ISPARTA
UYGULAMALI BİLİMLER
ÜNİVERSİTESİ

# STAJ FAALİYET RAPORU

| TEKNOLOJİ FAKÜLTESİ/YO/MYO | |
|---|---|
| **STAJ YAPANIN** | |
| **Bölümü** | : Bilgisayar Mühendisliği |
| **Adı ve Soyadı** | : Yusuf Aras |
| **Okul Numarası** | : 2112729013 |
| **Staj Konusu** | : Yazılım |

# ISPARTA UYGULAMALI BİLİMLER ÜNİVERSİTESİ
## TEKNOLOJİ FAKÜLTESİ/YO/MYO

**Adı ve Soyadı** **:Yusuf Aras**

**Okul Numarası** **: 2112729013**

**Okuduğu Yarıyıl** **:**

**Staj Konusu** **: Yazılım**

**Staj Yaptığı Kurum** **: UNNAB İNNOVATİON TEKNOLOJİ PROJE SAN.TİC.AŞ**

**İşyeri Telefonu** **: 05302203505**

**Öğrenci Telefonu** **: 05378761621**

| Kurumun Adı | Staj Konusu | Çalışma Süresi | | İş günü |
|---|---|---|---|---|
| | | Tarihinden | Tarihine | |
| UNNAB İNNOVATİON TEKNOLOJİ PROJE SAN.TİC.AŞ | YAZILIM | 14.07.2025 | 29.08.2025 | 34 |

**İş yerinde kontrol edenin**
**Kaşe ve imza**

**Staj Komisyonu Üyesi** **Staj Komisyonu Üyesi** **Staj Komisyonu Başkanı**

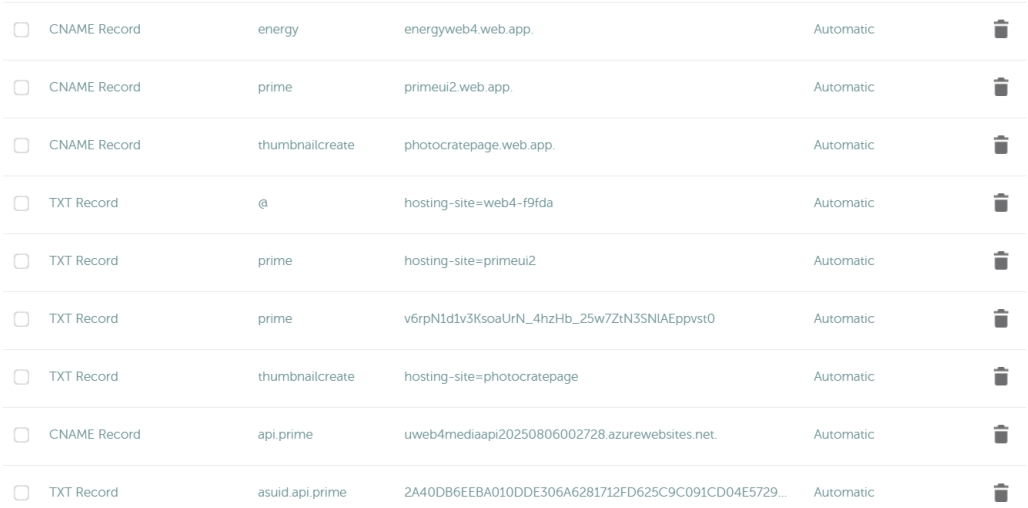
# PRATİK ÇALIŞMANIN GÜNLERE GÖRE DAĞITIM CETVELİ

| Tarih | Öğrencinin Çalıştığı Konular | Sayfa No |
|---|---|---|
| 14.07.2025 | Domain ve Hosting Yönlendirme, UI Tasarımına Başlangıç | 1 |
| 15.07.2025 | Resmi Tatil | 2 |
| 16.07.2025 | Veri Tabanı Entity Planlaması ve Tanımlamaları | 3 |
| 17.07.2025 | CQRS ve MediatR ile Kullanıcı Oluşturma Komutunun Kurulması | 4 |
| 18.07.2025 | Kampanya, Şirket ve Kanal Oluşturma Komutları, Controller ve Swagger Testleri | 5 |
| 21.07.2025 | Google ve GitHub ile OAuth Giriş (Login) Entegrasyonu | 6 |
| 22.07.2025 | Microsoft ile Giriş (OAuth) ve Azure Blob Storage ile Fotoğraf Yükleme Entegrasyonu | 7 |
| 23.07.2025 | Video İçerikleri için API Controller Oluşturulması | 8 |
| 24.07.2025 | Beğeni (Like), Yorum ve Takip (Subscribe) İçin CQRS ve API Controllerlarının Oluşturulması | 9 |
| 25.07.2025 | Sektör, Kanal ve Plan Ekleme-Güncelleme için CQRS Controllerları ve Yetkilendirme Kısıtlamaları | 10 |
| 28.07.2025 | Stripe ile Ödeme Entegrasyonu, Hesap Açılışı ve Altyapı Hazırlığı | 11 |
| 29.07.2025 | Stripe Ödeme API Entegrasyonlarının Tamamlanması ve Test Edilmesi | 12 |
| 30.07.2025 | Controllerlarda Yetkilendirme (Authentication & Authorization) Düzenlemeleri | 13 |
| 31.07.2025 | Admin Paneli UI Entegrasyonu ve API Bağlantısı | 14 |
| 01.07.2025 | Admin Panelinde Video Yükleme, Güncelleme, Silme Fonksiyonlarının Geliştirilmesi ve Tabloyla Listeleme | 15 |
| 04.08.2025 | Admin Panelinde Firma Yönetimi Modülünün Geliştirilmesi, Azure Logo Yükleme Entegrasyonu | 16 |
| 05.08.2025 | Yorum Yönetimi ve Sektör/Kanal Ekleme Sayfalarının Geliştirilmesi, Admin Yetkilendirme ile Güvenli Erişim | 17 |
| 06.08.2025 | Kullanıcı Yönetimi ve Abone Kullanıcı Sayfalarında Listeleme ve Güncelleme Fonksiyonlarının Geliştirilmesi | 18 |
| 07.08.2025 | Plan ve Token Planı Ekleme Fonksiyonlarının Geliştirilmesi | 19 |
| 08.08.2025 | Yapay Zeka Destekli Teknik Destek Modülü Entegrasyonu | 20 |
| 11.08.2025 | Prime.uweb4.com Login Paneli Entegrasyonu ve Hata Düzeltmeleri | 21 |
| 12.08.2025 | Google ve Microsoft ile Giriş (OAuth) Entegrasyonu, Backend ve Frontend Düzenlemeleri | 22 |
| 13.08.2025 | Kullanıcı Kayıt Sonrası E-Posta Doğrulama ve Frontend Entegrasyonu | 23 |
| 14.08.2025 | Kullanıcı Sidebar Geliştirme | 24 |
| 15.08.2025 | Kullanıcı Profil Sayfası Geliştirme | 25 |

| | | |
|---|---|---|
| 18.08.2025 | Planların Sıralanması ve Seçim Fonksiyonlarının Backend ile Entegrasyonu | 26 |
| 19.08.2025 | Abonelik Yönetimi, Faturalandırma ve Planlar Ekranının Tamamlanması | 27 |
| 20.08.2025 | Ödeme Öncesi E-Posta Doğrulama ve Kod ile Güvenli Onay Akışı | 28 |
| 21.08.2025 | Stripe ile Güvenli Ödeme Entegrasyonu ve Seçili Planların Ödeme Akışına Aktarılması | 29 |
| 22.08.2025 | Subscription Sayfasında Veri Akışının ve Kayıtlarının Düzeltilmesi | 30 |
| 25.08.2025 | Video Oluşturma Sayfası ve Token ile İçerik Üretimi Akışı | 31 |
| 26.08.2025 | Kullanıcı Postları için Thumbnail Tasarım Sayfası Geliştirme | 32 |
| 27.08.2025 | Ana Sayfa Tasarımı ve Postların Kartlar Halinde Gösterilmesi | 33 |
| 28.08.2025 | Postlar için Detay Sayfası Tasarımı ve Gerçek Veritabanı Entegrasyonu | 34 |
| 29.08.2025 | Arama Alanı (Search Bar) ve Sonuçların Dinamik Gösterimi | 35 |
| Toplam İş Günü | İş Yeri Amirinin Onayı | Öğrenci İmzası |
| 34 | | |

| Yapılan İş: Domain ve Hosting Yönlendirme, UI Tasarımına Başlangıç | Sayfa No:1 |
|---|---|
| | Tarih: 14.07.2025 |

Stajımın birinci gününde, daha önce bu şirkette çalışmış olmam nedeniyle herhangi bir tanışma, iş anlatımı veya oryantasyon süreci yaşanmadı. Güne, mevcut projelerimizde kullandığımız domainlerin (prime.uweb4.com ve admin.prime.uweb4.com) Azure ve Firebase üzerindeki hostinglerine doğru yönlendirme işlemlerini gerçekleştirerek başladım. Domainlerin doğru şekilde yönlendirilmesi için DNS ve hosting ayarlarını detaylıca kontrol ettim ve gerekli güncellemeleri uyguladım. Şekil 1' de de Namecheap üzerindeki DNS ayarları gözükmektedir.



| | | | |
|---|---|---|---|
| CNAME Record | energy | energyweb4.web.app. | Automatic |
| CNAME Record | prime | primeui2.web.app. | Automatic |
| CNAME Record | thumbnailcreate | photocratepage.web.app. | Automatic |
| TXT Record | @ | hosting-site=web4-f9fda | Automatic |
| TXT Record | prime | hosting-site=primeui2 | Automatic |
| TXT Record | prime | v6rpN1d1v3KsoaUrN_4hzHb_25w7ZtN3SNlAEppvst0 | Automatic |
| TXT Record | thumbnailcreate | hosting-site=photocratepage | Automatic |
| CNAME Record | api.prime | uweb4mediaapi20250806002728.azurewebsites.net. | Automatic |
| TXT Record | asuid.api.prime | 2A40DB6EEBA010DDE306A6281712FD625C9C091CD04E5729... | Automatic |

Şekil 1. Namecheap DNS Yönlendirmeleri

Yönlendirme işlemlerinin ardından, prime.uweb4.com projesinin kullanıcı arayüzü (UI) tasarımı için ön hazırlıklara başladım. Projenin ihtiyaçlarını ve genel konseptini göz önünde bulundurarak ilk taslaklar üzerinde çalıştım. Renk paleti ve temel bileşenlerin yerleşimiyle ilgili fikirlerimi dijital ortamda somutlaştırmaya başladım.

Günün sonunda, domain ve hosting yönlendirmelerinin başarılı şekilde tamamlandığını ve prime.uweb4.com'un yeni arayüz tasarımına yönelik ilk adımların atıldığını raporladım. Bugünkü çalışmalarım tamamen teknik uygulama ve tasarım odaklı geçti.

KONTROL:

| Yapılan İş: Resmi Tatil | Sayfa No:2 |
| --- | --- |
| | Tarih: 15.07.2025 |

Resmi Tatil

KONTROL:

| Yapılan İş:Veri Tabanı Entity Planlaması ve Tanımlamaları | Sayfa No:3 |
| --- | --- |
| | Tarih: 16.07.2025 |

Stajımın üçüncü gününde, yeni projenin veri tabanı tasarımına yönelik entity planlaması ve tanımlamalarını gerçekleştirdim. Öncelikle, uygulamanın gereksinimlerine göre ana tabloları ve aralarındaki ilişkileri belirledim. Bu kapsamda, kullanıcı rolleri, kullanıcılar, içerikler, yorumlar, beğeniler, bildirimler, ödemeler ve abonelikler gibi temel alanlar için entity sınıflarını oluşturdum.

Örneğin:
- AppRole: Kullanıcı rollerini temsil eden bu entity, sistemdeki her bir rol için bir kayıt barındırır. Her rolün benzersiz bir ID'si ve adı bulunur. Ayrıca, bu role sahip kullanıcıların listesini tutar.
- AppUser: Sistemdeki kullanıcıları temsil eden bu entity, kullanıcıya ait temel bilgileri (kullanıcı adı, şifre, ad, soyad, avatar, e-posta vb.) ve abonelik, doğrulama, rol ve üçüncü parti kimlikler gibi alanları içerir. Ayrıca, kullanıcı ile ilişkili içerikler, yorumlar, beğeniler, bildirimler, kampanyalar ve şirketler gibi birçok koleksiyonel ilişkiyi de tutar. Kullanıcıların rolü, AppRole entity'si ile ilişkilendirilmiştir (AppRoleID).
- RolesType Enum: Uygulamada kullanılacak rol tiplerini sabitler halinde tanımlar (örn: Admin, Member, İçerik Yöneticisi gibi). Bu sayede, rollerin sistem genelinde tutarlı şekilde kullanılmasını sağlar.

Entity dosyalarını organize ederken, projenin klasör yapısında (Admin, StripePayment, Enums gibi) mantıksal bir ayrım gerçekleştirdim. Böylece, örneğin kampanya, şirket yönetimi, kanal, sektör ve video gibi yönetimsel entityler `Admin` klasöründe, ödemelerle ilgili yapılar ise `StripePayment` altında tutuldu.

Günün sonunda, veri tabanı modellemesinin temel taşlarını oluşturarak, uygulamanın hem işlevsel hem de ilişkisel açıdan doğru bir şekilde kurgulanmasını sağladım. Bu süreç, ilerleyen aşamalarda hem yazılım geliştirme sürecinde kolaylık, hem de bakım ve genişletilebilirlik açısından büyük avantaj sağlayacaktır.

KONTROL:

| Yapılan İş: CQRS ve MediatR ile Kullanıcı Oluşturma Komutunun Kurulması | Sayfa No:4 |
|---|---|
| | Tarih: 17.07.2025 |

Stajımın dördüncü gününde, uygulamanın Application katmanında CQRS ve MediatR yapısını kurmaya başladım. Projenin sürdürülebilirliği ve geliştirilebilirliğini artırmak için katmanlı ve ayrık bir mimari tercih ettim. Bu kapsamda, görseldeki gibi Features klasörü altında CQRS ve Mediator yapıları için ayrı ayrı Commands, Handlers, Queries ve Results klasörleri oluşturdum.

