Siliconmade Academy JT101 - Backend Grubu

Staj Öncesi Hazırlık Proje Görevi

Giriş:

Backend case study'mizde bir e-ticaret uygulaması gerçekleştireceğiz. Proje, parça parça ve kolayca gerçekleştirilebilmesi amacıyla 7 ana adım halinde hazırlanmıştır. Bu adımları sırasıyla takip ettiğinizde bazen projeye yeni bir kısım/özellik ekleyecek, bazen de projenin daha modüler bir hale gelmesi için daha önceden yaptığınız kısımlarda değişiklik yapacaksınız.

Bu klavuzda belirtilen katman, controller, action, view, servis, entity, dto, rol, senaryo vs. tüm kısımlar sadece size kapsamlı bir kılavuz olması amacıyla belirtilmiştir, *projenin ihtiyaç duyduğu diğer yapıları sizin tespit edip oluşturmanız beklenmektedir.*

Hepinize başarılar dilerim

ADIM-1

Amaç:

Bu adımda bir e-Ticaret uygulaması taslağı hazırlanacaktır. Aşağıdaki tablolarda verilen örneklere göre controller ve action'lar oluşturulacak. Uygulama içerisindeki view'lar ise aşağıda verilen template dosyaları yardımıyla oluşturulacaktır. Template'lerde ilgili view'a uygun bir sayfa yok ise genel tasarıma uygunluğu gözetilerek sizin tarafınızdan oluşturulabilir.

Buradaki controller, action ve view'lar örnek nitelikte olup projenin ihtiyacı olan diğer controller, action, view, partial view, view component gibi yapıları sizin eklemeniz gerekmektedir.

Analiz/Tasarım

Solution içerisinde e-Ticaret ve Admin olmak üzere iki farklı MVC projesi olmalıdır.
Aşağıdaki tablolarda bu MVC'lerde bulunması gereken örnek Controller ve Action'lar gösterilmektedir.

e-Ticaret MVC

Açıklama	Controller Action
Anasayfa	Home Index
Hakkımızda	Home AboutUs
İletişim	Home Contact
Ürün Listeleme	Home Listing
Ürün Görüntüleme	Home ProductDetail
Kullanıcı Kaydı	Auth Register
Kullanıcı Girişi	Auth Login
Şifremi Unuttum	Auth ForgotPassword
Kullanıcı Çıkış	Auth Logout
Kullanıcı Bilgileri Görüntüleme Kullanıcı Bilgileri Güncelleme Siparişlerim	Profile Details Profile Edit Profile MyOrders
Satıcı Kendi Ürünlerini Listeleme	Profile MyProducts
Ürün Oluşturma	Product Create
Ürün Düzenleme	Product Edit
Ürün Silme	Product Delete
Ürüne Yorum ve Yıldız Yapma Sepete Ürün Ekleme Sepetteki Ürünleri Düzenleme Sipariş Oluşturma Sipariş Detayları	Product Comment Cart AddProduct Cart Edit Order Create Order Details

Admin MVC

Açıklama	Controller Action
Anasayfa	Home Index
Kategori Oluşturma	Category Create
Kategori Düzenleme Kategori Silme	Category Edit Category Delete
Ürünü Silme	Product Delete
Kullanıcı Listeleme	User List
Satıcı Olma Onayı	User Approve

Yorum Listeleme	Comment List
Yorum Onaylama	Comment Approve

Route yapısında arama motoru dostu URL routing ayarlanabilir. Aşağıdaki tabloda örnek route yapıları gösterilmiştir. Uygun olacağını düşündüğünüz route tanımlamalarını diğer action'lara uygulayabilirsiniz.

Proje	Açıklama	Controller Action	Route
e-Ticaret	Ürün Görüntüleme	Home ProductDetail	/product/{categoryName}-{title}-{id}/details
e-Ticaret	Hakkımızda	Home AboutUs	/about-us
e-Ticaret	Sipariş Detayları	Order Details	/order/{id}/details
Admin	Kategori Düzenleme	Category Edit	/category/{id}/edit

Site Şablonları

- https://themewagon.com/themes/free-bootstrap-4-html5-responsive-ecommerce-website-template-ogani/
- https://startbootstrap.com/theme/sb-admin-2

Yukarıda verilen örnek şablonlar projede kullanılabilir. Bu şablonları indirerek kendi projelerinize dahil edebilirsiniz.

- Css, JS, Görseller gibi şablon dosyalarını wwwroot klasörü içerisine düzenli ekleyin.
- Html dosyalarından ilgili html kodlarını view'larınıza aktarın.
- Html'deki bağlantılarını (css, JS, görseller) wwwroot içerisindeki doğru dosya yollarına göre düzenleyin, tilde (~) sembolünü root için kullanın.
- Her sayfada tekrar eden header ve footer gibi kısımları layout içerisine alın.
- Gerekli gördüğünüz yerlerde partialview ve viewcomponent ekleyin.
- Projenizdeki view'ların doğru bir şekilde görüntülendiğini test edin.

ADIM-2

Amaç:

Bu adımda Entity Framework Core kullanarak MVC uygulamasına veri katmanı ekleyerek veritabanı işlemlerine yönelik tecrübe kazanmanız hedeflenmektedir. Aşağıda verilen tabloları kullanarak “Yapılacaklar” başlığı altındaki maddeleri uygulayabilirsiniz. *Bu tablolardaki senaryo, yetki ve entity’ler örnek nitelikte olup projenin ihtiyacı olan diğer senaryo, yetki ve entity’leri sizin eklemeniz gerekmektedir.*

Örnek Kullanım Senaryoları

Projede hangi kullanıcının hangi işlemleri yapabilecekleri aşağıdaki örnek tabloda verilmiştir, duruma göre yeni işlem ve yetkileri sizin eklemeniz gerekebilir:

No	Proje	Modül	Senaryo	Rol: Buyer	Rol: Seller	Rol: Admin
1	e-ticaret	Auth	Kullanıcı sisteme kayıt olabilecektir	✓	✗	✗
2	e-ticaret, admin	Auth	Kullanıcı sisteme email ve şifre ile giriş yapabilecektir	✓	✓	✓
3	e-ticaret, admin	Auth	Kullanıcı sistemden çıkış yapabilecektir	✓	✓	✓
4	e-ticaret, admin	Profile	Kullanıcı kendi profil sayfasını görüntüleyebilecektir	✓	✓	✓
5	e-ticaret	Profile	Kullanıcı, satıcı olmak isteğini talep edebilecektir	✓	✗	✗
6	e-ticaret	Cart	Kullanıcı stokta varsa ürünü sepetine ekleyebilecektir	✓	✓	✗
7	e-ticaret	Cart	Kullanıcı ürünü sepettinden çıkarabilecektir	✓	✓	✗
8	e-ticaret	Cart	Kullanıcı sepetteki ürününün adedini güncelleyebilecektir	✓	✓	✗
9	e-ticaret	Cart	Kullanıcı sepetinin detaylarını görüntüleyebilecektir	✓	✓	✗
10	e-ticaret	Order	Kullanıcı ödeme yaptıktan sonra adres bilgisi girecek, ardından siparişi otomatik oluşacaktır	✓	✓	✗
11	e-ticaret	Order	Kullanıcı siparişlerinin listesini görebilecektir	✓	✓	✗
12	e-ticaret	Order	Kullanıcı tek bir siparişinin detaylarını görebilecektir	✓	✓	✗
13	admin	Category	Kullanıcı sisteme yeni bir kategori ekleyebilecektir	✗	✗	✓
14	e-ticaret	Category	Kullanıcı bir kategori altındaki ürünleri listeyebilecektir	✓	✓	✗
15	e-ticaret	Product	Kullanıcı sisteme yeni ürün ekleyebilecektir	✗	✓	✗
16	e-ticaret	Product	Kullanıcı satıştaki kendi ürünlerini listeleyebilecektir	✗	✓	✗
17	e-ticaret	Product	Kullanıcı sistemdeki mevcut kendi ürününün fiyatını güncelleyebilecektir	✗	✓	✗

18 19	e-ticaret e-ticaret	Product Product	Kullanıcı sistemdeki mevcut kendi ürününün stoğunu güncelleyebilecektir ✗ ✓ Kullanıcı sipariş oluşturduktan sonra ürüne yorum ve yıldız verebilecektir ✓ ✓	✗ ✗
20 21	e-ticaret e-ticaret	Product Product	Kullanıcı tek bir ürünün detaylarını görebilecektir ✓ ✓ Kullanıcı genel ürün araması yapabilecektir ✓ ✓	✗ ✗
22	e-ticaret, admin	Product	Kullanıcı, ürünü pasif hale getirebilecektir ✗ ✓	✓
23	e-ticaret	ProductComment	Kullanıcı satın aldığı ürünlere yorum ve yıldız verebilecektir ✓ ✓	✗
24	admin	ProductComment	Kullanıcı, ürün yorumlarını herkesin görebilmesi için onaylayabilecektir ✗ ✗	✓
25	admin	User	Kullanıcı, sistemdeki tüm kullanıcıları listeleyebilecektir ✗ ✗	✓

26	admin	User	Kullanıcı, diğer kullanıcıları aktif/pasif yapabilecektir ✗ ✗	✓
27	admin	User	Kullanıcı, satıcı olma taleplerini onaylayabilecektir ✗ ✗	✓

Entity'ler

Projede olması gereken entity'ler aşağıda tablolar halinde belirtilmiştir. *İhtiyaç halinde kendi entity'lerinizi eklemeniz gerekecektir (Ör: Blog ve blog yorumlama altyapısı için gerekli entity'ler).*

UserEntity	
Property Adı	Type Rules
Id	int PK, Identity, required
Email	string Email, required
FirstName	string min:2, max:50, required
LastName	string min:2, max:50, required
Password	string min:1, required
RoleId	int RoleFK, required
Enabled	bool default:true, required
CreatedAt	DateTime required

RoleEntity

Property Adı	Type Rules
Id	int PK, Identity, required

Name string min:2, max:10, required

CreatedAt DateTime required

ProductEntity	
Property Adı	Type Rules
Id	int PK, Identity, required
SellerId	int UserFK, required
CategoryId	int CategoryFK, required
Name	string min:2 , max:100, required
Price	decimal data type:currency, required
Details	string max: 1000
StockAmount	byte required
CreatedAt	DateTime required

Enabled	bool required, default:true

ProductImageEntity	
Property Adı	Type Rules
Id	int PK, Identity, required
ProductId	int ProductFK, required
Url	string DataType:url, min:10, max:250, required
CreatedAt	DateTime required

ProductCommentEntity	
Property Adı	Type Rules
Id	int PK, Identity, required
ProductId	int ProductFK, required
UserId	int UserFK, required
Text	string min:2 , max:500, required

StarCount	byte min:1 , max:5, required
IsConfirmed	bool default:false, required
CreatedAt	DateTime required

CategoryEntity	
Property Adı	Type Rules
Id	int PK, Identity, required
Name	string min:2, max:100, required
Color	string min:3, max:6, required
IconCssClass	string min:2 , max:50, required
CreatedAt	DateTime required

CartItemEntity	
Property Adı	Type Rules
Id	int PK, Identity, required
UserId	int UserFK, required
ProductId	int ProductFK, required
Quantity	byte min:1, required
CreatedAt	DateTime required

OrderEntity	
Property Adı	Type Rules
Id	int PK, Identity, required
UserId	int UserFK, required
OrderCode	string min:2, required
Address	string min:2, max:250, required
CreatedAt	DateTime required

OrderItemEntity	
Property Adı	Type Rules
Id	int PK, Identity, required

OrderId	int OrderFK, required
ProductId	int ProductFK, required
Quantity	byte min:1, required
UnitPrice	decimal DataType:currency, required
CreatedAt	DateTime required

Yapılacaklar:

1. Adımda hazırlamış olduğunuz e-Ticaret uygulamasına aşağıdaki maddeler eklenerek devam edilecektir;

- Solution altına “App.Data” isminde bir class library projesi oluşturun. Entity ve DbContext class’larımızı bu proje altına ekleyeceğiz.
- Yukarıdaki tablolarda verilen örneklere göre gerekli entity class’larını tasarlayın, ilişkileri ile birlikte oluşturun.
- Uygulamada kullanılacak DbContext class’ını oluşturun ve OnModelCreating metodunu override ederek seed edilecek verileri aşağıdaki gibi belirleyin;
  - 3 adet rol (seller, buyer, admin),
  - 1 adet admin user’ı,
  - 10 adet ürün kategorisi (siz belirleyin)
- Uygun connection string'i hem e-ticaret hem de admin projelerindeki appsetting.json dosyalarına ekleyin.
- App.Data projesini, proje referansı olarak iki MVC projesine de ekleyin.
- İki MVC projelesinde de Program.cs dosyalarında DbContext'i projelere ekleyin (AddDbContext<TContext> metodu ile).
- e-ticaret uygulamasının Program.cs dosyasında “app.Run()” komutundan hemen önce EnsureCreated ya da EnsureCreatedAsync yöntemlerinden birisi ile veritabanının oluşturulması için gerekli kodları yazın.
- e-ticaret uygulamasını çalıştırarak veritabanının ve tabloların, oluşturduğunuz entity’lere göre doğru bir şekilde oluşturulduğundan emin olun.

ADIM-3

Amaç:

Bu adımda, 1. ve 2. adımlarda oluşturduğunuz controller, action, view ve entity’ler üzerine eklemeler yaparak devam edeceksiniz. Adım 3 ve 4 tamamlandığında action ve view’lar ile veritabanı işlemlerini birbirine bağlayarak CRUD işlemlerinin gerçekleştirilmesi amaçlanmıştır. Ayrıca kullanıcıdan alınan verilerin validasyonlardan geçirilerek kısıtlamalara uygunluğu kontrol edilecektir. 3. adımda controller ve action’ların bir bölümü, 4. adımda ise geri kalan kısımlar benzer şekilde tamamlanacaktır. Aşağıda 3. adımda tamamlanması beklenen kısımlar belirtilmiştir;

e-Ticaret MVC

Açıklama	Controller Action
Anasayfa	Home Index
Hakkımızda	Home AboutUs
İletişim	Home Contact
Ürün Listeleme	Home Listing
Ürün Görüntüleme	Home ProductDetail
Ürün Oluşturma	Product Create
Ürün Düzenleme	Product Edit
Ürün Silme	Product Delete
Ürüne Yorum ve Yıldız Yapma	Product Comment

Admin MVC

Açıklama	Controller Action
Anasayfa	Home Index
Kategori Oluşturma Kategori Düzenleme	Category Create Category Edit
Kategori Silme	Category Delete
Ürünü Silme	Product Delete

Yapılacaklar:

- DbContext'i ilgili Controller'ların constructor'larına parametre olarak ekleyin
- CRUD işlemi olan action'larda kullanıcıdan veri alınan yerlerde **"{ControllerAdı}{ActionAdı}Viewmodel"** formatına benzer şekildeki view model'ler aracılığıyla verinin alınmasını sağlayın (*model binding*). View modelleri proje içerisinde ayrı bir klasörde oluşturun (Models veya ViewModels klasörü olabilir)
- View modellerdeki property'lere, amacına uygun ve ilgili entity'nin validasyon kurallarıyla uyumlu validasyon attribute'larını ekleyin,
- View'larda validasyon sonuçlarının görüntülenebilmesi için ilgili "asp-validation-summary" kısımlarını ekleyin,

● Client side ve server side validasyon işlemleri için gerekli kodları ekleyin, ● İlgili action'larda, View model'lerden gelen bilgilerle gerekiyorsa entity ya da entity'ler oluşturup bu entity'ler ve DbContext aracılığıyla veritabanı işlemlerini gerçekleştirin ● View'a ilgili işlem sonucunun sadece mesaj olarak döneceği durumlarda ViewBag veya ViewData aracılığı ile ilgili mesajı ekleyip View'da görünmesini sağlayın (bootstrap'deki alert componenti bu iş için uygun olabilir)

- CRUD işlemi sonrası entity'nin son halinin View'da görünmesi gereken durumlarda ise sadece View'da görünecek alanları içeren bir ViewModel oluşturup bu model aracılığıyla View'da verilenin görünmesini sağlayın

İpuçları:

● View'larda, "@model" keyword'ü ile doğru ViewModel'in kullanılmasını sağlayabilirsiniz, ● View'lardaki form ve input elementlerinde tag helper'lar aracılığı ile model binding ve data validasyon işlemlerinin otomatik ve sorunsuz bir şekilde gerçekleştirilebilmesini sağlayabilirsiniz,

- Client-side validasyon için, yeni bir MVC projesi oluşturulduğunda otomatik olarak oluşan "Views>Shared" klasörü altındaki "_ValidationScriptsPartial.cshtml" partial view'ını gerekli yerlerde kullanmanız işinizi kolaylaştırabilir

ADIM-4

Amaç:

Bu adımda, 1. ve 2. adımlarda oluşturduğunuz controller, action, view ve entity'ler üzerine eklemeler yaparak devam edeceksiniz. Adım 3 ve 4 tamamlandığında action ve view'lar ile veritabanı işlemlerini birbirine bağlayarak CRUD işlemlerinin gerçekleştirilmesi amaçlanmıştır. Ayrıca kullanıcıdan alınan verilerin validasyonlardan geçirilerek kısıtlamalara uygunluğu kontrol edilecektir. 3. adımda controller ve action'ların bir bölümü, 4. adımda ise geri kalan kısımlar benzer şekilde tamamlanacaktır. Aşağıda 4. adımda tamamlanması beklenen kısımlar belirtilmiştir;

e-Ticaret MVC

Açıklama	Controller Action
Kullanıcı Kaydı	Auth Register
Kullanıcı Girişi	Auth Login
Şifremi Unuttum	Auth ForgotPassword
Kullanıcı Çıkış	Auth Logout
Kullanıcı Bilgileri Görüntüleme	Profile Details

Kullanıcı Bilgileri Güncelleme	Profile Edit
Siparişlerim	Profile MyOrders
Satıcı Kendi Ürünlerini Listeleme	Profile MyProducts
Sepete Ürün Ekleme	Cart AddProduct
Sepetteki Ürünleri Düzenleme	Cart Edit
Sipariş Oluşturma	Order Create
Sipariş Detayları	Order Details

Admin MVC

Açıklama	**Controller Action**
Kullanıcı Listeleme	User List
Satıcı Olma Onayı	User Approve
Yorum Listeleme	Comment List
Yorum Onaylama	Comment Approve

Yapılacaklar:

- DbContext'i ilgili Controller'ların constructor'larına parametre olarak ekleyin
- CRUD işlemi olan action'larda kullanıcıdan veri alınan yerlerde **"{ControllerAdı}{ActionAdı}Viewmodel"** formatına benzer şekildeki view model'ler aracılığıyla verinin alınmasını sağlayın (*model binding*). View modelleri proje içerisinde ayrı bir klasörde oluşturun (Models veya ViewModels klasörü olabilir)
- View modellerdeki property'lere, amacına uygun ve ilgili entity'nin validasyon kurallarıyla uyumlu validasyon attribute'larını ekleyin,
- View'larda validasyon sonuçlarının görüntülenebilmesi için ilgili "asp-validation-summary" kısımlarını ekleyin,
- Client side ve server side validasyon işlemleri için gerekli kodları ekleyin,
- İlgili action'larda, View model'lerden gelen bilgilerle gerekiyorsa entity ya da entity'ler oluşturup bu entity'ler ve DbContext aracılığıyla veritabanı işlemlerini gerçekleştirin ● View'a ilgili işlem sonucunun sadece mesaj olarak döneceği durumlarda ViewBag veya ViewData aracılığı ile ilgili mesajı ekleyip View'da görünmesini sağlayın (bootstrap'deki alert componenti bu iş için uygun olabilir)
- CRUD işlemi sonrası entity'nin son halinin View'da görünmesi gereken durumlarda ise sadece View'da görünecek alanları içeren bir ViewModel oluşturup bu model aracılığıyla View'da verilenin görünmesini sağlayın

İpuçları:

- View'larda, "@model" keyword'ü ile doğru ViewModel'in kullanılmasını sağlayabilirsiniz, ● View'lardaki form ve input elementlerinde tag helper'lar aracılığı ile model binding ve data validasyon işlemlerinin otomatik ve sorunsuz bir şekilde gerçekleştirilebilmesini sağlayabilirsiniz,
- Client-side validasyon için, yeni bir MVC projesi oluşturulduğunda otomatik olarak oluşan "Views>Shared" klasörü altındaki "_ValidationScriptsPartial.cshtml" dosyasını kullanmanız işinizi kolaylaştırabilir

ADIM-5

Amaç:

Bu adımda amacımız veri işlemlerini yapabileceğimiz ayrı bir katman oluşturmak ve web uygulamalarımızda bu katmandaki yapıları kullanmak olacaktır. Ayrıca projelerimize yetkilendirme altyapısını ekleyerek, rollere göre erişim kısıtlamaları yapacağız. Bu sayede bir kullanıcı sadece kendi rolünün yetkileri kapsamındaki işlemleri gerçekleştirebilecektir.

Yapılacaklar:

Data Katmanı için yapılacak işlemler:

- Data projesi içerisine "DataRepository<TEntity>" şeklinde bir generic repository servisi ekleyerek bu class aracılığıyla tüm entity'ler üzerinde CRUD işlemleri yapabilmesini sağlayan veritabanı işlemlerini bu repository class'ına metodlar olarak ekleyin.
- MVC projelerinde bu repository'nin kullanılabilmesi amacıyla iki MVC projesi için de Program.cs dosyalarında dependency injection ile DataRepository'leri ekleyin. ● Veritabanı işlemlerinin yapıldığı tüm action'larda doğrudan DbContext üzerinden işlem yapılması yerine, yeni eklediğimiz DataRepository class'ı üzerinden CRUD işlemlerini gerçekleştirmek için gerekli kod değişikliklerini yapın. Bu işlemler sonucunda hiçbir MVC Controller'ında DbContext kullanılmıyor, CRUD işlemleri sadece ve sadece DataRepository'ler üzerinden yapılıyor olmalıdır.
- Projenizi çalıştırıp veritabanı işlemi yapan tüm action'ların öncesinde olduğu gibi sorunsuz çalıştığını tekrar test edin.

Yetkilendirme için yapılacak işlemler:

- MVC projelerinin Program.cs dosyalarında Cookie Authentication için gerekli altyapıyı ekleyin.
- Login ve logout işlemlerinin yapıldığı action'larda cookie authentication ile giriş yapılmasını sağlayan kodları ekleyin.

- Aşağıda verilen yetkiler tablosuna göre kısıtlandırılması gereken tüm action ve controller'lara uygun şekilde "Authorize" attribute'unu ekleyerek sadece yetkisi olan kişilerin bu action'lara istek yapabilmesini sağlayın.
- Burada önemli bir husus da login işlemlerinin yapıldığı action'lara unauthorized bir şekilde istek yapılabilmesine dikkat edilmesidir.
- Her rol için bir kullanıcı seed edilecek şekilde veritabanınızı ayarlayın. ● Tüm roller için ayrı ayrı giriş yapıp sadece yetkisi olan action'lara istekte bulunabildiğini test edin.
- İlgili View, Partial View veya ViewComponent'lardaki menü seçeneklerinde sadece aktif kullanıcının rolünün yetkisi olduğu seçeneklerin görünmesini, diğer seçeneklerin arayüzde bulunmamasını sağlayacak değişiklikleri C# ve HTML kodlarını beraber kullanarak yapın.

Örnek Yetkiler Tablosu:

No	Proje	Modül	User Story	Role: Buyer	Role: Seller	Role: Admin
1	e-ticaret	Auth	Kullanıcı sisteme kayıt olabilecektir	✓	✗	✗
2	e-ticaret, admin	Auth	Kullanıcı sisteme email ve şifre ile giriş yapabilecektir	✓	✓	✓
3	e-ticaret, admin	Auth	Kullanıcı sistemden çıkış yapabilecektir	✓	✓	✓
4	e-ticaret, admin	Profile	Kullanıcı kendi profil sayfasını görüntüleyebilecektir	✓	✓	✓
5	e-ticaret	Profile	Kullanıcı, satıcı olmak isteğini talep edebilecektir	✓	✗	✗
6	e-ticaret	Cart	Kullanıcı stokta varsa ürünü sepetine ekleyebilecektir	✓	✓	✗
7	e-ticaret	Cart	Kullanıcı ürünü sepettinden çıkarabilecektir	✓	✓	✗
8	e-ticaret	Cart	Kullanıcı sepetteki ürününün adedini güncelleyebilecektir	✓	✓	✗
9	e-ticaret	Cart	Kullanıcı sepetinin detaylarını görüntüleyebilecektir	✓	✓	✗
10	e-ticaret	Order	Kullanıcı ödeme yaptıktan sonra adres bilgisi girecek, ardından siparişi otomatik oluşacaktır	✓	✓	✗
11	e-ticaret	Order	Kullanıcı siparişlerinin listesini görebilecektir	✓	✓	✗
12	e-ticaret	Order	Kullanıcı tek bir siparişinin detaylarını görebilecektir	✓	✓	✗
13	admin	Category	Kullanıcı sisteme yeni bir kategori ekleyebilecektir	✗	✗	✓

14	e-ticaret	Category	Kullanıcı bir kategori altındaki ürünleri listeyebilecektir ✓	✓	✗
15	e-ticaret	Product	Kullanıcı sisteme yeni ürün ekleyebilecektir ✗	✓	✗
16	e-ticaret	Product	Kullanıcı satıştaki kendi ürünlerini listeleyebilecektir ✗	✓	✗
17	e-ticaret	Product	Kullanıcı sistemdeki mevcut kendi ürününün fiyatını güncelleyebilecektir ✗	✓	✗
18	e-ticaret	Product	Kullanıcı sistemdeki mevcut kendi ürününün stoğunu güncelleyebilecektir ✗	✓	✗
19	e-ticaret	Product	Kullanıcı sipariş oluşturduktan sonra ürüne yorum ve yıldız verebilecektir ✓	✓	✗
20	e-ticaret	Product	Kullanıcı tek bir ürünün detaylarını görebilecektir ✓	✓	✗
21	e-ticaret	Product	Kullanıcı genel ürün araması yapabilecektir ✓	✓	✗
22	e-ticaret, admin	Product	Kullanıcı, ürünü pasif hale getirebilecektir ✗	✓	✓
23	e-ticaret	ProductComment	Kullanıcı satın aldığı ürünlere yorum ve yıldız verebilecektir ✓	✓	✗
24	admin	ProductComment	Kullanıcı, ürün yorumlarını herkesin görebilmesi için onaylayabilecektir ✗	✗	✓
25	admin	User	Kullanıcı, sistemdeki tüm kullanıcıları listeleyebilecektir ✗	✗	✓
26	admin	User	Kullanıcı, diğer kullanıcıları aktif/pasif yapabilecektir ✗	✗	✓
27	admin	User	Kullanıcı, satıcı olma taleplerini onaylayabilecektir ✗	✗	✓

ADIM-6

Amaç:

Bu adımda; mevcuttaki projelere iki tane daha WebAPI projesi ekleyerek e-Ticaret ve Admin MVC projelerinin bu API'leri kullanılması sağlanacaktır. Bu API projelerinden birisi DB CRUD işlemlerini gerçekleştirmek ve JWT oluşturmaktan, diğeri ise sadece dosya upload ve download işlemlerini yapmaktan sorumlu olacaktır. Böylelikle amacımız; Web API, JWT Authentication ve dosya upload/download işlemlerini gerçekleştirebilmek olacaktır.

Yapılacaklar:

Data API Projesi için yapılacak işlemler:

- Solution altına “App.Api.Data” isimli bir ASP.NET Core Web API projesi ekleyin. Projeyi tamamladığınızda bu proje hem e-Ticaret hem de Admin MVC projelerindeki tüm veritabanı işlemlerini, hem de JWT oluşturma işini üstlenecek olup MVC projeleri doğrudan veritabanına erişemeyecek, veritabanı ve JWT işlemleri için bu API projesine istekte bulunacaktır.
- Data katmanını oluşturan class library projesini bu API projesine proje referansı olarak ekleyin.
- Hem e-Ticaret hem de Admin projelerindeki controller action’larını gözden geçirerek daha önceden oluşturmuş olduğumuz DataRepository üzerinden veritabanı işlemi yapan tüm action’lardaki tespit edin.
- Tespit ettiğiniz veritabanı işlemi yapan tüm kodlar için yeni eklediğiniz Data API projesine uygun API Controller’ları oluşturun (ör: OrderController, ProfileController gibi).
- Data API içerisinde oluşturduğunuz bu controller’lar içerisine uygun action’lar ekleyerek MVC projelerindeki tüm veritabanı işlemlerini tek tek ilgili API action’ına taşıyın.
- Her bir taşıma işlemi sonrasında MVC projelerinde ilgili veritabanı işlemini yapan yerde DataRepository kodlarını kaldırarak yerine .NET Framework’de dahili olarak bulunan HttpClient class’ını kullanın ve yeni oluşturduğunuz API action’ına uygun HTTP metodu ve URL ile (gerekliyse response body’yi de ekleyerek) HTTP isteğinde bulunun. Cevap olaran dönen response’dan status kod ve response body ile MVC action’larının öncekine benzer şekilde sonuçlanmasını sağlayın.
- Bu işlemler tamamlandığında tüm MVC action’larını tek tek deneyerek önceki durumuyla aynı şekilde davrandığını test edin.

JWT için yapılacak işlemler:

- Data API (*App.Api.Data*) projesi içerisindeki Program.cs dosyasında Bearer JWT Authentication altyapısı için gerekli değişiklikleri yapın.

● Data API içerisine “AuthController” isimli bir controller oluşturun. ● Bu controller içerisine e-mail ve password alıp kullanıcı varsa ilgili kullanıcının kişisel bilgilerini ve rollerini claim olarak içerek bir token oluşturarak response olarak döndüren bir action ekleyin. e-mail ve password ikilisinin uyuştuğu bir kayıt yoksa 404-NotFound sonucu döndürün.

● Bu action’ı “AllowAnonymous” attribute’u ile niteleyin.

● Token oluşturma işlemini Postman ile test edip gelen response içerisindeki token’ı jwt.io sitesinde deneyerek ayarladığınız claim’lere sahip olduğundan emin olun. ● Data API controller ve/veya action’larına aşağıda verilen **Yetkiler Tablosuna** uygun şekilde “*Authorize*” attribute’u ile rol tabanlı yetkilendirmeleri ekleyin. ● Postman aracılığı ile API’den önce token alıp ardından token sahibinin yetkili olduğu ve olmadığı endpoint’lere isteklerde bulunup beklenildiği gibi response döndürüldüğünden emin olun (Ör: 200-OK, 201-Created veya 401-Unauthorized) ● e-Ticaret ve Admin MVC

projelerinden Data API projesine HttpClient ile yapılan isteklerde login olunduysa token'ın header içerisinde gönderilmesini sağlayın. Kılavuz olması için örnek bir senaryo aşağıdaki gibi olabilir:

a. e-Ticaret MVC'den login olunduğunda Data API'nin login endpoint'ine e-mail ve şifre ile POST isteği gönderilmesi,

b. Gelen cevap 404-NotFound ise View'da "e-mail veya şifre yanlış" mesajının gösterilmesi,

c. Gelen cevap 200-OK ise response body'den JWT okunarak MVC response'unda Cookie olarak set edilmesi,

d. Bu noktada Data API'den token alınmış ve MVC aracılığı ile tarayıcıda cookie olarak tutuluyor durumda olacaktır. Bu cookie, tarayıcı tarafından her istekte MVC uygulamasına otomatik olarak gönderileceğinden her istek öncesi araya girilip, bu cookie okunup token olarak ayarlanması gerekmektedir. Böylelikle MVC projelerindeki Authentication altyapısı, fazla bir değişiklik yapmadan eskisi gibi çalışmaya devam edecektir. Sadece Cookie authentication'ının JWT authentication'a çevrilmesi ve JWT authentication konfigürasyonundaki *OnMessageReceived* event'i ile her istekte JWT içeren cookie'nin varsa okunup Token olarak ayarlanması gerekecektir. (Bkz: https://spin.atomicobject.com/net-core-jwt-cookie-authentication/)

e. MVC projesinden logout yapıldığında sadece ilgili cookie'nin (JWT içeren) silinmesi yeterli olacaktır.

f. Bu adımların ardından Data API'ye HttpClient ile istek gönderen her yerde cookie'deki bu token, request oluşturulurken "Authorization: Bearer {token}" formatında request header'ına eklenecek şekilde değişiklikler yapılmalıdır. Böylelikle Data API de aynı token ile sadece yetkilendirilmiş isteklere cevap verebilecektir.

- Data API uygulaması hem e-Ticaret hem de Admin MVC projeleri için aynı action ile token üreteceğinden sizden:
  a. Seller ve Buyer rollerindeki kullanıcıların Admin projesine login olamaması,
  b. Admin rolündeki kullanıcıların da e-Ticaret projesine login olamaması

  için gerekli bir altyapı sağlamanız beklenmektedir.

File API Projesi için yapılacak işlemler:

- Solution altına "App.Api.File" isimli bir ASP.NET Core Web API projesi ekleyin. Projeyi tamamladığınızda bu proje, dosyaları kaydetmekle sorumlu olacak, istenildiğinde dosyaları response olarak döndürecektir.
- Bu projeye "FileController" isminde bir controller ekleyin.
- FileController içerisine "Upload" isminde bir action oluşturun. Bu action HTTP POST metodu ile çalışmalıdır.
- Upload action'ında dosyaların "IFormFile" arayüzü ile alınıp daha önceden belirlediğiniz bir klasöre kaydedilmesini sağlayın. Kaydedilme başarılıysa

201-Created response kodu ile cevap döndürün.

- FileController içerisine “Download” isminde bir action oluşturun. Bu action HTTP GET metodu ile çalışmalıdır.
- Download action’ı Query String olarak dosya adı almalıdır. Buna göre upload action’ında kayıt yapılan klasörde bu isimde bir dosya varsa “return File(...)” ile dosyayı response ile göndermelidir. Bu isimde bir dosya yoksa 404-NotFound cevabı verilmelidir.
- e-Ticaret ve Admin projelerinde ürün görselleri ve (varsa) profil resimlerinin kaydedilmesi ve View’larda gösterilmesi için bu endpoint’ler kullanılmalıdır. Bunun için MVC projesinde gerekli değişikleri yapın.
- Bu işlemlerin ardından satıcı (*Seller*) rolündeki bir kullanıcı ile bir ürüne görseller ekleyin. Ardından ürün sayfasında bu görsellerin File API’den alınarak görüntülenebildiğini test edin.

ADIM-7

Amaç:

Bu adımda şimdiye kadar yapılan projelerdeki business işlemlerini servisler aracılığıyla yapılacak şekilde düzenlenerek katmanlı mimariye uygun, modüler bir proje yapısı oluşturulması amaçlanmıştır. Servisler ve web uygulamaları arasındaki veri aktarımı DTO modelleri aracılığı ile olacağı için ilgili servis metotlarının dönüş tipi ve parametrelerinde kullanılacak DTO modelleri de oluşturulacaktır.

Yapılacaklar:

- MVC projelerindeki action’larda, API’lere istek yapan kodlar için kullanılan HttpClient class’ını dependency injection ile singleton olarak kullanacak şekilde güncelleyin. Bunun için Microsoft tarafından önerilen yöntem, HttpClient’ın *IHttpClientFactory* interface’i aracılığı ile kullanılmasıdır (Bkz: Referans 1)
- Solution altına “**App.Services**” isimli bir class library oluşturun. Bu class library projesi servis interface ve class’larını içerecektir.
- Oluşturduğunuz bu class library içerisine “**Abstract**” ve “**Concrete**” isimli iki klasör oluşturun. Abstract klasörü servis interface’lerini, Concrete klasörü de bu interface’lerin implementasyonları olan class’ları içerecektir.
- MVC ve servis projelerine **Ardalis.Result** NuGet paketini ekleyin. Oluşturulacak servis metotlarının döndürdüğü değerler bu kütüphane içerisindeki *Result* tipinde olacaktır. (Bkz: Örnek, Referanslar 2, 3)
- Solution altına **“App.Models.DTO”** isimli bir class library projesi ekleyin. Bu proje, DTO modellerini içerecektir. (Bkz: Referanslar 4, 5)
- **App.Services** projesinin proje referanslarına **App.Models.DTO** projesini ekleyerek

ileride burada oluşturulacak DTO modellerini servislerin kullanabilmesini sağlayın. ● MVC projelerindeki action metotlarındaki business kodları için **App.Services** projesindeki Abstract klasörü içerisinde servis interface'lerini, Concrete klasörü içerisine de bu interface'leri implemente eden somut servis sınıflarını oluşturun. Servis interface'lerini oluştururken benzer domain işlemlerini yapan kodları gruplayarak aynı interface içerisinde toplamaya özen gösterin. Bunun için aşağıda verilen **Yetkiler ve Modüller** tablosundaki modül sütununa göre interface'ler oluşturabilirsiniz. Servis metotlarının aldığı ve geri döndürdüğü DTO modellerini de App.Models.DTO class library projesi içerisine oluşturun. (Bkz: Örnek) ● Abstract ve concrete servisler ve DTO'lar oluşturulup proje referansları düzenlendikten sonra MVC projelerine bu servisleri DI (dependency injection) yapılmak üzere ekleyin. Bunun için ilgili projelerdeki Program.cs dosyalarında değişiklik yapılmalıdır.

- MVC controller action'larındaki işlemlerin, oluşturduğunuz servis interface'leri üzerinden yapılması için gerekli değişiklikleri yapın. Servis metotlarından dönen *Result* tipindeki sonuçları, ilgili action'ın dönüş tipine uygun olarak kullanın.
- API metodlarındaki kodlar için de benzer adımları uygulayarak abstract ve concrete servisleri ile DTO modellerini oluşturun, dependency injection için gerekli güncellemeleri yapın ve servislerden gelen Result tipindeki sonuçları, ilgili action'ın dönüş tipine uygun şekilde kullanın.

Örnek (ASP.NET Core 8.0):

App.Models.DTO projesi içerisinde:

```
// NewOrderRequest.cs dosyası:
public class NewOrderRequestDto {
        public List<NewOrderItemDto> Items { get; set; }
        public string DeliveryAddress { get; set; }
}

// NewOrderItemDto.cs dosyası:
public class NewOrderItemDto {
        public int ProductId { get; set; }
        public byte Quantity { get; set; }
}

// NewOrderResponseDto.cs dosyası:
public class NewOrderResponseDto {
        public int OrderId { get; set; }
}
```

App.Services projesindeki Abstract klasörü içerisinde:

```
// IOrderService.cs dosyası:
using Ardalis.Result;
```

```
public interface IOrderService {
        Task<Result<NewOrderResponseDto>> PlaceOrderAsync(string jwt,
                NewOrderRequestDto newOrderRequest);
        Task<Result<MyOrdersResponseDto>> GetMyOrders(string jwt);
        Task<Result<MyOrderDetailResponseDto>> GetMyOrderDetails(string jwt,
               MyOrderDetailRequestDto myOrderDetailRequest);
}
```

App.Services projesindeki Concrete klasörü içerisinde:

```
// OrderService.cs dosyası:
using System.Net.Http;
using Ardalis.Result;

public class OrderService : IOrderService {
        private readonly HttpClient client;
        public OrderService(IHttpClientFactory httpClientFactory) {
                client = httpClientFactory.CreateClient("DataApi");
        }

        // JWT header'ını ekleyerek istek yapan yardımcı metot (ayrı bir servis
şeklinde yapılarak DI ile de kullanılabilir):
        protected async Task<HttpResponseMessage> SendApiRequestAsync(string apiRoute,
                HttpMethod method,
                string jwt,
                object payload = null)
        {
                var httpRequestMessage = new HttpRequestMessage(
                        method,
                        apiRoute)
                        {
                            Headers =
                            {
                                { HeaderNames.Authorization, $"Bearer {jwt}" }
                            }
                        };
                if (payload is not null) {
                        httpRequestMessage.Content = JsonContent.Create(payload);
                }

                return await client.SendAsync(httpRequestMessage);
        }

        public async Task<Result<NewOrderResponseDto>> PlaceOrderAsync(string jwt,
                NewOrderRequestDto newOrderRequest)
        {
```

```
                var response = await SendApiRequestAsync("api/order",
                        HttpMethod.Post,
                        jwt,
                        newOrderRequest);

                if (!response.IsSuccessStatusCode) {
                         // Başarısız işlemler için uygun erken dönüş (early return)
                       // ör 1: Result.Unauthorized();
                        // ör 2: Result.NotFound();
                }
                // Başarılı işlem:
                var newOrderResponse = await
                response.Content.ReadFromJsonAsync<NewOrderResponseDto>();

                return Result.Success(newOrderResponse);
        }

        // interface implementasyonu devamı ...
}
```

Referanslar:

1. Microsoft: Make HTTP requests using IHttpClientFactory in ASP.NET Core
2. Github: Ardalis.Result
3. Ardalis.Result Kullanımı: Getting Started With Ardalis.Result
4. Wikipedia: Data transfer object
5. Microsoft: Create Data Transfer Objects (DTOs)