxe':
      return '/assets/RoomsImage/DeluxeRoom.jpeg';
    default:
      return '/assets/RoomsImage/default.jpeg'; // Optional: Add a fallback image
  }
}
```

}

Booking.service.ts

```
// src/app/services/booking/booking.service.ts
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';
import { environment } from '../../../environments/environment';
import { CookieService } from 'ngx-cookie-service';

export interface Booking {
  roomID: number;
  checkInDate: string;
  checkOutDate: string;
  status: boolean;
}

@Injectable({
  providedIn: 'root'
})
export class BookingService {
  constructor(private http: HttpClient, private cookieService: CookieService) { }

  createBooking(booking: Booking): Observable<any> {
    const token = this.cookieService.get('token'); // ✅ Get token from cookie

    const headers = new HttpHeaders({
      'Authorization': `Bearer ${token}`,
      'Content-Type': 'application/json'
```

```
    });

    return this.http.post<any>(`${environment.apiBaseUrl}/Bookings`, booking, { headers });
  }

  cancelBooking(bookingId: number): Observable<any> {
    const token = this.cookieService.get('token'); // ✅ Get token from cookie

    const headers = new HttpHeaders({
      'Authorization': `Bearer ${token}`,
      'Content-Type': 'application/json'
    });

    return this.http.delete<any>(`${environment.apiBaseUrl}/Bookings/${bookingId}`, {
headers });
  }

  // Method to fetch all the Bookings of LoggedIn User
  getMyBookings(): Observable<any> {
    const token = this.cookieService.get('token');

    const headers = new HttpHeaders({
      'Authorization': `Bearer ${token}`,
      'Content-Type': 'application/json'
    });

    return this.http.get<any>(`${environment.apiBaseUrl}/Bookings/my`, { headers });
  }
}
```

Backend – code:

BookingController.cs

```
using System.Security.Claims;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Cors;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Controllers
{
   [Route("api/[controller]")]
   [ApiController]
   [EnableCors("AllowFrontend")]
   public class BookingsController : ControllerBase
   {
      private readonly IBookingService _bookingService;

      public BookingsController(IBookingService bookingService)
      {
         _bookingService = bookingService;
      }
      [HttpPost]
      [Authorize(Roles = "User")]
      public async Task<IActionResult> Create(CreateBookingDto dto)
      {
```

```
    var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
    if (userIdClaim == null)
        return Unauthorized("User ID not found in token.");

    int userId = int.Parse(userIdClaim.Value);

    try
    {
        var result = await _bookingService.CreateBookingAsync(dto, userId);
        return Ok(result);
    }
    catch (Exception ex)
    {
        Console.WriteLine("❌ Booking Error: " + ex.Message);
        return StatusCode(500, "Server Error: " + ex.Message);
    }
}



// ✅ Secure "My Bookings" - always uses logged-in user's ID
[HttpGet("my")]
[Authorize(Roles = "User")]
public async Task<IActionResult> GetMyBookings()
{
    var userIdClaim = User.FindFirst(ClaimTypes.NameIdentifier);
    if (userIdClaim == null)
        return Unauthorized("User ID not found in token.");

    int userId = int.Parse(userIdClaim.Value);
```

```
            var bookings = await _bookingService.GetBookingsByUserIdAsync(userId);
            return Ok(bookings);
        }
    }
}
```

BookingService.cs

```
using AutoMapper;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Services.Implementations
{
    public class BookingService : IBookingService
    {
        private readonly IBookingRepository _bookingRepository;
        private readonly IRoomRepository _roomRepository;
        private readonly IRoomService _roomService;
        private readonly IPaymentRepository _paymentRepository;
        private readonly IMapper _mapper;

        public BookingService(IBookingRepository bookingRepository, IRoomRepository
roomRepository,
                              IRoomService roomService, IPaymentRepository paymentRepository,
IMapper mapper)
        {
            _bookingRepository = bookingRepository;
            _roomRepository = roomRepository;
```

```
        _roomService = roomService;
        _paymentRepository = paymentRepository;
        _mapper = mapper;
    }

    public async Task<BookingDTO> GetBookingByIdAsync(int bookingId)
    {
        var booking = await _bookingRepository.GetByIdAsync(bookingId);
        return _mapper.Map<BookingDTO>(booking);
    }

    public async Task<IEnumerable<BookingDTO>> GetAllBookingsAsync()
    {
        var bookings = await _bookingRepository.GetAllAsync();
        return _mapper.Map<IEnumerable<BookingDTO>>(bookings);
    }

    public async Task<IEnumerable<BookingDTO>> GetBookingsByUserIdAsync(int
userId)
    {
        var bookings = await _bookingRepository.GetBookingsByUserIdAsync(userId);
        return _mapper.Map<IEnumerable<BookingDTO>>(bookings);
    }

    public async Task<BookingDTO> CreateBookingAsync(CreateBookingDto
bookingDto, int userId)

    {
```

```
        var room = await _bookingRepository.GetRoomByIdAsync(bookingDto.RoomID);

        if (room == null)

            throw new Exception("Room not found.");

        // Check for overlapping bookings

        var existingBookings = await
_bookingRepository.GetBookingsByRoomIdAsync(bookingDto.RoomID);

        bool isOverlapping = existingBookings.Any(b =>

            b.CheckInDate <= DateOnly.FromDateTime(bookingDto.CheckOutDate) &&

            b.CheckOutDate >= DateOnly.FromDateTime(bookingDto.CheckInDate));

        if (isOverlapping)

            throw new Exception("Room is already booked for the selected dates.");

        var booking = _mapper.Map<Booking>(bookingDto);

        booking.UserId = userId;

        booking.HotelID = room.HotelId;

        await _bookingRepository.AddAsync(booking);
```

```
        await _bookingRepository.SaveChangesAsync();

        return _mapper.Map<BookingDTO>(booking);

    }



    // ✅ New Method to Check Room Availability
    private async Task<bool> IsRoomAvailableAsync(int roomId, DateTime checkIn, DateTime checkOut)
    {
        DateOnly checkInDateOnly = DateOnly.FromDateTime(checkIn);
        DateOnly checkOutDateOnly = DateOnly.FromDateTime(checkOut);

        var conflictingBookings = await _bookingRepository.GetAllAsync();

        return !conflictingBookings.Any(b =>
            b.RoomId == roomId &&
            b.CheckInDate.HasValue && b.CheckOutDate.HasValue && // ✅ Ensure dates are not null
            ((checkInDateOnly >= b.CheckInDate.Value && checkInDateOnly < b.CheckOutDate.Value) ||  // Check-in conflicts
             (checkOutDateOnly > b.CheckInDate.Value && checkOutDateOnly <= b.CheckOutDate.Value) || // Check-out conflicts
             (checkInDateOnly <= b.CheckInDate.Value && checkOutDateOnly >= b.CheckOutDate.Value))); // Enclosing date conflicts
    }
```

```
        public async Task<bool> UpdateBookingAsync(int bookingId, UpdateBookingDto
bookingDto)
        {
            var existingBooking = await _bookingRepository.GetByIdAsync(bookingId);
            if (existingBooking == null) return false;

            _mapper.Map(bookingDto, existingBooking);
            _bookingRepository.Update(existingBooking);
            await _bookingRepository.SaveChangesAsync();
            return true;
        }

        public async Task<bool> DeleteBookingAsync(int bookingId)
        {
            var booking = await _bookingRepository.GetByIdAsync(bookingId);
            if (booking == null) return false;

            _bookingRepository.Delete(booking);
            await _bookingRepository.SaveChangesAsync();
            return true;
        }
    }
}
```

BookingRepository.cs

```
using Microsoft.EntityFrameworkCore;

using SmartHotelBooking.Models;
```

```
using SmartHotelBooking.Repositories.Interfaces;

namespace SmartHotelBooking.Repositories.Implementations

{

   public class BookingRepository : IBookingRepository

   {

      private readonly HotelBookingContext _context;

      public BookingRepository(HotelBookingContext context)

      {

         _context = context;

      }

      public async Task<Booking> GetByIdAsync(int bookingId) =>

         await _context.Bookings.FindAsync(bookingId);

      public async Task<IEnumerable<Booking>> GetAllAsync() =>

         await _context.Bookings.ToListAsync();
```

```
        //public async Task<IEnumerable<Booking>> GetBookingsByUserIdAsync(int userId)
=>

        //    await _context.Bookings.Where(b => b.UserId == userId).ToListAsync();

        public async Task<IEnumerable<Booking>> GetBookingsByUserIdAsync(int userId)

