Caselet: Building an API Core Project for a Hotel Booking System

**Client: XYZ Hotels**

**Overview:**XYZ Hotels is a renowned hotel chain operating globally. To streamline their reservation process and provide an efficient booking experience to their customers, XYZ Hotels plans to develop an API Core Projectusing a code-first approach. The projectshould include CRUD operations, filteringcapabilities, count functionality, JWT token authentication, and handle the one-to-many relationship between hotels and rooms.

**Challenge:**XYZ Hotels needs a comprehensive API Core Project that enables customers to make reservations, hotel staff to manage room availability, and provides secure access through JWT token authentication. Additionally, the project should handle the one-to-many relationship between hotels and rooms, where each hotel canhave multiple rooms.

**Objectives:**

1. **CRUD Operations:**Develop APIs to supportCRUD operations for managing hotel information, including creating new hotels,updating hotel details,retrieving hotel information, and deleting hotels.
2. **Filtering:**Implement filtering capabilities to allow customers to search and filter hotelsbased on criteria such as location, price range, or amenities.
3. **Count Functionality:**Enable users to obtain counts of available rooms in specific hotels, providing insights into room availability for betterdecision-making.
4. **JWT Token Authentication**: Implementa secure authentication mechanism using JWT tokens to ensure that only authorized users can accessthe API endpoints, safeguarding customerand hotel data.
5. **One-to-Many Relationship:**Establish an efficient solutionto handle the one-to-many relationship between hotels and rooms, where each hotel can have multiple rooms.
6. **Exception Handling**: Implement try-catchblocks to handleexceptions gracefully, providingmeaningful error messagesand ensuring the system's stability.
7. **Repository Pattern:**Apply the repository patternto separate the data accesslayer from the business logic, promoting code modularity and maintainability.

**Approach:**

1. **Database**Design: Define the database schema, including tables for hotels, rooms, and the necessary foreign key relationships to handle the one-to-many relationship.
2. **Code-First Development:**Employ a code-first approach to generate the database schema based on the defined modelsand relationships, utilizing frameworks like EntityFramework.
3. **Repository Implementation**: Implement repositories for hotels and rooms, following the repository pattern to handle data access operations and encapsulate data persistence logic.
4. **CRUD Operations**: Implement APIs to support creating, reading, updating, and deleting hotel information, allowing hotelstaff to manage hotel detailseffectively.
5. **Filtering:**Develop APIs that enable customers to search and filter hotels based on specified criteria, such as location, price range, or amenities, to facilitate the booking process.
6. **Count Functionality:**Implement APIs to provide counts of available rooms in specific hotels, allowing users to retrieve the number of available rooms in each hotel for accurate room availability information.
7. **JWT Token Authentication:**Integrate JWT token authentication to secure the API endpoints, ensuring that only authenticated users, such as registered customers and authorized hotel staff, can access the protected resources.
8. **Exception Handling:**Implement try-catch blocks to handle exceptions gracefully, capturing specific exceptions, logging the errors, and providing meaningful error responses to the API consumers.
9. **One-to-Many Relationship:**Design and implement the necessarydatabase mappings and APIs to handle the one-to-many relationship between hotels and rooms, allowing each hotel to have multiple rooms and facilitating efficient room management.

By successfully implementing this API Core Project, XYZ Hotels will have an advanced reservation system that allowscustomers to searchand book hotelseasily, hotel staff to manageroom availability effectively, and provides secureaccess through JWT token authentication. The project will handle the one-to-many relationship between hotels and rooms seamlessly, ensure accurate room availability information, and maintain code modularity and stability through the

Models

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace XYZHotels.Models

{

public class Booking

{

[Key] public int BookingId { get; set; }

public int RoomNo { get; set; }

[ForeignKey("RoomNo")]

public Room? Room { get; set; }

//public int Id { get; set; }

//[ForeignKey("Id")]

//public Hotel? Hotel { get; set; }

public DateTime BookingDateTime { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

namespace XYZHotels.Models

{

public class Hotel

{

public int Id { get; set; }

[Required(ErrorMessage = "Hotel name is manditory")]

public string Name { get; set; }

[Required(ErrorMessage = "Address name is manditory")]

public string Location { get; set; }

[Required(ErrorMessage = "Hotel phone number is manditory")]

public string Phone { get; set; }

public ICollection<Room>? Rooms { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace XYZHotels.Models

