BMT-308
WEB PROGRAMLAMA
ASP.NET CORE PROJE

Film Arşiv Uygulama Raporu

Hazırlayanlar:

Sena Nur Türkmen	-	22181616006
Ecemnur Sünnetcioğlu	-	22181617007
İrem Şaziye Koç	-	22181616020
Şura Özkan	-	22181616503
Ceyda Emre	-	201816707
Gülse Kayıkcı	-	22181616014

Dersin Öğretmeni : Sinan Toklu

İçindekiler

1. **Giriş**

2. **Kullanılan Yöntem ve Teknolojiler**
    - **2.1. Yazılım Geliştirme Yaklaşımı**
    - **2.2. Kullanılan Teknolojiler**
    - **2.3. Proje Yapısı**
    - **2.4. Veritabanı Yapılandırması**
    - **2.5. Proje Bağımlılıkları**

3. **Sistem Mimarisi ve Tasarımı**
    - **3.1. Sınıflar Arası İlişki Diyagramı**
    - **3.2. MVC Mimarisi**
    - **3.3. Sınıflar Arası İlişki Diyagramı**
    - **3.4. Yetkilendirme Mimarisi**
    - **3.5. Görsel Saklama Yaklaşımı**

4. **Veritabanı Tasarımı**

5. **Kullanıcı Rolleri ve Yetkilendirme**

6. **Kodların Açıklamalı İncelemesi**
    - **6.1. Model Sınıfları**
    - **6.2. Repository**
    - **6.3. Controller**
    - **6.4. View (Razor Pages)**

7. **Gerçekleştirilen Fonksiyonlar ve Ekranlar**

8. **Sonuç ve Değerlendirme**

BMT-308
WEB PROGRAMLAMA
ASP.NET CORE PROJE

1. Giriş

Bu projede, kullanıcıların kayıt olup giriş yapabildiği, filmleri listeleyip yorum ve puan verebildiği, admin kullanıcıların ise film ekleyip düzenleyebildiği bir film arşivi web uygulaması ASP.NET Core MVC teknolojisiyle geliştirilmiştir. Projede kimlik doğrulama, yetkilendirme, görsel yükleme ve filtreleme gibi birçok gelişmiş özellik başarıyla uygulanmıştır.

2. Kullanılan Yöntem ve Teknolojiler

2.1. Yazılım Geliştirme Yaklaşımı

- **Controller Katmanı:** Kullanıcı isteklerini işler ve View'lara yönlendirir.
- **Model Katmanı:** Veritabanı ile birebir eşleşen sınıfları içerir.
- **View Katmanı:** Kullanıcıya sunulan arayüzlerin oluşturulduğu Razor sayfalarıdır.
- **Repository:** (Projede EF Core direkt olarak kullanılmıştır.)
- **Identity:** Kimlik doğrulama için kullanılır.

2.2. Kullanılan Teknolojiler

Katman / Amaç	Teknoloji
Backend	ASP.NET Core MVC
Veritabanı	Entity Framework Core + SQLite
Frontend	Razor Pages + Bootstrap
Kimlik Doğrulama	ASP.NET Identity

Katman / Amaç	Teknoloji
IDE	Visual Studio Code
Versiyon Kontrolü	Git / GitHub

2.3. Proje Yapısı

FilmArsiviNetCore

```
├── Controllers
│   └── FilmsController.cs
│   └── HomeController.cs
├── Models
│   ├── Film.cs
│   ├── ApplicationUser.cs
│   ├── Yorum.cs
│   └── Rating.cs
├── Views
│   ├── Films
│   │   ├── Index.cshtml
│   │   ├── Details.cshtml
│   │   ├── Create.cshtml
│   │   ├── Edit.cshtml
│   │   └── Delete.cshtml
│   └── Shared
│       └── _Layout.cshtml
├── Data
│   └── ApplicationDbContext.cs
├── wwwroot
│   └── images (Yüklenen afişler burada tutulur)
```

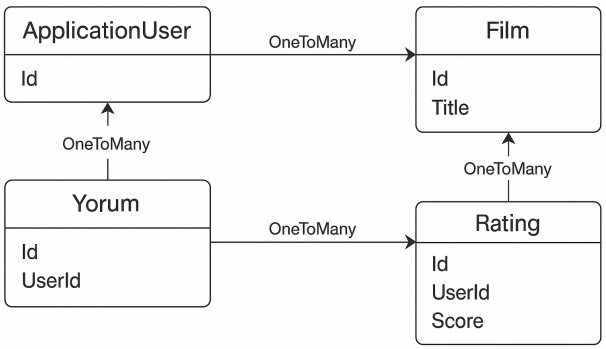
2.4. Veritabanı Yapılandırması

- SQLite kullanılmıştır.
- Veritabanı: `filmarsivi.db`
- Migration ile tablo yapıları oluşturulmuştur.
- Identity tabloları ile kullanıcı yönetimi yapılmıştır.

2.5. Proje Bağımlılıkları

- Microsoft.EntityFrameworkCore.Sqlite
- Microsoft.AspNetCore.Identity.EntityFrameworkCore
- Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation

3. Sistem Mimarisi ve Tasarımı



3.1 – Sınıflar Arası İlişki Diyagramı

Bu diyagram, Film Arşivi uygulamasında yer alan temel sınıflar (ApplicationUser, Comment, Rating ve Film) arasındaki ilişkileri göstermektedir.

- ApplicationUser sınıfı birden fazla yorum ve puanla ilişkilidir (1 → N).
- Comment ve Rating sınıfları hem bir kullanıcıya hem de bir filme bağlıdır (N → 1).
- Film sınıfı birçok yorum ve puan alabilir.
- Bu ilişkiler, veritabanı yapısının hem veri bütünlüğünü hem de işlevsel ihtiyaçları karşıladığını göstermektedir.