        {

            return await _context.Bookings

                .Include(b => b.Room)

                .ThenInclude(r => r.Hotel)

                .Where(b => b.UserId == userId && b.Status == true)

                .ToListAsync();

        }



        public async Task<IEnumerable<Booking>> GetBookingsByRoomIdAsync(int roomId)

        {

            return await _context.Bookings

                .Where(b => b.RoomId == roomId)
```

```
        .ToListAsync();
}
public void UpdateRoom(Room room)
{
    _context.Rooms.Update(room);
}
public async Task<Room> GetRoomByIdAsync(int roomId)
{
    return await _context.Rooms.FirstOrDefaultAsync(r => r.RoomId == roomId);
}
public async Task AddAsync(Booking booking) =>
    await _context.Bookings.AddAsync(booking);
public void Update(Booking booking) =>
    _context.Bookings.Update(booking);
```

```
        public void Delete(Booking booking) =>

            _context.Bookings.Remove(booking);

        public async Task SaveChangesAsync() =>
            await _context.SaveChangesAsync();

    }
}
```

Booking.cs (model)

```
using System;
using System.Collections.Generic;

namespace SmartHotelBooking.Models;

public partial class Booking
{
    public int BookingId { get; set; }

    public int? UserId { get; set; }

    public int? RoomId { get; set; }

    public DateOnly? CheckInDate { get; set; }

    public DateOnly? CheckOutDate { get; set; }

    public bool Status { get; set; } = false; // ✅ Changed from string to boolean
```

```
    public int? HotelID { get; set; }

    public virtual ICollection<Payment> Payments { get; set; } = new List<Payment>();

    public virtual ICollection<Redemption> Redemptions { get; set; } = new
List<Redemption>();

    public virtual Room? Room { get; set; }

    public virtual User? User { get; set; }
}
```

BookingDto.cs

```
namespace SmartHotelBooking.DTOs
{
    public class BookingDTO
    {
        public int BookingID { get; set; }
        public int UserID { get; set; }
        public int RoomID { get; set; }
        public int HotelID { get; set; }
        public DateOnly CheckInDate { get; set; }
        public DateOnly CheckOutDate { get; set; }
        public string? Status { get; set; }

        public string? HotelName { get; set; }  // ✅ New
        public string? RoomType { get; set; }   // ✅ New
    }
```

```
}
```

CreateBookingDTO.cs

```
namespace SmartHotelBooking.DTOs
{
    public class CreateBookingDto
    {
        //public int BookingID { get; set; }
        //public int UserID { get; set; }
        public int RoomID { get; set; }
        //public int HotelID { get; set; }
        public DateTime CheckInDate { get; set; }
        public DateTime CheckOutDate { get; set; }

        public bool Status { get; set; } = false;

        //public int? PaymentID { get; set; }
    }
}
```

MY_Booking -UI

My-bookings.html

```
<!-- src/app/components/user/my-bookings/my-bookings.html -->
<div class="container mt-5 mb-5">
  <h2 class="text-center mb-4">📖 My Bookings</h2>

  <ng-container *ngIf="bookings$ | async as bookings">
    <div *ngIf="bookings.length === 0" class="alert alert-warning text-center">
```

```
    No bookings found.
  </div>

  <div *ngIf="bookings.length > 0" class="table-responsive">
    <table class="table table-striped table-hover table-bordered align-middle shadow-sm">
      <thead class="table-dark text-center">
        <tr>
          <th>Hotel Name</th>
          <th>Room Type</th>
          <th>Check-In</th>
          <th>Check-Out</th>
          <th>Status</th>
        </tr>
      </thead>
      <tbody class="text-center">
        <tr *ngFor="let booking of bookings">
          <td>{{ booking.hotelName }}</td>
          <td>{{ booking.roomType }}</td>
          <td>{{ booking.checkInDate }}</td>
          <td>{{ booking.checkOutDate }}</td>
          <td>
            <span class="badge" [ngClass]="booking.status === 'Confirmed' ? 'bg-success' : 'bg-
danger'">
              {{ booking.status }}
            </span>
          </td>
        </tr>
      </tbody>
    </table>
```

```
    </div>
  </ng-container>
</div>
```

My-booking.css

```
.table {
    border-radius: 0.5rem;
    overflow: hidden;
  }

  .badge {
    font-size: 0.9rem;
    padding: 0.5em 1em;
  }

  h2 {
    font-weight: bold;
  }

  .alert {
    font-size: 1.1rem;
  }
```

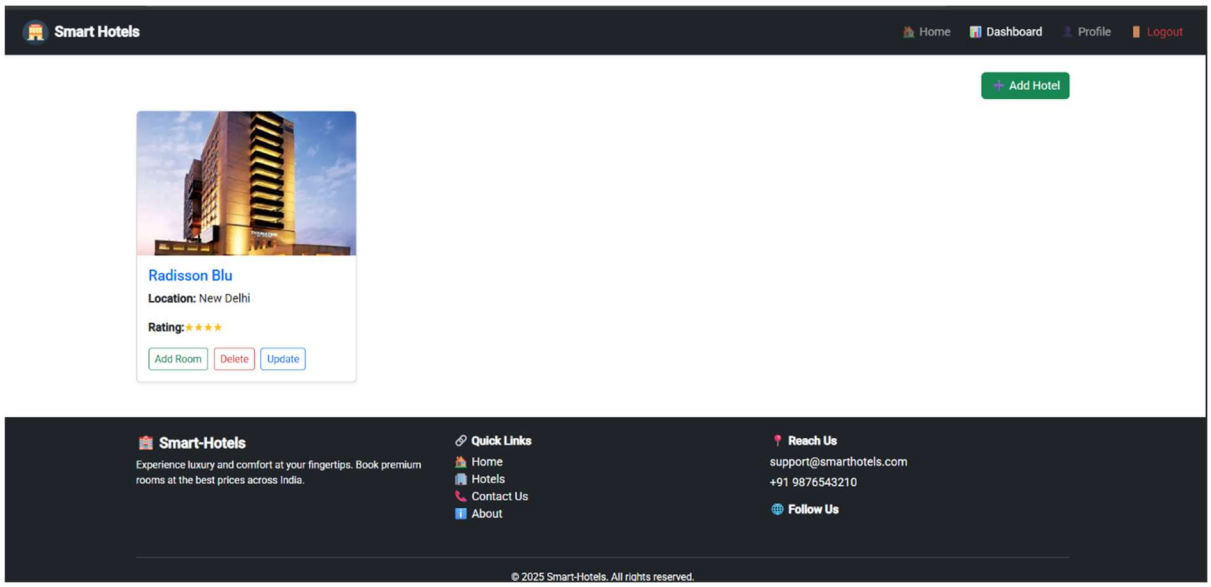
My-bookings.ts

```
import { Component, OnInit, ChangeDetectorRef } from '@angular/core';
import { CommonModule } from '@angular/common';
import { BookingService } from '../../../services/booking/booking.service';
import { Observable } from 'rxjs';
```

```
interface Booking {
  bookingID: number;
  hotelName: string;
  roomType: string;
  checkInDate: string;
  checkOutDate: string;
  status: string;
}

@Component({
  selector: 'app-my-bookings',
  standalone: true,
  templateUrl: './my-bookings.html',
  styleUrls: ['./my-bookings.css'],
  imports: [CommonModule],
})
export class MyBookings implements OnInit {
  bookings$!: Observable<Booking[]>;

  constructor(private bookingService: BookingService,  private cd: ChangeDetectorRef) {}

  ngOnInit(): void {
    this.bookings$ = this.bookingService.getMyBookings();
    console.log('Bookings:', this.bookings$);
    this.cd.detectChanges();
  }
}
```

Manager Dashboard

Hotel



Hotel.service.ts

```
// src/app/services/hotel.service.ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { environment } from '../../../environments/environment';

export interface Hotel {
  hotelID: number;
  name: string;
  location: string;
  managerID: number;
  amenities: string;
  rating: number;
  imageUrl: string;
}

@Injectable({
```

```
  providedIn: 'root',
})
export class HotelService {
  constructor(private http: HttpClient) { }

  // Method to get all hotels
  getAllHotels(): Observable<Hotel[]> {
    return this.http.get<Hotel[]>(`${environment.apiBaseUrl}/Hotels`, { withCredentials: true });
  }

  // Method to add Hotel by Manager Only
  addHotel(formData: FormData, managerID: number): Observable<any> {
    return this.http.post(`${environment.apiBaseUrl}/Hotels/manager/${managerID}`, formData, {
      withCredentials: true,
    });
  }