Özellikle kullanıcı işlemleri için, MediatR kütüphanesini kullanarak kullanıcı oluşturma (Create) komutunu hazırladım ve ilgili handler'ı yazdım. Handler içerisinde, yeni bir kullanıcı oluşturulmadan önce aynı kullanıcı adı veya e-posta ile daha önce kayıt yapılmamış olmasına dikkat ettim. E-posta formatının doğruluğunu kontrol ettim. Girilen şifreyi BCrypt ile güvenli bir şekilde hash'leyerek veritabanına kaydettim. Ayrıca, kullanıcıya özel bir e-posta doğrulama kodu oluşturarak, kayıt sonrası kullanıcıya e-posta ile gönderdim.

CQRS ve MediatR kullanımı sayesinde, komut ve sorgu işlemlerinin mantıksal olarak ayrışmasını sağladım ve kodun okunabilirliğini artırdım. Handler, repository üzerinden veri tabanı işlemlerini yönetti ve dış servis olarak e-posta servisiyle entegre çalıştı.

Günün sonunda, hem mantıksal klasör yapısını hem de kullanıcı oluşturma işlemini başarıyla MediatR üzerinden tamamlamış oldum.

```
public class CreateAppUserCommandHandler : IRequestHandler<CreateAppUserCommand>
{
    private readonly IRepository<AppUser> _repository;
    private readonly IEmailService _emailService;

    0 references
    public CreateAppUserCommandHandler(IRepository<AppUser> repository, IEmailService emailService)
    {
        _repository = repository;
        _emailService = emailService;
    }

    0 references
    public async Task Handle(CreateAppUserCommand request, CancellationToken cancellationToken)
    {
        var existingUser = await _repository.GetByFilterAsync(
            x => x.Username == request.Username || x.Email == request.Email);

        if (existingUser != null)
        {
            throw new Exception("Registration cannot be made with the same username or email address.");
        }

        if (string.IsNullOrEmpty(request.Email) || !request.Email.Contains("@"))
        {
            throw new Exception("Enter a valid email address. (There must be an @ in the email)");
        }

        string passwordHash = BCrypt.Net.BCrypt.HashPassword(request.Password);
        var verificationCode = new Random().Next(100000, 999999).ToString();

        var user = new AppUser
        {
            Username = request.Username,
            Password = passwordHash,
            AppRoleID = (int)RolesType.Member,
            Name = request.Name,
            Surname = request.Surname,
            Email = request.Email,
            SubscriptionStatus = "free",
            PostToken = 0,
            EmailVerificationCode = verificationCode,
            IsEmailVerified = false
        };

        await _repository.CreateAsync(user);

        // E-posta gönder
        await _emailService.SendEmailAsync(request.Email, "Email Verification", $"Your verification code is: {verificationCode}");
    }
}
```

Şekil 3. User Create Command Handler

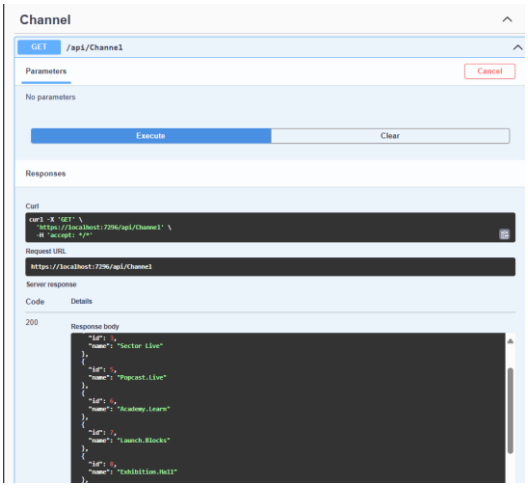
KONTROL:

| Yapılan İş: Kampanya, Şirket ve Kanal Oluşturma Komutları, Controller ve Swagger Testleri | Sayfa No:5 |
|---|---|
| | Tarih: 18.07.2025 |

Stajımın beşinci gününde, uygulamanın admin paneli için kampanya (Campaign), şirket (Company) ve kanal (Channel) ekleme işlemlerini gerçekleştirdim. CQRS mimarisi kapsamında, ilgili entity'ler için hem Command hem de Handler sınıflarını oluşturdum. Kodlamada, oluşturulan komutlar üzerinden gelen verileri alıp, uygun entity'ye dönüştürerek repository aracılığıyla veritabanına kaydedilmesini sağladım.

Örneğin;
- CreateCampaignCommandHandler: Kampanya oluşturma isteğinde, gelen parametreleri enum kontrolleriyle doğruladım ve yeni bir Campaign entity'si üretip kaydettim.
- CreateChannelCommandHandler: Kanal ekleme işlemi için, adı verilen yeni bir Channel entity'sini doğrudan kaydettim.
- CreateCompanyCommandHandler: Şirket ekleme işleminde, gelen status bilgisini enum'a çevirdim ve eksikse varsayılan değer atadım. Diğer alanları da uygun şekilde doldurarak yeni bir Company entity'si oluşturdum.

Bu işlemleri tamamladıktan sonra, her bir entity için API controller'larını yazdım ve ekleme işlemlerinin düzgün çalıştığını test ettim. Özellikle Channel ekleme fonksiyonunu, Swagger arayüzü üzerinden de test ederek doğruladım. Şekil 4'te görüldüğü gibi, Swagger'da Channel ekleme alanı sorunsuz şekilde görüntülenip kullanılabildi.



Şekil 4. Channel Get Methodu

Günün sonunda, hem kod tarafında mantıksal handler yapısını hem de uçtan uca testleri tamamlamış oldum. Böylece, admin paneli için temel ekleme fonksiyonları başarıyla hayata geçirilmiş oldu.

KONTROL:

| Yapılan İş: Google ve GitHub ile OAuth Giriş (Login) Entegrasyonu | Sayfa No:6 |
| --- | --- |
| | Tarih: 21.07.2025 |

Stajımın altıncı gününde, kullanıcıların uygulamaya Google ve GitHub hesapları ile giriş yapabilmesini sağlamak amacıyla OAuth entegrasyon çalışmalarını gerçekleştirdim. Bunun için öncelikle Google ve GitHub geliştirici konsolları üzerinden gerekli secret key ve client id bilgilerini aldım ve uygulamanın yapılandırma dosyalarında güvenli şekilde tanımladım.

Ardından, backend tarafında iki ayrı controller (GoogleAuthController ve GitHubAuthController) oluşturdum:
- GitHubAuthController: Kullanıcıyı GitHub girişine yönlendiriyor, dönüşte elde edilen kullanıcı bilgileriyle (e-posta, GithubId, kullanıcı adı vb.) sistemde daha önce kayıtlı olup olmadığını kontrol ediyor. Eğer kullanıcı ilk kez giriş yapıyorsa, veritabanında yeni bir AppUser kaydı oluşturuyor. Sonrasında JWT token üretip kullanıcıyı, token ile birlikte front-end'e yönlendiriyor.
- GoogleAuthController: Benzer şekilde, Google ile girişte kullanıcı bilgileri (e-posta, GoogleId, ad, soyad, profil fotoğrafı) ile sistemde kayıt kontrolü yapıyor, gerekirse yeni kullanıcı oluşturuyor ve JWT token ile yönlendirme işlemini tamamlıyor.

Bu entegrasyonlar sayesinde, kullanıcılar sistemde hızlıca Google veya GitHub hesaplarıyla kimlik doğrulaması yapabiliyor. Ayrıca tüm bu işlemleri gerçek ortamda test ederek, sorunsuz şekilde çalıştığını doğruladım.

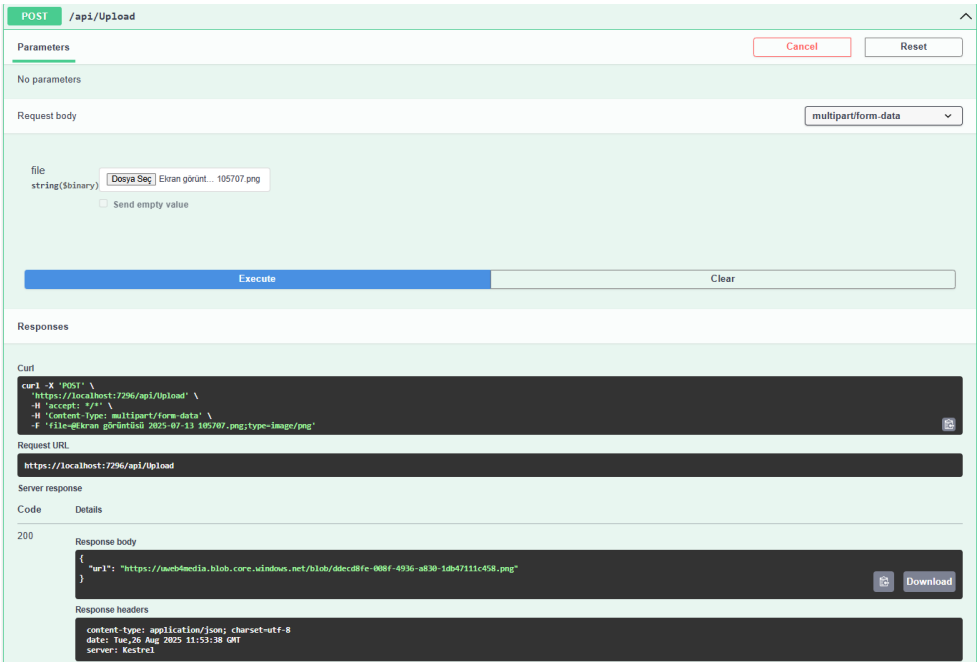
Günün sonunda, uygulamanın kullanıcı deneyimini ve güvenliğini artıracak sosyal giriş (social login) altyapısını başarıyla kurmuş oldum.

KONTROL:

| Yapılan İş: Microsoft ile Giriş (OAuth) ve Azure Blob Storage ile Fotoğraf Yükleme Entegrasyonu | Sayfa No:7 |
|---|---|
| | Tarih: 22.07.2025 |

Bugünkü işyeri eğitimimde, uygulamaya Microsoft hesabı ile giriş (OAuth) özelliğini ekledim. Öncelikle Microsoft platformundan gerekli istemci kimliği (client id) ve gizli anahtar (secret) bilgilerini alarak yapılandırma ayarlarına ekledim. Sonrasında, backend tarafında MicrosoftAuthController adında bir controller oluşturarak kullanıcıların Microsoft ile kimlik doğrulama işlemlerini yönetmesini sağladım. Kullanıcı Microsoft hesabıyla giriş yaptığında, sistemde daha önce kaydı yoksa otomatik olarak yeni kullanıcı oluşturuluyor ve JWT token üretildikten sonra frontend'e yönlendiriliyor.

Ek olarak, uygulamada kullanıcıların fotoğraf yükleyebilmesi için Azure Blob Storage entegrasyonunu gerçekleştirdim. UploadController adında bir API controller yazdım. Bu controller, yüklenen dosyayı Azure'daki ilgili container'a kaydediyor ve dosyanın herkese açık olacak şekilde bir bağlantı (public url) üretip kullanıcıya dönüyor. Şekil 5' de olduğu gibi yüklenen fotoğraflar sistemde Azure linkiyle kullanılabilir hale geliyor.




Şekil 5. Azure Blob Storage ile Fotoğraf Yükleme Entegrasyonu

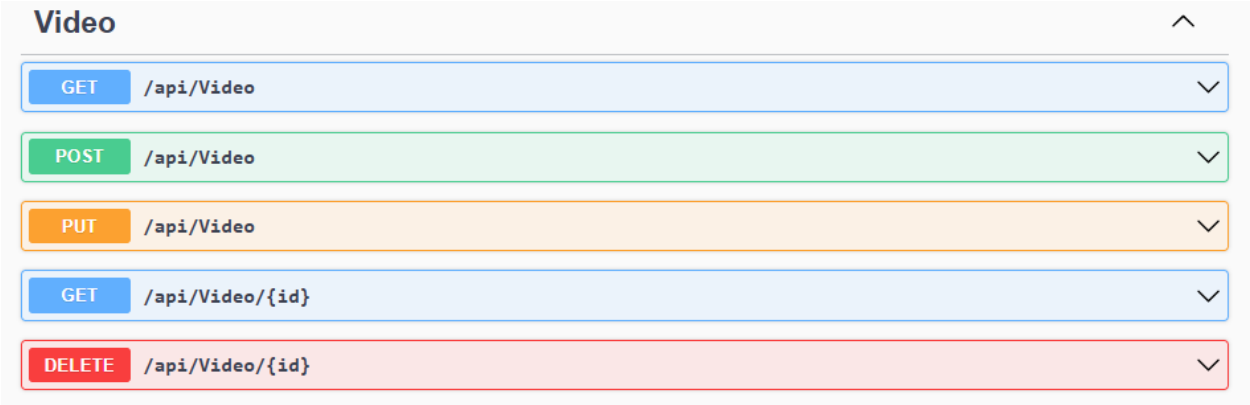
Günün sonunda, uygulamanın hem Microsoft OAuth ile sosyal girişini hem de dosya/fotoğraf yükleme altyapısını Azure üzerinde başarılı şekilde tamamlamış oldum.

KONTROL:

| Yapılan İş: Video İçerikleri için API Controller Oluşturulması | Sayfa No:8 |
| --- | --- |
| | Tarih: 23.07.2025 |

Stajımın sekizinci gününde, uygulamanın video yönetimi için API tarafında `VideoController`'ı oluşturdum. Controller katmanında, video ekleme, güncelleme, silme ve listeleme işlemleri için gerekli olan endpointleri hazırladım. Her bir işlem için ilgili handler sınıfları (`CreateVideoCommandHandler`, `UpdateVideoCommandHandler`, `RemoveVideoCommandHandler`, GetVideoQueryHandler`, `GetVideoByIdQueryHandler`) üzerinden servisleri kullandım.

