**MODELS**

**User**

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace HotelBookingApplication.Models

{

public class User

{

[Key]

public string Email { get; set; }

public string Password { get; set; }

public string Name { get; set; }

public string PhoneNumber { get; set; }

public string Address { get; set; }

public string Role { get; set; }

public byte[] Key { get; set; }

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Room**

using System.ComponentModel.DataAnnotations.Schema;

namespace HotelBookingApplication.Models

{

public class Room

{

public int RoomId { get; set; }

public string RoomType { get; set; }

public int HotelId { get; set; }

[ForeignKey("HotelId")]

public Hotel Hotel { get; set; }

public int TotalRooms { get; set; }

public int AvailableRooms { get; set; }

public float Price { get; set; }

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**RoomFacility**

using System.ComponentModel.DataAnnotations.Schema;

namespace HotelBookingApplication.Models

{

public class RoomFacility

{

public int RoomFacilityId { get; set; }

public int RoomId { get; set; }

[ForeignKey("RoomId")]

public Room Room{ get; set; }

public string Facilities { get; set; }

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Hotel**

using System.ComponentModel.DataAnnotations.Schema;

using System.ComponentModel.DataAnnotations;

namespace HotelBookingApplication.Models

{

public class Hotel

{

[Key]

public int HotelId { get; set; }

public string HotelName { get; set; }

public string UserId { get; set; }

[ForeignKey("UserId")]

public User user { get; set; }

public string Location { get; set; }

public string Address { get; set; }

public string PhoneNumber { get; set; }

public string Description { get; set; }

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Booking**

using System.ComponentModel.DataAnnotations.Schema;

namespace HotelBookingApplication.Models

{

public class Booking

{

public int BookingId { get; set; }

public String UserId { get; set; }

[ForeignKey("UserId")]

public User user { get; set; }

public string Date { get; set; }

public string CheckIn { get; set; }

public string CheckOut { get; set; }

public int RoomId { get; set; }

[ForeignKey("RoomId")]

public Room room { get; set; }

public int TotalRooms { get; set; }

public float Price { get; set; }

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DTOs**

**UserDTO**

using System.ComponentModel.DataAnnotations;

namespace HotelBookingApplication.Models.DTOs

{

public class UserDTO

{

[Required(ErrorMessage ="Email cannot be empty")]

public string Email { get; set; }

[Required(ErrorMessage ="Password cannot be empty")]

public string Password { get; set; }

public string? Role { get; set; }

public string? Token { get; set; }

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**UserRegisterDTO**

using System.ComponentModel.DataAnnotations;

namespace HotelBookingApplication.Models.DTOs

{

public class UserRegisterDTO

{

[Required(ErrorMessage = "Email cannot be empty")]

public string Email { get; set; }

public string Password { get; set; }

[Required (ErrorMessage ="Retype Password Cannot be Empty")]

[Compare("Password",ErrorMessage ="Password and Retyped password does not match")]

public string ReTypePassword { get; set; }

[Required(ErrorMessage ="Name cannot be empty")]

public string Name { get; set; }

[Required(ErrorMessage ="Phone number cannot be empty")]

public string PhoneNumber { get; set; }

[Required(ErrorMessage ="Address cannot be empty")]

public string Address { get; set; }

public object Token { get; internal set; }

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**RoomDTO**

namespace HotelBookingApplication.Models.DTOs

{

public class RoomDTO

{

public string RoomType { get; set; }

public string HotelId { get; set;}

public float Price { get; set;}

public int Capacity { get; set;}

public int TotalRooms { get; set;}

public string Desription { get; set;}

public List<string> RoomFacilities { get; set;}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**HotelDTO**

namespace HotelBookingApplication.Models.DTOs

{

public class HotelDTO

{

public string HotelName { get; set; }

public string UserId { get; set; }

public string Location { get; set; }

public string Address { get; set; }

public string PhoneNumber { get; set; }

public string Description { get; set; }

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**BookingDTO**

namespace HotelBookingApplication.Models.DTOs

{

public class BookingDTO

{

public string UserId { get; set; }

public string CheckIn { get; set; }

public string CheckOut { get; set; }

public int RoomId { get; set; }

public int TotalRooms { get; set; }

}

}

**CONTROLLERS**

**UserController**

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models.DTOs;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

namespace HotelBookingApplication.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class UserController : ControllerBase

{

private readonly IUserService \_userService;

private readonly ILogger \_logger;

public UserController(IUserService userService, ILogger<UserController> logger)