{

public class Room

{

[Key] public int RoomNo { get; set; }

public string Details { get; set; }

[Required(ErrorMessage = "price is mandatory")]

public double price { get; set; }

public int Id { get; set; }

[ForeignKey("Id")]

public Hotel? Hotel { get; set; }

}

}

using System.ComponentModel.DataAnnotations;

namespace XYZHotels.Models

{

public class User

{

[Key]

public string Username { get; set; }

public byte[] Password { get; set; }

public byte[] Key { get; set; }

public string? Role { get; set; }

}

}

namespace XYZHotels.Models.DTOs

{

public class BookingCheckAvalibilityDTO

{

public int RoomNo { get; set; }

public DateTime BookingDateTime { get; set; }

}

}

namespace XYZHotels.Models.DTOs

{

public class HotelLocationDTO

{

public int Id { get; set; }

public string Location { get; set; }

}

}

namespace XYZHotels.Models.DTOs

{

public class HotelPhoneDTO

{

public int Id { get; set; }

public string Phone { get; set; }

}

}

namespace XYZHotels.Models.DTOs

{

public class RoomPriceDTO

{

public int RoomNo { get; set; }

public double price { get; set; }

}

}

namespace XYZHotels.Models.DTOs

{

public class UserDTO

{

public string? Username { get; set; }

public string? Password { get; set; }

public string? Token { get; set; }

public string? Role { get; set; }

}

}

Interfaces

using XYZHotels.Models;

using XYZHotels.Models.DTOs;

namespace XYZHotels.Interfaces

{

public interface IBookingService

{

Booking AddBooking(Booking booking);

public bool CheckAvailability(BookingCheckAvalibilityDTO booking);

public Booking CancelBooking(int id);

public IList<Booking> GetAll();

}

}

using XYZHotels.Models;

using XYZHotels.Models.DTOs;

namespace XYZHotels.Interfaces

{

public interface IHotelService

{

List<Hotel> GetAllHotel();

Hotel AddNewHotel(Hotel hotel);

Hotel Delete(int id);

Hotel UpdateHotelLocation(HotelLocationDTO hotel);

Hotel UpdateHotelPhone(HotelPhoneDTO hotel);

}

}

namespace XYZHotels.Interfaces

{

public interface IRepository<K ,T>

{

public List<T> GetAll();

public T Get(K key);

public T Add(T item);

public T Delete(K key);

public T Update(T item);

}

}

using XYZHotels.Models;

using XYZHotels.Models.DTOs;

namespace XYZHotels.Interfaces

{

public interface IRoomService

{

List<Room> GetAllRooms();

List<Room> GetInPriceRange(float min , float max);

Room AddNewRoom(Room room);

Room UpdatePrice(RoomPriceDTO room);

}

}

namespace XYZHotels.Interfaces

{

public interface ITokenServices

{

public string GenerateToken(string username, string role);

}

}

using XYZHotels.Models.DTOs;

namespace XYZHotels.Interfaces

{

public interface IUserServices

{

public UserDTO Login(UserDTO userDTO);

public UserDTO Register(UserDTO userDTO);

}

}

Context

using Microsoft.EntityFrameworkCore;

using System.Numerics;

using XYZHotels.Models;

namespace XYZHotels.Context

{

public class HContext : DbContext

{

public HContext(DbContextOptions opts) : base(opts)

{

}

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

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

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

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

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Hotel>().HasData(

new Hotel

{

Id = 1,

Name = "TAJ",

Location="Mumbai",

Phone = "9955668855",

},

new Hotel

{

Id = 2,

Name = "IBIS",

Location = "Mumbai",

Phone = "256854658",

}

);

modelBuilder.Entity<Room>().HasData(

new Room

{

RoomNo = 101,

Details="AC Room",

price=2000,

Id=1,

},

new Room

{

RoomNo = 102,

Details = "Non AC Room",

price=1500,

Id = 2,

}

);

}

}

}

Repository

using XYZHotels.Context;

using XYZHotels.Interfaces;

using XYZHotels.Models;

namespace XYZHotels.Repositories

{

public class BookingRepository : IRepository<int, Booking>

{

private readonly HContext \_context;

public BookingRepository(HContext context)

{

\_context = context;