Kullanıcıların video listesini veya belirli bir videoyu görüntülemesi için [HttpGet] metotlarını, yeni video ekleme için [HttpPost], mevcut videoyu güncellemek için [HttpPut], silmek için ise [HttpDelete] endpointlerini geliştirdim. Silme işlemi için, sadece video sahibi ya da admin olan kullanıcıların işlem yapabilmesini sağlamak amacıyla token ve claim kontrolleri ekledim. Video ekleme işlemini ise şimdilik tüm kullanıcılara açık tuttum, ileride sadece adminlere açmak için ilgili authorize satırını yorum satırı olarak ekledim. Şekil 6' da swagger ekranında gözükmektedir.




Şekil 6. Video test için swagger ekranı

Günün sonunda, video içeriklerinin API üzerinden yönetilebilmesi için gerekli tüm temel işlemlerin controller seviyesinde tamamlandığını ve başarıyla test edildiğini kaydettim.

KONTROL:

| Yapılan İş: Beğeni (Like), Yorum ve Takip (Subscribe) İçin CQRS ve API Controllerlarının Oluşturulması | Sayfa No:9 |
| --- | --- |
| | Tarih: 24.07.2025 |

Bugünkü staj günümde, uygulamanın sosyal etkileşim fonksiyonları olan beğeni (like), yorum (comment) ve takip (subscribe) için hem CQRS mimarisinde gerekli komut ve handlerları hem de API controllerlarını oluşturdum.

- LikeController: Kullanıcıların videoları beğenip beğenmemesini (like/unlike) sağlayan, ayrıca belirli bir kullanıcıya veya tüm kullanıcılara ait beğenileri listeleyebilen endpointler yazdım. Like işlemi için hem klasik oluşturma (POST) hem de aynı endpoint üzerinden toggle edilebilen (beğen/geri al) bir yapı kurdum.
- CommentController: Videolar üzerinde yorum ekleme, silme, tüm yorumları veya belirli bir videoya ait yorumları getirme işlemleri için handler ve endpointleri geliştirdim. Yorumların videoId ile filtrelenebilmesini sağladım.
- SubscriptionController: Kullanıcıların birbirlerini takip edebilmesi (subscribe/unsubscribe) için de benzer şekilde oluşturma, silme ve listeleme işlemleri için gerekli API fonksiyonlarını kodladım.

Tüm bu işlemlerde CQRS mantığıyla, komutlar (Commands) ve sorgular (Queries) ayrı handlerlar tarafından yönetildi. Her bir controller, istenen işlemi ilgili command veya query handler'a yönlendirerek iş akışını tamamladı. Geliştirdiğim endpointleri Postman üzerinde de test ederek düzgün çalıştıklarını doğruladım.

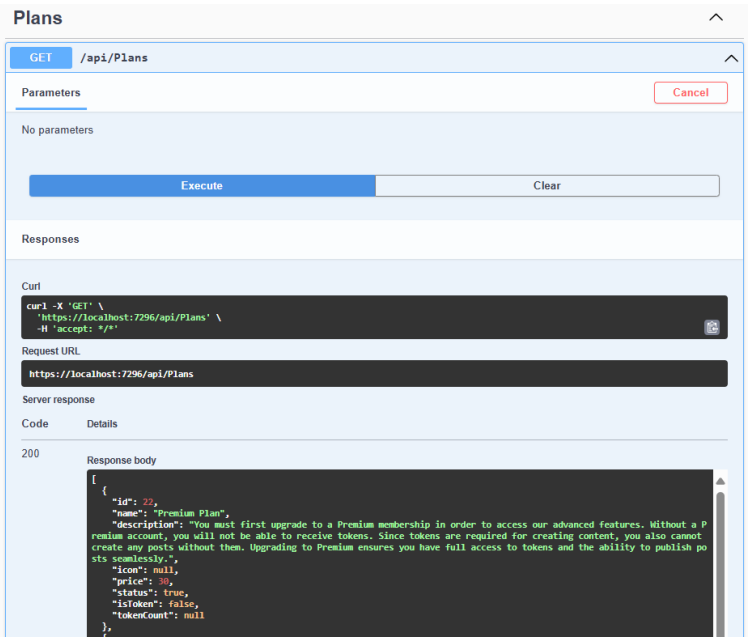
Günün sonunda, sosyal etkileşim altyapısı için gerekli olan temel API fonksiyonlarını başarılı şekilde tamamlamış oldum.

KONTROL:

| Yapılan İş: Sektör, Kanal ve Plan Ekleme-Güncelleme için CQRS Controllerları ve Yetkilendirme Kısıtlamaları | Sayfa No:10<br>Tarih: 25.07.2025 |
|---|---|

Bugünkü staj günümde, uygulamanın yönetim paneli için sektör (Sector), kanal (Channel) ve plan (Plans) ekleme, güncelleme, silme işlemlerine yönelik controller'larımı oluşturdum. CQRS mimarisini kullanarak, her bir işlem için ayrı command ve handler sınıfları tanımladım ve ilgili API controller'larında bu yapıları kullanarak endpointler geliştirdim.

- ChannelController: Kanal ekleme, silme ve güncelleme işlemlerini sadece admin rolündeki kullanıcılara açtım. Listeleme ve detay görüntüleme işlemleri tüm kullanıcılara açık şekilde yapılandırıldı.
- SectorController: Sektörler için de benzer şekilde, sadece adminlerin sektör ekleyip düzenleyebilmesi, diğer kullanıcıların ise sadece listeleme ve detay görme hakkına sahip olması sağlandı.
-PlansController: Plan ekleme, güncelleme ve silme işlemleri adminlere, planları görüntüleme ise tüm kullanıcılara açık şekilde ayarlandı. Şekil 7' de planlar için swagger arayüzü görülmektedir.



Şekil 7. Swagger Plans API get methodu

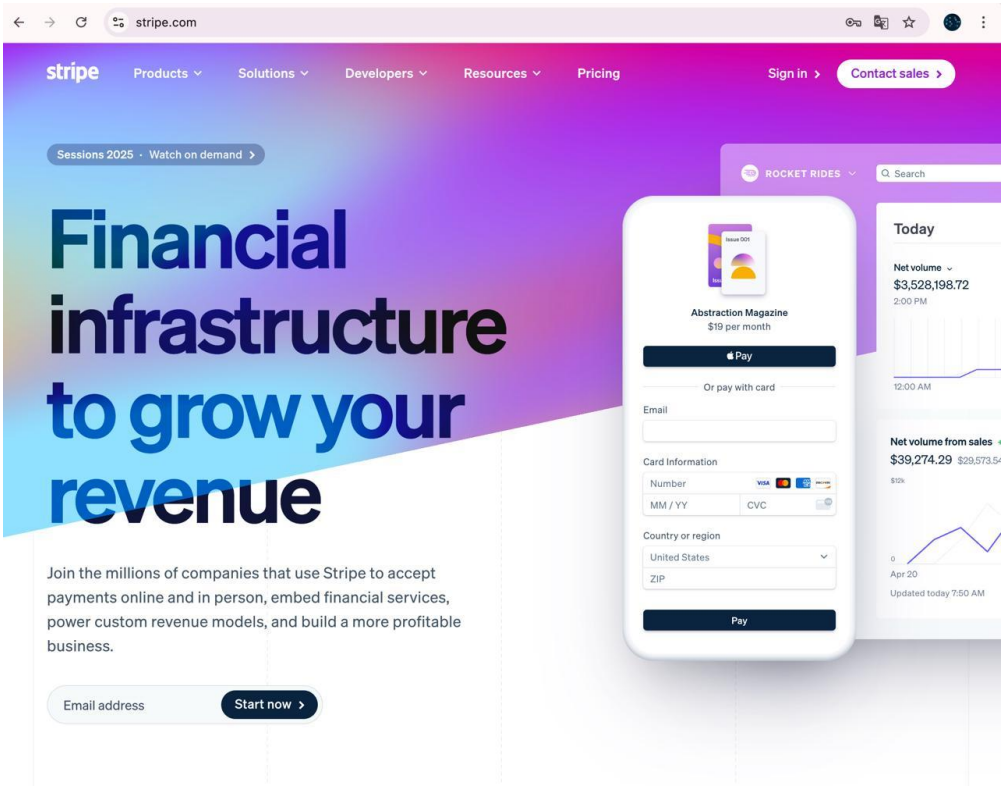
Tüm bu işlemlerde, `[Authorize(Roles = "Admin")]` attribute'unu kullanarak güvenlik ve yetkilendirme kısıtlamalarını başarıyla uyguladım. Böylece sistemin sadece yetkili kişiler tarafından yönetilebilmesini sağladım.

Günün sonunda, sektör, kanal ve plan işlemlerine yönelik API altyapısını hem CQRS mimarisiyle hem de rol bazlı erişim kontrolleriyle eksiksiz şekilde tamamladım.

KONTROL:

| Yapılan İş: Stripe ile Ödeme Entegrasyonu, Hesap Açılışı ve Altyapı Hazırlığı | Sayfa No:11 |
|---|---|
| | Tarih: 28.07.2025 |

Bugünkü staj günümde, uygulama için Stripe ödeme altyapısını hazırlamaya başladım. İlk olarak şirket adına Stripe üzerinden yeni bir hesap açtım ve gerekli olan tüm şirket bilgilerini eksiksiz şekilde sisteme girerek hesabın onaylanma sürecini başlattım. Onay süreci için gereken evrak ve bilgilerin takibini yaptım, Stripe panelinde eksik veya sorunlu alanların kontrolünü sağlayarak başvurunun sorunsuz ilerlemesini sağladım.




Şekil 8. Stripe Arayüzü

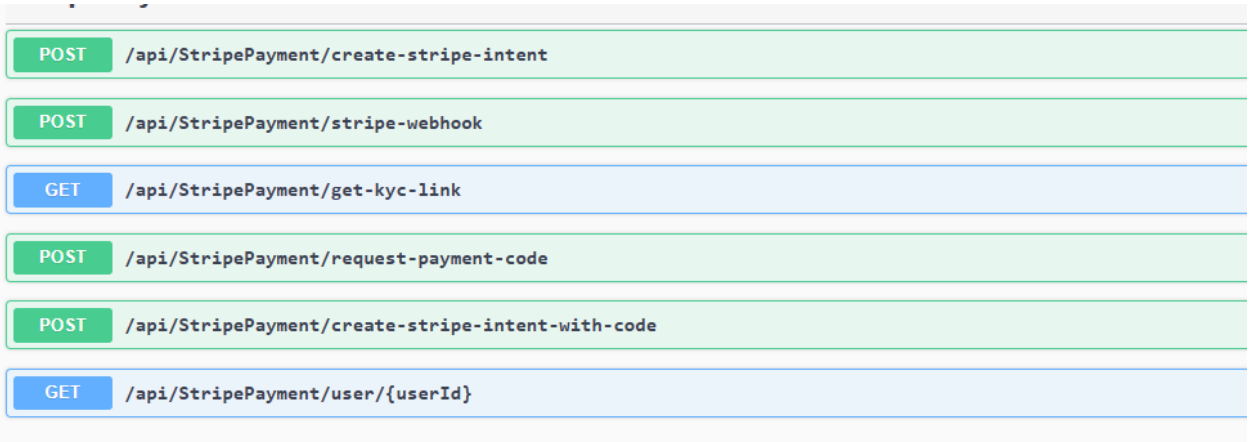
Ayrıca, Stripe üzerinden ödeme alınabilmesi için backend tarafında altyapı düzenlemelerine başladım. API anahtarlarının güvenli şekilde yönetilmesi, ödeme endpointlerinin ve gerekli servislerin hazırlanması için ön analiz ve hazırlık çalışmaları yaptım. Günün sonunda, Stripe hesabının kurulumu ve onay süreçlerinin tamamlanması için gerekli adımları tamamladım ve ödeme entegrasyonunun ilk fazını başarıyla başlatmış oldum.

KONTROL:

| Yapılan İş: Stripe Ödeme API Entegrasyonlarının Tamamlanması ve Test Edilmesi | Sayfa No:12<br>Tarih: 29.07.2025 |
|---|---|

Bugünkü staj günümde, Stripe ödeme entegrasyonu için gerekli olan API işlemlerini tamamladım ve test ettim. Backend tarafında, ödeme akışı ve kullanıcı doğrulama süreçlerine yönelik yeni endpointler oluşturdum. Özellikle; ödeme başlatma, ödeme kodu oluşturma, ödeme webhooks, KYC (kimlik doğrulama) bağlantısı alma gibi işlemleri API üzerinden yönetilebilir hale getirdim.

Şekil 9'da görüldüğü gibi;
- `/api/StripePayment/create-stripe-intent` ile ödeme işlemini başlatmak için intent oluşturulabiliyor,
- `/api/StripePayment/stripe-webhook` endpointi ile Stripe'dan gelen bildirimleri (webhook) almak mümkün,
- `/api/StripePayment/get-kyc-link` ile KYC doğrulama bağlantısı alınabiliyor,
- `/api/StripePayment/request-payment-code` ve `/api/StripePayment/create-stripe-intent-with-code` ile ödeme kodu oluşturma ve kod ile intent oluşturma işlemleri gerçekleştirilebiliyor,
- `/api/StripePayment/user/{userId}` endpointi ile de kullanıcının Stripe ödeme bilgilerine erişilebiliyor.




Şekil 9. Stripe Ödeme API Entegrasyonu

Tüm bu işlemleri Swagger arayüzünde tek tek test ederek, Stripe ile ödeme altyapısının uçtan uca sorunsuz çalıştığını doğruladım. Günün sonunda, ödeme ve kimlik doğrulama süreçlerini sistemde güvenli ve entegre şekilde tamamlamış oldum.

