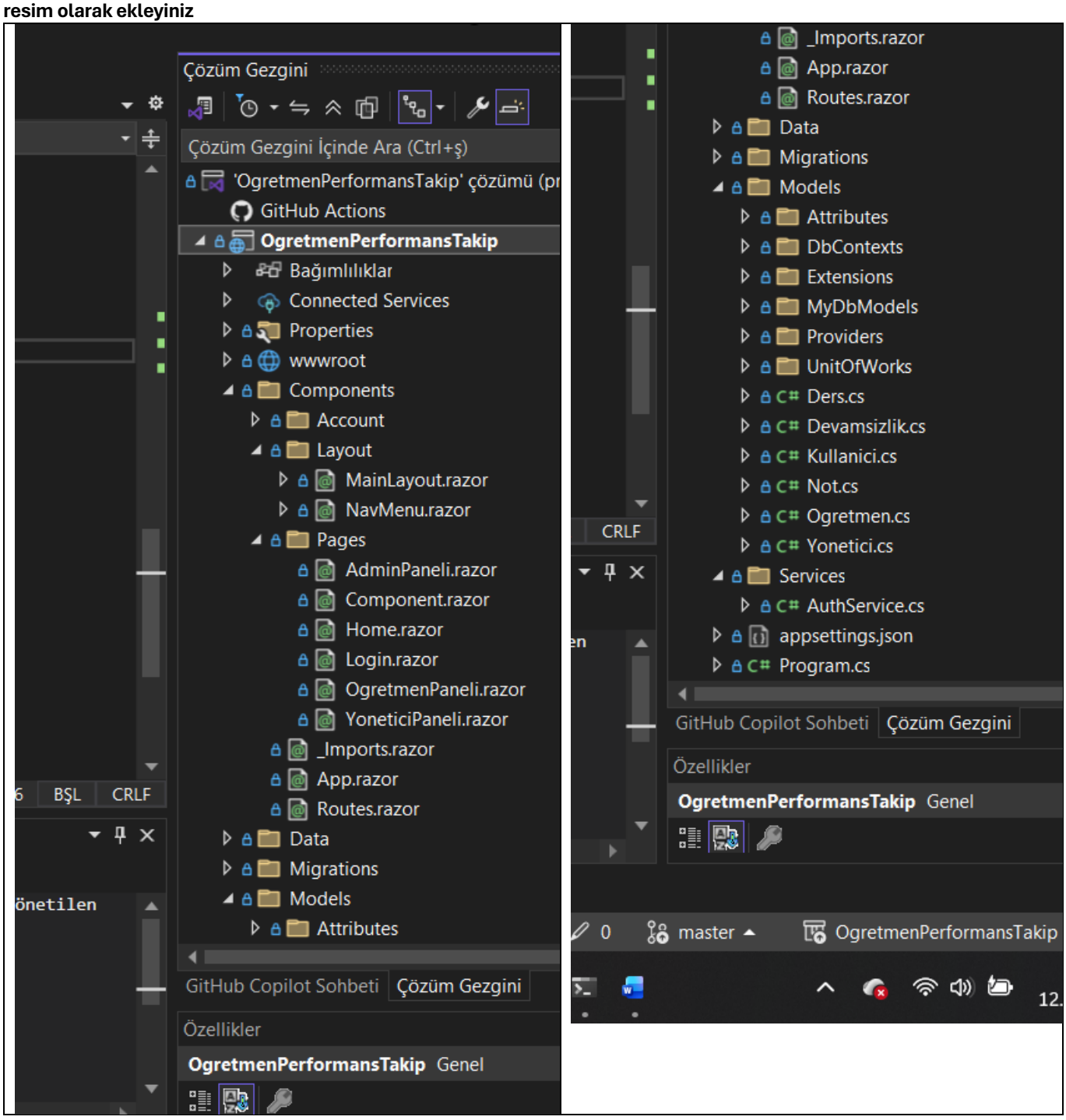
Proje Adı	Öğretmen Performans Takip
Proje Konusu	Öğretmenlerin işini düzgün yapıp yapmadığını kontrol etmek.
Proje Çözüm Adı	SAT242516060