}

public Booking Add(Booking item)

{

\_context.bookings.Add(item);

\_context.SaveChanges();

return item;

}

public Booking Delete(int key)

{

var book = Get(key);

if (book != null)

{

\_context.bookings.Remove(book);

\_context.SaveChanges();

return book;

}

return null;

}

public Booking Get(int key)

{

var book = \_context.bookings.FirstOrDefault(x=>x.BookingId == key);

return book;

}

public List<Booking> GetAll()

{

return \_context.bookings.ToList();

}

public Booking Update(Booking item)

{

\_context.Entry<Booking>(item).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

\_context.SaveChanges();

return item;

}

}

}

using XYZHotels.Context;

using XYZHotels.Interfaces;

using XYZHotels.Models;

namespace XYZHotels.Repositories

{

public class HotelRepository : IRepository<int, Hotel>

{

private readonly HContext \_context;

public HotelRepository(HContext context)

{

\_context = context;

}

public Hotel Add(Hotel item)

{

\_context.hotel.Add(item);

\_context.SaveChanges();

return item;

}

public Hotel Delete(int key)

{

var hotel = Get(key);

if (hotel != null)

{

\_context.hotel.Remove(hotel);

\_context.SaveChanges();

return hotel;

}return null;

}

public Hotel Get(int key)

{

var Hotel = \_context.hotel.FirstOrDefault(x => x.Id == key);

return Hotel;

}

public List<Hotel> GetAll()

{

return \_context.hotel.ToList();

}

public Hotel Update(Hotel item)

{

\_context.Entry<Hotel>(item).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

\_context.SaveChanges();

return item;

}

}

}

using XYZHotels.Context;

using XYZHotels.Interfaces;

using XYZHotels.Models;

namespace XYZHotels.Repositories

{

public class RoomRepository : IRepository<int, Room>

{

private readonly HContext \_context;

public RoomRepository(HContext context)

{

\_context=context;

}

public Room Add(Room item)

{

\_context.rooms.Add(item);

\_context.SaveChanges();

return item;

}

public Room Delete(int key)

{

var room = Get(key);

if (room == null)

{

\_context.rooms.Remove(room);

\_context.SaveChanges();

return room;

}return null;

}

public Room Get(int key)

{

var room = \_context.rooms.FirstOrDefault(x=>x.RoomNo == key);

return room;

}

public List<Room> GetAll()

{

return \_context.rooms.ToList();

}

public Room Update(Room item)

{

\_context.Entry<Room>(item).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

\_context.SaveChanges();

return item;

}

}

}

using XYZHotels.Context;

using XYZHotels.Interfaces;

using XYZHotels.Models;

namespace XYZHotels.Repositories

{

public class UserRepository : IRepository<string , User>

{

private readonly HContext \_context;

public UserRepository(HContext context)

{

\_context = context;

}

public User Add(User item)

{

\_context.users.Add(item);

\_context.SaveChanges();

return item;

}

public User Delete(string key)

{

var user = Get(key);

if (user != null)

{

\_context.users.Remove(user);

\_context.SaveChanges();

}

return null;

}

public User Get(string key)

{

var user = \_context.users.FirstOrDefault(u => u.Username == key);

return user;

}

public User Get(int key)

{

throw new NotImplementedException();

}

public List<User> GetAll()

{

return \_context.users.ToList();

}

public User Update(User item)

{

\_context.Entry<User>(item).State = Microsoft.EntityFrameworkCore.EntityState.Modified;

\_context.SaveChanges();

return item;

}

}

}

Services

using XYZHotels.ExceptionHandle;

using XYZHotels.Interfaces;

using XYZHotels.Models;

using XYZHotels.Models.DTOs;

namespace XYZHotels.Services

{

public class BookingService : IBookingService

{

private readonly IRepository<int, Booking> \_repo;

public BookingService(IRepository <int , Booking> repository)

{

\_repo=repository;

}

public Booking AddBooking(Booking booking)

{

BookingCheckAvalibilityDTO bca= new BookingCheckAvalibilityDTO

{

BookingDateTime = booking.BookingDateTime,

RoomNo = booking.RoomNo

};

if(CheckAvailability(bca) == true)

{

var book = \_repo.Add(booking);

return book;

}

throw new RoomNotAvailableExceptions();

}

public Booking CancelBooking(int id)