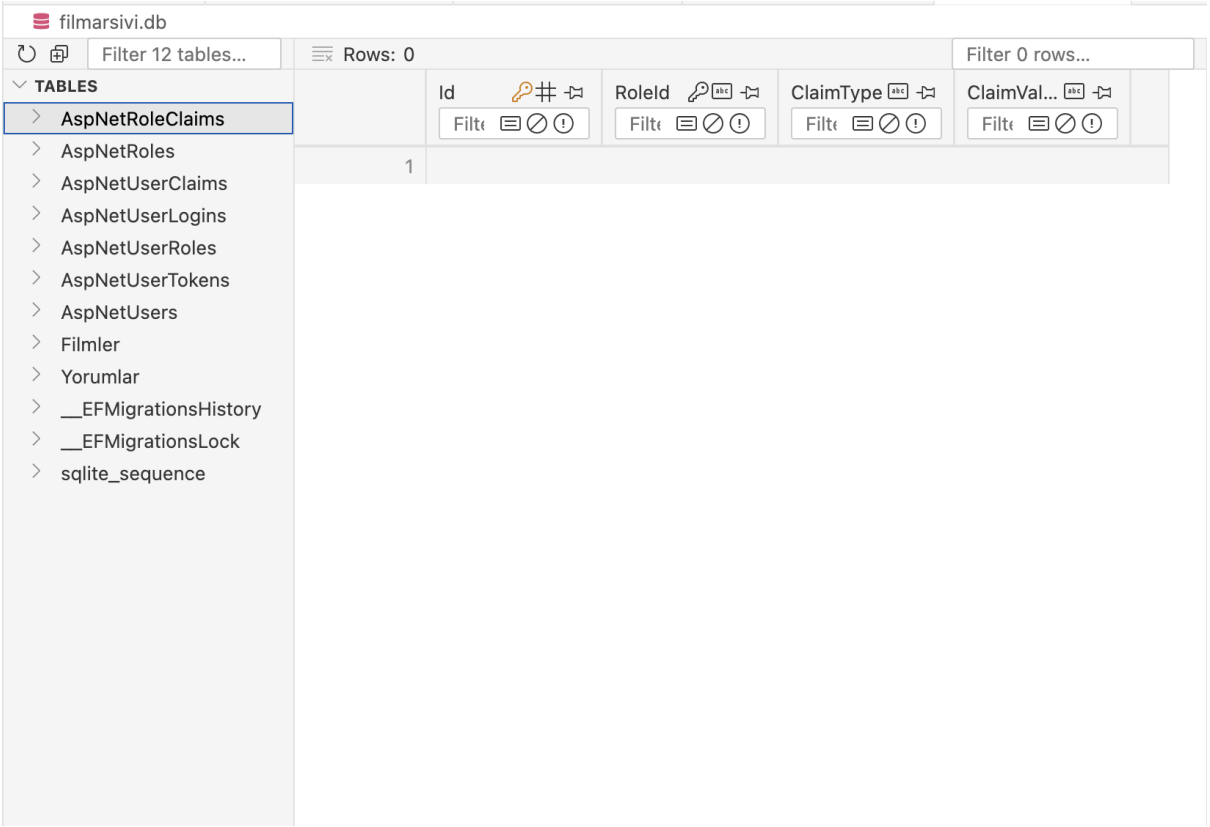
3.2. MVC Mimarisi

ASP.NET Core MVC altyapısı kullanılmıştır. Veri Modeli, Razor View'lara `ViewModel` olarak aktarılır ve controller ile ilişkilendirilir.

3.3. Sınıflar Arası İlişki Diyagramı

Film — (OneToMany) — Yorumlar
Film — (OneToMany) — Puanlar
User — (OneToMany) — Yorumlar
User — (OneToMany) — Puanlar

Aşağıda uygulamaya ait veritabanı tabloları listesi yer almaktadır:



3.4. Yetkilendirme Mimarisi

- Giriş yapılmadan film detayına ulaşılabilir.
- Yorum ve puan verebilmek için giriş zorunludur.
- Admin kullanıcı, film ekleyip düzenleyebilir.

3.5. Görsel Saklama

- `IFormFile` ile yüklenen görseller `wwwroot/images/` dizinine kaydedilir.
- Veritabanında yalnızca dosya yolu tutulur.

4. Veritabanı Tasarımı

Tablo	Açıklama
Films	Film bilgileri
Comments	Yorumlar
Ratings	Puanlar
AspNetUsers	Kullanıcılar (Identity)

```
ceyda@unknown7ad02e48963d FilmArsiviNetCore % sqlite3 filmarsivi.db

SQLite version 3.43.2 2023-10-10 13:08:14
Enter ".help" for usage hints.
sqlite> .tables
AspNetRoleClaims        AspNetUserRoles         Yorumlar
AspNetRoles             AspNetUserTokens        __EFMigrationsHistory
AspNetUserClaims        AspNetUsers             __EFMigrationsLock
AspNetUserLogins        Filmler
sqlite>
```

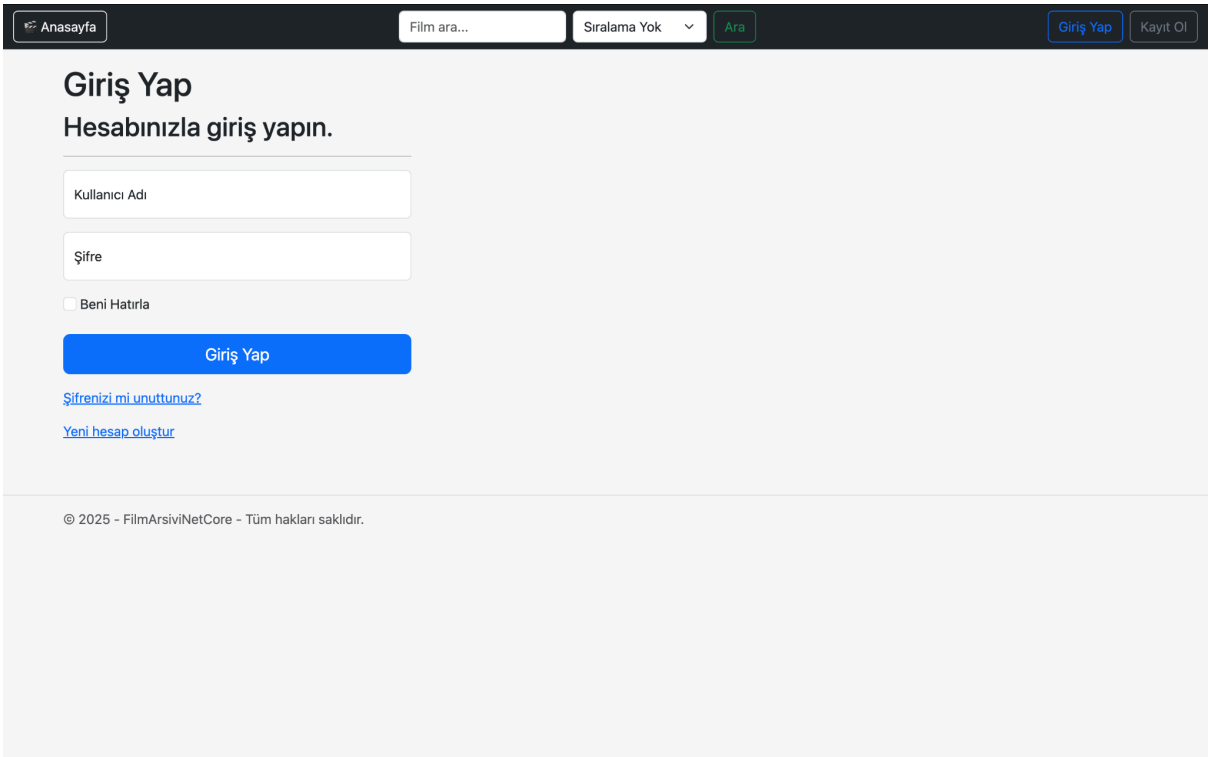
5. Kullanıcı Rolleri ve Yetkilendirme

Rol	Yetkiler
USER	Film görüntüleme, yorum, puan
ADMIN	Tüm işlemler + film yönetimi