  // Method to get all hotels created by the LoggedIn Manager Only.
  getHotelsByManagerId(managerID: number): Observable<Hotel[]> {
    return this.http.get<Hotel[]>(`${environment.apiBaseUrl}/Hotels/manager/${managerID}`, {
      withCredentials: true,
    });
  }

  // Method to delete a hotel by ID and Manager ID
  deleteHotel(hotelID: number, managerID: number): Observable<any> {
```

```
  return
this.http.delete(`${environment.apiBaseUrl}/Hotels/${hotelID}/manager/${managerID}
`, {
    withCredentials: true,
  });
 }

 // Method to update a hotel by ID and Manager ID
 updateHotel(hotelID: number, managerID: number, hotelData: any): Observable<any>
{
  return this.http.put(
    `${environment.apiBaseUrl}/Hotels/${hotelID}/manager/${managerID}`,
    hotelData,
    {
      withCredentials: true,
      headers: { 'Content-Type': 'application/json' }
    }
  );
 }

 // Method to get a Hotel by it's ID.
 getHotelById(hotelID: number): Observable<Hotel> {
  return this.http.get<Hotel>(`${environment.apiBaseUrl}/Hotels/${hotelID}`, {
    withCredentials: true
  });
 }



}
```

Hotel.html

```
<!-- src/app/components/core/hotel/hotel.html -->

<div class="container my-4">

 <div class="row">

  <ng-container *ngIf="hotels$ | async as hotels; else loading">

   <div *ngIf="hotels.length > 0; else noHotels">

    <div class="row">

     <div *ngFor="let hotel of hotels" class="col-12 col-sm-6 col-md-4 col-lg-3 mb-4">

      <div class="card shadow-sm h-100 hotel-card"
(click)="viewRooms(hotel.hotelID)" style="cursor: pointer;">

       <img [src]="getImageUrl(hotel.imageUrl)" class="card-img-top" alt="{{
hotel.name }}"

        style="height: 200px; object-fit: cover;">

       <div class="card-body">

        <h5 class="card-title text-primary">{{ hotel.name }}</h5>

        <p class="card-text"><strong>Location:</strong> {{ hotel.location }}</p>

        <p class="card-text">

         <strong>Rating:</strong>

         <span class="text-warning">

          <ng-container *ngFor="let star of [].constructor(hotel.rating)">★</ng-
container>

         </span>

        </p>


        <!-- ⭐ Add this below rating -->

        <div class="mt-2">

         <button class="btn btn-outline-primary btn-sm"
(click)="viewReviews(hotel.hotelID)">

          💬 Reviews
```

```
            </button>
          </div>

        </div>
      </div>
    </div>
  </div>
 </ng-container>

 <ng-template #loading>
  <p>Loading hotels...</p>
 </ng-template>
 <ng-template #noHotels>
  <p>No hotels available.</p>
 </ng-template>
</div>
</div>
```

Hotel.ts file :-

```
// src/app/components/hotel/hotel.ts
import {
 ChangeDetectionStrategy,
 Component,
 OnInit
} from '@angular/core';
import { HotelService, Hotel } from '../../../services/hotel/hotel.service';
import { Router } from '@angular/router';
```

```
import { CommonModule } from '@angular/common';
import { Observable } from 'rxjs';
import { AuthService } from '../../../services/auth/auth.service';

@Component({
  selector: 'app-hotel',
  templateUrl: './hotel.html',
  styleUrls: ['./hotel.css'],
  changeDetection: ChangeDetectionStrategy.OnPush,
  standalone: true,
  imports: [CommonModule],
})
export class HotelComponent implements OnInit {
  hotels$!: Observable<Hotel[]>;
  userRole: string = '';

  constructor(
    private hotelService: HotelService,
    private router: Router,
    private authService: AuthService
  ) { }

  ngOnInit(): void {
    this.userRole = this.authService.getRole() ?? '';
    this.hotels$ = this.hotelService.getAllHotels();

    // Debugging API response
    this.hotels$.subscribe({
```

```
    next: (hotels) => console.log('Hotels fetched:', hotels),
    error: (err) => console.error('Error fetching hotels:', err)
  });
}



viewRooms(hotelID: number): void {
  this.router.navigate(['/rooms', hotelID]);
}

addHotel(): void {
  this.router.navigate(['/add-hotel']);
}

// Method to construct the image URL
getImageUrl(imagePath: string): string {
  const baseUrl = 'http://localhost:5281'; // Replace with your actual base URL
  return `${baseUrl}${imagePath}`;
}



// ✅ Fix this in hotel.ts
addRoom(hotelID: number): void {
  const managerID = this.authService.getUserId();
  console.log('Redirecting to add-room with:', { hotelID, managerID });

  // ✅ Use lowercase keys to match add-room.ts
  this.router.navigate(['/add-room'], {
```

```
    queryParams: {
      hotelId: hotelID,     // lowercase
      managerId: managerID   // lowercase
    }
  });
 }

 viewReviews(hotelID: number): void {
  this.router.navigate(['/hotel-reviews', hotelID]);
 }
}
```

Hotel.css file:-

```
.hotel-container {
 padding: 20px;
}

.add-hotel-button {
 margin-bottom: 20px;
}

.add-hotel-button button {
 padding: 10px 20px;
 font-size: 16px;
 background-color: #007bff;
 color: white;
 border: none;
 border-radius: 4px;
```

```
  cursor: pointer;
}

.hotel-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 20px;
}
.hotel-card {
  border: 1px solid #ddd;
  border-radius: 8px;
  overflow: hidden;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  cursor: pointer;
  transition: transform 0.2s ease;
}
.hotel-card:hover {
  transform: scale(1.02);
}
.hotel-card img {
  width: 100%;
  height: 200px;
  object-fit: cover;
}
.hotel-info {
  padding: 15px;
}
.hotel-info h5 {
```

```
  margin: 0 0 10px;
}

.stars {
  color: #ffc107;
}
```

Backend :-

CreateHotelDto.cs file

```
using System.ComponentModel.DataAnnotations;

namespace SmartHotelBooking.DTOs
{
    public class CreateHotelDto
    {
        [Required]
        public string? Name { get; set; }

        [Required]
        public string? Location { get; set; }

        [Required]
        public int ManagerID { get; set; }

        [Required]
        public string? Amenities { get; set; }
```

```
        [Range(0, 5)]
        public decimal Rating { get; set; }

        public IFormFile? Image { get; set; }
    }
}
```

HotelsController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [EnableCors("AllowFrontend")]
    public class HotelsController : ControllerBase
    {
        private readonly IHotelService _hotelService;

        public HotelsController(IHotelService hotelService)
        {
            _hotelService = hotelService;
        }
```

```
    [HttpGet("{hotelId}")]
    [Authorize(Roles = "User,Manager")]
    public async Task<IActionResult> GetHotelById(int hotelId)
    {
      var hotel = await _hotelService.GetHotelByIdAsync(hotelId);

      if (hotel == null)
        return NotFound(new { Message = "Hotel not found" });

      return Ok(hotel);
    }

    [HttpGet]
    [Authorize(Roles = "User,Manager")]
    public async Task<IActionResult> GetAll() => Ok(await
_hotelService.GetAllHotelsAsync());

    [HttpGet("manager/{managerId}")]
    public async Task<IActionResult> GetHotelsByManager(int managerId)
    {
      var hotels = await _hotelService.GetHotelsByManagerAsync(managerId);
      return Ok(hotels);
    }

    [HttpPost("manager/{managerId}")]
    [Authorize(Roles = "Manager")]
    public async Task<IActionResult> CreateHotel(int managerId, [FromForm]
CreateHotelDto dto)
```

```
    {
        if (!ModelState.IsValid)
            return BadRequest(ModelState);

        try
        {
            var created = await _hotelService.CreateHotelAsync(managerId, dto);
            return Ok(new { Message = "Hotel created successfully", Hotel = created });
        }
        catch (InvalidOperationException ex)
        {
            return Conflict(new { Message = ex.Message });
        }
    }



    [HttpPut("{hotelId}/manager/{managerId}")]
    [Authorize(Roles = "Manager")]
    public async Task<IActionResult> UpdateHotel(int hotelId, int managerId,
UpdateHotelDto dto)
    {
        var result = await _hotelService.UpdateHotelAsync(hotelId, managerId, dto);
        if (!result)
            return NotFound(new { Message = "Hotel not found or unauthorized" });

        return Ok(new { Message = "Hotel updated successfully" });
    }
```

```
    [HttpDelete("{hotelId}/manager/{managerId}")]
    [Authorize(Roles = "Manager")]
    public async Task<IActionResult> DeleteHotel(int hotelId, int managerId)
    {

      var result = await _hotelService.DeleteHotelAsync(hotelId, managerId);
      if (!result)
        return NotFound(new { Message = "Hotel not found or unauthorized" });

      return Ok(new { Message = "Hotel deleted successfully" });
    }
  }
}
```

IHotelService.cs

```
using SmartHotelBooking.DTOs;

namespace SmartHotelBooking.Services.Interfaces
{
  public interface IHotelService
  {
    Task<IEnumerable<HotelDTO>> GetAllHotelsAsync();
    Task<HotelDTO?> GetHotelByIdAsync(int hotelId);
    Task<List<HotelDTO>> GetHotelsByManagerAsync(int managerId);
    Task<HotelDTO> CreateHotelAsync(int managerId, CreateHotelDto dto);
    Task<bool> UpdateHotelAsync(int hotelId, int managerId, UpdateHotelDto dto);
    Task<bool> DeleteHotelAsync(int hotelId, int managerId);
  }
```

```
}
```

HotelService.cs file:-

```
using AutoMapper;
using Microsoft.EntityFrameworkCore;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Services.Implementations
{
  public class HotelService : IHotelService
  {
    private readonly HotelBookingContext _context;
    private readonly IMapper _mapper;


    public HotelService(HotelBookingContext context, IMapper mapper)
    {
      _context = context;
      _mapper = mapper;
    }

    public async Task<IEnumerable<HotelDTO>> GetAllHotelsAsync()
    {
      var hotels = await _context.Hotels.ToListAsync();
      return _mapper.Map<IEnumerable<HotelDTO>>(hotels);
    }
```

```
    public async Task<HotelDTO> GetHotelByIdAsync(int hotelId)
    {
        var hotel = await _context.Hotels.FindAsync(hotelId);
        return _mapper.Map<HotelDTO>(hotel);
    }

    public async Task<List<HotelDTO>> GetHotelsByManagerAsync(int managerId)
    {
        var hotels = await _context.Hotels
            .Where(h => h.ManagerId == managerId)
            .ToListAsync();
        return _mapper.Map<List<HotelDTO>>(hotels);
    }

    public async Task<HotelDTO> CreateHotelAsync(int managerId, CreateHotelDto
hoteldto)
    {
        var existingHotel = await _context.Hotels.ToListAsync();

        if (existingHotel.Any(h => h.Name == hoteldto.Name && h.Location ==
hoteldto.Location && h.ManagerId == managerId))
            throw new InvalidOperationException("Hotel already exists for this manager.");

        var hotel = _mapper.Map<Hotel>(hoteldto);
        hotel.ManagerId = managerId;

        // ✅ Save image
```

```
        if (hoteldto.Image != null && hoteldto.Image.Length > 0)
        {
            var uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot/images/hotels");
            Directory.CreateDirectory(uploadsFolder); // Ensure folder exists

            var fileName = $"{Guid.NewGuid()}_{hoteldto.Image.FileName}";
            var filePath = Path.Combine(uploadsFolder, fileName);

            using (var stream = new FileStream(filePath, FileMode.Create))
            {
                await hoteldto.Image.CopyToAsync(stream);
            }

            hotel.ImageUrl = $"/images/hotels/{fileName}";
        }

        await _context.Hotels.AddAsync(hotel);
        await _context.SaveChangesAsync();

        return _mapper.Map<HotelDTO>(hotel);
    }

    public async Task<bool> UpdateHotelAsync(int hotelId, int managerId,
UpdateHotelDto dto)
    {
        // Check if manager exists
        var managerExists = await _context.Users.AnyAsync(u => u.UserId == managerId);
        if (!managerExists)
```

```
        return false;

    var hotel = await _context.Hotels
        .FirstOrDefaultAsync(h => h.HotelId == hotelId && h.ManagerId == managerId);

    if (hotel == null)
        return false;

    _mapper.Map(dto, hotel);

    try
    {
        await _context.SaveChangesAsync();
        return true;
    }
    catch (DbUpdateException)
    {
        return false;
    }
}

public async Task<bool> DeleteHotelAsync(int hotelId, int managerId)
{
    var hotel = await _context.Hotels
        .Include(h=>h.Rooms)
        .FirstOrDefaultAsync(h => h.HotelId == hotelId && h.ManagerId == managerId);

    if (hotel == null)
```

```
            return false;



        if (hotel.Rooms.Any())
            return false; // Or return a custom message like "Hotel has rooms and cannot be deleted."
        _context.Hotels.Remove(hotel);

        try
        {
            await _context.SaveChangesAsync();
            return true;
        }
        catch (DbUpdateException)
        {
            // Log or handle exception
            return false;
        }
      }
   }
}
```

IHotelRepository.cs file :-

```
using SmartHotelBooking.Models;


namespace SmartHotelBooking.Repositories.Interfaces
{
   public interface IHotelRepository
   {
      Task<Hotel> GetByIdAsync(int hotelId);
```

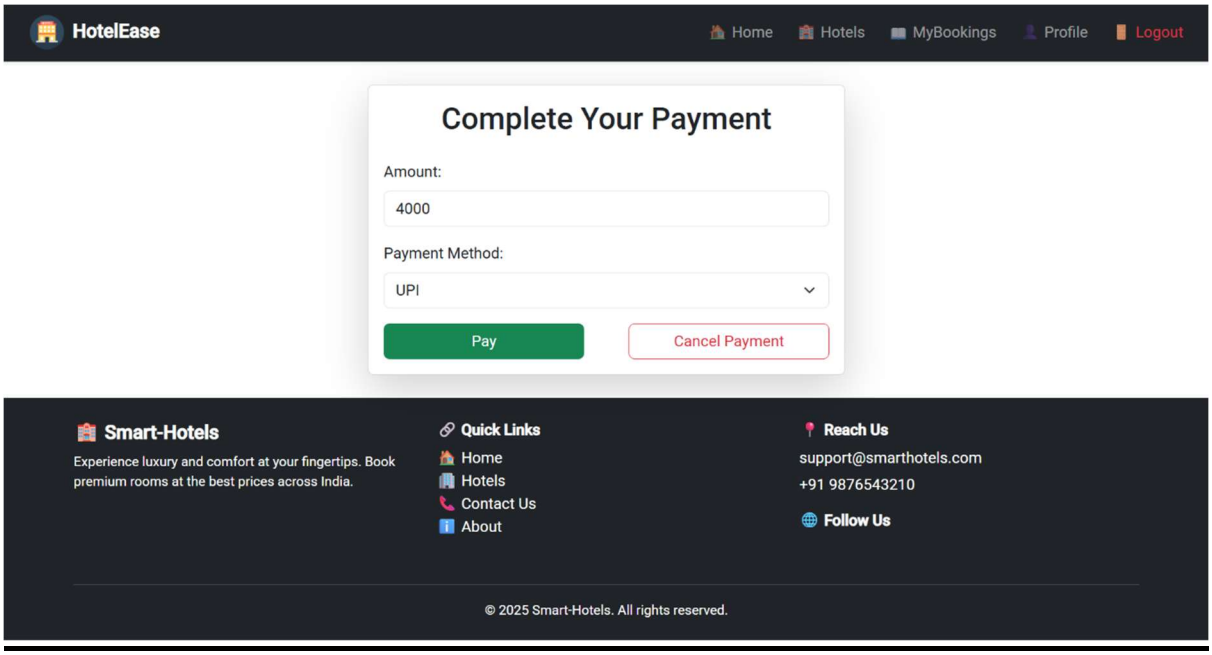
```
    Task<IEnumerable<Hotel>> GetAllAsync();
    Task AddAsync(Hotel hotel);
    void Update(Hotel hotel);
    void Delete(Hotel hotel);
    Task SaveChangesAsync();
  }
}
```

HotelRepository :-

```
using Microsoft.EntityFrameworkCore;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;

namespace SmartHotelBooking.Repositories.Implementations
{
  public class HotelRepository : IHotelRepository
  {
    private readonly HotelBookingContext _context;

    public HotelRepository(HotelBookingContext context)
    {
      _context = context;
    }

    public async Task<Hotel> GetByIdAsync(int hotelId) =>
      await _context.Hotels.FindAsync(hotelId);

    public async Task<IEnumerable<Hotel>> GetAllAsync() =>
      await _context.Hotels.ToListAsync();
```

```
    public async Task AddAsync(Hotel hotel) =>
        await _context.Hotels.AddAsync(hotel);

    public void Update(Hotel hotel) =>
        _context.Hotels.Update(hotel);

    public void Delete(Hotel hotel) =>
        _context.Hotels.Remove(hotel);

    public async Task SaveChangesAsync() =>
        await _context.SaveChangesAsync();
  }
}
```

PAYMENT – UI



Payment – Html

```
<!-- src/app/components/user/payment/payment.html -->
<div class="container py-4">
   <div class="card shadow-lg mx-auto" style="max-width: 500px;">
     <div class="card-body">
       <h2 class="card-title text-center mb-4">Complete Your Payment</h2>

       <div class="mb-3">
         <label class="form-label">Amount:</label>
         <input type="number" [(ngModel)]="amount" class="form-control" readonly />
       </div>

       <div class="mb-3">
         <label class="form-label">Payment Method:</label>
         <select [(ngModel)]="paymentMethod" class="form-select">
           <option value="UPI">UPI</option>
           <option value="Cash">Card</option>
         </select>
       </div>

       <div class="d-flex justify-content-between">
         <button class="btn btn-success w-45" (click)="pay()">Pay</button>
         <button class="btn btn-outline-danger w-45" (click)="cancel()">Cancel
Payment</button>
       </div>
     </div>
```

```
  </div>
 </div>
```

Payment.css

```
.w-45 {
   width: 45%;
}
```

Payment.ts

```
// src/app/components/user/payment/payment.component.ts
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { PaymentService } from '../../../services/payment/payment.service';
import { BookingService } from '../../../services/booking/booking.service';
import { FormsModule } from '@angular/forms';

@Component({
  selector: 'app-payment',
  templateUrl: './payment.html',
  styleUrl: './payment.css',
  standalone: true, // ✅ Mark as standalone
  imports: [FormsModule], // ✅ Include FormsModule
})
export class Payment implements OnInit {
  hotelID!: number;
  userID!: number;
  bookingID!: number;
  amount!: number;
  paymentMethod: string = 'UPI';
```

```
constructor(
  private route: ActivatedRoute,
  private router: Router,
  private paymentService: PaymentService,
  private bookingService: BookingService
) {}

ngOnInit(): void {
  this.route.queryParams.subscribe(params => {
    this.hotelID = +params['hotelId'];
    this.userID = +params['userId'];
    this.bookingID = +params['bookingId'];
    this.amount = +params['amount'];

    console.log('Received amount:', this.amount);
  });
}

pay(): void {
  const paymentData = {
    paymentID: 0,
    hotelID: this.hotelID,
    userID: this.userID,
    bookingID: this.bookingID,
    amount: this.amount,
    // status: true,
    paymentMethod: this.paymentMethod
  };
```

```
    // Log the payment data to the console for debugging
    console.log('Sending payment data:', paymentData);
    this.paymentService.makePayment(paymentData).subscribe(response => {
      alert('Pay for Booking the Room');
      this.router.navigate(['/my-bookings']);
    });
  }

  cancel(): void {
    this.bookingService.cancelBooking(this.bookingID).subscribe(() => {
      this.router.navigate(['/home']);
    });
  }
}
```

Backend – code

PaymentController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Cors;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Controllers
{
    [Route("api/[controller]")]
```

```
    [ApiController]
    [EnableCors("AllowFrontend")]
    public class PaymentsController : ControllerBase
    {
        private readonly IPaymentService _paymentService;
        public PaymentsController(IPaymentService paymentService)
        {
            _paymentService = paymentService;
        }
        [HttpPost]
        [Authorize(Roles = "Manager,User,Admin")]
        public async Task<IActionResult> MakePayment(CreatePaymentDto dto) => Ok(await
_paymentService.CreatePaymentAsync(dto));
    }
}
```

paymentService.cs

```
using AutoMapper;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Services.Implementations
{
    public class PaymentService : IPaymentService
    {
        private readonly IPaymentRepository _paymentRepository;
        private readonly IMapper _mapper;
```

```
    public PaymentService(IPaymentRepository paymentRepository, IMapper mapper)
    {
        _paymentRepository = paymentRepository;
        _mapper = mapper;
    }

    public async Task<PaymentDTO> GetPaymentByIdAsync(int paymentId)
    {
        var payment = await _paymentRepository.GetByIdAsync(paymentId);
        return _mapper.Map<PaymentDTO>(payment);
    }

    public async Task<IEnumerable<PaymentDTO>> GetAllPaymentsAsync()
    {
        var payments = await _paymentRepository.GetAllAsync();
        return _mapper.Map<IEnumerable<PaymentDTO>>(payments);
    }

    public async Task<IEnumerable<PaymentDTO>> GetPaymentsByUserIdAsync(int
userId)
    {
        var payments = await _paymentRepository.GetPaymentsByUserIdAsync(userId);
        return _mapper.Map<IEnumerable<PaymentDTO>>(payments);
    }


    public async Task<PaymentDTO> CreatePaymentAsync(CreatePaymentDto
paymentDto)
    {
```

```
        var payment = _mapper.Map<Payment>(paymentDto);

        // Save payment first
        await _paymentRepository.AddAsync(payment);
        await _paymentRepository.SaveChangesAsync();

        // Now update Booking.Status = true and Room.Availability = false
        var booking = await
_paymentRepository.GetBookingByIdAsync(payment.BookingId.Value);
        if (booking != null)
        {
            booking.Status = true;
            var room = await
_paymentRepository.GetRoomByIdAsync(booking.RoomId.Value);
            if (room != null)
            {
                room.Availability = false;
                _paymentRepository.UpdateRoom(room);
            }

            _paymentRepository.UpdateBooking(booking);
            await _paymentRepository.SaveChangesAsync();
        }

        return _mapper.Map<PaymentDTO>(payment);
    }
    public async Task<bool> UpdatePaymentAsync(int paymentId, UpdatePaymentDto
paymentDto)
    {
        var existingPayment = await _paymentRepository.GetByIdAsync(paymentId);
```

```
            if (existingPayment == null) return false;

            _mapper.Map(paymentDto, existingPayment);
            _paymentRepository.Update(existingPayment);
            await _paymentRepository.SaveChangesAsync();
            return true;
        }

        public async Task<bool> DeletePaymentAsync(int paymentId)
        {
            var payment = await _paymentRepository.GetByIdAsync(paymentId);
            if (payment == null) return false;

            _paymentRepository.Delete(payment);
            await _paymentRepository.SaveChangesAsync();
            return true;
        }
    }
}
```

PaymentRepository:

```
using Microsoft.EntityFrameworkCore;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;

namespace SmartHotelBooking.Repositories.Implementations
{
    public class PaymentRepository : IPaymentRepository
    {
```

```
    private readonly HotelBookingContext _context;

    public PaymentRepository(HotelBookingContext context)
    {
        _context = context;
    }

    public async Task<Payment> GetByIdAsync(int paymentId) =>
        await _context.Payments.FindAsync(paymentId);

    public async Task<IEnumerable<Payment>> GetAllAsync() =>
        await _context.Payments.ToListAsync();

    public async Task<IEnumerable<Payment>> GetPaymentsByUserIdAsync(int userId)
=>
        await _context.Payments.Where(p => p.UserId == userId).ToListAsync();

    public async Task AddAsync(Payment payment) =>
        await _context.Payments.AddAsync(payment);

    public void Update(Payment payment) =>
        _context.Payments.Update(payment);

    public void Delete(Payment payment) =>
        _context.Payments.Remove(payment);

    public async Task SaveChangesAsync() =>
        await _context.SaveChangesAsync();
```

```
        // Add these Methods for new changes in logics
        public async Task<Booking> GetBookingByIdAsync(int bookingId)
        {
            return await _context.Bookings.FirstOrDefaultAsync(b => b.BookingId ==
bookingId);
        }

        public async Task<Room> GetRoomByIdAsync(int roomId)
        {
            return await _context.Rooms.FirstOrDefaultAsync(r => r.RoomId == roomId);
        }

        public void UpdateRoom(Room room)
        {
            _context.Rooms.Update(room);
        }

        public void UpdateBooking(Booking booking)
        {
            _context.Bookings.Update(booking);
        }

    }
}
```

Payment.cs-(model)

```
using System;
using System.Collections.Generic;

namespace SmartHotelBooking.Models;
```

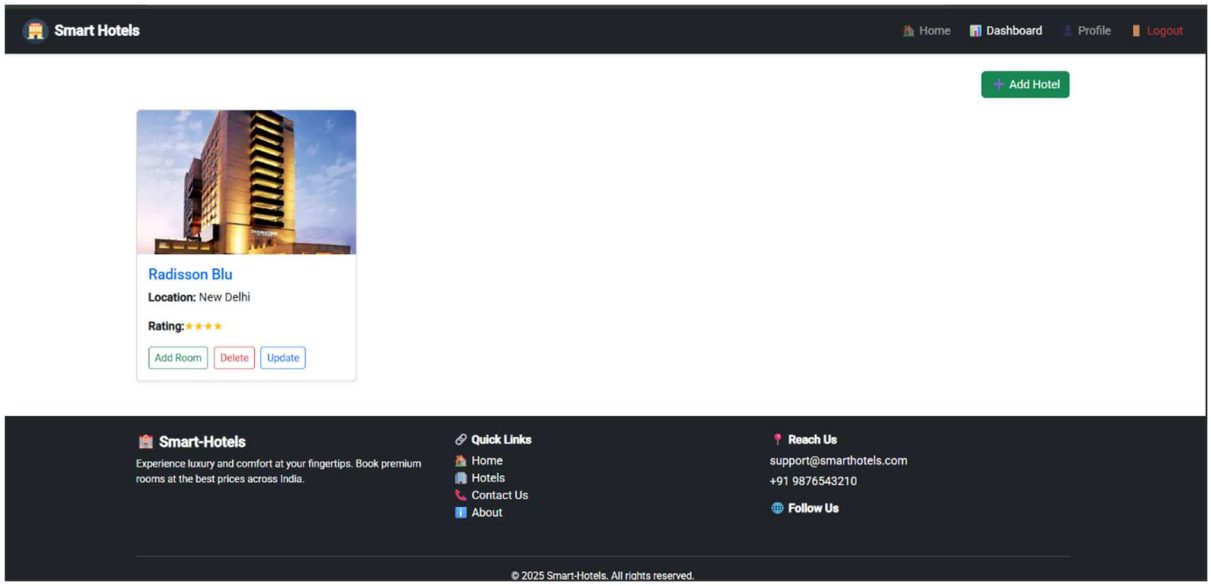
```
public partial class Payment
{
    public int PaymentId { get; set; }

    public int? UserId { get; set; }

    public int? BookingId { get; set; }
    public decimal? Amount { get; set; }
    //public string? Status { get; set; }
    public string? PaymentMethod { get; set; }
    public virtual Booking? Booking { get; set; }

    public virtual User? User { get; set; }
}
```

CreatePayment.dto

```
namespace SmartHotelBooking.DTOs
{
    public class CreatePaymentDto
    {
        //public int PaymentID { get; set; }
        public int HotelID { get; set; }
        public int UserID { get; set; }
        public int BookingID { get; set; }
        public decimal Amount { get; set; }
        //public string Status { get; set; }
        public string? PaymentMethod { get; set; }
    }
}
```

Manager Dashboard

Hotel



Hotel.service.ts

```
// src/app/services/hotel.service.ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { environment } from '../../../environments/environment';


export interface Hotel {
  hotelID: number;
  name: string;
  location: string;
  managerID: number;
  amenities: string;
  rating: number;
  imageUrl: string;
}
```

```
@Injectable({
  providedIn: 'root',
})
export class HotelService {
  constructor(private http: HttpClient) { }

  // Method to get all hotels
  getAllHotels(): Observable<Hotel[]> {
    return this.http.get<Hotel[]>(` ${environment.apiBaseUrl}/Hotels` , { withCredentials: true });
  }

  // Method to add Hotel by Manager Only
  addHotel(formData: FormData, managerID: number): Observable<any> {
    return this.http.post(` ${environment.apiBaseUrl}/Hotels/manager/${managerID}` , formData, {
      withCredentials: true,
    });
  }

  // Method to get all hotels created by the LoggedIn Manager Only.
  getHotelsByManagerId(managerID: number): Observable<Hotel[]> {
    return this.http.get<Hotel[]>(` ${environment.apiBaseUrl}/Hotels/manager/${managerID}` , {
      withCredentials: true,
    });
  }
```

```
  // Method to delete a hotel by ID and Manager ID
  deleteHotel(hotelID: number, managerID: number): Observable<any> {
    return
this.http.delete(` ${environment.apiBaseUrl}/Hotels/${hotelID}/manager/${managerID}
`, {
      withCredentials: true,
    });
  }

  // Method to update a hotel by ID and Manager ID
  updateHotel(hotelID: number, managerID: number, hotelData: any): Observable<any>
{
    return this.http.put(
      `${environment.apiBaseUrl}/Hotels/${hotelID}/manager/${managerID}`,
      hotelData,
      {
        withCredentials: true,
        headers: { 'Content-Type': 'application/json' }
      }
    );
  }

  // Method to get a Hotel by it's ID.
  getHotelById(hotelID: number): Observable<Hotel> {
    return this.http.get<Hotel>(`${environment.apiBaseUrl}/Hotels/${hotelID}`, {
      withCredentials: true
    });
  }
```

}

Hotel.html

```
<!-- src/app/components/core/hotel/hotel.html -->

<div class="container my-4">

 <div class="row">

  <ng-container *ngIf="hotels$ | async as hotels; else loading">

   <div *ngIf="hotels.length > 0; else noHotels">

    <div class="row">

     <div *ngFor="let hotel of hotels" class="col-12 col-sm-6 col-md-4 col-lg-3 mb-4">

      <div class="card shadow-sm h-100 hotel-card"
(click)="viewRooms(hotel.hotelID)" style="cursor: pointer;">

       <img [src]="getImageUrl(hotel.imageUrl)" class="card-img-top" alt="{{
hotel.name }}"

        style="height: 200px; object-fit: cover;">

       <div class="card-body">

        <h5 class="card-title text-primary">{{ hotel.name }}</h5>

        <p class="card-text"><strong>Location:</strong> {{ hotel.location }}</p>

        <p class="card-text">

         <strong>Rating:</strong>

         <span class="text-warning">

          <ng-container *ngFor="let star of [].constructor(hotel.rating)">★</ng-
container>

         </span>

        </p>


        <!-- ⭐ Add this below rating -->

        <div class="mt-2">
```

```
            <button class="btn btn-outline-primary btn-sm"
(click)="viewReviews(hotel.hotelID)">
              💬 Reviews
            </button>
          </div>

        </div>
      </div>
    </div>
  </div>
 </ng-container>

 <ng-template #loading>
  <p>Loading hotels...</p>
 </ng-template>
 <ng-template #noHotels>
  <p>No hotels available.</p>
 </ng-template>
</div>
</div>
```

Hotel.ts file :-

```
// src/app/components/hotel/hotel.ts
import {
 ChangeDetectionStrategy,
 Component,
 OnInit
```

```
} from '@angular/core';
import { HotelService, Hotel } from '../../../services/hotel/hotel.service';
import { Router } from '@angular/router';
import { CommonModule } from '@angular/common';
import { Observable } from 'rxjs';
import { AuthService } from '../../../services/auth/auth.service';

@Component({
  selector: 'app-hotel',
  templateUrl: './hotel.html',
  styleUrls: ['./hotel.css'],
  changeDetection: ChangeDetectionStrategy.OnPush,
  standalone: true,
  imports: [CommonModule],
})
export class HotelComponent implements OnInit {
  hotels$!: Observable<Hotel[]>;
  userRole: string = '';

  constructor(
    private hotelService: HotelService,
    private router: Router,
    private authService: AuthService
  ) { }

  ngOnInit(): void {
    this.userRole = this.authService.getRole() ?? '';
    this.hotels$ = this.hotelService.getAllHotels();
```

```
  // Debugging API response
  this.hotels$.subscribe({
    next: (hotels) => console.log('Hotels fetched:', hotels),
    error: (err) => console.error('Error fetching hotels:', err)
  });
}



viewRooms(hotelID: number): void {
  this.router.navigate(['/rooms', hotelID]);
}

addHotel(): void {
  this.router.navigate(['/add-hotel']);
}

// Method to construct the image URL
getImageUrl(imagePath: string): string {
  const baseUrl = 'http://localhost:5281'; // Replace with your actual base URL
  return `${baseUrl}${imagePath}`;
}



// ✅ Fix this in hotel.ts
addRoom(hotelID: number): void {
  const managerID = this.authService.getUserId();
  console.log('Redirecting to add-room with:', { hotelID, managerID });
```