KONTROL:

| Yapılan İş: Controllerlarda Yetkilendirme (Authentication & Authorization) Düzenlemeleri | Sayfa No:13 |
| --- | --- |
| | Tarih: 30.07.2025 |

Bugünkü staj günümde, uygulamanın API katmanında güvenlik ve erişim kontrollerini güçlendirmek için önemli düzenlemeler yaptım. Özellikle, sistemde sadece admin yetkisine sahip kullanıcıların işlem yapabilmesi gereken noktalara `[Authorize(Roles = "Admin")]` attribute'unu ekleyerek, kritik işlemlerin güvenliğini üst seviyeye çıkardım. Böylece, örneğin video ekleme, video güncelleme ve silme gibi önemli operasyonlar artık yalnızca admin rolündeki kullanıcılar tarafından gerçekleştirilebiliyor. Sıradan kullanıcılar ise yalnızca video listeleme ve görüntüleme işlemlerini yapabiliyor.

Bu kapsamda, kod tarafında controllerlar üzerinde detaylı inceleme yaparak, hangi endpointlerin herkes tarafından erişilebilir, hangilerinin ise sadece admin yetkisiyle çalıştırılabilir olması gerektiğini belirledim. Ardından, ilgili endpointlerin başına `[Authorize(Roles = "Admin")]` veya `[AllowAnonymous]` gibi gerekli attribute'ları ekledim. Bununla birlikte, bazı işlemlerde daha esnek bir yapı gerektirdiği için, token içerisindeki claim'ler üzerinden kullanıcının admin olup olmadığını kontrol eden ek mekanizmalar geliştirdim. Özellikle silme gibi işlemlerde, hem admin olup olmadığına hem de kullanıcının kendi kaydı üzerinde işlem yapıp yapmadığına bakarak, güvenliği ve veri bütünlüğünü artırdım.

Ayrıca, authentication ve authorization konusu üzerinde çalışırken, uygulamanın genel güvenlik politikasını ve kullanıcı rolleriyle ilgili kurguyu da gözden geçirdim. Böylece, hem yönetici hem de sıradan kullanıcıların sistemde hangi işlemleri yapabileceği net bir şekilde ayrılmış oldu. Bu değişikliklerin ardından, farklı rollerle sisteme giriş yaparak ilgili endpoint'leri test ettim ve yetki sınırlarının doğru şekilde uygulandığını doğruladım.

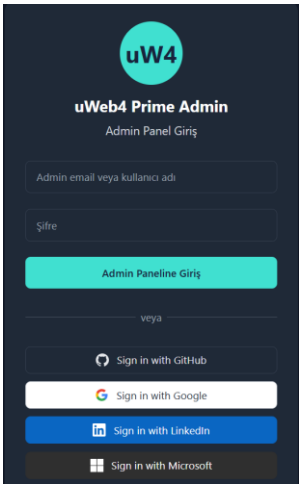
Günün sonunda, uygulamanın güvenliğini ve veri bütünlüğünü sağlayacak şekilde, rol bazlı erişim kısıtlamalarını ve authentication & authorization altyapısını başarıyla tamamladım. Bu düzenlemeler sayesinde sistemdeki kritik işlemler yalnızca yetkili kişiler tarafından yapılabilir hale geldi ve uygulamanın genel güvenlik seviyesi önemli ölçüde yükseldi.

KONTROL:

| Yapılan İş: Admin Paneli UI Entegrasyonu ve API Bağlantısı | Sayfa No:14 |
|---|---|
| | Tarih: 31.07.2025 |

Bugünkü staj günümde, admin paneli için özel olarak tasarlanmış ve tarafıma iletilen yeni kullanıcı arayüzünü (UI), uygulamanın mevcut API altyapısına entegre ettim. Şekil 10'da görüldüğü gibi modern ve kullanıcı dostu bir arayüz ile giriş ekranı tasarlanmış. Bu ekranda, adminlerin e-posta/kullanıcı adı ve şifre ile giriş yapabildiği alanlar ve ayrıca sosyal giriş seçenekleri (GitHub, Google, LinkedIn, Microsoft) yer almakta.

Entegrasyon sürecinde, React tabanlı frontend kodunu projeme aldım ve API'mize bağlanabilmesi için gerekli endpoint bağlantılarını düzenledim. Özellikle:
- Giriş işlemlerinde `login` fonksiyonunu, email ve şifre bilgisini API'ye iletecek şekilde bağladım.
- Kullanıcı doğrulama ve hata yönetimini Toast bildirim sistemi ile destekledim.
- Sosyal giriş butonlarını, ilerleyen dönemde OAuth akışına yönlendirmek için hazır hale getirdim.
- Kullanıcıdan gelen bilgiler, API'ye güvenli şekilde gönderildi ve login sonrası başarılı girişte admin paneline yönlendirme işlemlerini test ettim. Şekil 10' da login ekranı arayüzü görülmektedir.




Şekil 10. Admin Login Ekranı

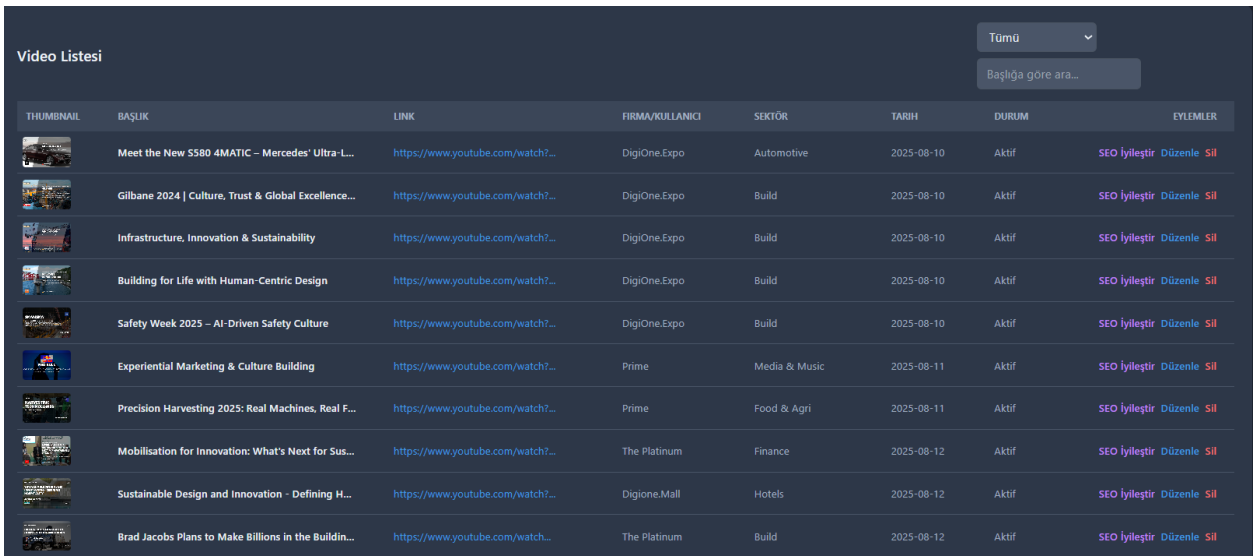
Ayrıca, sosyal giriş (GitHub, Google, LinkedIn, Microsoft) akışları için gerekli butonları ve ikonları ekleyerek, ileride kolayca OAuth bağlantısı kurulabilecek şekilde ön hazırlığımı tamamladım.

Tasarım tarafında, koyu tema (dark mode) ve responsive (her ekrana uyumlu) yapı ile kullanıcı deneyimini artırdım. Günün sonunda, hem görsel olarak hem de fonksiyonel olarak güçlü bir admin paneli giriş ekranı ve panel temelini başarıyla uygulamış oldum.

KONTROL:

| Yapılan İş: Admin Panelinde Video Yükleme, Güncelleme, Silme Fonksiyonlarının Geliştirilmesi ve Tabloyla Listeleme | Sayfa No:15<br>Tarih: 01.08.2025 |
|---|---|

Bugünkü staj günümde, admin panelinde video yönetimi ile ilgili önemli fonksiyonları geliştirdim. Özellikle video yükleme, video güncelleme ve video silme işlemlerini başarıyla tamamladım. Tüm bu işlemler için kullanıcı arayüzünde (UI) gerekli formları ve butonları oluşturdum. Ek olarak, yüklenen veya var olan videoların sistemde bir tablo halinde listelenmesini sağladım. Şekil 11' de görülmektedir.




Şekil 11. Video Ekleme, Silme, Düzenleme Sayfası.

Video listesini oluştururken, her video için görsel (thumbnail), başlık, link, firma/kullanıcı, sektör, tarih, durum ve işlemler sütunlarını ekledim. "Düzenle" ve "Sil" butonları ile adminler ilgili videoyu güncelleyebiliyor veya silebiliyor. Ayrıca, "SEO İyileştir" gibi ek işlemler de tablo üzerinden erişilebilir durumda. Tablo üzerinde başlığa göre arama ve filtreleme gibi kullanışlı özellikler ekledim.

Güvenlik tarafında, video ekleme, düzenleme ve silme işlemlerini sadece admin yetkisine sahip kullanıcıların yapabilmesini sağladım. Authentication ve Authorization kontrollerini kullanarak, admin dışındaki kullanıcıların bu sayfaya ve fonksiyonlara erişimini engelledim. Böylece sistemde veri bütünlüğü ve güvenlik tam anlamıyla sağlanmış oldu.

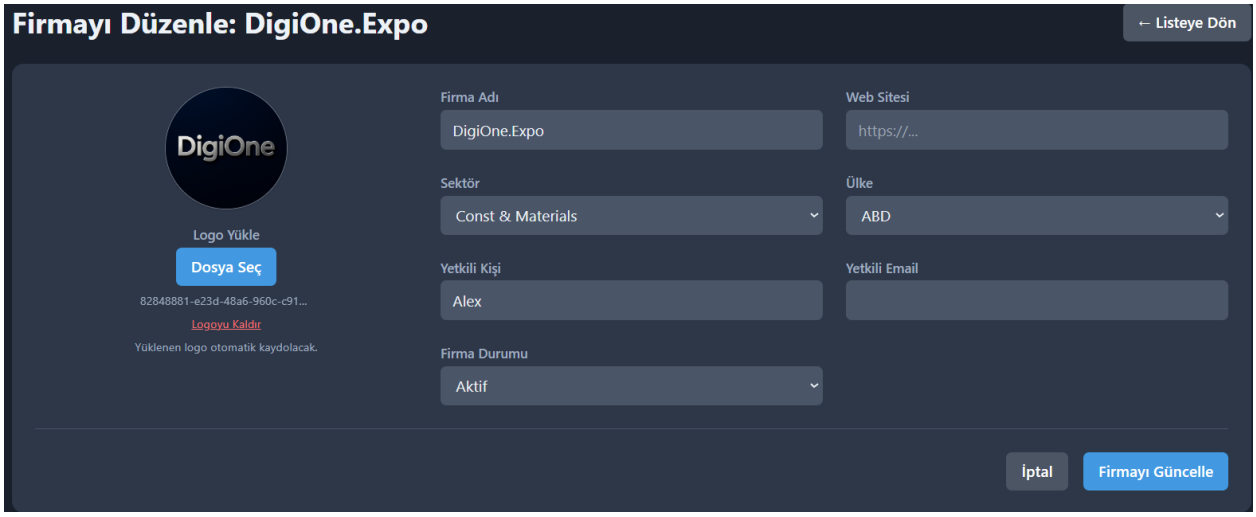
Günün sonunda, admin panelinde video yönetimine dair tüm temel CRUD işlemlerini tamamladım ve kullanıcı dostu bir arayüzle listelenmesini sağladım.

KONTROL:

| Yapılan İş: Admin Panelinde Firma Yönetimi Modülünün Geliştirilmesi, Azure Logo Yükleme Entegrasyonu | Sayfa No:16 |
|---|---|
| | Tarih: 04.08.2025 |

Bugünkü staj günümde, admin panelinde firma yönetimi modülünü geliştirdim ve videolarda olduğu gibi firmaların da tablo halinde listelenmesini sağladım. Kullanıcı arayüzünde, firmaların bilgileri bir tablo halinde listeleniyor ve her firma için düzenleme ve silme işlemleri kolaylıkla yapılabiliyor.

Firma düzenleme ekranında (Şekil 12), seçilen firmanın adı, web sitesi, sektör, ülke, yetkili kişi ve iletişim e-posta adresi gibi alanların güncellenmesini sağlayan bir form oluşturdum. Firma durumu (aktif, pasif, incelemede vs.) seçimli olarak ayarlanabiliyor.




Şekil 12. Firma Düzenleme Ekranı

En dikkat çekici yenilik ise firma logolarının yönetimi oldu. Kullanıcı, "Logo Yükle" butonunu kullanarak yeni bir logo seçebiliyor ve yüklenen logolar doğrudan Azure Storage üzerinde depolanıyor. Yükleme tamamlandığında logo otomatik olarak kaydediliyor ve gerekirse "Logoyu Kaldır" butonuyla kaldırılabiliyor. Böylece merkezi ve güvenli bir logo yönetimi sistemi kurmuş oldum.

Bu ekran üzerinde yapılan değişiklikler, API ile haberleşerek backend'e kaydediliyor ve güncellenen firma bilgileri anlık olarak tabloya yansıyor. Firma düzenleme ve silme işlemlerinde de sadece admin yetkisine sahip kullanıcıların işlem yapabilmesini sağladım.