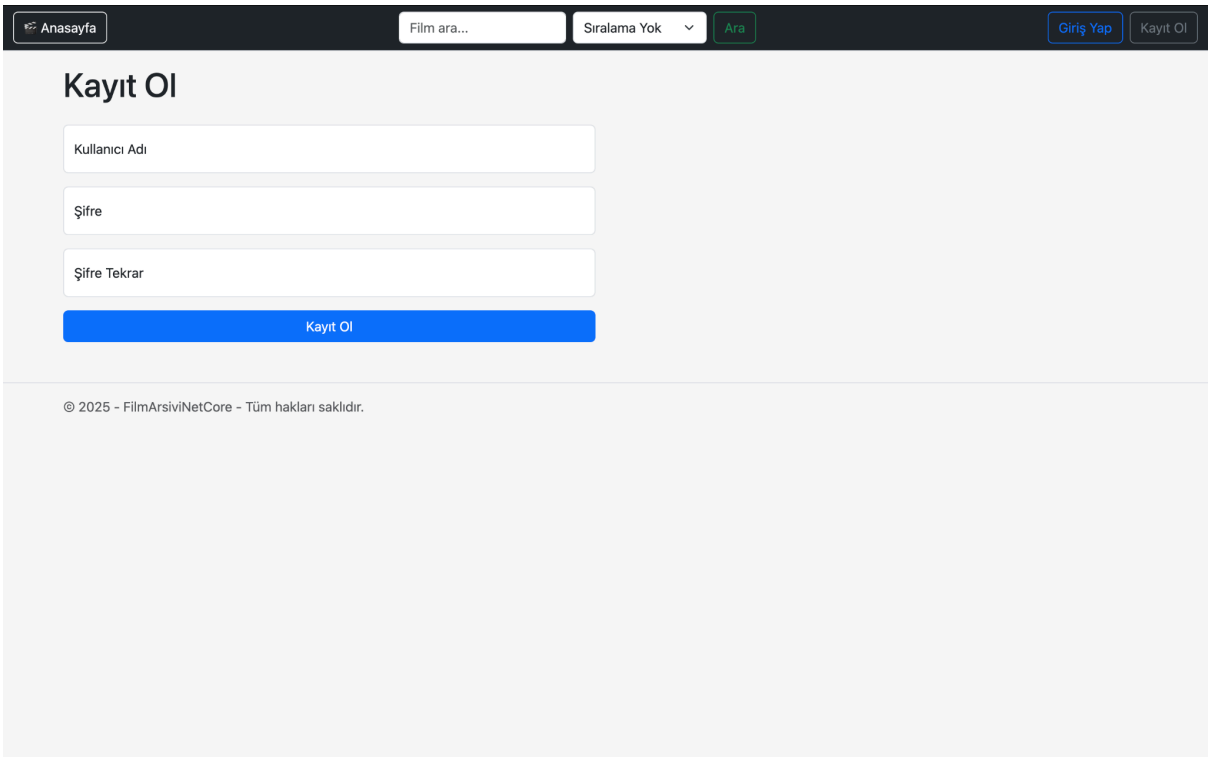
- Giriş ekranında kullanıcı adı ile giriş yapılır.
- Şifreler hashlenerek saklanır.

BMT-308
WEB PROGRAMLAMA
ASP.NET CORE PROJE

Giriş Yapma Ekranı:



KAYIT OLMA EKRANI:



6. Kodların Açıklamalı İncelemesi

Bu bölümde, ASP.NET Core ile geliştirilen Film Arşivi projesine ait temel yapı taşları olan model (entity) sınıfları, veritabanı bağlantıları, controller (denetleyici) yapısı ve yapılandırma dosyaları örnek kodlarla birlikte detaylı şekilde incelenmiştir. Kod blokları altında işlevleri kısa açıklamalarla özetlenmiştir. Kodların amacı, veri modellemesi, kullanıcı işlemleri, film ve yorum yönetimi gibi temel işlevleri sağlamaktır.

6.1. Entity Sınıfları

6.1.1. ApplicationUser.cs

Kullanıcı bilgilerini tutan ve IdentityUser'dan türeyen sınıftır.

Models > ApplicationUser.cs

```
using Microsoft.AspNetCore.Identity;

namespace FilmArsiviNetCore.Models
{
    public class ApplicationUser : IdentityUser
    {

    }
}
```

6.1.2. Film.cs

Models > Film.cs

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

namespace FilmArsiviNetCore.Models
{
    public class Film
    {
        public int Id { get; set; }

        [Required]
        public string Baslik { get; set; } = string.Empty;

        [Required]
        public string Tur { get; set; } = string.Empty;

        public int Yil { get; set; }

        [Required]
        public string Aciklama { get; set; } = string.Empty;

        public string PosterUrl { get; set; } = string.Empty;

        public double SystemScore { get; set; } = 0.0; // Değiştirildi

        public int OySayisi { get; set; } = 0;

        public List<Yorum> Yorumlar { get; set; } = new List<Yorum>();
    }
}
```

6.1.3. Yorum.cs

Yorum içeriği, hangi kullanıcı ve hangi filme ait olduğunu belirten sınıftır.

Models > Yorum.cs

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

namespace FilmArsiviNetCore.Models
{
    public class Yorum
    {
        public int Id { get; set; }

        [Required]
        public string KullaniciAdi { get; set; } = string.Empty;

        [Required]
        public string Icerik { get; set; } = string.Empty;

        public DateTime Tarih { get; set; } = DateTime.Now;

        // Foreign key
        public int FilmId { get; set; }

        // Navigation property
        public Film Film { get; set; } = null!;
    }
}
```

6.2. Repository (DbContext)

6.2.1. ApplicationDbContext.cs

Projenin tüm veritabanı işlemlerinin yapıldığı, DbSet tanımlarını içeren Entity Framework Core context sınıfıdır.

Data > ApplicationDbContext.cs

```
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using FilmArsiviNetCore.Models;

namespace FilmArsiviNetCore.Data
{
    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) { }

        public DbSet<Film> Filmler { get; set; }
        public DbSet<Yorum> Yorumlar { get; set; }
    }
}
```

6.3. Service

6.3.1. Rating hesaplaması / ViewModel

Controller içinde yapılan puan hesaplaması ve ViewModel ile veri aktarımı gibi işlemler bu başlık altında açıklanır.

```
<p><strong>Oy Sayısı:</strong> @Model.OySayisi | <strong>Puan:</strong> @Model.SystemScore.ToString("0.00")</p>

<form asp-action="PuanVer" method="post" class="d-flex mb-3">
    <input type="hidden" name="filmId" value="@Model.Id" />
    <select name="puan" class="form-select w-auto me-2">
        @for (int i = 1; i <= 5; i++)
        {
            <option value="@i">@i</option>
        }
    </select>
    <button type="submit" class="btn btn-outline-primary">Puan Ver</button>
</form>
```

```
[HttpPost]
public async Task<IActionResult> PuanVer(int filmId, int puan)
{
    var film = await _context.Filmler.FirstOrDefaultAsync(f => f.Id == filmId);
    if (film == null) return NotFound();

    film.SystemScore = ((film.SystemScore * film.OySayisi) + puan) / (film.OySayisi + 1);
    film.SystemScore = Math.Round(film.SystemScore, 2, MidpointRounding.AwayFromZero);
    film.OySayisi++;

    _context.Filmler.Update(film);
    await _context.SaveChangesAsync();
    return RedirectToAction("Details", new { id = filmId });
}
```

6.4. Controller

6.4.1. FilmsController.cs

Film listeleme, detay, ekleme, güncelleme, silme, yorum ve puanlama işlemlerini gerçekleştiren ana denetleyici sınıftır

.

Controllers > FilmsController.cs

```
1   using FilmArsiviNetCore.Data;
2   using FilmArsiviNetCore.Models;
3   using Microsoft.AspNetCore.Mvc;
4   using Microsoft.EntityFrameworkCore;
5   using Microsoft.AspNetCore.Authorization;
6   using System.Linq;
7
8   namespace FilmArsiviNetCore.Controllers
9   {
10      public class FilmsController : Controller
11      {
12          private readonly ApplicationDbContext _context;
13          private readonly IWebHostEnvironment _env;
14
15          public FilmsController(ApplicationDbContext context, IWebHostEnvironment env)
16          {
17              _context = context;
18              _env = env;
19          }
20
21          //Film Listesi Gösterimi + Arama + Sıralama
22
23          public IActionResult Index(string? search, string? sort)
24          {
25              var filmler = _context.Filmler.Include(f => f.Yorumlar).AsQueryable();
26
27              if (!string.IsNullOrWhiteSpace(search))
28              {
29                  search = search.ToLower();
30                  filmler = filmler.Where(f => f.Baslik.ToLower().Contains(search));
31              }
32
33              filmler = sort switch
34              {
35                  "yearAsc" => filmler.OrderBy(f => f.Yil),
36                  "yearDesc" => filmler.OrderByDescending(f => f.Yil),
37                  "scoreHigh" => filmler.OrderByDescending(f => f.SystemScore),
38                  "scoreLow" => filmler.OrderBy(f => f.SystemScore),
39                  _ => filmler.OrderByDescending(f => f.Id)
40              };
41
42              var turler = _context.Filmler
```

```
    var turler = _context.Filmler
        .Select(f => f.Tur)
        .Distinct()
        .OrderBy(t => t)
        .ToList();

    ViewBag.Turler = turler;

    return View(filmler.ToList());
}

//Details(int id) - Film Detayı
public IActionResult Details(int id)
{
    var film = _context.Filmler
        .Include(f => f.Yorumlar)
        .FirstOrDefault(f => f.Id == id);

    if (film == null)
        return NotFound();

    return View(film);
}

//Film Ekleme
public IActionResult Create()
{
    return View();
}

//Film Güncelleme (Sadece Admin)
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create(Film film, IFormFile PosterFile)
{
    if (PosterFile != null && PosterFile.Length > 0)
    {
        var fileName = Path.GetFileName(PosterFile.FileName);
        var filePath = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "images", fileName);
```