{

var mybook = \_repo.Get(id);

if(mybook == null)

{

throw new NoEntriesAvailable("Rooms");

}

mybook =\_repo.Delete(mybook.BookingId);

return mybook;

}

public bool CheckAvailability(BookingCheckAvalibilityDTO booking)

{

try

{

var book = \_repo.GetAll();

var checkBooking = book.FirstOrDefault(x=>x.RoomNo== booking.RoomNo && x.BookingDateTime==booking.BookingDateTime);

return checkBooking == null;

}

catch (RoomNotAvailableExceptions e )

{

return true;

}

catch(NoEntriesAvailable e)

{

return true;

}

}

public IList<Booking> GetAll()

{

return \_repo.GetAll().ToList();

}

}

}

using XYZHotels.ExceptionHandle;

using XYZHotels.Interfaces;

using XYZHotels.Models;

using XYZHotels.Models.DTOs;

namespace XYZHotels.Services

{

public class HotelService : IHotelService

{

private readonly IRepository<int, Hotel> \_repo;

public HotelService(IRepository <int , Hotel> repository)

{

\_repo=repository;

}

public Hotel AddNewHotel(Hotel hotel)

{

return \_repo.Add(hotel);

}

public List<Hotel> GetAllHotel()

{

return \_repo.GetAll();

}

public Hotel UpdateHotelLocation(HotelLocationDTO hotel)

{

var myHotel =\_repo.Get(hotel.Id);

if(myHotel != null)

{

myHotel.Location= hotel.Location;

return \_repo.Update(myHotel);

}return null;

}

public Hotel UpdateHotelPhone(HotelPhoneDTO hotel)

{

var myHotel = \_repo.Get(hotel.Id);

if (myHotel != null)

{

myHotel.Phone = hotel.Phone;

return \_repo.Update(myHotel);

}

return null;

}

public Hotel Delete(int id)

{

var hotel = \_repo.Get(id);

{

if(hotel == null)

{

throw new NoEntriesAvailable("Hotel");

}

return \_repo.Delete(hotel.Id);

}

}

}

}

using XYZHotels.Interfaces;

using XYZHotels.Models;

using XYZHotels.Models.DTOs;

namespace XYZHotels.Services

{

public class RoomService : IRoomService

{

private readonly IRepository<int, Room> \_repo;

public RoomService(IRepository <int ,Room> repository)

{

\_repo=repository;

}

public Room AddNewRoom(Room room)

{

return \_repo.Add(room);

}

public List<Room> GetAllRooms()

{

return \_repo.GetAll();

}

public List<Room> GetInPriceRange(float min, float max)

{

var room =\_repo.GetAll().Where(x=>x.price>=min && x.price<=max).ToList();

if(room.Count()>0)

{

return room.ToList();

}

return null;

}

public Room UpdatePrice(RoomPriceDTO room)

{

var rooms = \_repo.Get(room.RoomNo);

if(rooms!= null)

{

rooms.price = room.price;

return \_repo.Update(rooms);

}return null;

}

}

}

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using XYZHotels.Interfaces;

namespace XYZHotels.Services

{

public class TokenServices : ITokenServices

{

byte[] key;

public TokenServices(IConfiguration configuration)

{

key = Encoding.UTF8.GetBytes(configuration.GetValue(typeof(string), "TokenKey").ToString());

}

public string GenerateToken(string username, string role)

{

string token = string.Empty;

//Username for the token - from the parameter

//Role is the role from parameter

var subject = new ClaimsIdentity(new[]

{

new Claim(ClaimTypes.Name, username),

new Claim(ClaimTypes.Role,role)

});

var tokenDescription = new SecurityTokenDescriptor();

//Describing the token

tokenDescription.Subject = subject;

tokenDescription.Expires = DateTime.UtcNow.AddDays(1);

var signature = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature);

tokenDescription.SigningCredentials = signature;

var tokenHandler = new JwtSecurityTokenHandler();

//Generating the token object

var tokenObject = tokenHandler.CreateToken(tokenDescription);

//Getting the t oken as string from the object

token = tokenHandler.WriteToken(tokenObject);

return token;