{

\_userService = userService;

\_logger = logger;

}

[HttpPost("register")]

public ActionResult Register(UserRegisterDTO userRegisterDTO)

{

string message = "";

try

{

var user = \_userService.Register(userRegisterDTO);

if (user != null)

{

\_logger.LogInformation("User Registerd");

return Ok(user);

}

}

catch (DbUpdateException exp)

{

message = "Duplicate username";

}

catch (Exception)

{

\_logger.LogError("Could not register user");

}

return BadRequest(message);

}

[HttpPost("login")]

public ActionResult Login(UserDTO userDTO)

{

var result = \_userService.Login(userDTO);

if (result != null)

{

\_logger.LogInformation("Login Successful");

return Ok(result);

}

\_logger.LogError("Login failed");

return Unauthorized("Invalid username or password");

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**RoomController**

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models.DTOs;

using HotelBookingApplication.Services;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

namespace HotelBookingApplication.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class RoomController : ControllerBase

{

private readonly IRoomService \_roomService;

private readonly ILogger \_logger;

public RoomController(IRoomService roomService, ILogger<RoomController> logger)

{

\_roomService = roomService;

\_logger = logger;

}

[HttpPost(" GetAvailableRooms")]

public ActionResult GetAvailableRooms(int hotelId, string checkIn, string checkOut)

{

string errorMessage = "";

try

{

var result = \_roomService.GetRooms(hotelId, checkIn, checkOut);

if (result != null)

{

\_logger.LogInformation("Rooms Displayed");

return Ok(result);

}

}

catch (NoRoomsAvailableException e)

{

errorMessage = e.Message;

}

\_logger.LogError("Unable to display rooms");

return BadRequest(errorMessage);

}

[HttpPost("CreateRooms")]

[Authorize(Roles = "Admin")]

public ActionResult CreateRooms(RoomDTO roomDTO)

{

var room = \_roomService.AddRoom(roomDTO);

if (room != null)

{

\_logger.LogInformation("Room Created");

return Ok(room);

}

\_logger.LogError("Unable to add room");

return BadRequest("Could not add rooms");

}

[HttpDelete("DeleteRooms")]

[Authorize(Roles = "Admin")]

public ActionResult DeleteRooms(int id)

{

bool roomId = \_roomService.RemoveRoom(id);

if (roomId)

{

\_logger.LogInformation("Room Deleted");

return Ok("The room has been deleted successfully");

}

\_logger.LogError("Unable to delete room");

return BadRequest("Invalid roomId");

}

[HttpPost("PromoteRooms")]

[Authorize(Roles = "Admin")]

public ActionResult PromoteRooms(int id, RoomDTO roomDTO)

{

var room = \_roomService.UpdateRoom(id, roomDTO);

if (room != null)

{

\_logger.LogInformation("Room Updated");

return Ok("Room updated successfully");

}

\_logger.LogError("Unable to update room");

return BadRequest("Unable to update");

}

}

}

**HotelController**

using HotelBookingApplication.Exceptions;

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models.DTOs;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Hosting;

namespace HotelBookingApplication.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class HotelController : ControllerBase

{

private readonly IHotelService \_hotelService;

private readonly ILogger \_logger;

public HotelController(IHotelService hotelService, ILogger<HotelController> logger)

{

\_hotelService = hotelService;

\_logger = logger;

}

[HttpPost("AddHotel")]

[Authorize(Roles = "Admin")]

public ActionResult AddHotel(HotelDTO hotelDTO)

{

var hotel = \_hotelService.AddHotel(hotelDTO);

if (hotel != null)

{

\_logger.LogInformation("Hotel Added");

return Ok(hotel);

}

\_logger.LogError("Could not add hotel");

return BadRequest("Could not add hotel");

}

[HttpGet]

public ActionResult GetHotel(string city)

{

string message = string.Empty;

try

{

var result = \_hotelService.GetHotels(city);

\_logger.LogInformation("Displayed Hotels");

return Ok(result);

}

catch (NoHotelsAvailableException ex)

{

message = ex.Message;

}

\_logger.LogError("Could not display hotels");

return BadRequest(message);

}

[HttpDelete("RemoveHotel")]

[Authorize(Roles = "Admin")]

public ActionResult RemoveHotel(int id)

{

var result = \_hotelService.RemoveHotel(id);

if (result)

{

\_logger.LogInformation("Hotel Removed");

return Ok("Hotel removed successfully");