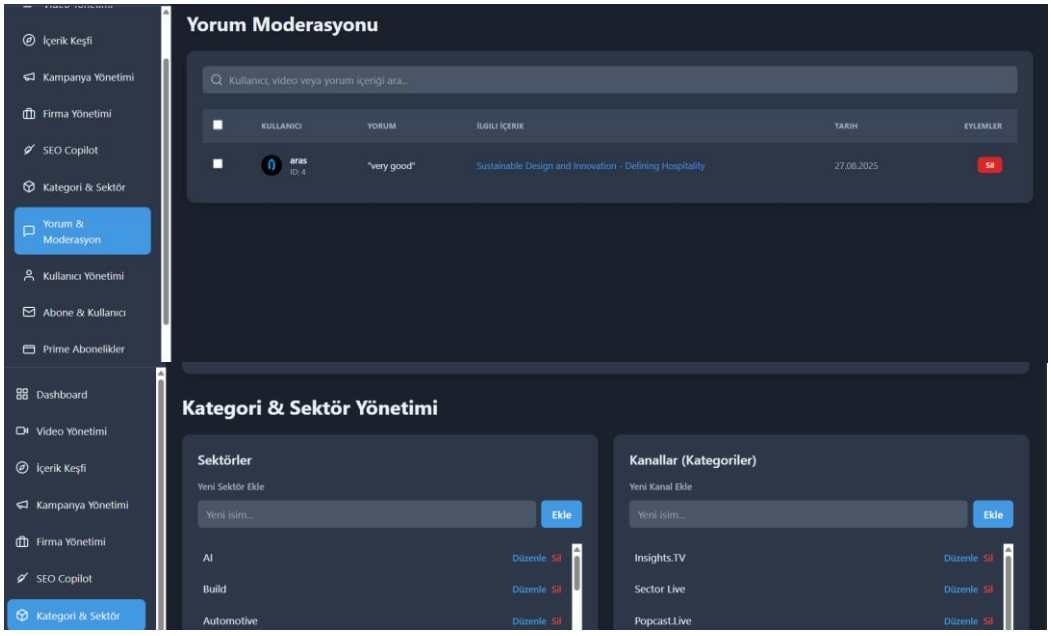
Günün sonunda, admin panelinde firma yönetimi için kapsamlı ve kullanıcı dostu bir modül geliştirmiş oldum. Artık firmalar kolayca eklenip, güncellenip, silinebiliyor ve logoları Azure üzerinde güvenle saklanıyor.

KONTROL:

| Yapılan İş: Yorum Yönetimi ve Sektör/Kanal Ekleme Sayfalarının Geliştirilmesi, Admin Yetkilendirme ile Güvenli Erişim | Sayfa No:17<br>Tarih: 05.08.2025 |
|---|---|

Bugünkü staj günümde, admin panelinde hem yorum yönetimi ekranını hem de sektör ve kanal ekleme/güncelleme sayfalarını geliştirdim. Yorum yönetimi (moderasyon) sayfasında, kullanıcıların yaptığı tüm yorumları tablo halinde listeleyip, adminlerin bu yorumları kolayca silebilmesini sağladım. Arama çubuğu sayesinde kullanıcıya, videoya veya yorum içeriğine göre filtreleme yapılabiliyor. Ayrıca, yorumlara ait kullanıcı adı, yorum metni, ilgili video ve tarih bilgilerini detaylı şekilde ekranda gösterecek şekilde tasarımı geliştirdim.

Bunun yanında, kategori & sektör yönetimi ekranında hem yeni sektör hem de yeni kanal (kategori) ekleme fonksiyonlarını hayata geçirdim. Mevcut sektör ve kanalların listelenmesini, her biri için ayrı ayrı düzenleme ve silme işlemlerinin yapılabilmesini sağladım. Eklenen sektör ve kanallar anında listeye yansıyor ve güncellenebiliyor. Şekil 13' de görüldüğü gibi işlemler için sade ve anlaşılır bir arayüz oluşturdum.



Şekil 13. Yorum, Kategori ve Sektör Yönetimi

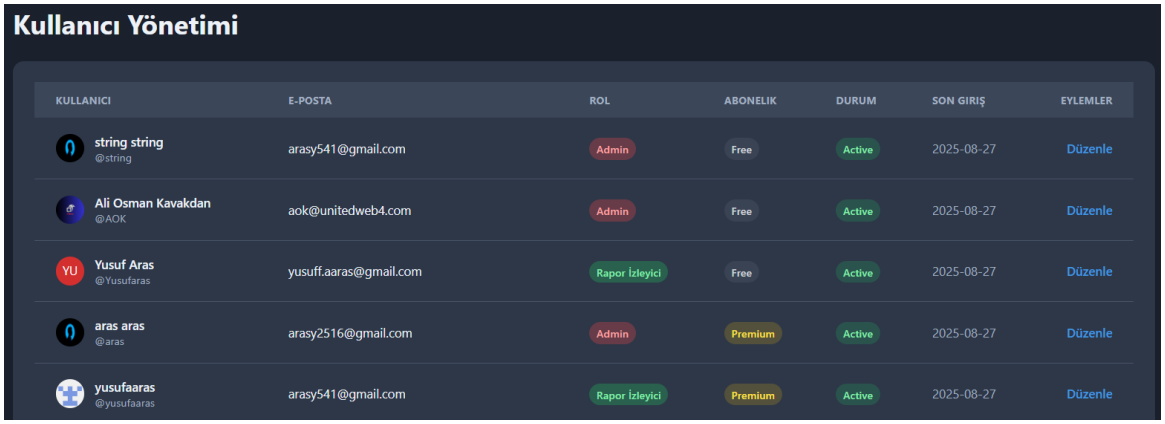
Güvenlik tarafında, admin dışındaki kullanıcıların bu sayfalara erişimini engellemek için token (JWT) tabanlı authentication ve authorization yapılandırmasını uyguladım. Her ekleme, silme veya güncelleme işlemi öncesinde kullanıcının admin olup olmadığı kontrol ediliyor. Böylece, sistemde hem yorum moderasyonu hem de kategori/kanal yönetimi tamamen güvenli ve sadece yetkili kişiler tarafından yapılabilir hale geldi.

Günün sonunda, admin panelinde içerik ve kategori yönetimi süreçlerini büyük ölçüde tamamlamış oldum. Kullanıcı dostu ve modern bir arayüzle, sistemin yönetimsel işlevleri artık daha etkin bir şekilde sürdürülebiliyor.

KONTROL:

| Yapılan İş: Kullanıcı Yönetimi ve Abone Kullanıcı Sayfalarında Listeleme ve Güncelleme Fonksiyonlarının Geliştirilmesi | Sayfa No:18<br>Tarih: 06.08.2025 |
|---|---|

Bugünkü işyeri eğitimimde, admin panelinde kullanıcı yönetimi ve abone kullanıcı yönetimi ekranlarını geliştirerek verilerin listelenmesini ve güncellenebilmesini sağladım. Kullanıcı yönetimi sayfasında, sistemde yer alan tüm kullanıcıların ad, e-posta, rol, abonelik tipi, durum ve son giriş tarihi gibi bilgileri tablo halinde gösterilecek şekilde arayüzü tasarladım. Her bir kullanıcı satırı için "Düzenle" butonu ile kullanıcı bilgilerinin güncellenebilmesini sağladım. Kullanıcıların rol ve abonelik durumları gibi kritik alanlarını hızlıca görebilmek için etiketler ve renkli gösterimler kullandım (Şekil 14).



Şekil 14. Kullanıcı Yönetimi Sayfası

Ayrıca, abone ve kullanıcı yönetimi başlığı altında, sisteme kayıtlı tüm kullanıcı ve abonelerin detaylı olarak listelendiği ayrı bir ekran daha tasarladım. Burada kullanıcıların kimlik kaynağı, konum, abonelik durumu ve son aktivite bilgileri gibi ek detaylar da tabloya eklendi. Her bir kullanıcı için "Detay" butonuyla, ilgili kullanıcının daha fazla bilgisine erişilebiliyor. Toplu e-posta gönderimi ve dışa aktarma gibi gelişmiş yönetimsel fonksiyonların butonlarını da ekledim.

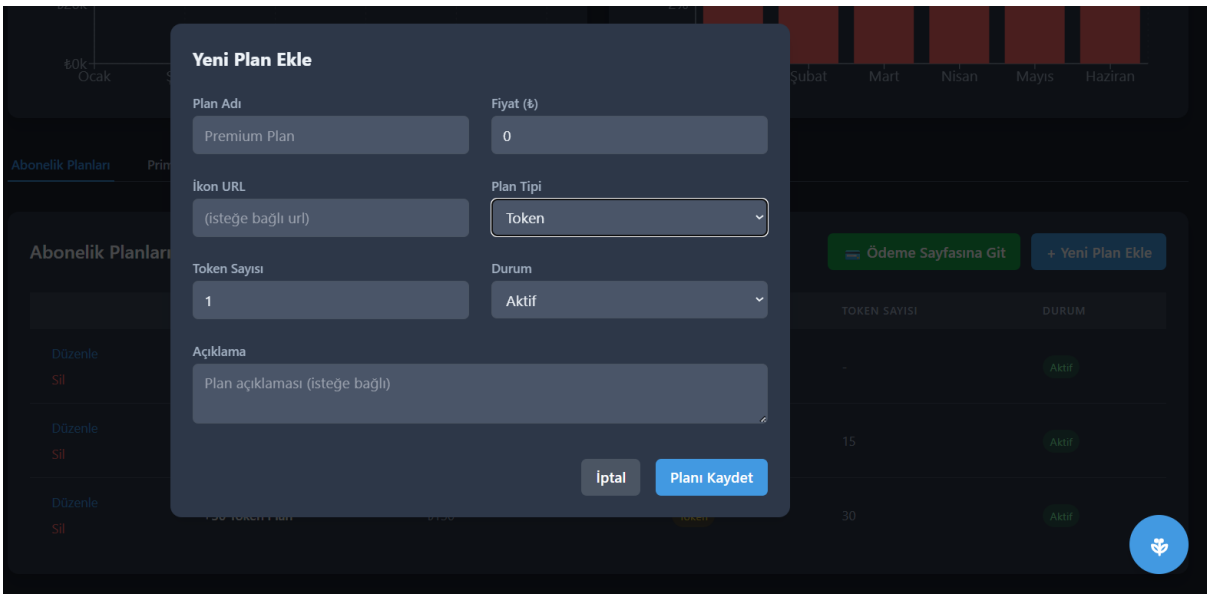
Tüm bu işlemlerde, güncelleme fonksiyonlarını API ile entegre ederek, yapılan değişikliklerin anlık olarak backend'e kaydedilmesini sağladım. Böylece, hem kullanıcı bilgilerinin güncel tutulması hem de sistem yönetiminin daha verimli yürütülmesi mümkün oldu.

Günün sonunda, admin panelinin kullanıcı ve abone yönetimi modüllerini işlevsel ve kullanıcı dostu bir şekilde tamamlamış oldum.

KONTROL:

| Yapılan İş: Plan ve Token Planı Ekleme Fonksiyonlarının Geliştirilmesi | Sayfa No:19 |
|---|---|
| | Tarih: 07.08.2025 |

Bugünkü staj günümde, admin panelinde kullanıcıların abonelik satın alabilmesi veya içerik/post satın alabilmesi için gerekli olan plan ve token planı ekleme ekranlarını geliştirdim. Şekil 15'de görüldüğü gibi, yeni bir plan eklemek için açılan formda plan adı, fiyatı, ikon url'si, plan tipi (abonelik veya token), token sayısı, durum ve açıklama gibi alanlar yer almakta.




Şekil 15. Yeni Plan Ekleme Modalı

Bu ekranda admin kullanıcılar, sistemde kullanılacak yeni bir abonelik planı veya token bazlı plan tanımlayabiliyor. Token planları; kullanıcıların belirli sayıda token satın alarak, bu tokenları içerik veya diğer özellikler için harcayabilmesini sağlıyor. Plan tipi seçimi ile birlikte formda ilgili alanlar dinamik olarak açılıp kapanıyor.

Eklediğim bu arayüzde, plan bilgileri API'ye kaydediliyor ve anında güncelleniyor. Adminler tüm planları tablo halinde görebiliyor, düzenleyip silebiliyor. Kullanıcılar ise abonelik veya token planlarını seçerek diledikleri gibi satın alma işlemlerini gerçekleştirebilecekler.

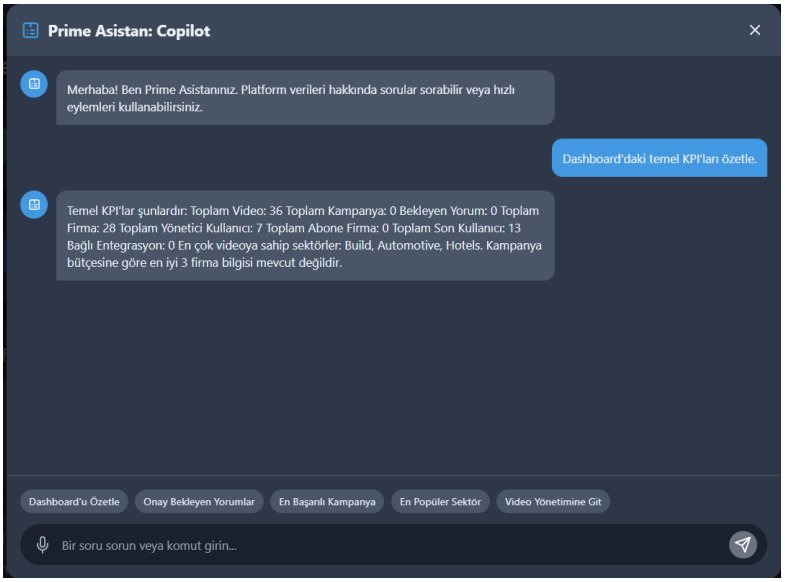
Günün sonunda, sistemde esnek ve yönetilebilir plan altyapısını tamamlamış oldum. Bu sayede, kullanıcıların abonelik ya da token satın alma süreçleri admin paneli üzerinden kolayca yönetilebilir hale geldi.

KONTROL:

| Yapılan İş: Yapay Zeka Destekli Teknik Destek Modülü Entegrasyonu | Sayfa No:20 |
|---|---|
| | Tarih: 08.08.2025 |

Bugünkü staj günümde, admin paneli için teknik destek modülü olan Prime Asistan'a yapay zeka entegrasyonu gerçekleştirdim. Bu sayede, platformda admin kullanıcılarının karşılaştığı tüm sorulara anında cevap bulabilmeleri ve kimseye ihtiyaç duymadan platformu etkin bir şekilde kullanabilmeleri sağlanmış oldu.

Entegrasyon kapsamında OpenAI GPT-4.1 mini modeli ve embedding teknolojisini kullandım. Sık sorulan sorular, platformun veri yapısı, temel KPI analizleri ve yönetimsel işlemler gibi konularda kullanıcının sorduğu sorulara anlık ve doğru yanıtlar verilmesi için yapay zekanın bilgi tabanını hazırladım. Yapay zekanın, platformdaki güncel verilere erişip özetler sunabilmesi için arayüzde özet, yönlendirme ve bilgi kartı butonları da ekledim. Şekil 16' da gözükmektedir.