}

}

}

using System.Security.Cryptography;

using System.Text;

using XYZHotels.Interfaces;

using XYZHotels.Models.DTOs;

using XYZHotels.Models;

namespace XYZHotels.Services

{

public class UserServices : IUserServices

{

private readonly IRepository<string, User> \_userRepository;

private readonly ITokenServices \_tokenSevice;

public UserServices(IRepository<string, User> repository, ITokenServices tokenService)

{

\_userRepository = repository;

\_tokenSevice = tokenService;

}

public UserDTO Login(UserDTO userDTO)

{

var user = \_userRepository.Get(userDTO.Username);

if (user != null)

{

var dbPass = user.Password;

HMACSHA512 hMACSHA512 = new HMACSHA512(user.Key);

var userPass = hMACSHA512.ComputeHash(Encoding.UTF8.GetBytes(userDTO.Password));

if (userPass.Length == dbPass.Length)

{

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

{

if (userPass[i] != dbPass[i])

return null;

}

var loggedinUser = new UserDTO

{

Username = user.Username,

Token = \_tokenSevice.GenerateToken(user.Username, user.Role)

};

return loggedinUser;

}

}

return null;

}

public UserDTO Register(UserDTO userDTO)

{

HMACSHA512 hMACSHA512 = new HMACSHA512();

User user = new User();

user.Username = userDTO.Username;

user.Password = hMACSHA512.ComputeHash(Encoding.UTF8.GetBytes(userDTO.Password));

user.Role = userDTO.Role;

user.Key = hMACSHA512.Key;

\_userRepository.Add(user);

var regiteredUser = new UserDTO

{

Username = user.Username,

Token = \_tokenSevice.GenerateToken(user.Username, user.Role)

};

return regiteredUser;

}

}

}

Program

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.EntityFrameworkCore;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

using XYZHotels.Context;

using XYZHotels.Interfaces;

using XYZHotels.Models;

using XYZHotels.Repositories;

using XYZHotels.Services;

namespace XYZHotels

{

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();

#region Swagger Generation

builder.Services.AddSwaggerGen(option =>

{

option.SwaggerDoc("v1", new OpenApiInfo { Title = "Demo API", Version = "v1" });

option.AddSecurityDefinition("Bearer", new OpenApiSecurityScheme

{

In = ParameterLocation.Header,

Description = "Please enter a valid token",

Name = "Authorization",

Type = SecuritySchemeType.Http,

BearerFormat = "JWT",

Scheme = "Bearer"

});

option.AddSecurityRequirement(new OpenApiSecurityRequirement

{

{

new OpenApiSecurityScheme

{

Reference = new OpenApiReference

{

Type=ReferenceType.SecurityScheme,

Id="Bearer"

}

},

new string[]{}

}

});

});

var stringkey = builder.Configuration.GetValue(typeof(string), "TokenKey").ToString();

var key = Encoding.UTF8.GetBytes(stringkey);

builder.Services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(opts =>

{

opts.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = false,

ValidateAudience = false,

ValidateLifetime = true,

ValidateIssuerSigningKey = true,

IssuerSigningKey = new SymmetricSecurityKey(key)

};

});

#endregion

#region Context

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

{

opts.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection"));

});

#endregion

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 , Booking>, BookingRepository>();

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

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

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

builder.Services.AddScoped<ITokenServices , TokenServices>();

builder.Services.AddScoped<IUserServices, UserServices>();

var app = builder.Build();

AppContext.SetSwitch("Npgsql.EnableLegacyTimestampBehavior", true); /// For DateTime

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

}

}

}

Controller

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using XYZHotels.Interfaces;

using XYZHotels.Models;

namespace XYZHotels.Controllers

{

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

[ApiController]

public class BookingController : ControllerBase

{

private readonly IBookingService \_service;

public BookingController(IBookingService service)

{

\_service= service;

}

[HttpGet("Get")]

public ActionResult Get()

{

var result = \_service.GetAll();

if (result == null)

{

return NotFound("No Booking Available");

}

return Ok(result);

}

[HttpPost("AdBooking")]

public ActionResult AdBooking(Booking book)

{

if(ModelState.IsValid)

{

try

{

var result = \_service.AddBooking(book);

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

return BadRequest(ModelState.Keys);

}

[HttpDelete("CanclBooking")]

public ActionResult CanclBooking(int id)

{

try

{

var result = \_service.CancelBooking(id);

if (result == null)

return NotFound();