}

\_logger.LogError("Could not remove hotel");

return BadRequest("Could not remove hotel");

}

[HttpPost("UpdateHotel")]

[Authorize(Roles = "Admin")]

public ActionResult UpdateHotel(int id, HotelDTO hotelDTO)

{

var result = \_hotelService.UpdateHotel(id, hotelDTO);

if (result != null)

{

\_logger.LogInformation("Hotel Updated");

return Ok(result);

}

\_logger.LogError("Could not update hotel");

return BadRequest("Could not update");

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**BookingController**

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models.DTOs;

using HotelBookingApplication.Services;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace HotelBookingApplication.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class BookingController : ControllerBase

{

private readonly IBookingService \_bookingService;

private readonly ILogger<UserController> \_logger;

public BookingController(IBookingService bookingService, ILogger<UserController> logger)

{

\_bookingService = bookingService;

\_logger = logger;

}

[HttpPost("addBooking")]

[Authorize(Roles = "User")]

public ActionResult AddBooking(BookingDTO bookingDTO)

{

var booking = \_bookingService.AddBookingDetails(bookingDTO);

if (booking != null)

{

\_logger.LogInformation("Booked successfully");

return Ok(booking);

}

\_logger.LogError("Could not book rooms");

return BadRequest("Could not book");

}

[HttpGet("adminBooking")]

[Authorize(Roles = "Admin")]

public ActionResult GetAdminBooking(int id)

{

var booking = \_bookingService.GetBooking(id);

if (booking != null)

{

\_logger.LogInformation("Admin booking details displayed");

return Ok(booking);

}

\_logger.LogError("Could not display admin bookings");

return BadRequest("No bookings found");

}

[HttpGet("userBooking")]

[Authorize(Roles = "User")]

public ActionResult GetUserBooking(string id)

{

var booking = \_bookingService.GetUserBooking(id);

if (booking != null)

{

\_logger.LogInformation("User booking details displayed");

return Ok(booking);

}

\_logger.LogError("Could not display user bookings");

return BadRequest("No bookings found");

}

[HttpPost("Update")]

public ActionResult UpdateBooking(int id, string status)

{

var booking = \_bookingService.UpdateBookingStatus(id, status);

if (booking != null)

{

\_logger.LogInformation("Booking status updated");

return Ok(booking);

}

\_logger.LogError("Could not update booing status");

return BadRequest("couldn't update");

}

}

}

**CONTEXTS**

**BookingContext**

using HotelBookingApplication.Models;

using Microsoft.EntityFrameworkCore;

namespace HotelBookingApplication.Contexts

{

public class BookingContext : DbContext

{

public BookingContext(DbContextOptions options) : base(options)

{

}

public DbSet<User> Users { get; set; }

public DbSet<Hotel> Hotels { get; set; }

public DbSet<Room> Rooms { get; set; }

public DbSet<RoomFacility> RoomFacilities { get; set; }

public DbSet<Booking> Bookings { get; set; }

}

}

INTERFACES

IUserService

using HotelBookingApplication.Models.DTOs;

namespace HotelBookingApplication.Interfaces

{

public interface IUserService

{

UserDTO Login(UserDTO userDTO);

UserDTO Register(UserRegisterDTO userRegisterDTO);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ITokenService**

using HotelBookingApplication.Models.DTOs;

namespace HotelBookingApplication.Interfaces

{

public interface ITokenService

{

string GetToken(UserDTO user);

object GetToken(UserRegisterDTO userRegisterDTO);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**IRepository**

namespace HotelBookingApplication.Interfaces

{

public interface IRepository<K, T>

{

T GetById(K key);

IList<T> GetAll();

T Add(T entity);

T Update(T entity);

T Delete(K key);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**IRoomService**

using HotelBookingApplication.Models;

using HotelBookingApplication.Models.DTOs;

namespace HotelBookingApplication.Interfaces

{

public interface IRoomService

{

List<Room> GetRooms(int hotelid, string checkIn, string checkOut);

RoomDTO AddRoom(RoomDTO room);

RoomDTO UpdateRoom(int id, RoomDTO room);

bool RemoveRoom(int id);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**IHotelService**

using HotelBookingApplication.Models;

using HotelBookingApplication.Models.DTOs;

namespace HotelBookingApplication.Interfaces

{

public interface IHotelService

{

List<Hotel> GetHotels(string city);

HotelDTO AddHotel(HotelDTO hotelDTO);

HotelDTO UpdateHotel(int id, HotelDTO hotelDTO);

bool RemoveHotel(int id);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**IBookingService**

using HotelBookingApplication.Models;

using HotelBookingApplication.Models.DTOs;

namespace HotelBookingApplication.Interfaces

{

public interface IBookingService

{

public BookingDTO AddBookingDetails(BookingDTO bookingDTO);

public Booking UpdateBookingStatus(int bookingId, string status);

public List<Booking> GetUserBooking(string userId);

public List<Booking> GetBooking(int hotelId);

}

}

**EXCEPTIONS**

**NoBookingsAvailableException**

using System.Runtime.Serialization;

namespace HotelBookingApplication.Exceptions

{

[Serializable]

public class NoBookingsAvailableException : Exception

{

string message;

public NoBookingsAvailableException()

{

message = "No bookings available to display";

}

public override string Message => message;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NoHotelsAvailableException**

namespace HotelBookingApplication.Exceptions

{

public class NoHotelsAvailableException : Exception

{

string message;

public NoHotelsAvailableException()

{

message = "No hotels are available for display";

}

public override string Message => message;

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**NoRoomsAvailableException**

using System.Runtime.Serialization;

﻿using System.Runtime.Serialization;

namespace HotelBookingApplication.Services

{

[Serializable]

internal class NoRoomsAvailableException : Exception

{

string message;

public NoRoomsAvailableException()

{

message = "No rooms are availble to display";

}

public override string Message => message;

}

}

**REPOSITORIES**

**UserRepository**

using HotelBookingApplication.Contexts;

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models;

using Microsoft.EntityFrameworkCore;

namespace HotelBookingApplication.Repositories

{

public class UserRepository : IRepository<String, User>

{

private readonly BookingContext \_context;

public UserRepository(BookingContext context)

{

\_context = context;

}

public User Add(User entity)

{

\_context.Users.Add(entity);

\_context.SaveChanges();

return entity;

}

public User Delete(string key)

{

var user = GetById(key);

if (user != null)

{

\_context.Users.Remove(user);

\_context.SaveChanges();

return user;

}

return null;

}

public IList<User> GetAll()

{

if (\_context.Users.Count() == 0)

return null;

return \_context.Users.ToList();

}

public User GetById(string key)

{

var user = \_context.Users.SingleOrDefault(u => u.Email == key);

return user;

}

public User Update(User entity)

{

var user = GetById(entity.Email);

if (user != null)

{

\_context.Entry<User>(user).State = EntityState.Modified;

\_context.SaveChanges();

return user;

}

return null;

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**RoomRepository**

using HotelBookingApplication.Contexts;

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models;

using Microsoft.Data.SqlClient;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Internal;

using static Microsoft.EntityFrameworkCore.DbLoggerCategory.Database;

namespace HotelBookingApplication.Repositories

{

public class RoomRepository : IRepository<int, Room>

{

private readonly BookingContext \_context;

public RoomRepository(BookingContext context)

{

\_context = context;

}

public Room Add(Room entity)

{

\_context.Rooms.Add(entity);

\_context.SaveChanges();

return entity;

}

public Room Delete(int key)

{

var room = GetById(key);

if (room != null)

{

\_context.Rooms.Remove(room);

\_context.SaveChanges();

return room;

}

return null;

}

public IList<Room> GetAll()

{

if (\_context.Rooms.Count() == 0)

return null;

return \_context.Rooms.ToList();

}

public Room GetById(int key)

{

var room = \_context.Rooms.SingleOrDefault(r => r.RoomId == key);

return room;

}

public Room Update(Room entity)

{

var room = GetById(entity.RoomId);

if (room != null)

{

\_context.Entry<Room>(room).State = EntityState.Modified;

\_context.SaveChanges();

return room;

}

return null;

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**RoomFacilityRepository**

using HotelBookingApplication.Contexts;

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models;

using Microsoft.EntityFrameworkCore;

namespace HotelBookingApplication.Repositories

{

public class RoomFacilityRepository : IRepository<int, RoomFacility>

{

private readonly BookingContext \_context;

public RoomFacilityRepository(BookingContext context)

{

\_context = context;

}

public RoomFacility Add(RoomFacility entity)

{

\_context.RoomFacilities.Add(entity);

\_context.SaveChanges();

return entity;

}

public RoomFacility Delete(int key)

{

var facilities = GetById(key);

if (facilities != null)