Şekil 16. Yapay Zeka Modalı

Kullanıcılar, teknik destek modülünde doğal dilde soru sorabildiği gibi, hızlı aksiyonlar için butonları da kullanabiliyor. Örneğin; "Dashboard'u özetle", "En popüler sektör", "Onay bekleyen yorumlar", "Video yönetimine git" gibi komutlarla platformun farklı bölümlerine hızlı erişim sağlanabiliyor.

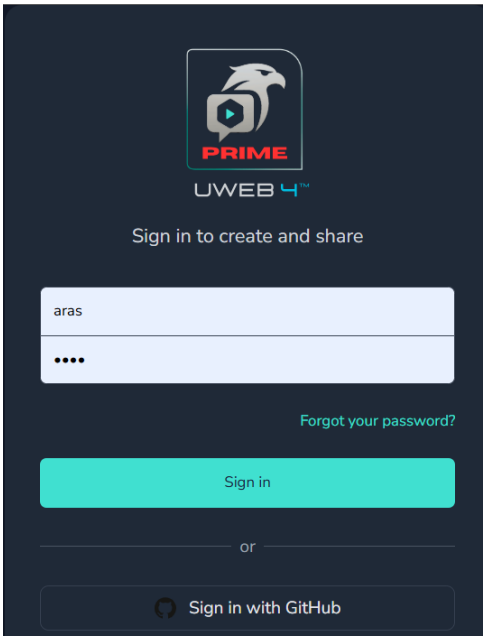
Günün sonunda, Prime Asistan sayesinde platformun yönetiminde otomasyon seviyesi yükseldi ve kullanıcı deneyimi ciddi şekilde iyileştirildi. Artık adminler, teknik destek almak için başka birine ihtiyaç duymadan platformla ilgili her türlü bilgiye ulaşabiliyor ve işlemlerini yapay zeka desteğiyle kolayca gerçekleştirebiliyorlar.

KONTROL:

| Yapılan İş: Prime.uweb4.com Login Paneli Entegrasyonu ve Hata Düzeltmeleri | Sayfa No:21 |
|---|---|
| | Tarih: 11.08.2025 |

Bugünkü staj günümde prime.uweb4.com adresi için yeni login panelinin entegrasyonunu gerçekleştirdim. Tüm kullanıcı arayüzü (UI) tarafı bana şirketin frontend tasarımcıları tarafından hazır şekilde gönderildi. Ben de bu tasarımları alarak projeye ekledim ve login panelini işlevsel hale getirdim.

Login panelinde, kullanıcı adı ve şifre ile girişin yanı sıra GitHub ile giriş seçeneği de mevcut. Özellikle GitHub OAuth entegrasyonu için gerekli API bağlantılarını kurdum ve güvenli giriş akışını tamamladım. Kullanıcılar, klasik giriş yerine GitHub hesaplarıyla da kolayca oturum açabiliyorlar.




Şekil 17. Prime Login Sayfası

Geliştirme sürecinde gün boyunca karşıma çıkan çeşitli hata ve eksiklikleri giderdim. Özellikle doğrulama, form validasyonu, hata mesajı gösterimi ve login akışındaki yönlendirme gibi konularda detaylı debugging ve testler yaptım. Ayrıca “Forgot your password?” (Şifrenizi mi unuttunuz?) bağlantısının işlevselliğini kontrol ettim ve UI/UX tarafında kullanıcı deneyimini arttıracak düzenlemeler yaptım.

Günün sonunda, modern ve kullanışlı bir login panelini prime.uweb4.com'da başarıyla devreye aldım ve platformun kullanıcı giriş süreçlerini hem güvenli hem de kullanıcı dostu hale getirdim.

KONTROL:

| Yapılan İş: Google ve Microsoft ile Giriş (OAuth) Entegrasyonu, Backend ve Frontend Düzenlemeleri | Sayfa No:22 |
| --- | --- |
| | Tarih: 12.08.2025 |

Bugünkü staj günümde, prime.uweb4.com login paneline Google ve Microsoft ile giriş (OAuth) adımlarının tamamlanması üzerine çalıştım. Hem backend hem de frontend tarafında gerekli entegrasyonları ve düzenlemeleri gerçekleştirdim.

Google ile giriş için Google Console üzerinden, Microsoft ile giriş için ise Azure portal üzerinden gerekli uygulamaları oluşturdum ve istemci kimlik bilgilerini aldım. Backend tarafında OAuth akışını yönetecek endpointleri düzenledim ve güvenli token doğrulama mekanizmasını ekledim. Frontend tarafında ise, login ekranına Google ve Microsoft ile giriş butonlarını ekleyerek kullanıcıların tek tıkla hızlı ve güvenli şekilde giriş yapabilmesini sağladım.

Geliştirme sürecinde, OAuth akışındaki yönlendirme, hata yönetimi ve token alımı gibi detayları dikkatlice test ettim ve yaşadığım problemleri giderdim. Artık kullanıcılar, dilerlerse Google, Microsoft veya GitHub hesaplarıyla kolayca platforma giriş yapabiliyorlar.

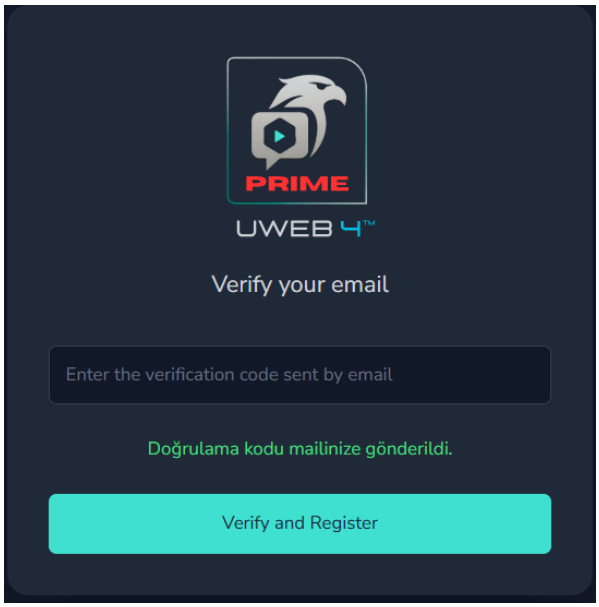
Günün sonunda, platforma kurumsal ve bireysel kullanıcıların sorunsuzca erişebileceği, çoklu kimlik sağlayıcı desteği olan modern bir giriş panelini başarıyla devreye almış oldum.

KONTROL:

| Yapılan İş: Kullanıcı Kayıt Sonrası E-Posta Doğrulama ve Frontend Entegrasyonu | Sayfa No:23 |
| --- | --- |
| | Tarih: 13.08.2025 |

Bugünkü staj günümde, kullanıcıların sisteme kayıt olurken e-posta adreslerinin doğrulanmasını sağlayan mekanizmayı geliştirdim. Kullanıcı register olduktan sonra, sistem tarafından e-posta adresine bir doğrulama kodu gönderiliyor. Frontend tarafında ise kullanıcıdan bu doğrulama kodunu girmesini isteyen bir arayüz hazırladım.

Kullanıcı, e-posta adresine gelen doğrulama kodunu ilgili alana girip "Verify and Register" butonuna bastığında, backend'e yapılan istek ile hem kodun doğruluğu kontrol ediliyor hem de kullanıcının e-posta adresinin gerçekten ona ait olduğu doğrulanıyor. Böylece, yalnızca geçerli ve doğrulanmış e-posta adresine sahip kullanıcılar sisteme kayıt olabiliyor. Şekil 18' de kod girme alanı gözükmektedir.




Şekil 18. Doğrulama Kodu Girme Alanı

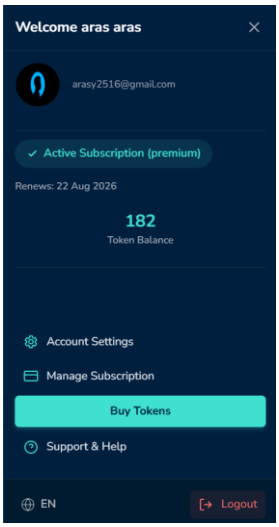
Bu işlem sayesinde, hem platformun güvenliği arttı hem de spam veya yanlış e-posta ile kayıtların önüne geçilmiş oldu. Kullanıcı deneyimini iyileştirmek için doğrulama kodunun gönderildiğine dair bilgilendirme mesajı da ekledim.

Günün sonunda, kullanıcı kayıt sürecinin önemli bir adımını daha tamamladım ve platformun güvenilirliğini bir adım ileriye taşıdım.

KONTROL:

| Yapılan İş: Kullanıcı Sidebar Geliştirme | Sayfa No:24 |
|---|---|
| | Tarih: 14.08.2025 |

Bugünkü staj günümde, kullanıcılar için özel olarak tasarlanmış sidebar (yan menü) komponentini tamamladım. Bu sidebar'da Şekil 19' da görüldüğü üzere kullanıcıların ismi, e-posta adresi, aktif abonelik durumu (ör. premium), aboneliğin yenilenme tarihi, güncel token bakiyesi gibi bilgilerin tamamı anlaşılır ve görsel olarak öne çıkan bir şekilde gösterilmektedir. Kullanıcı arayüzünde, "Buy Tokens" (Token Satın Al), "Manage Subscription" (Aboneliği Yönet) ve "Account Settings" (Hesap Ayarları) gibi hızlı erişim butonlarını ekledim.




Şekil 19. Kullanıcı Sidebarı

Ayrıca, kullanıcı platformdan çıkış (logout) yaptığında, hem backend hem de frontend tarafında kullanıcıya ait token bilgisinin güvenli şekilde silinmesini sağladım. Böylece kullanıcı, tekrar giriş yapana kadar sistemde oturum bilgisi ve token saklanmıyor, güvenlik artırılmış oldu.

Bunun yanında, sidebar'da destek ve yardım, dil seçimi gibi ek fonksiyonlara da yer verdim. Görsel ve işlevsellik açısından modern, sade ve kullanıcı dostu bir yapıya sahip olan bu modül ile kullanıcı deneyimini önemli ölçüde iyileştirdim.

Günün sonunda, kullanıcıların hem kendi bilgilerini hem de abonelik ve token durumlarını hızlıca görebileceği, güvenli bir şekilde çıkış yapabileceği fonksiyonel bir sidebar geliştirilmiş oldu.

KONTROL:

| Yapılan İş: Kullanıcı Profil Sayfası Geliştirme | Sayfa No:25 |
| --- | --- |
| | Tarih: 15.08.2025 |

Bugünkü staj günümde, kullanıcılar için modern ve detaylı bir profil sayfası geliştirdim. Bu sayfanın amacı, kullanıcının kişisel bilgilerini, abonelik durumunu ve platformdaki kendi içeriklerini tek bir yerde kolayca görüntülemesini ve yönetmesini sağlamaktır.

Profil sayfasının genel yapısı ve fonksiyonları:

1. Kullanıcı Kartı ve Genel Bilgiler:
Sayfanın üst kısmında kullanıcının profil fotoğrafı (veya avatarı), kullanıcı adı ve e-posta adresi büyük bir kart üzerinde gösteriliyor. Ayrıca kullanıcının rolü (örneğin Admin) burada etiket olarak yer almakta. Kullanıcı, sağ üstteki “Edit Profile” butonuna tıklayarak profil bilgilerini düzenleyebiliyor.

2. Hesap Detayları (Account Details):
Bu bölümde, kullanıcının e-posta adresi ve profil biyografi (bio) alanı yer alıyor. Kayıt esnasında veya daha sonra eklenen biyografi buradan düzenlenebiliyor.

3. Abonelik Bilgileri (Subscription):
Kullanıcının mevcut abonelik planı (örneğin “Premium Plan”) burada gösteriliyor. “Manage” butonuna tıklandığında kullanıcı abonelik veya ödeme ayarlarına yönlendirilebiliyor. Böylece kullanıcı, aboneliğini kolayca kontrol edebiliyor ve gerekli değişiklikleri yapabiliyor.

4. Kullanıcının Yayınladığı Videolar (Your Published Videos):
Sayfanın altında, kullanıcının platformda yayınladığı tüm videolar küçük kartlar halinde listeleniyor. Her video kartında başlık, yayın tarihi ve bir silme (Delete) butonu bulunuyor. Kullanıcı isterse, kendi yüklediği videoları kolayca silebiliyor.

Bu profil sayfası sayesinde kullanıcılar, hem kendi bilgilerini hem de platformdaki aktif içeriklerini merkezi ve kullanıcı dostu bir arayüz üzerinden yönetebiliyor. Arayüz tasarımında sadelik ve modernlik ön planda tutuldu, kullanıcı deneyimi için tüm bilgilerin anlaşılır şekilde sunulmasına özen gösterildi.

KONTROL:

| Yapılan İş: Planların Sıralanması ve Seçim Fonksiyonlarının Backend ile Entegrasyonu | Sayfa No:26 |
| --- | --- |
| | Tarih: 18.08.2025 |

Bugünkü staj günümde, kullanıcıların token ve normal (abonelik) planlarını kolayca sıralayabilmesi, seçebilmesi ve bu seçimlerin backend'e başarılı bir şekilde iletilip işlenebilmesini sağladım.

Arayüzün üst kısmında, kullanıcının abone olduğu veya olabileceği kanallar gösteriliyor. Her kanalın yanında, abone olunan veya olunabilecek durumu ve ilgili aksiyon butonları yer almakta. Kullanıcı istediği kanalı seçebiliyor, bu seçimler ödeme sepetine ekleniyor.

Alt bölümde ise "Subscription Plan" ve "Token Plan" başlıkları altında iki farklı plan tipi detaylı şekilde listeleniyor:
- Subscription Plan: Kullanıcıya platformun premium özelliklerinden yararlanabilmesi için abonelik sunuyor. Premium plan aktif değilse, gelişmiş özelliklere erişilemiyor.
- Token Plan: Kullanıcı, abone olduktan sonra ekstra token paketleri satın alabiliyor. Token planlarını seçmek için önce abonelik planının aktif olması gerekiyor. Her token planı için açıklama, fiyat ve alınacak token miktarı görünüyor.