Proje 'Çözüm Gezgini (Solution Explorer)' penceresinin (alt klasörler ve dosyalar dahil olacak şekilde) ekran görüntüsünü resim olarak ekleyiniz



Projede oluşturulan arabirimlerin (interface) C# kodlarını yazınız.

```
public interface IMyDbModel
{
    string Message { get; set; }
    IMyDbModel_Parameter Parameters { get; set; }
    IDictionary<object, object> OrderByItems { get; set; }
}
```

```
public interface IMyDbModel<T> : IMyDbModel where T : class, new()
{
    IEnumerable<T> Items { get; set; }
}
```

```
public interface IMyDbModel_Parameter
{
    string OrderBy { get; set; }
    int PageNumber { get; set; }
    int PageSize { get; set; }
    int TotalPageCount { get; }
    int TotalRecordCount { get; set; }
    IDictionary<string, object> Params { get; set; }
    IDictionary<string, string> Where { get; set; }
}
```

```
public interface IMyDbModel_Result_KeyValue<TKey,TValue>
{
    TKey Key { get; set; }
    TValue Value { get; set; }
}
```

```
public interface IMyDbModel_Provider
{
    ValueTask<IMyDbModel<TResult>> Execute<TResult>(IMyDbModel<TResult>
myResultModel,
        string spName = "",
        bool isPagination = true) where TResult : class, new();

    ValueTask<IMyDbModel<TResult>> Execute<TResult>(string spName = "",
        params (string Key, object Value)[] parameters)
        where TResult : class, new();

    ValueTask<IEnumerable<TResult>> GetItems<TResult>(string spName = "",
        params (string Key, object Value)[] parameters)
        where TResult : class, new();

    ValueTask<IEnumerable<TResult>> SetItems<TResult>(string spName = "",
        params (string Key, object Value)[] parameters)
        where TResult : class, new();
}
```

```
public interface IMyDbModel_UnitOfWork
{
    Task Execute<T>(IMyDbModel<T> myDbModel, string spName = "", bool isPagination =
true)
        where T : class, new();
}
```

Projede oluşturulan sınıfların (class) C# kodlarını yazınız.