{

\_context.RoomFacilities.Remove(facilities);

\_context.SaveChanges();

return facilities;

}

return null;

}

public IList<RoomFacility> GetAll()

{

if (\_context.RoomFacilities.Count() == 0)

return null;

return \_context.RoomFacilities.ToList();

}

public RoomFacility GetById(int key)

{

var facility = \_context.RoomFacilities.SingleOrDefault(u => u.RoomFacilityId == key);

return facility;

}

public RoomFacility Update(RoomFacility entity)

{

var facility = GetById(entity.RoomFacilityId);

if (facility != null)

{

\_context.Entry<RoomFacility>(facility).State = EntityState.Modified;

\_context.SaveChanges();

return facility;

}

return null;

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**HotelRepository**

using HotelBookingApplication.Contexts;

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models;

using Microsoft.EntityFrameworkCore;

namespace HotelBookingApplication.Repositories

{

public class HotelRepository : IRepository<int, Hotel>

{

private readonly BookingContext \_context;

public HotelRepository(BookingContext context)

{

\_context = context;

}

public Hotel Add(Hotel entity)

{

\_context.Hotels.Add(entity);

\_context.SaveChanges();

return entity;

}

public Hotel Delete(int key)

{

var hotel = GetById(key);

if (hotel != null)

{

\_context.Hotels.Remove(hotel);

\_context.SaveChanges();

return hotel;

}

return null;

}

public IList<Hotel> GetAll()

{

if (\_context.Hotels.Count() == 0)

return null;

return \_context.Hotels.ToList();

}

public Hotel GetById(int key)

{

var hotel = \_context.Hotels.SingleOrDefault(u => u.HotelId == key);

return hotel;

}

public Hotel Update(Hotel entity)

{

var hotel = GetById(entity.HotelId);

if (hotel != null)

{

\_context.Entry<Hotel>(hotel).State = EntityState.Modified;

\_context.SaveChanges();

return hotel;

}

return null;

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**BookingRepository**

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Contexts;

using HotelBookingApplication.Models;

using Microsoft.EntityFrameworkCore;

namespace HotelBookingApplication.Repositories

{

public class BookingRepository : IRepository<int, Booking>

{

private readonly BookingContext \_context;

public BookingRepository(BookingContext context)

{

\_context = context;

}

public Booking Add(Booking entity)

{

\_context.Bookings.Add(entity);

\_context.SaveChanges();

return entity;

}

public Booking Delete(int key)

{

throw new NotImplementedException();

}

public IList<Booking> GetAll()

{

if (\_context.Bookings.Count() == 0)

return null;

return \_context.Bookings.ToList();

}

public Booking GetById(int key)

{

var booking = \_context.Bookings.SingleOrDefault(u => u.BookingId == key);

return booking;

}

public Booking Update(Booking entity)

{

var booking = GetById(entity.BookingId);

if (booking != null)

{

\_context.Entry<Booking>(booking).State = EntityState.Modified;

\_context.SaveChanges();

return booking;

}

return null;

}

}

}

**SERVICES**

**UserService**

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models.DTOs;

using HotelBookingApplication.Models;

using System.Security.Cryptography;

using System.Text;

namespace HotelBookingApplication.Services

{

public class UserService : IUserService

{

private readonly IRepository<string, User> \_repository;

private readonly ITokenService \_tokenService;

public UserService(IRepository<string, User> repository, ITokenService tokenService)

{

\_repository = repository;

\_tokenService = tokenService;

}

public UserDTO Login(UserDTO userDTO)

{

var user = \_repository.GetById(userDTO.Email);

if (user != null)

{

HMACSHA512 hmac = new HMACSHA512(user.Key);

var userpass = hmac.ComputeHash(Encoding.UTF8.GetBytes(userDTO.Password));

for (int i = 0; i < userpass.Length; i++)

{

if (user.Password[i] != userpass[i])

return null;

}

userDTO.Role = user.Role;

userDTO.Token = \_tokenService.GetToken(userDTO);

userDTO.Password = "";

return userDTO;

}

return null;

}

public UserDTO Register(UserRegisterDTO userRegisterDTO)

{

HMACSHA512 hmac = new HMACSHA512();

User user = new User()

{

Email = userRegisterDTO.Email,

Password = hmac.ComputeHash(Encoding.UTF8.GetBytes(userRegisterDTO.Password)),

PhoneNumber = userRegisterDTO.PhoneNumber,

Name = userRegisterDTO.Name,

Address = userRegisterDTO.Address,

Key = hmac.Key,

};

var result = \_repository.Add(user);

if (result != null)

{

userRegisterDTO.Token = \_tokenService.GetToken(userRegisterDTO);

userRegisterDTO.Password = "";

userRegisterDTO.ReTypePassword = "";

return userRegisterDTO;

}

return null;

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TokenService**

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models.DTOs;

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

namespace HotelBookingApplication.Services

{

public class TokenService : ITokenService

{

private readonly SymmetricSecurityKey \_key;

public TokenService(IConfiguration configuration)

{

var secretKey = configuration["SecretKey"].ToString();

\_key = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(secretKey));

}

public string GetToken(UserDTO user)

{

var claims = new List<Claim>()

{

new Claim(JwtRegisteredClaimNames.Name,user.Email),

new Claim("role",user.Role)

};

var cred = new SigningCredentials(\_key, SecurityAlgorithms.HmacSha512Signature);

var tokenDescription = new SecurityTokenDescriptor

{

Subject = new ClaimsIdentity(claims),

Expires = DateTime.Now.AddDays(1),

SigningCredentials = cred

};

var tokenHandler = new JwtSecurityTokenHandler();

var token = tokenHandler.CreateToken(tokenDescription);

return tokenHandler.WriteToken(token);

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**RoomService**

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models;

using HotelBookingApplication.Models.DTOs;

using HotelBookingApplication.Repositories;

using System.Runtime.Serialization;

namespace HotelBookingApplication.Services

{

public class RoomService : IRoomService

{

private readonly IRepository<int, Room> \_roomrepository;

private readonly IRepository<int, RoomFacility> \_roomFacilityRepository;

private readonly IRepository<int, Booking> \_bookingRepository;

public RoomService(IRepository<int, Room> repository, IRepository<int, RoomFacility> roomAmenityRepository, IRepository<int, Booking> bookingRepository)

{

\_roomrepository = repository;

\_roomFacilityRepository = roomAmenityRepository;

\_bookingRepository = bookingRepository;

}

public RoomDTO AddRoom(RoomDTO roomDTO)

{

Room room = new Room()

{

RoomType = roomDTO.RoomType,

Price = roomDTO.Price,

HotelId = roomDTO.HotelId,

TotalRooms = roomDTO.TotalRooms,

};

var result = \_roomrepository.Add(room);

int id = room.RoomId;

foreach (string a in roomDTO.RoomFacilities)

{

RoomFacility roomfacility = new RoomFacility()

{

RoomId = id,

Facilities = a,

};

\_roomFacilityRepository.Add(roomFacility);

}

if (result != null)

{

return roomDTO;

}

return null;

}

public List<Room> GetRooms(int hotelId, string checkIn, string checkOut)

{

var room = \_roomrepository.GetAll().Where(r => r.HotelId == hotelId).ToList();

var availableRoom = CheckAvailableRooms(room, checkIn, checkOut);

if (room.Count != 0)

{

return availableRoom;

}

throw new NoRoomsAvailableException();

}

private List<Room> CheckAvailableRooms(List<Room> room, string checkIn, string checkOut)

{

List<Room> roomList = new List<Room>();

foreach (var a in room)

{

var booking = (from Booking in \_bookingRepository

.GetAll()

.Where(booking =>

booking.RoomId == a.RoomId &&

(DateTime.Parse(checkIn).Date >= DateTime.Parse(booking.CheckIn).Date &&

DateTime.Parse(checkIn).Date <= DateTime.Parse(booking.CheckOut).Date ||

DateTime.Parse(checkOut).Date <= DateTime.Parse(booking.CheckOut).Date &&

DateTime.Parse(checkOut).Date >= DateTime.Parse(booking.CheckIn).Date))

select Booking

)

.ToList();

int count = 0;

foreach (var b in booking)

{

count += b.TotalRooms;

}

a.TotalRooms -= count;

roomList.Add(a);

}

return roomList;

}

public bool RemoveRoom(int id)

{

var roomcheck = \_roomrepository.Delete(id);

if (roomcheck != null)

{

return true;

}

return false;

}

public RoomDTO UpdateRoom(int id, RoomDTO roomDTO)

{

var room = \_roomrepository.GetById(id);

if (room != null)

{

room.Price = roomDTO.Price;

room.TotalRooms = roomDTO.TotalRooms;

room.RoomType = roomDTO.RoomType;

var result = \_roomrepository.Update(room);

return roomDTO;

}

return null;

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**HotelService**

using HotelBookingApplication.Exceptions;

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models;

using HotelBookingApplication.Models.DTOs;

namespace HotelBookingApplication.Services

{

public class HotelService : IHotelService

{

private readonly IRepository<int, Hotel> \_hotelRepository;

private readonly IRepository<int, Room> \_roomRepository;

public HotelService(IRepository<int, Hotel> repository, IRepository<int, Room> roomRepository)

{

\_hotelRepository = repository;

\_roomRepository = roomRepository;

}

public HotelDTO AddHotel(HotelDTO hotelDTO)

{

Hotel hotel = new Hotel()

{

HotelName = hotelDTO.HotelName,

Location = hotelDTO.Location,

Address = hotelDTO.Address,

UserId = hotelDTO.UserId,

PhoneNumber = hotelDTO.PhoneNumber,

Description = hotelDTO.Description,

};

var result = \_hotelRepository.Add(hotel);

if (result != null)

{

return hotelDTO;

}

return null;

}

public List<Hotel> GetHotels(string city)

{

var hotels = \_hotelRepository.GetAll().Where(c => c.Address.Contains(city, StringComparison.OrdinalIgnoreCase)).ToList();

foreach (var a in hotels)

{

int id = a.HotelId;

if (\_roomRepository.GetAll().Where(r => r.HotelId == id).ToList().Count != 0)

{

float price = (from Room in \_roomRepository.GetAll()

where Room.HotelId == id

select (Room.Price))

.Min();

}

}

if (hotels != null)

{

return hotels;

}

throw new NoHotelsAvailableException();

}

public bool RemoveHotel(int id)

{

var result = \_hotelRepository.Delete(id);

if (result != null)

{

return true;

}

return false;

}

public HotelDTO UpdateHotel(int id, HotelDTO hotelDTO)

{

var hotel = \_hotelRepository.GetById(id);

if (hotel != null)

{

hotel.PhoneNumber = hotelDTO.PhoneNumber;

hotel.Address = hotelDTO.Address;

hotel.HotelName = hotelDTO.HotelName;

hotel.Location = hotelDTO.Location;

hotel.Description = hotelDTO.Description;

var result = \_hotelRepository.Update(hotel);

return hotelDTO;

}

return null;

}

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**BookingService**

using HotelBookingApplication.Exceptions;

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models;

using HotelBookingApplication.Models.DTOs;

using HotelBookingApplication.Repositories;

using System.Net.Mail;

using System.Net;

namespace HotelBookingApplication.Services

{

public class BookingService : IBookingService

{

private readonly IRepository<int, Booking> \_bookingRepository;

private readonly IRepository<int, Room> \_roomRepository;

private readonly IRepository<int, Hotel> \_hotelRepository;

private readonly IRepository<string, User> \_userRepository;

public BookingService(IRepository<int, Booking> bookingRepository, IRepository<int, Room> roomRepository, IRepository<int, Hotel> hotelRepository, IRepository<string, User> userRepository)

{

\_bookingRepository = bookingRepository;

\_roomRepository = roomRepository;

\_hotelRepository = hotelRepository;

\_userRepository = userRepository;

}

public BookingDTO AddBookingDetails(BookingDTO bookingDTO)

{

int roomId = bookingDTO.RoomId;

var room = \_roomRepository.GetById(roomId);

var hotel = \_hotelRepository.GetById(room.HotelId);

float amount = (bookingDTO.TotalRooms \* room.Price);

DateTime dateTime = DateTime.Now;

Booking booking = new Booking()

{

UserId = bookingDTO.UserId,

CheckIn = bookingDTO.CheckIn,

CheckOut = bookingDTO.CheckOut,

RoomId = bookingDTO.RoomId,

TotalRooms = bookingDTO.TotalRooms,

Date = dateTime.ToString(),

Price = amount

};

var result = \_bookingRepository.Add(booking);

var user = \_userRepository.GetById(bookingDTO.UserId);

string message = $"Dear {user.Name},\nThank you for choosing {hotel.HotelName}! Your reservation is confirmed, Your booking reference number is {result.BookingId}. \nSafe travels!\nBest regards,\nThe {hotel.HotelName} Team\n{hotel.PhoneNumber}";

string subject = $"Booking Confirmation - {hotel.HotelName}";

string body = $"Dear {user.Name},\nThank you for choosing {hotel.HotelName}! Your reservation is confirmed, \nBooking Details:-\nBooking ID: {result.BookingId}\nCheck-In Date: {result.CheckIn}\nCheck-Out Date: {result.CheckOut}\nRoom Type: {room.RoomType}\nTotal Price: {amount}\n\nWe look forward to making your stay at {hotel.HotelName} a memorable experience. Safe travels!\nBest regards,\nThe {hotel.HotelName} Team\n{hotel.PhoneNumber}";

if (result != null)