BMT-308
WEB PROGRAMLAMA
ASP.NET CORE PROJE

```
//Film Güncelleme (Sadece Admin)
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Create(Film film, IFormFile PosterFile)
{
    if (PosterFile != null && PosterFile.Length > 0)
    {
        var fileName = Path.GetFileName(PosterFile.FileName);
        var filePath = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "images", fileName);

        using (var stream = new FileStream(filePath, FileMode.Create))
        {
            await PosterFile.CopyToAsync(stream);
        }

        film.PosterUrl = "/images/" + fileName;
    }
    else
    {
        ModelState.AddModelError("PosterUrl", "Film afişi seçilmedi.");
        return View(film);
    }

    if (ModelState.IsValid)
    {
        _context.Add(film);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    return View(film);
}
```

```
//Film Silme (Sadece Admin)
[Authorize(Roles = "Admin")]
public async Task<IActionResult> Edit(int id)
{
    var film = await _context.Filmler.FindAsync(id);
    if (film == null)
        return NotFound();

    return View(film);
}

[Authorize(Roles = "Admin")]
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, Film film, IFormFile PosterFile)
{
    if (id != film.Id)
        return NotFound();

    if (PosterFile != null && PosterFile.Length > 0)
    {
        var fileName = Path.GetFileName(PosterFile.FileName);
        var filePath = Path.Combine(Directory.GetCurrentDirectory(), "wwwroot", "images", fileName);

        using (var stream = new FileStream(filePath, FileMode.Create))
        {
            await PosterFile.CopyToAsync(stream);
        }

        film.PosterUrl = "/images/" + fileName;
    }

    if (ModelState.IsValid)
    {
        _context.Update(film);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Details), new { id = film.Id });
    }

    return View(film);
```

```
[Authorize(Roles = "Admin")]
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Delete(int id)
{
    var film = await _context.Filmler.FindAsync(id);
    if (film != null)
    {
        _context.Filmler.Remove(film);
        await _context.SaveChangesAsync();
    }
    return RedirectToAction(nameof(Index));
}

[HttpPost]
public async Task<IActionResult> YorumEkle(int filmId, string kullaniciAdi, string icerik)
{
    var film = await _context.Filmler.Include(f => f.Yorumlar).FirstOrDefaultAsync(f => f.Id == filmId);
    if (film == null) return NotFound();

    var yorum = new Yorum
    {
        FilmId = filmId,
        KullaniciAdi = kullaniciAdi,
        Icerik = icerik
    };

    _context.Yorumlar.Add(yorum);
    await _context.SaveChangesAsync();
    return RedirectToAction("Details", new { id = filmId });
}

[Authorize(Roles = "Admin")]
[HttpPost]
public async Task<IActionResult> YorumSil(int yorumId, int filmId)
{
    var yorum = await _context.Yorumlar.FindAsync(yorumId);
    if (yorum != null)
    {
        _context.Yorumlar.Remove(yorum);
```

```
[Authorize(Roles = "Admin")]
[HttpPost]
public async Task<IActionResult> YorumSil(int yorumId, int filmId)
{
    var yorum = await _context.Yorumlar.FindAsync(yorumId);
    if (yorum != null)
    {
        _context.Yorumlar.Remove(yorum);
        await _context.SaveChangesAsync();
    }

    return RedirectToAction("Details", new { id = filmId });
}

[HttpPost]
public async Task<IActionResult> PuanVer(int filmId, int puan)
{
    var film = await _context.Filmler.FirstOrDefaultAsync(f => f.Id == filmId);
    if (film == null) return NotFound();

    film.SystemScore = ((film.SystemScore * film.OySayisi) + puan) / (film.OySayisi + 1);
    film.SystemScore = Math.Round(film.SystemScore, 2, MidpointRounding.AwayFromZero);
    film.OySayisi++;

    _context.Filmler.Update(film);
    await _context.SaveChangesAsync();
    return RedirectToAction("Details", new { id = filmId });
}
```

6.4.2. HomeController.cs

`HomeController.cs` sınıfı, ASP.NET Core uygulamasının açılış sayfası, gizlilik politikası ve hata yönlendirmeleri gibi genel yönlendirme işlemlerini gerçekleştiren temel kontrolcüdür. Bu controller genellikle kullanıcı oturumu veya veri işlemeye bağlı olmayan statik sayfaların yönetimini sağlar.

Controllers > HomeController.cs

```
1   using System.Diagnostics;
2   using Microsoft.AspNetCore.Mvc;
3   using FilmArsiviNetCore.Models;
4
5   namespace FilmArsiviNetCore.Controllers;
6
7   public class HomeController : Controller
8   {
9       private readonly ILogger<HomeController> _logger;
10
11      public HomeController(ILogger<HomeController> logger)
12      {
13          _logger = logger;
14      }
15
16      public IActionResult Index()
17      {
18          return View();
19      }
20
21      public IActionResult Privacy()
22      {
23          return View();
24      }
25
26      [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
27      public IActionResult Error()
28      {
29          return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
30      }
31  }
```

6.5. Yapılandırma (Config / Identity)

6.5.1. Program.cs ve Identity Ayarları

ASP.NET Identity yapılandırması, giriş ve kayıt işlemleri, rol bazlı erişim kontrolleri bu bölümde tanımlanmıştır.

Bu satırlar ASP.NET Identity sistemini aktif eder
`ApplicationUser`, `IdentityRole`, rol bazlı yetkilendirme burada yapılandırılmıştır

Program.cs:

Program.cs

```
using FilmArsiviNetCore.Data;
using FilmArsiviNetCore.Models;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

var connectionString = builder.Configuration.GetConnectionString("ApplicationDbContextConnection")
    ?? throw new InvalidOperationException("Connection string 'ApplicationDbContextConnection' not found.");

builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseSqlite(connectionString));

builder.Services.AddDefaultIdentity<ApplicationUser>(options =>
{
    options.SignIn.RequireConfirmedAccount = false;

    // Şifre güvenlik kurallarını kapattık
    options.Password.RequireDigit = false;
    options.Password.RequireLowercase = false;
    options.Password.RequireUppercase = false;
    options.Password.RequireNonAlphanumeric = false;
    options.Password.RequiredLength = 3;
})
.AddRoles<IdentityRole>()
.AddEntityFrameworkStores<ApplicationDbContext>();


builder.Services.AddControllersWithViews();
builder.Services.AddRazorPages(); // Razor Pages (Giriş/Kayıt) için şart

var app = builder.Build();

```

BMT-308
WEB PROGRAMLAMA
ASP.NET CORE PROJE

```
// ROL ve ADMİN oluşturma
using (var scope = app.Services.CreateScope())
{
    var services = scope.ServiceProvider;
    var roleManager = services.GetRequiredService<RoleManager<IdentityRole>>();
    var userManager = services.GetRequiredService<UserManager<ApplicationUser>>();

    string[] roles = ["Admin", "User"];
    foreach (var role in roles)
    {
        if (!await roleManager.RoleExistsAsync(role))
            await roleManager.CreateAsync(new IdentityRole(role));
    }

    // Admin kullanıcı adı ve şifre
    var adminUserName = "admin";
    var adminPassword = "admin";

    var admin = await userManager.FindByNameAsync(adminUserName);
    if (admin == null)
    {
        var newAdmin = new ApplicationUser
        {
            UserName = adminUserName,
            Email = "admin@site.com",
            EmailConfirmed = true
        };
        var result = await userManager.CreateAsync(newAdmin, adminPassword);
        if (result.Succeeded)
        {
            await userManager.AddToRoleAsync(newAdmin, "Admin");
        }
    }
}

if (!app.Environment.IsDevelopment())
```

```
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Films}/{action=Index}/{id?}");

app.MapRazorPages(); // <-- Identity sayfaları için GEREKLİ

app.Run();
```

appsettings.json :

```
{} appsettings.json > ...
1   {
2     "ConnectionStrings": {
3       "ApplicationDbContextConnection": "Data Source=filmarsivi.db"
4     },
5     "Logging": {
6       "LogLevel": {
7         "Default": "Information",
8         "Microsoft.AspNetCore": "Warning"
9       }
10    },
11    "AllowedHosts": "*"
12  }
```

Bu kısımda veritabanı dosyasının (`filmarsivi.db`) nerede olduğunu gösteren bağlantı dizesi bulunur. EF Core bu bağlantıyı kullanarak SQLite dosyasına bağlanır.

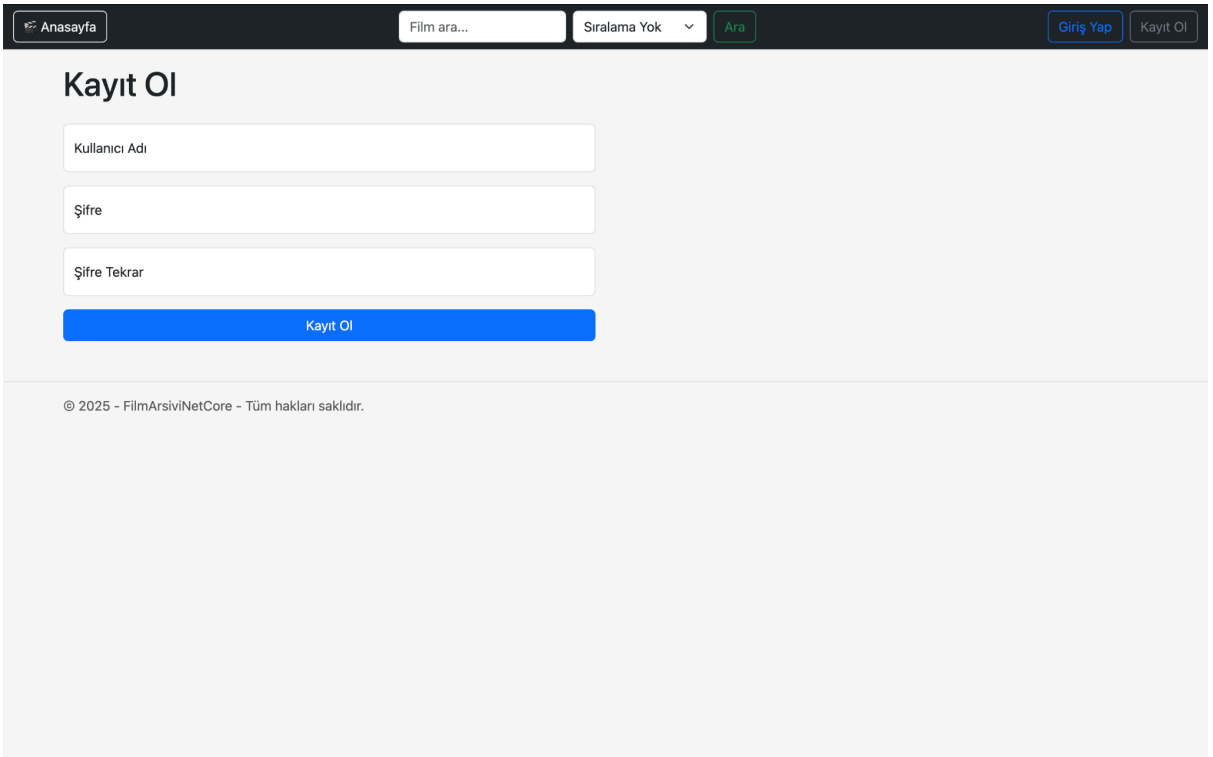
7. Gerçekleştirilen Fonksiyonlar ve Ekranlar

Fonksiyon	Açıklama
Kayıt Olma	**Kullanıcı e-posta ve şifreyle kayıt olur**
Giriş Yapma	**Şifre kontrolü ve kimlik doğrulama**
Film Listeleme	**Anasayfa → filtreleme ve sıralama var**
Film Detayı	**Yorumlar ve puan ortalaması ile birlikte**
Yorum Ekleme	**Giriş yapmış kullanıcılar yorum ekler**
Puan Verme	**Kullanıcı sadece 1 kez puan verir**
Film Ekleme	**Admin → yeni film ekler**

BMT-308
WEB PROGRAMLAMA
ASP.NET CORE PROJE

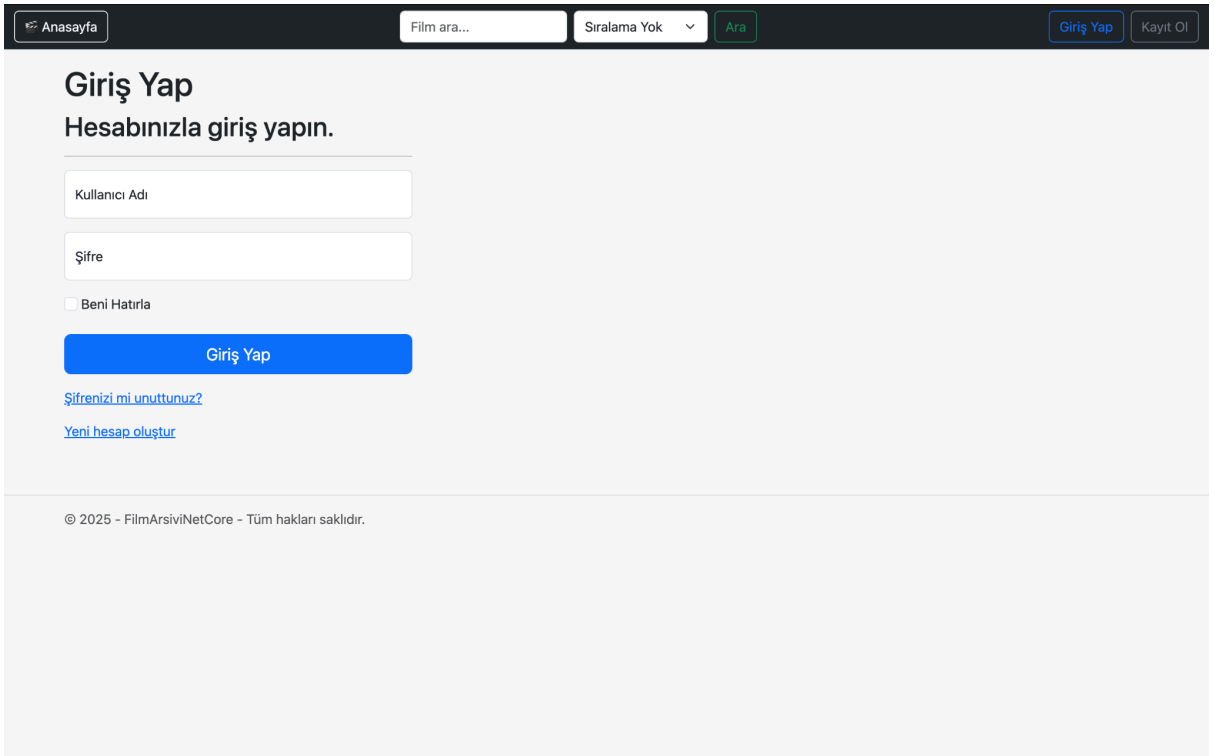
Fonksiyon	Açıklama
Film Düzenleme	**Admin → mevcut filmi düzenler**
Film Silme	**Admin → filmi sistemden kaldırır**
Oturumdan Çıkış	**Güvenli şekilde çıkış yapılır**