Uyarı : Sınıf içerisinde metotların sadece isim ve parametreleri yazılacak.
Örnek : public async Task MethodAsync(parameters...) { return null; }

```
  public sealed class MyDbModel<T> : IMyDbModel<T> where T : class, new()
{
    public MyDbModel() : this(1, 10, "")
    {
    }

    public MyDbModel(int pageNumber, int pageSize, string orderBy)
    {
        Parameters = MyDbModel_Parameter.Create(pageNumber, pageSize, orderBy);
        OrderByItems = this.GetOrderByItems();
        Items = new List<T>();
    }

    public IMyDbModel_Parameter Parameters { get; set; }
    public IDictionary<object, object> OrderByItems { get; set; }
    public IEnumerable<T> Items { get; set; }
    public string Message { get; set; }
}
```

```
internal sealed class MyDbModel_Parameter : IMyDbModel_Parameter
{
    public static MyDbModel_Parameter Create(int pageNumber, int pageSize, string
orderBy) => new(pageNumber, pageSize, orderBy);
    private MyDbModel_Parameter(int pageNumber, int pageSize, string orderBy)
    {
        PageNumber = pageNumber;
        PageSize = pageSize;
        OrderBy = orderBy;

        if (Params == null) Params = new Dictionary<string, object>();
        if (Where == null) Where = new Dictionary<string, string>();
    }

    public int PageNumber { get; set; }
    public int PageSize { get; set; }
    public int TotalRecordCount { get; set; }
    public int TotalPageCount => (int)Math.Ceiling(TotalRecordCount /
(double)(PageSize <= 0 ? 1 : PageSize));
    public string OrderBy { get; set; }
    public IDictionary<string, object> Params { get; set; }
    public IDictionary<string, string> Where { get; set; }
}
```

```
public class MyDbModel_Result_KeyValue<TKey, TValue> :
IMyDbModel_Result_KeyValue<TKey, TValue>
{
    public TKey Key { get; set; }
    public TValue Value { get; set; }
}
```

```
public static class MyDbModel_Extension
{
    public static IDictionary<object, object> GetOrderByItems<E>(this MyDbModel<E>
myDbModel) where E : class, new()
    {
        var sortByItems = new Dictionary<object, object>();

        return sortByItems;
    }

}
```

```
public class MyDbModel_Provider(IMyDbModel_UnitOfWork myDbModel_UnitOfWork) :
IMyDbModel_Provider
{
    #region Execute : Pagination=true

    public async ValueTask<IMyDbModel<TResult>> Execute<TResult>(IMyDbModel<TResult>
myResultModel,
        string spName = "",
        bool isPagination = true) where TResult : class, new()
    {
        if (myResultModel == null)
            myResultModel = new MyDbModel<TResult>();

        await myDbModel_UnitOfWork.Execute(myResultModel, spName, isPagination);

        return myResultModel;
    }
```

```
public sealed class MyDbModel_UnitOfWork<TDbContext>(TDbContext context) :
IMyDbModel_UnitOfWork where TDbContext : DbContext
{
    private readonly DbContext _context = context;

    public async Task Execute<T>(IMyDbModel<T> myDbModel,
        string spName = "",
        bool isPagination = true)
        where T : class, new()
    {
```

```
public class MyDbModel_DbContext(DbContextOptions<MyDbModel_DbContext> options) :
DbContext(options)
{
}
```

```
public static class Extensions_DataTable
{
    public static IEnumerable<T> DataTableToList<T>(this DataTable table) where T :
class
    {
        var list = new List<T>();
        try
        {
            var columnsNames = new List<string>();
            foreach (DataColumn DataColumn in table.Columns)
                columnsNames.Add(DataColumn.ColumnName);

            list = table.AsEnumerable().ToList()
                .ConvertAll(row => GetObject<T>(row, columnsNames));
        }
        catch (Exception)
        {
            // ...
        }

        return list;
    }
```

```
public static class Extensions_Enum
{
    public static string Color<T>(this T value)
    {
        var result = value.ToString();

        try
        {
            var fi = value
                .GetType()
                .GetField(value.ToString());

            if (fi != null)
            {
                var attributes =
(ColorAttribute[])fi.GetCustomAttributes(typeof(ColorAttribute), false);
                result = attributes != null && attributes.Length > 0
                    ? attributes[0].Color
                    : value.ToString();
            }
        }
        catch (Exception) { }

        return result;
    }
```

```
public static class Extensions_Json
{
    public static T JsonToItem<T>(this string jsonItem)
    {
        var json = default(T);
        try
        {
            json = JsonSerializer.Deserialize<T>(jsonItem ?? "");
        }
        catch (Exception)
        {
        }

        return json;
    }
```

```
public static class Extensions_SqlParameter
{
    #region ToSqlParameter_Table_Type_Dictionary

    public static SqlParameter ToSqlParameter_Table_Type_Dictionary<TKey, TValue>(
        this IDictionary<TKey, TValue> dictionary,
        string parameterName,
        string parameterTypeName = "",
        int length = 0,
        SqlDbType sqlDbType = SqlDbType.Structured,
        ParameterDirection direction = ParameterDirection.Input)
    {
```

```
namespace Attributes;

public class ColorAttribute(string color) : Attribute
{
    public string Color { get; set; } = color;
}
```

```
namespace Attributes;
public class TitleAttribute(string title) : Attribute
{
    public string Title { get; set; } = title;
}
```

appsettings.json dosyasında "DefaultConnection" ifadesini projenize göre yazınız

```
DefaultConnection : Server=localhost,1445; Database=OgretmenPerformansTakip; User
Id=sa; Password=Oo_454545; TrustServerCertificate=True;
```

Program.cs dosyasında, gerekli servis kayıtlarını yapan C# kodlarını yazınız.

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddRazorComponents()
    .AddInteractiveServerComponents();

builder.Services.AddCascadingAuthenticationState();
builder.Services.AddScoped<IdentityUserAccessor>();
builder.Services.AddScoped<IdentityRedirectManager>();
builder.Services.AddScoped<AuthenticationStateProvider,
IdentityRevalidatingAuthenticationStateProvider>();

builder.Services.AddAuthentication(options =>
    {
        options.DefaultScheme = IdentityConstants.ApplicationScheme;
        options.DefaultSignInScheme = IdentityConstants.ExternalScheme;
    })
    .AddIdentityCookies();

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection")
?? throw new InvalidOperationException("Connection string 'DefaultConnection' not
found.");
builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlServer(connectionString));
builder.Services.AddDatabaseDeveloperPageExceptionFilter();

builder.Services.AddIdentityCore<ApplicationUser>(options =>
options.SignIn.RequireConfirmedAccount = true)
    .AddEntityFrameworkStores<ApplicationDbContext>()
    .AddSignInManager()
    .AddDefaultTokenProviders();

builder.Services.AddSingleton<IEmailSender<ApplicationUser>,
IdentityNoOpEmailSender>();
// ... var builder = WebApplication.CreateBuilder(args);
// ... builder.Services.AddRazorComponents<App>();

// Bu satırı ekleyin:
// AuthService'i 'Singleton' olarak ekle.
// Yani uygulama boyunca tek bir örneği olacak.
builder.Services.AddSingleton<AuthService>();


var app = builder.Build();
```

App.razor bileşeninde, gerekli C# kodlarını yazınız

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <base href="/" />
    <link rel="stylesheet" href="bootstrap/bootstrap.min.css" />
    <link rel="stylesheet" href="app.css" />
    <link rel="stylesheet" href="OgretmenPerformansTakip.styles.css" />
    <link rel="icon" type="image/png" href="favicon.png" />
    <HeadOutlet />
</head>

<body>
    <Routes />
    <script src="_framework/blazor.web.js"></script>
</body>

</html>
```

Projenizde oluşturmuş olduğunuz bileşenlerin tasarım ve C# kodlarını yazınız (Blazor Component)

NOT: daha fazla bileşen için, aşağıdaki tabloyu çoğaltınız.

Bileşen Adı	Ogretmen.Razor

Tasarım Kodları

```
@page "/ogretmen-paneli"
@page "/ogretmen/not-gir"
@page "/ogretmen/devamsizlik-gir"
@page "/ogretmen/ders-programi"
@using OgretmenPerformansTakip.Services
@rendermode InteractiveServer

@inject AuthService AuthService
@inject NavigationManager NavigationManager

<h3>Öğretmen Paneli</h3>

@* Yetki Kontrolü: Eğer giriş yapılmamışsa veya rol yanlışsa, giriş
sayfasına at. *@
@if (!AuthService.IsLoggedIn || AuthService.UserRole != "ogretmen")
{
    <p class="text-danger">Bu sayfayı görmek için öğretmen olarak giriş
yapmalısınız.</p>
}
else
{
    <h4>Hoş geldin, @AuthService.CurrentUser.Ad</h4>
    <p>Burada öğretmen işlemleri (Not Girme, Devamsızlık Girme vb.)
için formlar yer alacak.</p>

    @* Buraya <NotGirmeFormu /> veya <DevamsizlikFormu /> gibi
        ayrı component'ler (bileşenler) ekleyebilirsin.
    *@
}

@code {
    // Sayfa yüklendiğinde çalış
    protected override void OnInitialized()
    {
        // Eğer kullanıcı öğretmen değilse, anında /login'e geri yolla
        if (AuthService.UserRole != "ogretmen")
        {
            NavigationManager.NavigateTo("/login");
        }
    }
}
```

C# Kodları

```
namespace OgretmenPerformansTakip.Models
{
    // 'Kullanici' sınıfından kalıtım alır
    public class Ogretmen : Kullanici
    {
        public Ogretmen(string id, string ad, string email) : base(id,
ad, email) { }
    }
}
```

Ekran Görüntüleri

UYARI: Resim boyutlarının çok büyük olmamasına dikkat ediniz