Kullanıcı, istediği plan veya planları seçip "Payment Basket" (Ödeme Sepeti) kısmında toplam tutarı görebiliyor ve ödemeyi başlatabiliyor. Seçilen planlar ve ödeme detayları, backend'e gönderilerek ilgili işlemler güvenli şekilde gerçekleştiriliyor. Kullanıcının yaptığı seçimler, ödeme tamamlanınca backend tarafından kaydediliyor ve kullanıcının hesabına planlar yansıtılıyor.

Bu geliştirme ile birlikte, kullanıcıların plan ve token alışverişi sırasında yaşadığı deneyim hem kolaylaştı hem de tüm işlemler güvenli ve hatasız şekilde tamamlanabilir hale geldi.

KONTROL:

| Yapılan İş: Abonelik Yönetimi, Faturalandırma ve Planlar Ekranının Tamamlanması | Sayfa No:27 |
|---|---|
| | Tarih: 19.08.2025 |

Bugünkü staj günümde, kullanıcıların abonelik ve token işlemlerini yönetebileceği, eksiksiz ve modern bir arayüz olan “Subscription Management” ve “Plans And Token” sayfalarını tamamen bitirdim.

Sayfanın ana başlıkları ve işlevleri:

1. Subscription Management (Abonelik Yönetimi):
- Kullanıcıların mevcut abonelik planını (ör. Free Plan) görebildiği, planla ilgili açıklama ve avantajların yer aldığı bir kart tasarlandı.
- Kullanıcılar, Free Plan aktif durumdayken buton üzerinden mevcut durumunu görebiliyor ve gerekirse Premium’a yükseltme işlemlerini başlatabiliyor.
- Sağ üstteki “Billing History” (Faturalandırma Geçmişi) bölümünde, kullanıcının geçmiş ödemeleri, tutarları ve tarihleri listeleniyor. Her ödeme için detaylı fatura (PDF vb.) indirilebiliyor.

2. Plans And Token (Planlar ve Tokenlar):
- Kullanıcılar abone oldukları kanalları sol üstteki listeden görebiliyor ve yeni kanal seçimleri yapabiliyor.
- “Payment Basket” (Ödeme Sepeti) kısmında, seçilen planlara göre toplam ödeme anlık olarak hesaplanıp, tek tıkla ödeme işlemi başlatılabiliyor.
- Sayfanın alt bölümünde iki ana plan türü detaylı şekilde gösteriliyor:
  - Subscription Plan:nPremium abonelik planı, kullanıcıya platformun gelişmiş özelliklerini ve içerik yönetimini sunuyor.
  - Token Plan: Sadece Premium kullanıcıların ulaşabildiği, ekstra token paketleri. Her bir planın fiyatı, sağladığı token miktarı ve açıklaması net biçimde belirtiliyor.

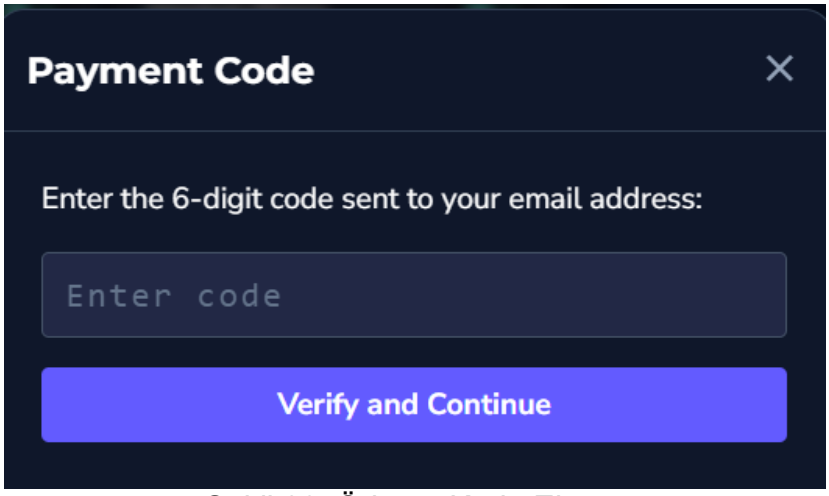
Bu ekranların tamamlanması ile birlikte, kullanıcılar platformda aboneliklerini ve token işlemlerini kolayca yönetebiliyor, ödeme ve fatura geçmişine ulaşabiliyor ve platformun tüm önemli ticari akışı eksiksiz olarak görsel ve işlevsel bir arayüze kavuşmuş oldu.

KONTROL:

| Yapılan İş: Ödeme Öncesi E-Posta Doğrulama ve Kod ile Güvenli Onay Akışı | Sayfa No:28 |
|---|---|
| | Tarih: 20.08.2025 |

Bugünkü staj günümde, kullanıcıların ödeme işlemi yapmadan önce e-posta adresleri üzerinden bir doğrulama adımı ekledim. Kullanıcı ödeme işlemini başlatınca, sistem tarafından kullanıcının e-posta adresine 6 haneli bir doğrulama kodu gönderiliyor. Kullanıcı, ödeme ekranında bu kodu girerek “Verify and Continue” butonuna tıklıyor ve ancak kod doğrulandıktan sonra ödeme işlemi devam edebiliyor.

Bu geliştirme ile birlikte:
- Kullanıcıların ödeme öncesinde gerçekten kendilerine ait bir e-posta adresi kullanıp kullanmadığı kontrol ediliyor.
- Kodun doğruluğu backend tarafında kontrol ediliyor, yanlış kod girildiğinde işlem başlatılamıyor.
- Bu güvenlik adımı sayesinde ödemelerde dolandırıcılık ve yetkisiz erişim riskleri azaltılmış oldu.




Şekil 20. Ödeme Kodu Ekranı

Kullanıcı deneyimini artırmak için, doğrulama kodu gönderildiğine dair bilgilendirme mesajları ve kolay anlaşılır bir arayüz tasarlandı. Kullanıcılar için ödeme akışı hem daha güvenli hem de daha profesyonel bir hale getirildi.

Günün sonunda, platformda ödeme işlemlerinin güvenliğini bir üst seviyeye çıkaran önemli bir adım tamamlanmış oldu.

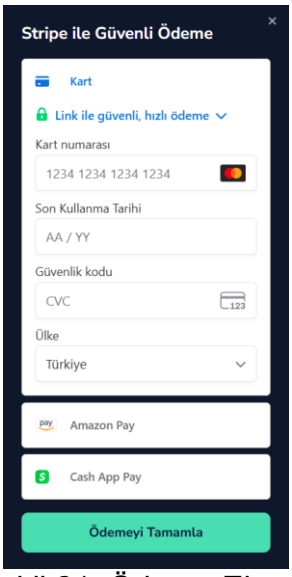
KONTROL:

| Yapılan İş: Stripe ile Güvenli Ödeme Entegrasyonu ve Seçili Planların Ödeme Akışına Aktarılması | Sayfa No:29<br>Tarih: 21.08.2025 |
|---|---|

Bugünkü staj günümde, kullanıcıların seçtiği token ve abonelik planlarının ödeme ekranına sorunsuz şekilde aktarılmasını ve Stripe ile güvenli ödeme entegrasyonunu tamamladım.

Geliştirdiğim ödeme ekranında, kullanıcılar seçili planları ödemeye geçtiğinde Stripe'ın sunduğu kart, Link, Amazon Pay ve Cash App Pay gibi farklı ödeme seçenekleri ile işlemlerini kolayca tamamlayabiliyorlar. Kullanıcıdan kart numarası, son kullanma tarihi, güvenlik kodu ve ülke bilgisi gibi bilgiler alınıyor ve Stripe API üzerinden ödeme işlemi güvenli şekilde gerçekleştiriliyor.

Stripe ile entegrasyonu kurarken:
- Kart bilgileri hiçbir şekilde sistemde saklanmadan doğrudan Stripe altyapısına aktarılıyor.
- Kullanıcılar yanlış veya eksik bilgi girdiğinde gerekli doğrulamalar ve hata mesajları gösteriliyor.
- Tüm ödeme süreci hem frontend hem de backend tarafında güvenli ve hızlı bir şekilde yönetiliyor.



Şekil 21. Ödeme Ekranı

Ayrıca, ödeme işlemi tamamlandığında seçilen token veya abonelik planının kullanıcının hesabına anında tanımlanmasını sağladım. Böylelikle kullanıcı deneyimi sorunsuz ve güvenli hale geldi.

Günün sonunda, platformun ticari altyapısı için kritik olan güvenli ödeme akışını ve Stripe ile tam entegre çalışan modern bir ödeme ekranını başarıyla devreye aldım.

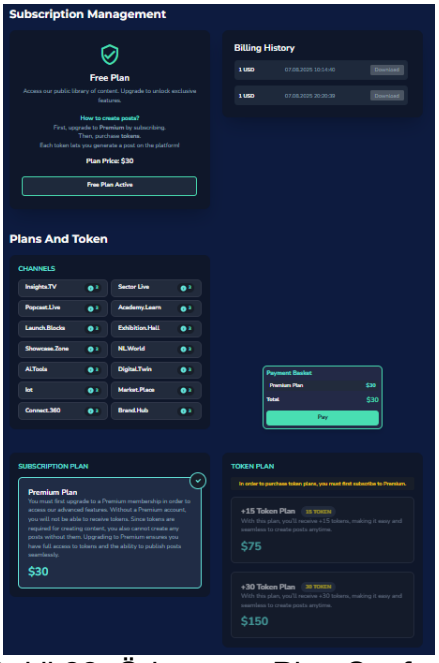
KONTROL:

| Yapılan İş: Subscription Sayfasında Veri Akışının ve Kayıtlarının Düzeltilmesi | Sayfa No:30<br>Tarih: 22.08.2025 |
|---|---|

Bugün subscription (abonelik) sayfasında mevcut olan tüm eksik ve hatalı veri akışlarını analiz ederek düzelttim. Kullanıcıların plan ve token seçimleri, ödeme işlemleri, kanal seçimleri gibi fonksiyonlarda zaman zaman veri eksikliği veya yanlış bilgi kaydı gibi sorunlar oluşuyordu. Bu problemleri tespit edip, backend ve frontend tarafında gerekli düzenlemeleri yaparak artık verilerin çok daha doğru ve eksiksiz şekilde kaydedilmesini ve kullanıcı arayüzünde sorunsuz şekilde görüntülenmesini sağladım.

Yaptığım geliştirmelerle birlikte, kullanıcıların yaptığı seçimler ve ödemeler sonrası bilgiler anında güncelleniyor ve arayüzde herhangi bir tutarsızlık kalmıyor. Ayrıca geçmiş fatura kayıtları, mevcut plan durumu ve kanal abonelikleri de doğru şekilde gösteriliyor. Böylece platformun abonelik yönetimi ve ticari veri akışı çok daha güvenli, hızlı ve hatasız bir hale geldi.

Aşağıdaki görselde subscription sayfasının güncellenmiş ve eksikleri giderilmiş son halini görebilirsiniz:




Şekil 22. Ödeme ve Plan Sayfası

KONTROL:

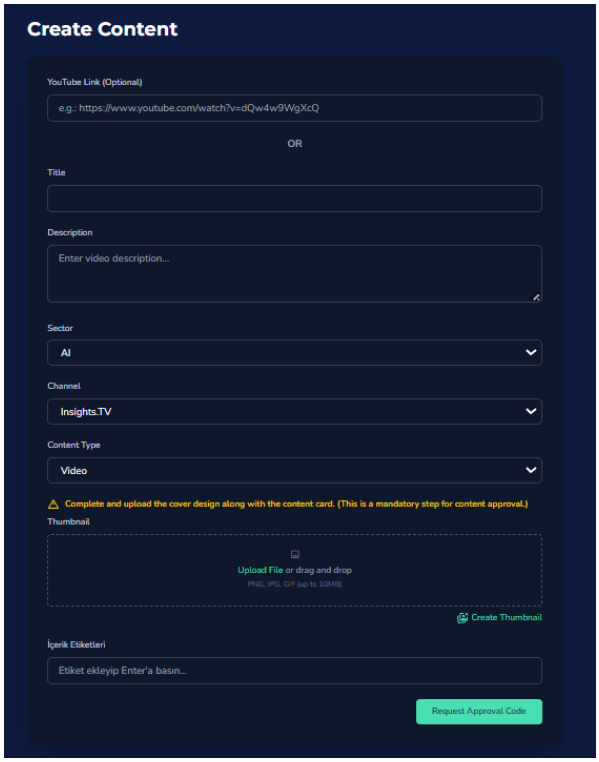
| Yapılan İş: Video Oluşturma Sayfası ve Token ile İçerik Üretimi Akışı | Sayfa No:31 |
|---|---|
| | Tarih: 25.08.2025 |

Bugün kullanıcılar için video oluşturma (content creation) sayfasını geliştirdim. Kullanıcılar, bu sayfa üzerinden YouTube linki ekleyerek veya doğrudan başlık, açıklama, sektör ve kanal seçerek kendi videolarını oluşturabiliyorlar. Ayrıca içerik türü olarak video seçimi yapılabiliyor ve onay süreci için gerekli olan kapak (thumbnail) görseli de yüklenebiliyor. İçeriklere etiketler eklenerek arama ve kategorilendirme fonksiyonları güçlendirildi.

Sayfanın en önemli kısmı, içerik oluşturulurken kullanılan token sistemi oldu. Kullanıcılar, video oluşturmak için sahip oldukları tokenlardan harcıyorlar ve içerik üretimi için token bakiyelerinin yeterli olması gerekiyor. Tüm bu süreçte, kullanıcıdan alınan bilgiler backend'e doğru ve eksiksiz şekilde iletiliyor ve içerik onay kodu istenerek güvenli bir içerik gönderim süreci sağlanıyor.