{

return bookingDTO;

}

return null;

}

public List<Booking> GetBooking(int hotelId)

{

var bookings = (from Booking in \_bookingRepository.GetAll()

join room in \_roomRepository.GetAll() on Booking.RoomId equals room.RoomId

where room.HotelId == hotelId

select new Booking

{

BookingId = Booking.BookingId,

Date = Booking.Date,

CheckIn = Booking.CheckIn,

CheckOut = Booking.CheckOut,

RoomId = Booking.RoomId,

TotalRooms = Booking.TotalRooms,

Price = Booking.Price,

UserId = Booking.UserId

})

.ToList();

if (bookings.Count > 0)

{

return bookings;

}

return null;

}

public List<Booking> GetUserBooking(string userId)

{

var user = \_bookingRepository.GetAll().Where(u => u.UserId == userId).ToList();

if (user != null)

{

return user;

}

throw new NoBookingsAvailableException();

}

public Booking UpdateBookingStatus(int bookingId, string status)

{

var booking = \_bookingRepository.GetById(bookingId);

if (booking != null)

{

var result = \_bookingRepository.Update(booking);

return booking;

}

return null;

}

}

}

**JSON**

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"ConnectionStrings": {

"Conn": "Data source=DESKTOP-T3F1D04\\DEMOINSTANCE;user id=sa;password=Kavi191201@;Initial catalog=HotelBooking"

},

"SecretKey": "this is my secret key",

"AllowedHosts": "\*"

}

**PROGRAM.CS**

using HotelBookingApplication.Contexts;

using HotelBookingApplication.Interfaces;

using HotelBookingApplication.Models;

using HotelBookingApplication.Repositories;

using HotelBookingApplication.Services;

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.EntityFrameworkCore;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

namespace HotelBookingApplication

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen(opt =>

{

opt.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

Name = "Authorization",

Type = SecuritySchemeType.Http,

Scheme = "Bearer",

BearerFormat = "JWT",

In = ParameterLocation.Header,

Description = "JWT Authorization header using the Bearer scheme."

});

opt.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type = ReferenceType.SecurityScheme,

Id = "Bearer"

}

},

new string[] {}

}

});

});

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = false,

ValidateAudience = false,

IssuerSigningKey = new SymmetricSecurityKey(Encoding.UTF8.GetBytes(builder.Configuration["SecretKey"])),

ValidateIssuerSigningKey = true

};

});

builder.Services.AddDbContext<BookingContext>(opts =>

{

opts.UseSqlServer(builder.Configuration.GetConnectionString("bookingCon"));

});

//builder.Logging.AddLog4Net();

var app = builder.Build();

builder.Services.AddScoped<IRepository<string, User>, UserRepository>();

builder.Services.AddScoped<IRepository<int, Hotel>, HotelRepository>();

builder.Services.AddScoped<IRepository<int, Room>, RoomRepository>();

builder.Services.AddScoped<IRepository<int, RoomFacility>, RoomFacilityRepository>();

builder.Services.AddScoped<IRepository<int, Booking>, BookingRepository>();

builder.Services.AddScoped<IUserService, UserService>();

builder.Services.AddScoped<ITokenService, TokenService>();

builder.Services.AddScoped<IRoomService, RoomService>();

builder.Services.AddScoped<IHotelService, HotelService>();

builder.Services.AddScoped<IBookingService, BookingService>();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseAuthorization();

var summaries = new[]

{

"Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching"

};

app.MapGet("/weatherforecast", (HttpContext httpContext) =>

{

var forecast = Enumerable.Range(1, 5).Select(index =>

new WeatherForecast

{

Date = DateTime.Now.AddDays(index),

TemperatureC = Random.Shared.Next(-20, 55),

Summary = summaries[Random.Shared.Next(summaries.Length)]

})

.ToArray();

return forecast;

})

.WithName("GetWeatherForecast");

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

}

}

}

**OUTPUT**