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

}

}

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using XYZHotels.Interfaces;

using XYZHotels.Models;

using XYZHotels.Models.DTOs;

namespace XYZHotels.Controllers

{

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

[ApiController]

public class HotelController : ControllerBase

{

private readonly IHotelService \_service;

public HotelController(IHotelService hotelService)

{

\_service = hotelService;

}

[HttpGet]

public ActionResult Get() {

var result = \_service.GetAllHotel();

if (result == null)

{

return NotFound("Hotels Not Found");

}

return Ok(result);

}

[Authorize(Roles = "Owner")]

[HttpPost]

public ActionResult AddHotels(Hotel hotel)

{

if(ModelState.IsValid)

{

try

{

var result = \_service.AddNewHotel(hotel);

return Created("", result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

return BadRequest(ModelState.Keys);

}

[Authorize(Roles = "Owner")]

[HttpPut("UpdateLocation")]

public ActionResult UpdateLocation(HotelLocationDTO hotel)

{

try

{

var result = \_service.UpdateHotelLocation(hotel);

if(result == null)

{

return NotFound();

}

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

[Authorize(Roles = "Owner")]

[HttpPut("UpdatePhone")]

public ActionResult UpdatePhone(HotelPhoneDTO hotelPhone)

{

try

{

var result = \_service.UpdateHotelPhone(hotelPhone);

if(result == null)

{

return NotFound();

}

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

[Authorize(Roles = "Owner")]

[HttpDelete]

public ActionResult Delete(int id)

{

try

{

var result = \_service.Delete(id);

if (result == null)

return NotFound();

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

}

}

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using XYZHotels.Interfaces;

using XYZHotels.Models;

using XYZHotels.Models.DTOs;

namespace XYZHotels.Controllers

{

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

[ApiController]

public class RoomController : ControllerBase

{

private readonly IRoomService \_service;

public RoomController(IRoomService service)

{

\_service = service;

}

[Authorize(Roles = "Manager")]

[HttpGet("Get")]

public ActionResult Get()

{

var result = \_service.GetAllRooms();

if(result == null)

{

return NotFound("No Rooms At the moment");

}

return Ok(result);

}

[Authorize(Roles = "Manager")]

[HttpPost("Addroom")]

public ActionResult Addroom(Room room)

{

if(ModelState.IsValid)

{

try

{

var result = \_service.AddNewRoom(room);

return Created("", result);

}

catch (Exception e )

{

return BadRequest(e.Message);

}

}

return BadRequest(ModelState.Keys);

}

[HttpGet("GetRangePrice")]

public ActionResult GetRangePrice(float min , float max) {

var result = \_service.GetInPriceRange(min, max);

if(result == null)

{

return NotFound("No Rooms in this Price Range");

}

return Ok(result);

}

[Authorize(Roles = "Manager")]

[HttpPut("UpdatePrice")]

public ActionResult UpdatePrice(RoomPriceDTO room)

{

try

{

var result = \_service.UpdatePrice(room);

if (result == null)

return NotFound();

return Ok(result);

}

catch (Exception e)

{

return BadRequest(e.Message);

}

}

}

}

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using XYZHotels.Interfaces;

using XYZHotels.Models.DTOs;

namespace XYZHotels.Controllers

{

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

[ApiController]

public class UserController : ControllerBase

{

private readonly IUserServices \_service;

public UserController(IUserServices service)

{

\_service = service;

}

[HttpPost("Login")]

public ActionResult Login(UserDTO userDTO)

{

var result = \_service.Login(userDTO);

if (result == null)

{

return Unauthorized();

}

return Ok(result);

}

[HttpPost("Register")]

public ActionResult Register(UserDTO userDTO)

{

var result = \_service.Register(userDTO);

if (result == null)

{

return BadRequest();

}

return Ok(result);

}

}

}

using Microsoft.AspNetCore.Mvc;

namespace XYZHotels.Controllers

{

[ApiController]

[Route("[controller]")]

public class WeatherForecastController : ControllerBase

{

private static readonly string[] Summaries = new[]

{

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

};

private readonly ILogger<WeatherForecastController> \_logger;

public WeatherForecastController(ILogger<WeatherForecastController> logger)

{

\_logger = logger;

}

[HttpGet(Name = "GetWeatherForecast")]

public IEnumerable<WeatherForecast> Get()

{

return 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();

}

}

}