```
  // ✅ Use lowercase keys to match add-room.ts
   this.router.navigate(['/add-room'], {
    queryParams: {
     hotelId: hotelID,     // lowercase
     managerId: managerID   // lowercase
    }
   });
  }

  viewReviews(hotelID: number): void {
   this.router.navigate(['/hotel-reviews', hotelID]);
  }
}
```

Hotel.css file:-

```
.hotel-container {
  padding: 20px;
}

.add-hotel-button {
  margin-bottom: 20px;
}

.add-hotel-button button {
  padding: 10px 20px;
  font-size: 16px;
  background-color: #007bff;
```

```
  color: white;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}

.hotel-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 20px;
}
.hotel-card {
  border: 1px solid #ddd;
  border-radius: 8px;
  overflow: hidden;
  box-shadow: 0 2px 8px rgba(0, 0, 0, 0.1);
  cursor: pointer;
  transition: transform 0.2s ease;
}
.hotel-card:hover {
  transform: scale(1.02);
}
.hotel-card img {
  width: 100%;
  height: 200px;
  object-fit: cover;
}
.hotel-info {
```

```
  padding: 15px;
}
.hotel-info h5 {
  margin: 0 0 10px;
}
.stars {
  color: #ffc107;
}
```

Backend :-

CreateHotelDto.cs file

```
using System.ComponentModel.DataAnnotations;

namespace SmartHotelBooking.DTOs
{
   public class CreateHotelDto
   {
      [Required]
      public string? Name { get; set; }

      [Required]
      public string? Location { get; set; }

      [Required]
      public int ManagerID { get; set; }
```

```
        [Required]
        public string? Amenities { get; set; }

        [Range(0, 5)]
        public decimal Rating { get; set; }

        public IFormFile? Image { get; set; }
    }
}
```

HotelsController.cs

```
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Cors;
using Microsoft.AspNetCore.Mvc;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    [EnableCors("AllowFrontend")]
    public class HotelsController : ControllerBase
    {
        private readonly IHotelService _hotelService;

        public HotelsController(IHotelService hotelService)
        {
            _hotelService = hotelService;
```

```
    }


    [HttpGet("{hotelId}")]
    [Authorize(Roles = "User,Manager")]
    public async Task<IActionResult> GetHotelById(int hotelId)
    {
      var hotel = await _hotelService.GetHotelByIdAsync(hotelId);

      if (hotel == null)
        return NotFound(new { Message = "Hotel not found" });

      return Ok(hotel);
    }


    [HttpGet]
    [Authorize(Roles = "User,Manager")]
    public async Task<IActionResult> GetAll() => Ok(await
_hotelService.GetAllHotelsAsync());

    [HttpGet("manager/{managerId}")]
    public async Task<IActionResult> GetHotelsByManager(int managerId)
    {
      var hotels = await _hotelService.GetHotelsByManagerAsync(managerId);
      return Ok(hotels);
    }
```

```
    [HttpPost("manager/{managerId}")]
    [Authorize(Roles = "Manager")]
    public async Task<IActionResult> CreateHotel(int managerId, [FromForm]
CreateHotelDto dto)
    {
      if (!ModelState.IsValid)
        return BadRequest(ModelState);

      try
      {
        var created = await _hotelService.CreateHotelAsync(managerId, dto);
        return Ok(new { Message = "Hotel created successfully", Hotel = created });
      }
      catch (InvalidOperationException ex)
      {
        return Conflict(new { Message = ex.Message });
      }
    }



    [HttpPut("{hotelId}/manager/{managerId}")]
    [Authorize(Roles = "Manager")]
    public async Task<IActionResult> UpdateHotel(int hotelId, int managerId,
UpdateHotelDto dto)
    {
      var result = await _hotelService.UpdateHotelAsync(hotelId, managerId, dto);
      if (!result)
        return NotFound(new { Message = "Hotel not found or unauthorized" });
```

```
        return Ok(new { Message = "Hotel updated successfully" });
    }

    [HttpDelete("{hotelId}/manager/{managerId}")]
    [Authorize(Roles = "Manager")]
    public async Task<IActionResult> DeleteHotel(int hotelId, int managerId)
    {

        var result = await _hotelService.DeleteHotelAsync(hotelId, managerId);
        if (!result)
            return NotFound(new { Message = "Hotel not found or unauthorized" });

        return Ok(new { Message = "Hotel deleted successfully" });
    }
  }
}
```

IHotelService.cs

```
using SmartHotelBooking.DTOs;

namespace SmartHotelBooking.Services.Interfaces
{
  public interface IHotelService
  {
    Task<IEnumerable<HotelDTO>> GetAllHotelsAsync();
    Task<HotelDTO?> GetHotelByIdAsync(int hotelId);
    Task<List<HotelDTO>> GetHotelsByManagerAsync(int managerId);
    Task<HotelDTO> CreateHotelAsync(int managerId, CreateHotelDto dto);
```

```
    Task<bool> UpdateHotelAsync(int hotelId, int managerId, UpdateHotelDto dto);
    Task<bool> DeleteHotelAsync(int hotelId, int managerId);
  }
}
```

HotelService.cs file:-

```
using AutoMapper;
using Microsoft.EntityFrameworkCore;
using SmartHotelBooking.DTOs;
using SmartHotelBooking.Models;
using SmartHotelBooking.Services.Interfaces;

namespace SmartHotelBooking.Services.Implementations
{
  public class HotelService : IHotelService
  {
    private readonly HotelBookingContext _context;
    private readonly IMapper _mapper;


    public HotelService(HotelBookingContext context, IMapper mapper)
    {
      _context = context;
      _mapper = mapper;
    }

    public async Task<IEnumerable<HotelDTO>> GetAllHotelsAsync()
    {
```

```
        var hotels = await _context.Hotels.ToListAsync();
        return _mapper.Map<IEnumerable<HotelDTO>>(hotels);
    }

    public async Task<HotelDTO> GetHotelByIdAsync(int hotelId)
    {
        var hotel = await _context.Hotels.FindAsync(hotelId);
        return _mapper.Map<HotelDTO>(hotel);
    }

    public async Task<List<HotelDTO>> GetHotelsByManagerAsync(int managerId)
    {
        var hotels = await _context.Hotels
            .Where(h => h.ManagerId == managerId)
            .ToListAsync();
        return _mapper.Map<List<HotelDTO>>(hotels);
    }

    public async Task<HotelDTO> CreateHotelAsync(int managerId, CreateHotelDto
hoteldto)
    {
        var existingHotel = await _context.Hotels.ToListAsync();

        if (existingHotel.Any(h => h.Name == hoteldto.Name && h.Location ==
hoteldto.Location && h.ManagerId == managerId))
            throw new InvalidOperationException("Hotel already exists for this manager.");

        var hotel = _mapper.Map<Hotel>(hoteldto);
        hotel.ManagerId = managerId;
```

```
        // ✅ Save image
        if (hoteldto.Image != null && hoteldto.Image.Length > 0)
        {
            var uploadsFolder = Path.Combine(Directory.GetCurrentDirectory(),
"wwwroot/images/hotels");
            Directory.CreateDirectory(uploadsFolder); // Ensure folder exists

            var fileName = $"{Guid.NewGuid()}_{hoteldto.Image.FileName}";
            var filePath = Path.Combine(uploadsFolder, fileName);

            using (var stream = new FileStream(filePath, FileMode.Create))
            {
                await hoteldto.Image.CopyToAsync(stream);
            }

            hotel.ImageUrl = $"/images/hotels/{fileName}";
        }

        await _context.Hotels.AddAsync(hotel);
        await _context.SaveChangesAsync();

        return _mapper.Map<HotelDTO>(hotel);
    }

    public async Task<bool> UpdateHotelAsync(int hotelId, int managerId,
UpdateHotelDto dto)
    {
```

```
    // Check if manager exists
    var managerExists = await _context.Users.AnyAsync(u => u.UserId == managerId);
    if (!managerExists)
        return false;

    var hotel = await _context.Hotels
        .FirstOrDefaultAsync(h => h.HotelId == hotelId && h.ManagerId == managerId);

    if (hotel == null)
        return false;

    _mapper.Map(dto, hotel);

    try
    {
        await _context.SaveChangesAsync();
        return true;
    }
    catch (DbUpdateException)
    {
        return false;
    }
}

public async Task<bool> DeleteHotelAsync(int hotelId, int managerId)
{
    var hotel = await _context.Hotels
        .Include(h=>h.Rooms)
```