Bu geliştirme sayesinde, hem içerik üretimi tamamen kullanıcı dostu ve profesyonel bir arayüze kavuştu hem de içeriklerin platform kurallarına uygun şekilde ve güvenli olarak oluşturulması sağlandı.

Aşağıdaki görsellerde video oluşturma sayfasının arayüzü yer almaktadır:



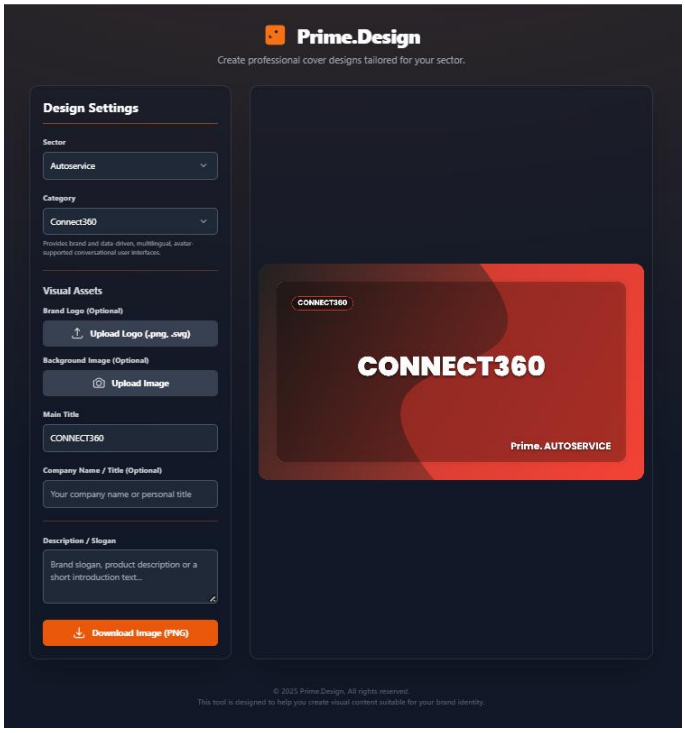

Şekil 23. Post Oluşturma Sayfası

KONTROL:

| Yapılan İş: Kullanıcı Postları için Thumbnail Tasarım Sayfası Geliştirme | Sayfa No:32 |
|---|---|
| | Tarih: 26.08.2025 |

Bugün, kullanıcıların kendi oluşturacakları postlara özel thumbnail (kapak görseli) tasarlayabilmeleri için https://thumbnailcreate.uweb4.com/ adresindeki özel sayfayı geliştirdim. Bu sayfa, kullanıcının seçtiği sektöre ve kategoriye göre profesyonel kapak görselleri oluşturmasına imkan tanıyor. Kullanıcılar, marka logosu ve arka plan görseli gibi görsel ögeleri yükleyebiliyor, ana başlık ve isteğe bağlı olarak şirket adı ya da kişisel unvan gibi bilgileri girebiliyorlar. Ayrıca, ürün açıklaması veya slogan gibi alanlara da içerik eklenebiliyor.

Sayfanın sağ tarafında, yapılan tüm değişiklikler anlık olarak büyük bir önizleme alanında gösteriliyor. Böylece kullanıcı, oluşturduğu tasarımın son halini görerek istediği gibi düzenleyebiliyor. Tasarım tamamlandığında ise "Download Image (PNG)" butonuna tıklayarak görseli yüksek kaliteli PNG formatında bilgisayarına indirebiliyor. Şekil 24' te görülmektedir.




Şekil 24. Fotoğraf Oluşturma Sayfası

Bu gelişmeyle birlikte, kullanıcılar artık postlarını öne çıkaracak profesyonel ve marka kimliğine uygun thumbnail'ları kolayca oluşturabiliyorlar. Bu da hem içeriklerin platformda daha dikkat çekici olmasını hem de kullanıcı deneyiminin artmasını sağladı.

KONTROL:

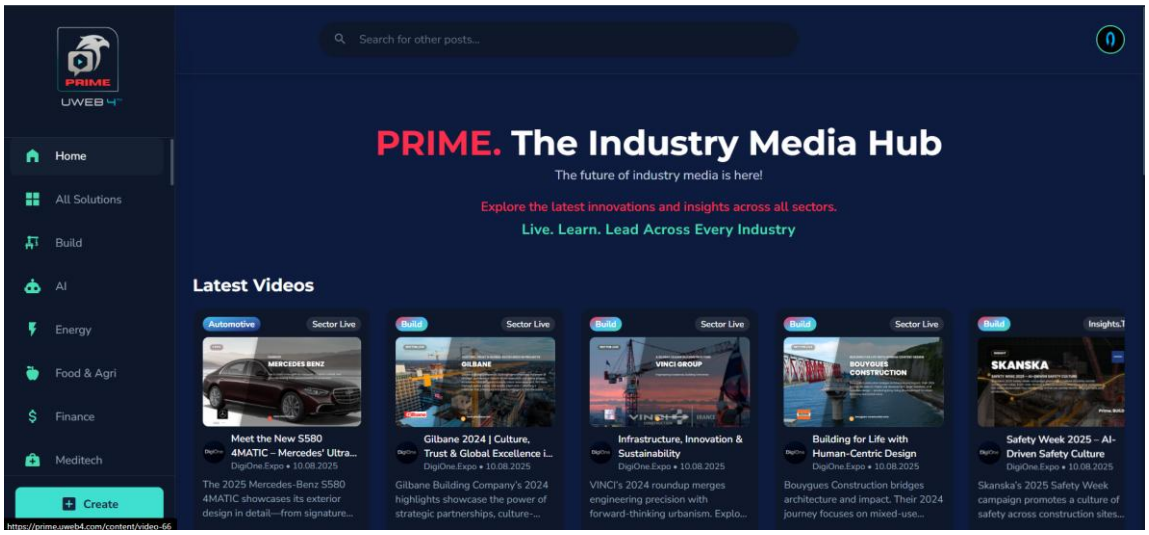
| Yapılan İş: Ana Sayfa Tasarımı ve Postların Kartlar Halinde Gösterilmesi | Sayfa No:33 |
|---|---|
| | Tarih: 27.08.2025 |

Bugün platformun ana sayfa tasarımını tamamladım. Ana sayfa artık kullanıcıya modern ve profesyonel bir arayüz sunuyor. Sol tarafta sektörel kategorilerin ve navigasyonun bulunduğu bir menü yer alıyor; kullanıcı istediği kategoriye hızlıca erişebiliyor. Sayfanın üst kısmında arama çubuğu ve dikkat çekici bir başlık bölümü var. Burada platformun sektörler arası bir medya merkezi olduğu vurgulanıyor.

Ana sayfanın en önemli kısmı ise, kullanıcıların en güncel paylaşımlarını ve videolarını kartlar halinde görebildiği “Latest Videos” bölümü oldu. Her post, görseliyle birlikte bir kart üzerinde başlık, özet, yayın tarihi ve hangi sektöre ait olduğu gibi bilgilerle gösteriliyor. Bu şekilde kullanıcılar, ilgi çekici içeriklere kolayca ulaşabiliyor ve farklı sektörlerden öne çıkan gelişmeleri anında takip edebiliyorlar.

Yaptığım bu tasarımla birlikte ana sayfa hem daha dinamik hem de kullanıcı dostu bir yapıya kavuştu. İçeriklerin kartlar halinde sunulması ise kullanıcı deneyimini ve platformun profesyonelliğini önemli derecede artırdı.

Aşağıda ana sayfa tasarımının son halini görebilirsiniz:



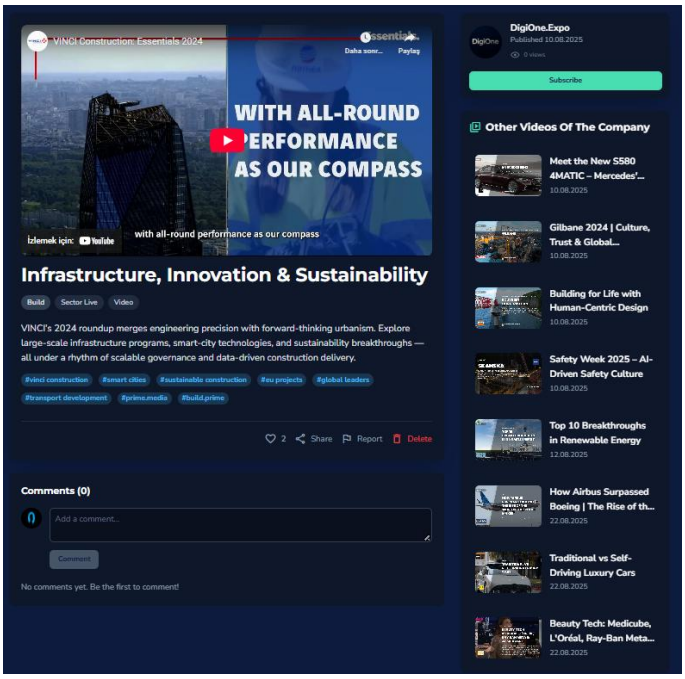

Şekil 25. Ana Sayfa

KONTROL:

| Yapılan İş: Postlar için Detay Sayfası Tasarımı ve Gerçek Veritabanı Entegrasyonu | Sayfa No:34 |
|---|---|
| | Tarih: 28.08.2025 |

Bugün, platformdaki postlar için detay sayfasını geliştirdim. Bu sayfa, seçilen bir postun tüm detaylarını ve ilişkili verileri kullanıcıya sunuyor. Artık postun başlığı, açıklaması, etiketleri, video içeriği ve ilgili diğer tüm bilgiler doğrudan veritabanından çekilerek dinamik olarak ekrana getiriliyor. Kullanıcılar, gerçek zamanlı olarak post detaylarını, etiketleri ve açıklamaları görebiliyorlar.

Ayrıca, detay sayfasında post sahibinin diğer videoları da sağ tarafta listeleniyor. Kullanıcılar bu alandan farklı videolara kolayca geçiş yapabiliyor. Postun altında, yorum yazma alanı da bulunuyor; burada kullanıcılar içerik hakkında görüşlerini paylaşabiliyorlar. Beğenme, paylaşma, raporlama veya silme gibi aksiyonlar da detay sayfasında erişilebilir durumda. Silme işlemini sadece adminler ve post sahibi kullanıcılar yapabilmektedr. Şekil 26' da post detay sayfası görülmektedir.



Şekil 26. Post Detay Sayfası

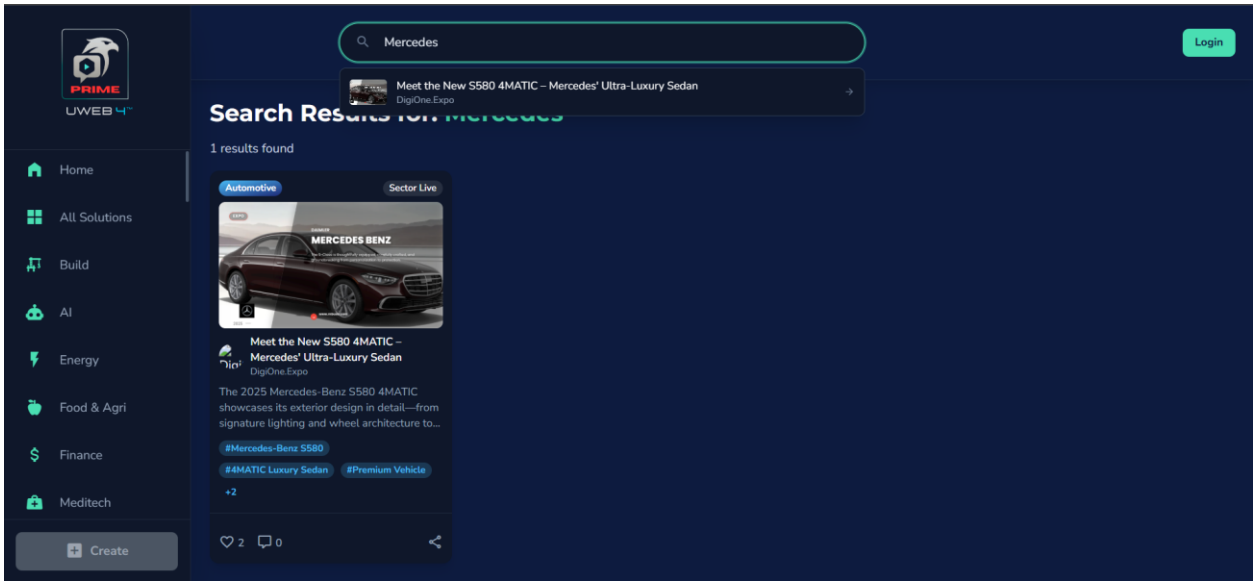
Bu geliştirme sayesinde platformdaki içeriklerin sunumu çok daha profesyonel ve dinamik bir hal aldı. Kullanıcılar artık gerçek verilerle güncel olarak oluşturulan, etkileşimli bir detay sayfası deneyimi yaşayabiliyor.

KONTROL:

| Yapılan İş: Arama Alanı (Search Bar) ve Sonuçların Dinamik Gösterimi | Sayfa No:35 |
|---|---|
| | Tarih: 29.08.2025 |

Bugün, platformda kullanıcıların anahtar kelimeyle arama yapabileceği arama alanını (search bar) geliştirdim. Artık kullanıcılar, üstteki arama kutusuna kelime girmeye başladıkça sistem otomatik olarak ilgili verileri getiriyor ve öneriler şeklinde kutunun altında gösteriyor. Kullanıcı Enter'a bastığında ise, arama yapılan kelimeyle ilgili tüm postlar aşağıda kartlar halinde listeleniyor.

Geliştirdiğim bu özellik sayesinde arama deneyimi hem hızlı hem de kullanıcı dostu bir hale geldi. Sonuçlar, anlık olarak veritabanından çekiliyor ve gerçek içeriklerle ekrana yansıtılıyor. Böylece kullanıcılar istedikleri bilgiye ya da paylaşıma kolayca ulaşabiliyorlar. Şekil 27' de arama sayfası gözükmektedir.




Şekil 27. Post Arama Sayfası

KONTROL: