XYZHotelsDBContext.cs

using Microsoft.EntityFrameworkCore;

using XYZHotelsApp.Models;

namespace XYZHotelsApp.Contexts

{

public class XYZHotelsDBContext : DbContext

{

public XYZHotelsDBContext(DbContextOptions options) : base(options)

{

}

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

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

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

public DbSet<Reservation> Reservations { get; set; }

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Hotel>()

.HasMany(h => h.Rooms)

.WithOne(r => r.Hotel)

.HasForeignKey(r => r.HotelId)

.OnDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<Hotel>()

.Property(h => h.PricePerNight)

.HasPrecision(18, 2);

modelBuilder.Entity<Room>()

.HasMany(r => r.Reservations)

.WithOne(res => res.Room)

.HasForeignKey(res => res.RoomId)

.OnDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<Reservation>()

.HasOne(r => r.Users)

.WithMany(u => u.Reservations)

.HasForeignKey(r => r.Username)

.OnDelete(DeleteBehavior.Cascade);

base.OnModelCreating(modelBuilder);

}

}

}

HotelController.cs

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using XYZHotelsApp.Interfaces;

using System.Net.Mime;

using XYZHotelsApp.Models;

using XYZHotelsApp.Exceptions;

namespace XYZHotelsApp.Controllers

{

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

[ApiController]

public class HotelController : ControllerBase

{

private readonly IHotelService \_hotelService;

public HotelController(IHotelService hotelService)

{

\_hotelService = hotelService;

}

[HttpGet]

public ActionResult Get()

{

string errorMessage = string.Empty;

try

{

var result = \_hotelService.GetHotels();

return Ok(result);

}

catch (NoHotelsAvailableException e)

{

errorMessage = e.Message;

}

return BadRequest(errorMessage);

}

[Authorize(Roles = "Admin")]

[HttpPost("Add")]

public ActionResult Create(Hotel hotel)

{

string errorMessage = string.Empty;

try

{

var result = \_hotelService.AddHotel(hotel);

return Ok(result);

}

catch (Exception e)

{

errorMessage = e.Message;

}

return BadRequest(errorMessage);

}

[HttpPost("Remove")]

public ActionResult Remove(Hotel hotel)

{

string errorMessage = string.Empty;

try

{

var result = \_hotelService.RemoveHotel(hotel);

return Ok(result);

}

catch (Exception e)

{

errorMessage = e.Message;

}

return BadRequest(errorMessage);

}

}

}

ReservationController.cs

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models;

using XYZHotelsApp.Services;

namespace XYZHotelsApp.Controllers

{

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

[ApiController]

public class ReservationController : ControllerBase

{

private readonly IReservationService \_reservationService;

public ReservationController(IReservationService reservationService)

{

\_reservationService = reservationService;

}

[HttpPost("Booking")]

public ActionResult BookRoom(Reservation reservation)

{

string errorMessage = string.Empty;

try

{

var result = \_reservationService.BookRoom(reservation);

return Ok(result);

}

catch (Exception e)

{

errorMessage = e.Message;

}

return BadRequest(errorMessage);

}

}

}

RoomController.cs

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using XYZHotelsApp.Exceptions;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models;

using XYZHotelsApp.Services;

namespace XYZHotelsApp.Controllers

{

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

[ApiController]

public class RoomController : ControllerBase

{

private readonly IRoomService \_roomService;

public RoomController(IRoomService roomService)

{

\_roomService = roomService;

}

[HttpGet]

public ActionResult GetRooms()

{

string errorMessage = string.Empty;

try

{

var result = \_roomService.GetAllRooms();

return Ok(result);

}

catch (NoRoomsAvailableException e)

{

errorMessage = e.Message;

}

return BadRequest(errorMessage);

}

[HttpPost("AddRoom")]

public ActionResult Add(Room room)

{

string errorMessage = string.Empty;

try

{

var result = \_roomService.AddRoom(room);

return Ok(result);

}

catch (Exception e)

{

errorMessage = e.Message;

}

return BadRequest(errorMessage);

}

}

}

UserController.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models.DTOs;

using XYZHotelsApp.Services;

namespace XYZHotelsApp.Controllers

{

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

[ApiController]

public class UserController : ControllerBase

{

private readonly IUserService \_userService;

public UserController(IUserService userService)

{

\_userService = userService;

}

[HttpPost("register")]

public ActionResult Register(UserDTO viewModel)

{

string message = "";

try

{

var user = \_userService.Register(viewModel);

if (user != null)

{

return Ok(user);

}

}

catch (DbUpdateException exp)

{

message = "Duplicate username";

}

catch (Exception)

{

}

return BadRequest(message);

}

[HttpPost("login")]

public ActionResult Login(UserDTO userDTO)

{

var user = \_userService.Login(userDTO);

if (user != null)

{

return Ok(user);

}

else

{

return Unauthorized("Username and Password Mismatch");

}

}

}

}

IHotelService.cs

using XYZHotelsApp.Models;

namespace XYZHotelsApp.Interfaces

{

public interface IHotelService

{

List<Hotel> GetHotels();

Hotel AddHotel(Hotel hotel);

Hotel RemoveHotel(Hotel hotel);

}

}

IRepository.cs

namespace XYZHotelsApp.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);

}

}

IReservationService.cs

using XYZHotelsApp.Models;

namespace XYZHotelsApp.Interfaces

{

public interface IReservationService

{

Reservation BookRoom(Reservation reservation);

}

}

IRoomService.cs

using XYZHotelsApp.Models;

namespace XYZHotelsApp.Interfaces

{

public interface IRoomService

{

Room AddRoom(Room room);

List<Room> GetAllRooms();

}

}

ITokenService.cs

using XYZHotelsApp.Models.DTOs;

namespace XYZHotelsApp.Interfaces

{

public interface ITokenService

{

string GetToken(UserDTO user);

}

}

IuserService.cs

using XYZHotelsApp.Models.DTOs;

namespace XYZHotelsApp.Interfaces

{

public interface IUserService

{

UserDTO Login(UserDTO userDTO);

UserDTO Register(UserDTO userDTO);

}

}

Hotel.cs(Model)

using System.Text.Json.Serialization;

namespace XYZHotelsApp.Models

{

public class Hotel

{

public int HotelId { get; set; }

public string Name { get; set; }

public string Location { get; set; }

public decimal PricePerNight { get; set; }

[JsonIgnore]

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

}

}

Reservation.cs(Model)

using System.ComponentModel.DataAnnotations.Schema;

using System.Text.Json.Serialization;

namespace XYZHotelsApp.Models

{

public class Reservation

{

public int ReservationId { get; set; }

public int RoomId { get; set; }

public DateTime? CheckInDate { get; set; }

public DateTime? CheckOutDate { get; set; }

[JsonIgnore]

public Room? Room { get; set; }

[JsonIgnore]

public User? Users { get; set; }

[ForeignKey("Username")]

public string Username { get; set; }

}

}

Room.cs(Model)

using System.Text.Json.Serialization;

namespace XYZHotelsApp.Models

{

public class Room

{

public int RoomId { get; set; }

public int RoomNumber { get; set; }

public int HotelId { get; set; }

[JsonIgnore]

public Hotel? Hotel { get; set; }

[JsonIgnore]

public List<Reservation>? Reservations { get; set; }

public RoomStatus Status { get; set; }

public DateTime? AvailableAgainAt { get; set; }

}

public enum RoomStatus

{

Available,

Reserved

}

}

User.cs(Model)

using System.ComponentModel.DataAnnotations;

namespace XYZHotelsApp.Models

{

public class User

{

[Key]

public string Username { get; set; }

public byte[] Password { get; set; }

public string Role { get; set; }

public byte[] Key { get; set; }

public ICollection<Reservation> Reservations { get; set; }

}

}

UserDTO.cs

using System.ComponentModel.DataAnnotations;

namespace XYZHotelsApp.Models.DTOs

{

public class UserDTO

{

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

public string Username { get; set; }

public string? Role { get; set; }

public string? Token { get; set; }

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

public string Password { get; set; }

}

}

UserViewModel.cs

using System.ComponentModel.DataAnnotations;

namespace XYZHotelsApp.Models.DTOs

{

public class UserViewModel : UserDTO

{

[Required(ErrorMessage = "Re type password cannot be empty")]

[Compare("Password", ErrorMessage = "Password and retype password do not match")]

public string ReTypePassword { get; set; }

}

}

HotelRepository.cs

using XYZHotelsApp.Contexts;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models;

namespace XYZHotelsApp.Repositories

{

public class HotelRepository : IRepository<int, Hotel>

{

private readonly XYZHotelsDBContext \_context;

public HotelRepository(XYZHotelsDBContext context)

{

\_context = context;

}

public Hotel Add(Hotel hotel)

{

\_context.Hotels.Add(hotel);

\_context.SaveChanges();

return hotel;

}

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(b => b.HotelId == key);

return hotel;

}

public Hotel Update(Hotel entity)

{

throw new NotImplementedException();

}

}

}

ReservationRepository.cs

using XYZHotelsApp.Contexts;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models;

namespace XYZHotelsApp.Repositories

{

public class ReservationRepository : IRepository<int, Reservation>

{

private readonly XYZHotelsDBContext \_context;

public ReservationRepository(XYZHotelsDBContext context)

{

\_context = context;

}

public Reservation Add(Reservation reservation)

{

\_context.Add(reservation);

\_context.SaveChanges();

return reservation;

}

public Reservation Delete(int key)

{

throw new NotImplementedException();

}

public IList<Reservation> GetAll()

{

throw new NotImplementedException();

}

public Reservation GetById(int key)

{

throw new NotImplementedException();

}

public Reservation Update(Reservation entity)

{

throw new NotImplementedException();

}

}

}

RoomRepository.cs

using XYZHotelsApp.Contexts;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models;

namespace XYZHotelsApp.Repositories

{

public class RoomRepository : IRepository<int, Room>

{

private readonly XYZHotelsDBContext \_context;

public RoomRepository(XYZHotelsDBContext context)

{

\_context = context;

}

public Room Add(Room rooom)

{

\_context.Rooms.Add(rooom);

\_context.SaveChanges();

return rooom;

}

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(b => b.RoomId == key);

return room;

}

public Room Update(Room entity)

{

throw new NotImplementedException();

}

}

}

UserRepository.cs

using Microsoft.EntityFrameworkCore;

using XYZHotelsApp.Contexts;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models;

namespace XYZHotelsApp.Repositories

{

public class UserRepository : IRepository<string, User>

{

private readonly XYZHotelsDBContext \_context;

public UserRepository(XYZHotelsDBContext context)

{

\_context = context;

}

public User Add(User user)

{

\_context.Users.Add(user);

\_context.SaveChanges();

return user;

}

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.Username == key);

return user;

}

public User Update(User entity)

{

var user = GetById(entity.Username);

if (user != null)

{

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

\_context.SaveChanges();

return user;

}

return null;

}

}

}

HotelService.cs

using XYZHotelsApp.Exceptions;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models;

namespace XYZHotelsApp.Services

{

public class HotelService : IHotelService

{

private readonly IRepository<int,Hotel> \_hotelRepository;

public HotelService(IRepository<int, Hotel> hotelRepository)

{

\_hotelRepository = hotelRepository;

}

public Hotel AddHotel(Hotel hotel)

{

var result =\_hotelRepository.Add(hotel);

if(result != null)

{

return result;

}

return null;

}

public List<Hotel> GetHotels()

{

var hotels = \_hotelRepository.GetAll();

if (hotels != null)

{

return hotels.ToList();

}

throw new NoHotelsAvailableException();

}

public Hotel RemoveHotel(Hotel hotel)

{

int hotelId = 0;

var checkHotel = \_hotelRepository.GetAll().FirstOrDefault(rh=>rh.HotelId==hotel.HotelId);

if (checkHotel != null)

{

hotelId=checkHotel.HotelId;

var result = \_hotelRepository.Delete(hotelId);

return result;

}

return null;

}

}

}

ReservationService.cs

using XYZHotelsApp.Exceptions;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models;

using XYZHotelsApp.Repositories;

namespace XYZHotelsApp.Services

{

public class ReservationService : IReservationService

{

private readonly IRepository<int,Hotel> \_hotelRepository;

private readonly IRepository<int,Reservation> \_reservationRepository;

private readonly IRepository<int, Room> \_roomRepository;

public ReservationService(IRepository<int, Hotel> hotelRepository, IRepository<int, Reservation> reservationRepository, IRepository<int, Room> roomRepository)

{

\_hotelRepository = hotelRepository;

\_reservationRepository = reservationRepository;

\_roomRepository = roomRepository;

}

public bool IsRoomAvailable(Room room)

{

return room.Status == RoomStatus.Available && (room.AvailableAgainAt == null || DateTime.UtcNow >= room.AvailableAgainAt);

}

public Reservation BookRoom(Reservation reservation)

{

var checkRoomNumber = \_roomRepository.GetAll().FirstOrDefault(r => r.RoomId == reservation.RoomId);

if (checkRoomNumber != null && IsRoomAvailable(checkRoomNumber))

{

reservation.CheckInDate = DateTime.UtcNow;

reservation.CheckOutDate = DateTime.UtcNow.AddHours(24);

checkRoomNumber.Status = RoomStatus.Reserved;

checkRoomNumber.AvailableAgainAt = DateTime.UtcNow.AddHours(24);

var result = \_reservationRepository.Add(reservation);

return result;

}

throw new RoomIsBooked();

}

}

}

RoomService.cs

using System.Reflection.Metadata;

using XYZHotelsApp.Exceptions;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models;

namespace XYZHotelsApp.Services

{

public class RoomService : IRoomService

{

private readonly IRepository<int,Room> \_roomRepository;

private readonly IRepository<int,Hotel> \_hotelRepository;

public RoomService(IRepository<int, Room> roomRepository, IRepository<int, Hotel> hotelRepository)

{

\_roomRepository = roomRepository;

\_hotelRepository = hotelRepository;

}

public Room AddRoom(Room room)

{

var checkHotel = \_hotelRepository.GetAll().FirstOrDefault(h => h.HotelId == room.HotelId);

var checkRoomNumber = \_roomRepository.GetAll().FirstOrDefault(r => r.RoomNumber == room.RoomNumber);

if (checkHotel != null)

{

if (checkRoomNumber == null)

{

var result = \_roomRepository.Add(room);

if (result != null)

{

result.AvailableAgainAt = DateTime.UtcNow;

result.Status = RoomStatus.Available;

return result;

}

else

throw new RoomCouldNotBeAdded();

}

else

throw new RoomNumberAlreadyExists();

}

else

throw new NoHotelsAvailableException();

}

public List<Room> GetAllRooms()

{

var rooms = \_roomRepository.GetAll();

if (rooms != null)

{

return rooms.ToList();

}

throw new NoRoomsAvailableException();

}

}

}

TokenService.cs

using Microsoft.IdentityModel.Tokens;

using System.IdentityModel.Tokens.Jwt;

using System.Security.Claims;

using System.Text;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models.DTOs;

namespace XYZHotelsApp.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.NameId,user.Username),

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);

}

}

}

UserService.cs

using System.Security.Cryptography;

using System.Text;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models;

using XYZHotelsApp.Models.DTOs;

namespace XYZHotelsApp.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.Username);

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.Token = \_tokenService.GetToken(userDTO);

userDTO.Password = "";

return userDTO;

}

return null;

}

public UserDTO Register(UserDTO userDTO)

{

HMACSHA512 hmac = new HMACSHA512();

User user = new User()

{

Username = userDTO.Username,

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

Key = hmac.Key,

Role = userDTO.Role

};

var result = \_repository.Add(user);

if (result != null)

{

userDTO.Password = "";

return userDTO;

}

return null;

}

}

}

Programs.cs

using Microsoft.AspNetCore.Authentication.JwtBearer;

using Microsoft.EntityFrameworkCore;

using Microsoft.IdentityModel.Tokens;

using Microsoft.OpenApi.Models;

using System.Text;

using XYZHotelsApp.Contexts;

using XYZHotelsApp.Interfaces;

using XYZHotelsApp.Models;

using XYZHotelsApp.Repositories;

using XYZHotelsApp.Services;

namespace XYZHotelsApp

{

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<XYZHotelsDBContext>(opts =>

{

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

});

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

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

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

builder.Services.AddScoped<IRepository<int, Reservation>, ReservationRepository>();

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

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

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

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

builder.Services.AddScoped<IReservationService, ReservationService>();

var app = builder.Build();

// Configure the HTTP request pipeline.

if (app.Environment.IsDevelopment())

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllers();

app.Run();

}

}

}

------------------------------------------------------------------------------------------------------------------------------------------

ScreenShots:

