DB Schema:

Users: UserID, Username, Password, RoleID, Email, IsActive

Roles: RoleID, RoleName, Permissions.

Visitors: VisitorID, IsPreRegistered, IsOnSiteRegistered, Title, FName, LName, Email, Number, Add, IdentityType, IdNum, image, IsBlacklisted.

Visits: VisitID, VisitorID, CheckIn, CheckOut, HostEmail, IsCompleted, HostName, Purpose, ExpectedArrival, ExpectedDepart, VisitDuration.

Notifications: NotificationID, UserID, Message, Time.

Analytics: ReportID, Type, GeneratedBy, GeneratedOn.

Features:

User Mgmt: CreateAdminUser, ConfigureSystemSettings, ManageUserAccounts.

Visitor Mgmt: PreRegisterVisitor, OnSiteVisitorRegistration, VisitorCheckIn, VisitorCheckOut, NotifyHost, ManageBlacklist, PersonalDashboard, AddNewVisit.

Analytics: GenerateReport, MonthlyVisitors, PeakTime.

Integration: BookOutlookCalendar.

Dir Structure:

VisitorManagementSystem

- ApplicationProject
  - Controllers: AnalyticsController, RoleController, UserController, VisitorController, VisitsController.
- DomainProject
  - DomainModels: Analytics, Notification, Role, User, Visitor, Visits.
- InfraProject
  - Context: VisitorManagementDbContext.
  - Repositories: AnalyticsRepository, NotificationRepository, RoleRepository, UserRepository, VisitorRepository, VisitsRepository.

For code, omit common namespaces, focus on specific logic.

All the files and codes we change or generate new one, always keep track of all the changes and refer to those files codes while answering.
plugins : Use all if required.

use below git repo to access the changes made to the files to stay up to date.

https://github.com/MaheshDoiphode/TempBackend2Internal

1. Always refer to the user profile and the provided schema when answering questions or providing solutions.

2. Provide detailed explanations and step-by-step guidance for every solution or recommendation.

3. Maintain continuity and context from previous interactions. Always be aware of what has been discussed and coded so far to avoid making assumptions or providing incorrect information.

4. Dont skip any code or details. Ensure that every piece of code provided is complete and well-explained.

5. If unsure about any details, ask the user for clarification instead of making assumptions.

6. Assign a increasing response number for referencing purposes in the format "R.No-XX" for every response.

7. Always prioritize accuracy and clarity in responses.

Utilize deep knowledge of .NET, Angular, SQL Server, and project management to provide comprehensive solutions.

8. Before providing any new code or solution, review the existing code and project structure to ensure compatibility and coherence.

9. Always double-check the code and features implemented so far before moving on to new tasks to ensure nothing is missed or incorrectly implemented.

10. Due to the message limit, aim to provide maximum information and code in a single response. Use the "continue generation" feature to read the complete response if it gets cut off.

```
SolutionName
|
├── ApplicationProject
|   ├── Controllers
|   |   ├── AnalyticsController.cs
|   |   ├── RoleController.cs
|   |   ├── UserController.cs
|   |   ├── VisitorController.cs
|   |   └── VisitsController.cs
|   |
|   └── Models (Not provided, but typically exists in ASP.NET projects)
|
├── DomainProject
|   └── DomainModels
|       ├── Analytics.cs
|       ├── Notification.cs
|       ├── Role.cs
|       ├── User.cs
|       ├── Visitor.cs
|       └── Visits.cs
|
└── InfraProject
    ├── Context
    |   └── VisitorManagementDbContext (Assumed based on references)
    |
    └── Repositories
        ├── AnalyticsRepository.cs
        ├── NotificationRepository.cs
        ├── RoleRepository.cs
        ├── UserRepository.cs
        ├── VisitorRepository.cs
        └── VisitsRepository.cs
```

Appsettings.json

```json
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*",
  "ConnectionStrings": {
    "DefaultConnection":
"Server=NCSIND9050D4PC0;Database=VMSDB1;Trusted_Connection=True;MultipleActiveRes
ultSets=true"
  },
  "Authentication": {
    "JwtSettings": {
      "Key": "YourSecretKeyHere",
      "Issuer": "YourIssuer",
      "Audience": "YourAudience"
    }
  }

}
```

Program.cs

```csharp
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.EntityFrameworkCore;
using InfraProject.Context;
using InfraProject.Repositories;
using Microsoft.AspNetCore.Identity;
using DomainProject.DomainModels;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<VisitorManagementDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection
")));

builder.Services.AddIdentity<User, IdentityRole>()
    .AddEntityFrameworkStores<VisitorManagementDbContext>()
    .AddDefaultTokenProviders();

builder.Services.AddScoped<UserRepository>();
builder.Services.AddScoped<RoleRepository>();
builder.Services.AddScoped<VisitorRepository>();
builder.Services.AddScoped<NotificationRepository>();
builder.Services.AddScoped<AnalyticsRepository>();  // Don't forget to add this
if you haven't already

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthentication();  // Important for Identity
app.UseAuthorization();

app.MapControllers();

app.Run();
```

_
AnalyticsController.cs

```
using System;
using System.Linq;
using InfraProject.Context;
using System.Collections.Generic;

namespace InfraProject.Repositories
{
    public class AnalyticsRepository
    {
        private readonly VisitorManagementDbContext _context;

        public AnalyticsRepository(VisitorManagementDbContext context)
        {
            _context = context;
        }

        public int GetTotalVisitors()
        {
            return _context.Visitors.Count();
        }

        public int GetVisitorsPerDay(DateTime date)
        {
            return _context.Visits.Count(v => v.CheckIn.Date == date.Date);
        }

        public double GetAverageVisitDuration()
        {
            return _context.Visits.Where(v => v.IsCompleted).Average(v =>
(v.CheckOut - v.CheckIn).TotalMinutes);
        }

        public Dictionary<string, int> GetVisitsPerHost()
        {
            return _context.Visits.GroupBy(v => v.HostEmail)
                                   .ToDictionary(g => g.Key, g => g.Count());
        }

        // New methods for advanced analytics
        public int GetMonthlyVisitors(int month, int year)
        {
            return _context.Visits.Count(v => v.CheckIn.Month == month &&
v.CheckIn.Year == year);
        }

        public TimeSpan GetPeakVisitTime()
        {
            return _context.Visits.GroupBy(v => v.CheckIn.TimeOfDay)
                                   .OrderByDescending(g => g.Count())
                                   .FirstOrDefault()?.Key ?? TimeSpan.Zero;
        }

        public string GetMostFrequentVisitor()
        {
            return _context.Visits.GroupBy(v => v.VisitorID)
                                   .OrderByDescending(g => g.Count())
                                   .FirstOrDefault()?.Key;
        }

        public string GetMostActiveHost()
        {
            return _context.Visits.GroupBy(v => v.HostEmail)
                                   .OrderByDescending(g => g.Count())
                                   .FirstOrDefault()?.Key;
        }
    }
}
```

_
RoleControlle.cs

```
using Microsoft.AspNetCore.Mvc;
using InfraProject.Repositories;
using DomainProject.DomainModels;

namespace ApplicationProject.Controller
{
    [ApiController]
    [Route("api/[controller]")]
    public class RoleController : ControllerBase
    {
        private readonly RoleRepository _roleRepository;

        public RoleController(RoleRepository roleRepository)
        {
            _roleRepository = roleRepository;
        }

        [HttpGet]
        public IActionResult GetAllRoles()
        {
            var roles = _roleRepository.GetAllRoles();
            return Ok(roles);
        }

        [HttpGet("{id}")]
        public IActionResult GetRoleById(string id)
        {
            var role = _roleRepository.GetRoleById(id);
            if (role == null)
            {
                return NotFound();
            }
            return Ok(role);
        }

        [HttpPost]
        public IActionResult AddRole([FromBody] Role role)
        {
            _roleRepository.AddRole(role);
            return CreatedAtAction(nameof(GetRoleById), new { id = role.RoleID },
role);
        }

        [HttpPut("{id}")]
        public IActionResult UpdateRole(string id, [FromBody] Role role)
        {
            if (id != role.RoleID)
            {
                return BadRequest();
            }

            _roleRepository.UpdateRole(role);
            return NoContent();
        }

        [HttpDelete("{id}")]
        public IActionResult DeleteRole(string id)
        {
            _roleRepository.DeleteRole(id);
            return NoContent();
        }

        [HttpPost("ConfigureRole")]
        public IActionResult ConfigureRole([FromBody] Role role)
        {
            _roleRepository.AddOrUpdateRole(role);
            return CreatedAtAction(nameof(GetRoleById), new { id = role.RoleID },
role);
        }

    }
}
```

—
UserController.cs

```
using Microsoft.AspNetCore.Mvc;
using InfraProject.Repositories;
using DomainProject.DomainModels;

namespace ApplicationProject.Controller
{
    [ApiController]
    [Route("api/[controller]")]
    public class UserController : ControllerBase
    {
        private readonly UserRepository _userRepository;

        public UserController(UserRepository userRepository)
        {
            _userRepository = userRepository;
        }

        [HttpGet]
        public IActionResult GetAllUsers()
        {
            var users = _userRepository.GetAllUsers();
            return Ok(users);
        }

        [HttpPost("CreateAdmin")]
        public IActionResult CreateAdminUser([FromBody] User adminUser)
        {
            _userRepository.CreateAdminUser(adminUser);
            return CreatedAtAction(nameof(GetUserById), new { id =
adminUser.UserID }, adminUser);
        }

        [HttpGet("{id}")]
        public IActionResult GetUserById(string id)
        {
            var user = _userRepository.GetUserById(id);
            if (user == null)
            {
                return NotFound();
            }
            return Ok(user);
        }


        [HttpPut("UpdateUserRole/{userId}/{newRoleId}")]
        public IActionResult UpdateUserRole(string userId, string newRoleId)
        {
            _userRepository.UpdateUserRole(userId, newRoleId);
            return NoContent();
        }

    }
}
```

_
VisitorController.cs

```
using Microsoft.AspNetCore.Mvc;
using InfraProject.Repositories;
using DomainProject.DomainModels;

namespace ApplicationProject.Controller
{
    [ApiController]
    [Route("api/[controller]")]
    public class VisitorController : ControllerBase
    {
        private readonly VisitorRepository _visitorRepository;
        private readonly NotificationRepository _notificationRepository;
        private readonly VisitsRepository _visitsRepository;

        public VisitorController(VisitsRepository visitsRepository, 
VisitorRepository visitorRepository, NotificationRepository 
notificationRepository)
        {
            _visitsRepository = visitsRepository;
            _visitorRepository = visitorRepository;
            _notificationRepository = notificationRepository;
        }

        [HttpGet]
        public IActionResult GetAllVisitors()
        {
            var visitors = _visitorRepository.GetAllVisitors();
            return Ok(visitors);
        }

        [HttpGet("{id}")]
        public IActionResult GetVisitorById(string id)
        {
            var visitor = _visitorRepository.GetVisitorById(id);
            if (visitor == null)
            {
                return NotFound();
            }
            return Ok(visitor);
        }

        [HttpPost]
        public IActionResult AddVisitor([FromBody] Visitor visitor)
        {
            _visitorRepository.AddVisitor(visitor);
            return CreatedAtAction(nameof(GetVisitorById), new { id = 
visitor.VisitorID }, visitor);
        }

        [HttpPut("{id}")]
        public IActionResult UpdateVisitor(string id, [FromBody] Visitor visitor)
        {
            if (id != visitor.VisitorID)
            {
                return BadRequest();
            }

            _visitorRepository.UpdateVisitor(visitor);
            return NoContent();
        }

        [HttpDelete("{id}")]
        public IActionResult DeleteVisitor(string id)
        {
            _visitorRepository.DeleteVisitor(id);
            return NoContent();
        }

        [HttpPost("PreRegister")]
        public IActionResult PreRegisterVisitor([FromBody] Visitor visitor)
        {
            _visitorRepository.PreRegisterVisitor(visitor);
            _notificationRepository.AddNotification(visitor.HostEmail, "A visitor
has pre-registered.");
            return CreatedAtAction(nameof(GetVisitorById), new { id =
visitor.VisitorID }, visitor);
        }

        [HttpPost("OnSiteRegister")]
        public IActionResult OnSiteRegisterVisitor([FromBody] Visitor visitor)
        {
            _visitorRepository.OnSiteRegisterVisitor(visitor);
            _notificationRepository.AddNotification(visitor.HostEmail, "A visitor
has registered on-site.");
            return CreatedAtAction(nameof(GetVisitorById), new { id =
visitor.VisitorID }, visitor);
        }

        [HttpPost("AddToBlacklist/{visitorId}")]
        public IActionResult AddToBlacklist(string visitorId)
        {
            var visitor = _visitorRepository.GetVisitorById(visitorId);
            if (visitor != null)
            {
                _visitorRepository.AddToBlacklist(visitorId);
                _notificationRepository.AddNotification(visitor.HostEmail, "A
visitor has been added to the blacklist.");
            }
            return Ok();
        }

        [HttpPost("RemoveFromBlacklist/{visitorId}")]
        public IActionResult RemoveFromBlacklist(string visitorId)
        {
            var visitor = _visitorRepository.GetVisitorById(visitorId);
            if (visitor != null)
            {
                _visitorRepository.RemoveFromBlacklist(visitorId);
                _notificationRepository.AddNotification(visitor.HostEmail, "A
visitor has been removed from the blacklist.");
            }
            return Ok();
        }

        [HttpPost("CheckIn")]
        public IActionResult CheckInVisitor([FromBody] Visits visit)
        {
            _visitsRepository.CheckInVisitor(visit);
            _notificationRepository.AddNotification(visit.HostEmail, "A visitor
has checked in.");
            return CreatedAtAction(nameof(CheckInVisitor), new { id =
visit.VisitID }, visit);
        }

        [HttpPost("AddVisit")]
        public IActionResult AddVisit([FromBody] Visits visit)
        {
            _visitsRepository.AddVisit(visit);
            return CreatedAtAction(nameof(AddVisit), new { id = visit.VisitID },
visit);
        }
    }
}
```

_

VisitsController.cs

```
using Microsoft.AspNetCore.Mvc;
using InfraProject.Repositories;
using DomainProject.DomainModels;

namespace ApplicationProject.Controller
{
    [ApiController]
    [Route("api/[controller]")]
    public class VisitsController : ControllerBase
    {
        private readonly VisitsRepository _visitsRepository;

        public VisitsController(VisitsRepository visitsRepository)
        {
            _visitsRepository = visitsRepository;
        }

        [HttpPost("CheckIn")]
        public IActionResult CheckInVisitor([FromBody] Visits visit)
        {
            _visitsRepository.CheckInVisitor(visit);
            return CreatedAtAction(nameof(CheckInVisitor), new { id =
visit.VisitID }, visit);
        }

        [HttpPost("CheckOut/{visitId}")]
        public IActionResult CheckOutVisitor(string visitId)
        {
            _visitsRepository.CheckOutVisitor(visitId);
            return Ok();
        }

        [HttpPost("AddVisit")]
        public IActionResult AddVisit([FromBody] Visits visit)
        {
            _visitsRepository.AddVisit(visit);
            return CreatedAtAction(nameof(AddVisit), new { id = visit.VisitID },
visit);
        }
    }
}
```

_
Analytics.cs

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace DomainProject.DomainModels
{
    public class Analytics
    {
        public int ReportID { get; set; }
        public string Type { get; set; }
        public int GeneratedBy { get; set; }
        public DateTime GeneratedOn { get; set; }
    }

}
```

_
Notification.cs

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace DomainProject.DomainModels
{
    public class Notification
    {
        public int NotificationID { get; set; }
        public int UserID { get; set; }
        public string Message { get; set; }
        public DateTime Time { get; set; }
    }

}
```

_
Role.cs

```
using System.ComponentModel.DataAnnotations;

namespace DomainProject.DomainModels
{
    public class Role
    {
        public int RoleID { get; set; }
        public string RoleName { get; set; }
        public string Permissions { get; set; }
    }

}
```

_
User.cs

```
using Microsoft.AspNetCore.Identity;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace DomainProject.DomainModels
{

    public class User
    {
        public int UserID { get; set; }
        public string Username { get; set; }
        public string Password { get; set; }
        public int RoleID { get; set; }
        public string Email { get; set; }
        public bool IsActive { get; set; }
        public Role Role { get; set; }
    }

}
```

_
Visitor.cs

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace DomainProject.DomainModels
{
    public class Visitor
    {
        public int VisitorID { get; set; }
        public bool IsPreRegistered { get; set; }
        public bool IsOnSiteRegistered { get; set; }
        public string Title { get; set; }
        public string FName { get; set; }
        public string LName { get; set; }
        public string Email { get; set; }
        public string Number { get; set; }
        public string Add { get; set; }
        public string IdentityType { get; set; }
        public string IdNum { get; set; }
        public byte[] Image { get; set; }
        public bool IsBlacklisted { get; set; }
    }

}
--
Visits.cs
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace DomainProject.DomainModels
{
    public class Visits
    {
        public int VisitID { get; set; }
        public int VisitorID { get; set; }
        public DateTime CheckIn { get; set; }
        public DateTime CheckOut { get; set; }
        public string HostEmail { get; set; }
        public bool IsCompleted { get; set; }
        public string HostName { get; set; }
        public string Purpose { get; set; }
        public DateTime ExpectedArrival { get; set; }
        public DateTime ExpectedDepart { get; set; }
        public int VisitDuration { get; set; }
    }

}


_
VisitorManagementDbContext.cs
using DomainProject.DomainModels;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace InfraProject.Context
{
    public class VisitorManagementDbContext : IdentityDbContext<User>
    {
        public
VisitorManagementDbContext(DbContextOptions<VisitorManagementDbContext> options)
            : base(options)
        {
        }

        public DbSet<Role> Roles { get; set; }
        public DbSet<Visitor> Visitors { get; set; }
        public DbSet<Visits> Visits { get; set; }
        public DbSet<Analytics> Analytics { get; set; }
        public DbSet<Notification> Notifications { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Visits>()
                .HasOne(v => v.Visitor)
                .WithMany()
                .HasForeignKey(v => v.VisitorID)
                .OnDelete(DeleteBehavior.NoAction);  // or DeleteBehavior.SetNull

            modelBuilder.Entity<Visits>()
                .HasOne(v => v.User)
                .WithMany()
                .HasForeignKey(v => v.HostEmail)
                .OnDelete(DeleteBehavior.NoAction);  // or DeleteBehavior.SetNull

            base.OnModelCreating(modelBuilder);  // for Identity configurations
        }



    }
}

_
AnalyticsRepository.cs
using System;
using System.Linq;
using InfraProject.Context;
using System.Collections.Generic;

namespace InfraProject.Repositories
{
    public class AnalyticsRepository
    {
        private readonly VisitorManagementDbContext _context;

        public AnalyticsRepository(VisitorManagementDbContext context)
        {
            _context = context;
        }

        public int GetTotalVisitors()
        {
            return _context.Visitors.Count();
        }

        public int GetVisitorsPerDay(DateTime date)
        {
            return _context.Visits.Count(v => v.CheckIn.Date == date.Date);
        }

        public double GetAverageVisitDuration()
        {
            return _context.Visits.Where(v => v.IsCompleted).Average(v =>
(v.CheckOut - v.CheckIn).TotalMinutes);
        }

        public Dictionary<string, int> GetVisitsPerHost()
        {
            return _context.Visits.GroupBy(v => v.HostEmail)
                                  .ToDictionary(g => g.Key, g => g.Count());
        }

        public int GetMonthlyVisitors(int month, int year)
        {
            return _context.Visits.Count(v => v.CheckIn.Month == month &&
v.CheckIn.Year == year);
        }

        public TimeSpan GetPeakVisitTime()
        {
            return _context.Visits.GroupBy(v => v.CheckIn.TimeOfDay)
                                  .OrderByDescending(g => g.Count())
                                  .FirstOrDefault()?.Key ?? TimeSpan.Zero;
        }

        public string GetMostFrequentVisitor()
        {
            return _context.Visits.GroupBy(v => v.VisitorID)
                                  .OrderByDescending(g => g.Count())
                                  .FirstOrDefault()?.Key;
        }

        public string GetMostActiveHost()
        {
            return _context.Visits.GroupBy(v => v.HostEmail)
                                  .OrderByDescending(g => g.Count())
                                  .FirstOrDefault()?.Key;
        }
        public Dictionary<string, int> GetAnalyticsByRole(string role)
        {
            return _context.Users.Where(u => u.RoleID == role)
                                .GroupBy(u => u.RoleID)
                                .ToDictionary(g => g.Key, g => g.Count());
        }
    }
```

```
}
_
NotificationRepository.cs
using DomainProject.DomainModels;
using InfraProject.Context;
using System;

namespace InfraProject.Repositories
{
    public class NotificationRepository
    {
        private readonly VisitorManagementDbContext _context;

        public NotificationRepository(VisitorManagementDbContext context)
        {
            _context = context;
        }

        public void AddNotification(string userId, string type)
        {
            string message = type switch
            {
                "CheckIn" => "A visitor has checked in.",
                "CheckOut" => "A visitor has checked out.",
                "PreRegister" => "A visitor has been pre-registered.",
                "OnSiteRegister" => "A visitor has been registered on-site.",
                "AddToBlacklist" => "A visitor has been added to the blacklist.",
                "RemoveFromBlacklist" => "A visitor has been removed from the
blacklist.",
                _ => "Unknown notification type."
            };

            var notification = new Notification
            {
                NotificationID = Guid.NewGuid().ToString(),
                UserID = userId,
                Message = message,
                Time = DateTime.Now
            };

            _context.Notifications.Add(notification);
            _context.SaveChanges();
        }
        public IEnumerable<Notification> GetNotificationsByUser(string userId)
        {
            return _context.Notifications.Where(n => n.UserID ==
userId).ToList();
        }
    }
}


_
RoleRepository.cs
using System.Collections.Generic;
using System.Linq;
using DomainProject.DomainModels;
using InfraProject.Context;
using Microsoft.EntityFrameworkCore;

namespace InfraProject.Repositories
{
    public class RoleRepository
    {
        private readonly VisitorManagementDbContext _context;

        public RoleRepository(VisitorManagementDbContext context)
        {
            _context = context;
        }

        public IEnumerable<Role> GetAllRoles()
        {
            return _context.Roles.ToList();
        }

        public Role GetRoleById(string id)
        {
            return _context.Roles.Find(id);
        }

        public void AddRole(Role role)
        {
            _context.Roles.Add(role);
            _context.SaveChanges();
        }

        public void UpdateRole(Role role)
        {
            _context.Entry(role).State = EntityState.Modified;
            _context.SaveChanges();
        }

        public void DeleteRole(string id)
        {
            var role = _context.Roles.Find(id);
            if (role != null)
            {
                _context.Roles.Remove(role);
                _context.SaveChanges();
            }
        }

        public void AddOrUpdateRole(Role role)
        {
            var existingRole = _context.Roles.Find(role.RoleID);
            if (existingRole != null)
            {
                _context.Entry(existingRole).CurrentValues.SetValues(role);
            }
            else
            {
                _context.Roles.Add(role);
            }
            _context.SaveChanges();
        }
        public IEnumerable<Role> GetRolesByPermission(string permission)
        {
            return _context.Roles.Where(r =>
r.Permissions.Contains(permission)).ToList();
        }
    }
}


_
UserRepository.cs
using System.Collections.Generic;
```

```csharp
using System.Linq;
using DomainProject.DomainModels;
using InfraProject.Context;
using Microsoft.AspNetCore.Identity;

namespace InfraProject.Repositories
{
    public class UserRepository
    {
        private readonly VisitorManagementDbContext _context;
        private readonly UserManager<User> _userManager;

        public UserRepository(VisitorManagementDbContext context,
UserManager<User> userManager)
        {
            _userManager = userManager;
            _context = context;
        }

        public void CreateAdminUser(User adminUser)
        {
            adminUser.RoleID = "Admin";
            _context.Users.Add(adminUser);
            _context.SaveChanges();
        }

        public IEnumerable<User> GetAllUsers()
        {
            return _context.Users.ToList();
        }

        public void UpdateUserRole(string userId, string newRoleId)
        {
            var user = _context.Users.Find(userId);
            if (user != null)
            {
                user.RoleID = newRoleId;
                _context.SaveChanges();
            }
        }

        public User GetUserById(string id)
        {
            return _context.Users.Find(id);
        }
        public IEnumerable<User> GetUsersByRole(string role)
        {
            return _context.Users.Where(u => u.RoleID == role).ToList();
        }
    }
}

_
VisitorRepository.cs
using System.Collections.Generic;
using System.Linq;
using DomainProject.DomainModels;
using InfraProject.Context;
using Microsoft.EntityFrameworkCore;

namespace InfraProject.Repositories
{
    public class VisitorRepository
```

```
{
    private readonly VisitorManagementDbContext _context;

    public VisitorRepository(VisitorManagementDbContext context)
    {
        _context = context;
    }

    public IEnumerable<Visitor> GetAllVisitors()
    {
        return _context.Visitors.ToList();
    }

    public Visitor GetVisitorById(string id)
    {
        return _context.Visitors.Find(id);
    }

    public void AddVisitor(Visitor visitor)
    {
        _context.Visitors.Add(visitor);
        _context.SaveChanges();
    }

    public void UpdateVisitor(Visitor visitor)
    {
        _context.Entry(visitor).State = EntityState.Modified;
        _context.SaveChanges();
    }

    public void DeleteVisitor(string id)
    {
        var visitor = _context.Visitors.Find(id);
        if (visitor != null)
        {
            _context.Visitors.Remove(visitor);
            _context.SaveChanges();
        }
    }
    public void PreRegisterVisitor(Visitor visitor)
    {
        visitor.IsPreRegistered = true;
        _context.Visitors.Add(visitor);
        _context.SaveChanges();
    }

    public void OnSiteRegisterVisitor(Visitor visitor)
    {
        visitor.IsOnSiteRegistered = true;
        _context.Visitors.Add(visitor);
        _context.SaveChanges();
    }
    public void AddToBlacklist(string visitorId)
    {
        var visitor = _context.Visitors.Find(visitorId);
        if (visitor != null)
        {
            visitor.IsBlacklisted = true;
            _context.SaveChanges();
        }
    }
    public void RemoveFromBlacklist(string visitorId)
    {
            var visitor = _context.Visitors.Find(visitorId);
            if (visitor != null)
            {
                visitor.IsBlacklisted = false;
                _context.SaveChanges();
            }
        }

        public IEnumerable<Visitor> GetVisitorsByHost(string hostEmail)
        {
            return _context.Visitors.Where(v => v.HostEmail ==
hostEmail).ToList();
        }


    }
}
```

_
VisitsRepository.cs

```
using System;
using DomainProject.DomainModels;
using InfraProject.Context;

namespace InfraProject.Repositories
{
    public class VisitsRepository
    {
        private readonly VisitorManagementDbContext _context;

        public VisitsRepository(VisitorManagementDbContext context)
        {
            _context = context;
        }

        public void CheckInVisitor(Visits visit)
        {
            visit.CheckIn = DateTime.Now;
            _context.Visits.Add(visit);
            _context.SaveChanges();
        }

        public void CheckOutVisitor(string visitId)
        {
            var visit = _context.Visits.Find(visitId);
            if (visit != null)
            {
                visit.CheckOut = DateTime.Now;
                visit.IsCompleted = true;
                _context.SaveChanges();
            }
        }
        public void AddVisit(Visits visit)
        {
            _context.Visits.Add(visit);
            _context.SaveChanges();
        }
        public IEnumerable<Visits> GetVisitsByDate(DateTime date)
        {
            return _context.Visits.Where(v => v.CheckIn.Date ==
date.Date).ToList();
        }
    }
}
```

_
ApplicationProject csproj
```
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net7.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore"
Version="6.0.21" />
    <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="7.0.10" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.Design"
Version="6.0.21">
      <PrivateAssets>all</PrivateAssets>
      <IncludeAssets>runtime; build; native; contentfiles; analyzers;
buildtransitive</IncludeAssets>
    </PackageReference>
    <PackageReference Include="Swashbuckle.AspNetCore" Version="6.4.0" />
    <PackageReference Include="Microsoft.EntityFrameworkCore" Version="6.0.21" />
    <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer"
Version="6.0.21" />
  </ItemGroup>

  <ItemGroup>
    <ProjectReference Include="..\DomainProject\DomainProject.csproj" />
    <ProjectReference Include="..\InfraProject\InfraProject.csproj" />
  </ItemGroup>

</Project>
```

_
DomainProject csproj
```
<Project Sdk="Microsoft.NET.Sdk">

	<PropertyGroup>
		<TargetFramework>net7.0</TargetFramework>
		<ImplicitUsings>enable</ImplicitUsings>
		<Nullable>enable</Nullable>
	</PropertyGroup>
	<ItemGroup>
		<PackageReference
Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore" Version="6.0.21" />
		<PackageReference Include="Microsoft.EntityFrameworkCore"
Version="6.0.21" />
		<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer"
Version="6.0.21" />
	</ItemGroup>
</Project>
```

_
InfraProject csproj
```
<Project Sdk="Microsoft.NET.Sdk">

	<PropertyGroup>
		<TargetFramework>net7.0</TargetFramework>
		<ImplicitUsings>enable</ImplicitUsings>
		<Nullable>enable</Nullable>
	</PropertyGroup>
```

```xml
	<ItemGroup>
		<PackageReference
Include="Microsoft.AspNetCore.Identity.EntityFrameworkCore" Version="6.0.21" />
		<PackageReference Include="Microsoft.EntityFrameworkCore"
Version="6.0.21" />
		<PackageReference Include="Microsoft.EntityFrameworkCore.Design"
Version="6.0.21">
		  <PrivateAssets>all</PrivateAssets>
		  <IncludeAssets>runtime; build; native; contentfiles; analyzers;
buildtransitive</IncludeAssets>
		</PackageReference>
		<PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer"
Version="6.0.21" />
	</ItemGroup>

	<ItemGroup>
		<ProjectReference Include="..\DomainProject\DomainProject.csproj" />
	</ItemGroup>

</Project>
```