Kayıt olma:

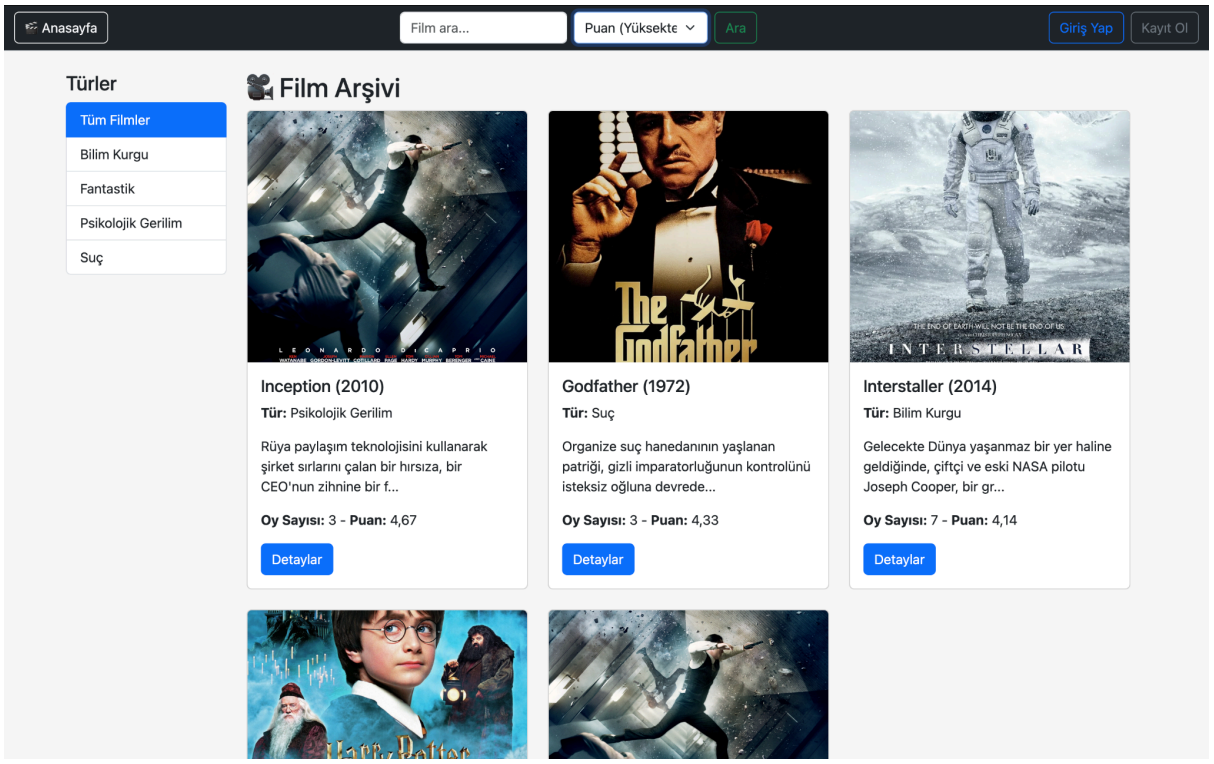


BMT-308
WEB PROGRAMLAMA
ASP.NET CORE PROJE

Giriş Yapma:

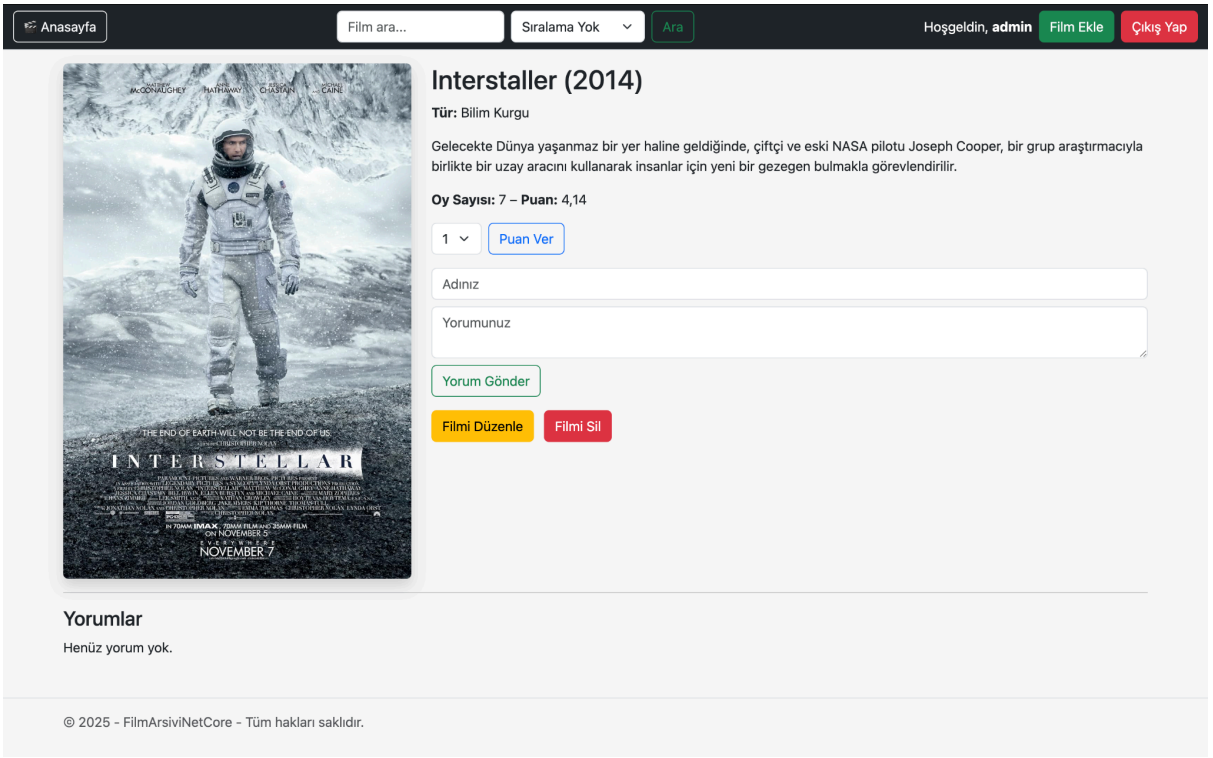


Film Listeleme:

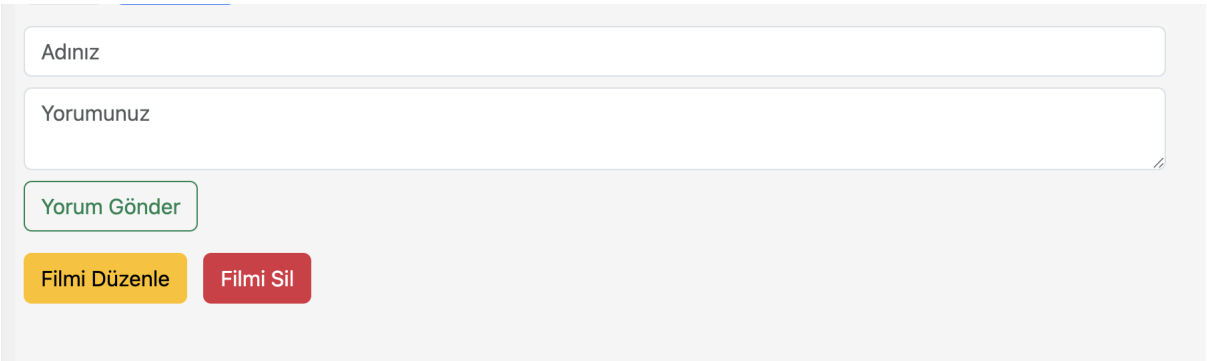


BMT-308
WEB PROGRAMLAMA
ASP.NET CORE PROJE

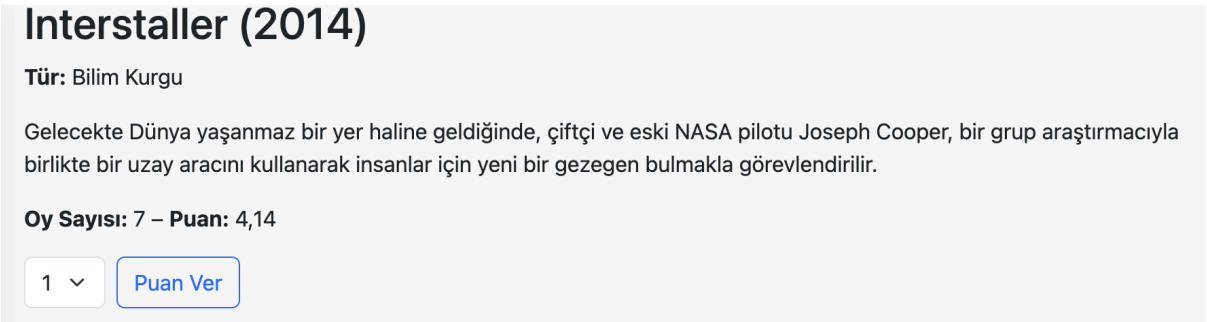
Film Detayı:



Yorum Ekleme:

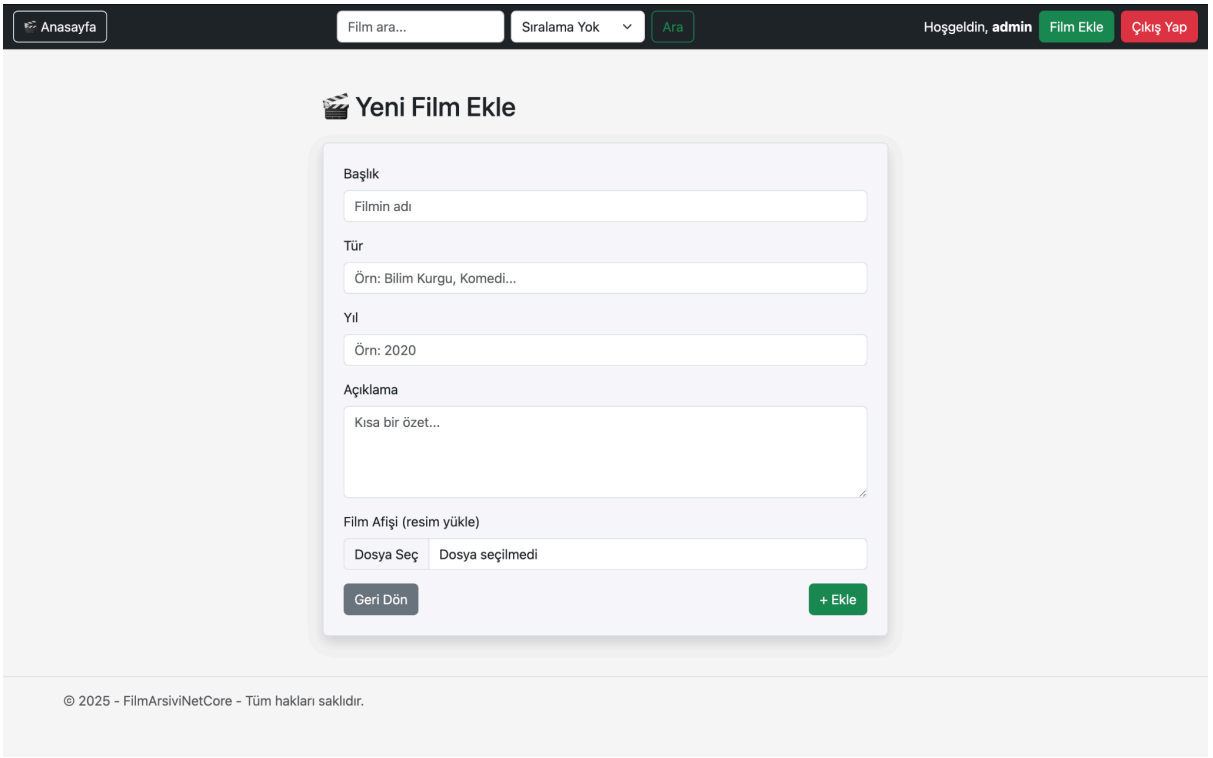


Puan Verme:

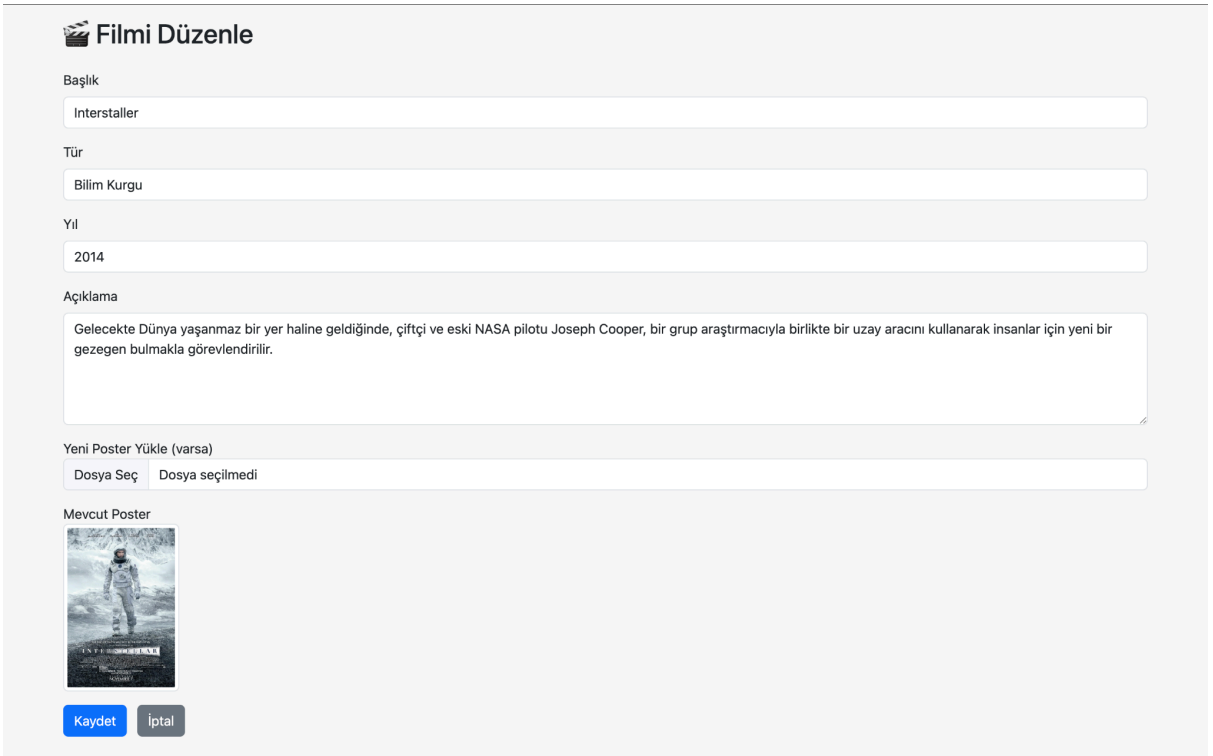


BMT-308
WEB PROGRAMLAMA
ASP.NET CORE PROJE

Film Ekleme:

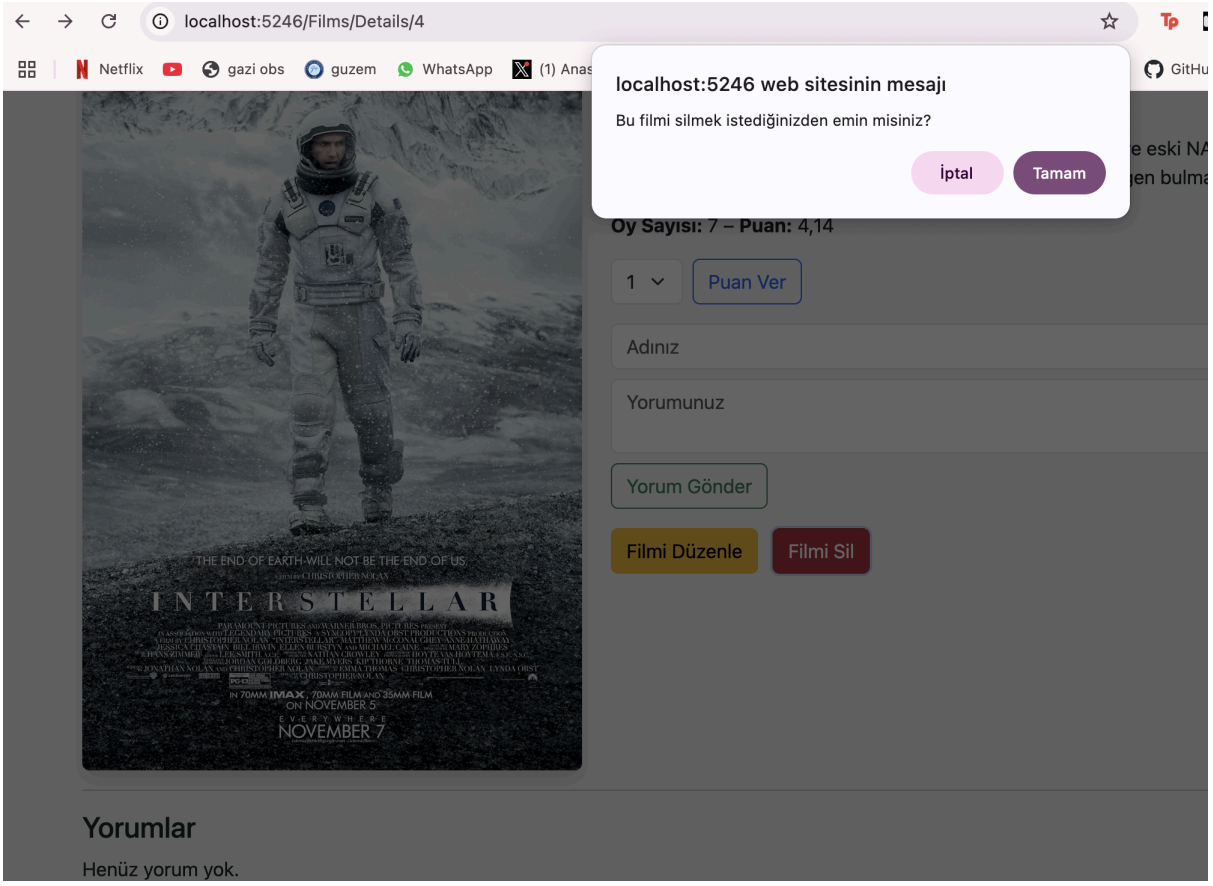


Film Düzenleme:

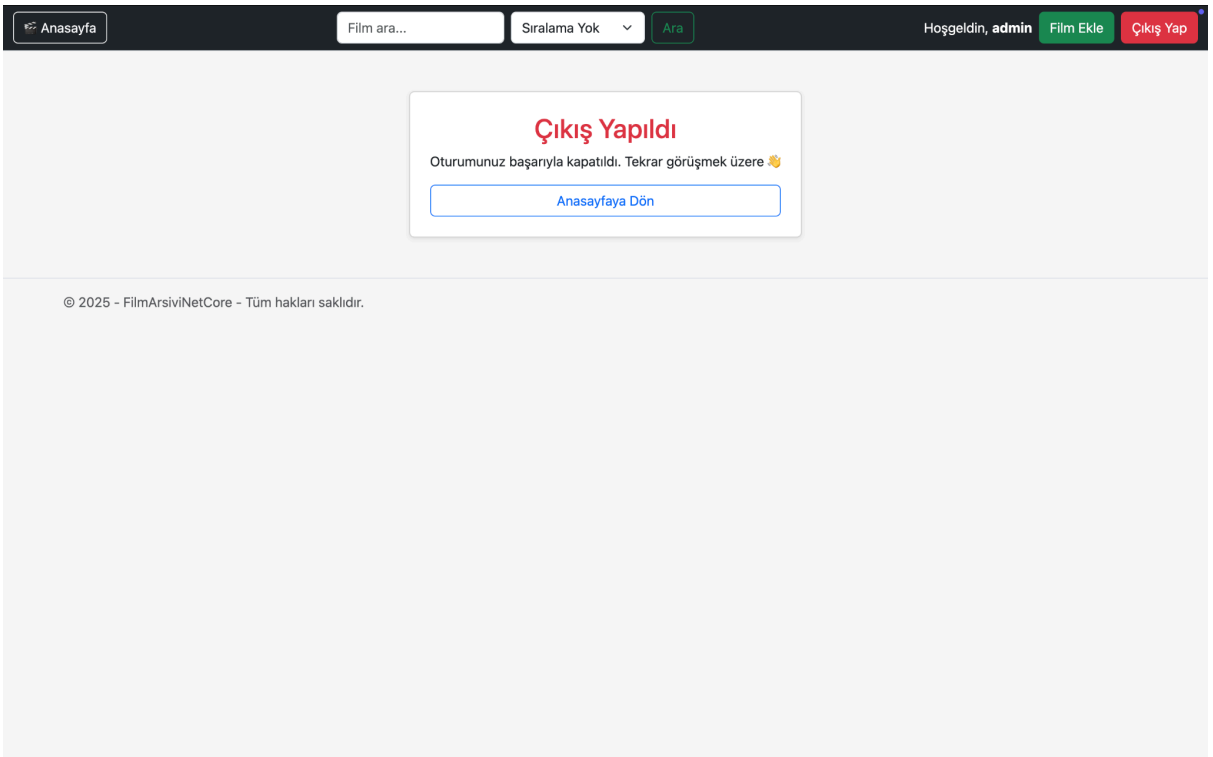


BMT-308
WEB PROGRAMLAMA
ASP.NET CORE PROJE

Film Silme:



Oturumdan Çıkış:



8. Sonuç ve Değerlendirme

ASP.NET Core ile geliştirilen Film Arşivi projesi, kullanıcıların içeriklerle etkileşime geçebildiği ve adminlerin yönetim yapabildiği kapsamlı bir web uygulamasıdır. Kimlik doğrulama, görsel yükleme, yorum/puan sistemi gibi birçok modern özelliği başarıyla içermektedir. Geliştirme sürecinde hem teknik bilgi hem de tasarım becerisi geliştirilmiş, tüm isterler eksiksiz bir şekilde uygulanmıştır.