```
            .FirstOrDefaultAsync(h => h.HotelId == hotelId && h.ManagerId == managerId);


        if (hotel == null)
            return false;



        if (hotel.Rooms.Any())
            return false; // Or return a custom message like "Hotel has rooms and cannot be deleted."
        _context.Hotels.Remove(hotel);


        try
        {
            await _context.SaveChangesAsync();
            return true;
        }
        catch (DbUpdateException)
        {
            // Log or handle exception
            return false;
        }
    }
  }
}
```

IHotelRepository.cs file :-

```
using SmartHotelBooking.Models;


namespace SmartHotelBooking.Repositories.Interfaces
{
```

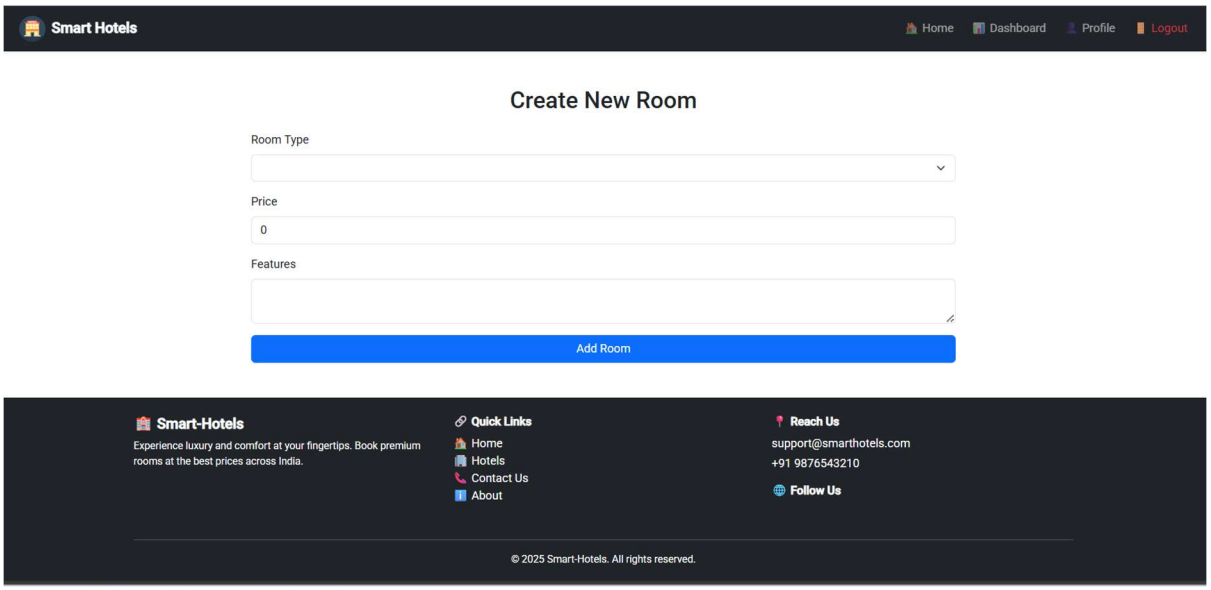
```
  public interface IHotelRepository
  {
    Task<Hotel> GetByIdAsync(int hotelId);
    Task<IEnumerable<Hotel>> GetAllAsync();
    Task AddAsync(Hotel hotel);
    void Update(Hotel hotel);
    void Delete(Hotel hotel);
    Task SaveChangesAsync();
  }
}
```

HotelRepository :-

```
using Microsoft.EntityFrameworkCore;
using SmartHotelBooking.Models;
using SmartHotelBooking.Repositories.Interfaces;

namespace SmartHotelBooking.Repositories.Implementations
{
  public class HotelRepository : IHotelRepository
  {
    private readonly HotelBookingContext _context;

    public HotelRepository(HotelBookingContext context)
    {
      _context = context;
    }

    public async Task<Hotel> GetByIdAsync(int hotelId) =>
      await _context.Hotels.FindAsync(hotelId);
```

```
    public async Task<IEnumerable<Hotel>> GetAllAsync() =>
      await _context.Hotels.ToListAsync();

    public async Task AddAsync(Hotel hotel) =>
      await _context.Hotels.AddAsync(hotel);

    public void Update(Hotel hotel) =>
      _context.Hotels.Update(hotel);

    public void Delete(Hotel hotel) =>
      _context.Hotels.Remove(hotel);

    public async Task SaveChangesAsync() =>
      await _context.SaveChangesAsync();
  }
}
```

ADD ROOM



C0DE

```
// src/app/services/room.service.ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { environment } from '../../../environments/environment';

export interface Room {
 roomID: number;
 hotelID: number;
 managerID: number;
 type: string;
 price: number;
 availability: boolean;
 features: string;
 imageUrl?: string;
}

export interface CreateRoom {
```

```
  hotelID: number;
  managerID: number;
  type: string;
  price: number;
  availability: boolean;
  features: string;
}

@Injectable({
  providedIn: 'root'
})
export class RoomService {

  constructor(private http: HttpClient) {}

  getRoomsByHotelId(hotelID: number): Observable<Room[]> {
    return this.http.get<Room[]>(`${environment.apiBaseUrl}/Rooms/hotel/${hotelID}`);
  }

  getRoomById(roomID: number): Observable<Room> {
    return this.http.get<Room>(`${environment.apiBaseUrl}/Rooms/${roomID}`);
  }

  addRoom(room: CreateRoom): Observable<Room> {
    return this.http.post<Room>(`${environment.apiBaseUrl}/Rooms`, room);
  }

  // delete(roomID: number): Observable<any> {
```

```
  //   return this.http.delete(` ${environment.apiBaseUrl}/Rooms/${roomID}` );
  // }


}
```

ROOM.HTML

```
<div class="container mt-5 mb-5">
  <h2 class="text-center mb-4">Create New Room</h2>
  <form (ngSubmit)="addRoom()" #roomForm="ngForm" class="w-75 mx-auto">
    <div class="mb-3">
      <label for="type" class="form-label">Room Type</label>
      <select id="type" class="form-select" required [(ngModel)]="type" name="type">
        <option value="Single">Single</option>
        <option value="Double">Double</option>
        <option value="Family">Family</option>
        <option value="Deluxe">Deluxe</option>
        <option value="Other">Other</option>
      </select>
      <div *ngIf="type === 'Other'" class="mt-2">
        <input type="text" class="form-control" placeholder="Enter custom type"
[(ngModel)]="type" name="customType">
      </div>
    </div>


    <div class="mb-3">
      <label for="price" class="form-label">Price</label>
      <input type="number" id="price" class="form-control" required [(ngModel)]="price"
name="price">
    </div>
```

```
  <div class="mb-3">
    <label for="features" class="form-label">Features</label>
    <textarea id="features" class="form-control" [(ngModel)]="features"
name="features"></textarea>
  </div>

  <button type="submit" class="btn btn-primary w-100">Add Room</button>
 </form>
</div>
```

ROOM.TS

```
// src/app/components/manager/add-room/add-room.ts
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { RoomService, CreateRoom } from '../../../services/room/room.service';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';

@Component({
 selector: 'app-add-room',
 templateUrl: './add-room.html',
 styleUrls: ['./add-room.css'],
 imports:[CommonModule, FormsModule]
})

export class AddRoomComponent implements OnInit {
 hotelId: number = 0;
 managerId: number = 0;
```

```
  type: string = '';
  price: number = 0;
  availability: boolean = true;
  features: string = '';

  constructor(private route: ActivatedRoute, private router: Router, private roomService:
RoomService) {}

  ngOnInit(): void {
    this.route.queryParams.subscribe(params => {
      this.hotelId = +params['hotelId'];
      this.managerId = +params['managerId'];

      console.log('💡 Parsed Hotel ID:', this.hotelId);
      console.log('💡 Parsed Manager ID:', this.managerId);
    });
  }

  addRoom(): void {
    const newRoom: CreateRoom = {
      hotelID: this.hotelId,
      managerID: this.managerId,
      type: this.type,
      price: this.price,
      availability: this.availability,
      features: this.features
    };
```

```
    console.log('Creating room with data:', newRoom);

    this.roomService.addRoom(newRoom).subscribe({
      next: (room) => {
        console.log('✅ Room created successfully:', room);
        alert('Room created successfully!');
        this.router.navigate(['/rooms', this.hotelId]);
      },
      error: (err) => {
        console.error('❌ Error creating room:', err);
        alert('Something went wrong!');
      }
    });
  }
}
```