ISE309 - WEB PROGRAMLAMA

Proje Ödevi: ASP.NET Core ile Web Uygulaması Geliştirme

1. Ödevin Amacı

Bu projenin amacı, öğrencilerin ASP.NET Core platformunu kullanarak bir web uygulamasını baştan sona geliştirmelerini sağlamaktır. Öğrenciler, Entity Framework Core (EF Core) kullanarak bir veritabanını yönetecek, ASP.NET Core Identity ile kullanıcı doğrulama (Authentication) ve yetkilendirme (Authorization) mekanizmalarını uygulayacak, MVC (Model-View-Controller) mimarisini etkin bir şekilde kullanacak ve projelerini Git ile versiyonlayacaklardır.

2. Genel Teknik Gereksinimler (Tüm Projeler İçin Zorunludur)

- **Platform:** ASP.NET Core (Güncel LTS sürümü tercih edilmelidir).
- **Mimari:** MVC (Model-View-Controller) mimarisi kullanılacaktır.
- **Veritabanı Erişimi:** Entity Framework Core (EF Core) kullanılacaktır.
  - Mutlaka **Code-First** yaklaşımı tercih edilmeli ve **Migrations** yapısı kullanılmalıdır.
  - Veritabanı olarak SQLite, MS SQL Server veya PostgreSQL kullanılabilir.
- **Güvenlik:** ASP.NET Core Identity kullanılacaktır.
  - Kullanıcı **kayıt (Register)** ve **giriş (Login)** işlemleri (Authentication) zorunludur.
  - Mutlaka **rol bazlı yetkilendirme (Role-based Authorization)** kullanılmalıdır (Örn: "Admin" ve "Kullanıcı" rolleri).
- **Arayüz:** Temel HTML/CSS ve **Bootstrap** (veya benzeri bir CSS framework) ile temiz ve kullanılabilir bir arayüz tasarlanmalıdır.

3. Proje Seçenekleri

Öğrenciler, aşağıda listelenen **dört (4) projeden yalnızca bir (1) tanesini** seçerek geliştirecektir. Projelerin temel işlevsellikleri ve rolleri tanımlanmıştır.

Seçenek 1: İkinci El Eşya Satış Platformu

- **Amaç:** Kullanıcıların ikinci el ürünlerini listeleyebildiği ve diğer kullanıcıların bu ürünlere göz atabildiği bir platform.
- **Roller:**
  - **Kullanıcı (User):** Profil oluşturabilir, ürün ekleyebilir (başlık, açıklama, fiyat, kategori, resim yükleme), kendi ürünlerini düzenleyebilir/silebilir. Diğer kullanıcıların ürünlerini listeleyebilir/filtreleyebilir.
  - **Admin:** Kategorileri yönetebilir (CRUD), uygunsuz bulunan ürünleri silebilir,

kullanıcıları yönetebilir.
- **Zorunlu Ek Özellik:** Resim yükleme (File Upload) işlevselliği.

Seçenek 2: Etkinlik Yönetim Portalı

- **Amaç:** Organizatörlerin etkinlikler (seminer, konser, atölye) oluşturabildiği ve kullanıcıların bu etkinliklere kayıt olabildiği bir portal.
- **Roller:**
  - **Organizatör (Organizer):** Yeni etkinlik oluşturabilir (tarih, yer, kontenjan, açıklama), kendi etkinliklerini yönetebilir, etkinliğe katılanların listesini görebilir.
  - **Kullanıcı (User):** Etkinlikleri listeleyebilir, filtreleyebilir, detaylarını görebilir ve kontenjanı dolmamış etkinliklere kayıt olabilir ("Katıl" butonu). Kendi katıldığı etkinlikleri görebilir.
  - **Admin:** Kategorileri ve etkinlik yerlerini (Venue) yönetebilir (CRUD), tüm etkinlikleri silebilir/düzenleyebilir.

Seçenek 3: Proje ve Görev Takip Sistemi

- **Amaç:** Ekiplerin projelerini ve bu projelere ait görevleri (task) yönetebileceği basit bir proje yönetim aracı.
- **Roller:**
  - **Proje Yöneticisi (Manager):** Proje oluşturabilir, projeye kullanıcı (Team Member) atayabilir. Proje içinde görevler oluşturabilir ve bu görevleri kullanıcılara atayabilir.
  - **Ekip Üyesi (Team Member):** Sadece kendisine atanan projeleri ve görevleri görebilir. Görevlerin durumunu güncelleyebilir (Örn: "Beklemede", "Yapılıyor", "Tamamlandı"). Görevlere yorum ekleyebilir.
  - **Admin:** Tüm projeleri, kullanıcıları ve görevleri yönetebilir. Kullanıcılara "Manager" veya "Team Member" rolü atayabilir.
- **Zorunlu Ek Özellik:** Görevlerin durum (Status) yönetimi (kanban benzeri).

Seçenek 4: Çevrimiçi Kurs Kayıt Sistemi

- **Amaç:** Eğitmenlerin kurslar açabildiği ve öğrencilerin bu kurslara kaydolabildiği bir e-öğrenme platformu taslağı.
- **Roller:**
  - **Eğitmen (Instructor):** Yeni kurs oluşturabilir (başlık, açıklama, kategori), kurslarına modüller/bölümler ekleyebilir (Örn: "Bölüm 1: Giriş"). Kurslarına kayıtlı öğrencileri listeleyebilir.
  - **Öğrenci (Student):** Kursları listeleyebilir, filtreleyebilir, detaylarını görebilir ve kurslara kaydolabilir. Kaydolduğu kursların detaylarını (modüllerini) görebilir.
  - **Admin:** Kategorileri yönetebilir (CRUD), tüm kursları düzenleyebilir/silebilir, kullanıcılara "Eğitmen" veya "Öğrenci" rolü atayabilir.

4. Uygulanması Gereken Zorunlu Teknik Konular (Checklist)

Projenizde aşağıdaki kavramların *tümü* uygulanmış olmalıdır:

1. **ASP.NET Core Identity:**
    - Register ve Login sayfaları.
    - [Authorize] attribute'ünün kullanımı (Örn: Sadece giriş yapanların "Profilim" sayfasını görmesi).
    - Rol bazlı yetkilendirme (Örn: [Authorize(Roles = "Admin")] attribute'ünün "Kategori Yönetimi" gibi sayfalarda kullanılması).
2. **Entity Framework Core:**
    - Tüm veri modelleri (Entities) için C# sınıflarının oluşturulması.
    - DbContext sınıfının yapılandırılması (Configuration).
    - Varlıklar arası ilişkilerin (One-to-Many, Many-to-Many) Code-First ile tanımlanması.
    - Veritabanı oluşturma ve güncelleme için Migrations kullanımı.
3. **Mimari ve Best Practices:**
    - **MVC Mimarisi:** Logic (Controller), Veri (Model) ve Görünüm (View) katmanlarının net ayrımı.
    - **Repository Pattern (veya Service Layer):** Controller'ların doğrudan DbContext kullanması *istenmemektedir*. Veri erişim mantığını ayırmak için bir Repository katmanı (veya Service katmanı) oluşturulması *şiddetle tavsiye edilir*.
    - **Dependency Injection (DI):** Repository/Service ve DbContext gibi servislerin Controller'lara Constructor Injection ile enjekte edilmesi.
    - **View Models (DTOs):** View'lara veri taşırken EF Core modellerinin (Entities) doğrudan gönderilmesi *istenmemektedir*. Bunun yerine, sadece View'ın ihtiyaç duyduğu verileri içeren View Model (veya DTO) sınıfları oluşturulmalı ve kullanılmalıdır.
4. **Temel İşlevsellik:**
    - Tüm temel varlıklar için (Örn: Category, Product, Event) **CRUD (Create, Read, Update, Delete)** işlemlerinin tam olarak uygulanması.
    - **Veri Doğrulama (Data Validation):** Model/View Model sınıflarında Data Annotations ([Required], [StringLength] vb.) kullanılması ve View tarafında asp-validation-for tag helper'ları ile hata mesajlarının gösterilmesi.

5. Proje Sunumu (Video Anlatım) - ZORUNLU

Proje kodları tamamlandıktan sonra, her öğrenci projesini anlatan bir video çekmelidir.

- Video **en fazla 15 dakika** olmalıdır.
- Videonun **başlangıcında** öğrencinin **yüzü net bir şekilde görünmeli** ve öğrenci **adını, soyadını ve öğrenci numarasını** sesli olarak söylemelidir.
- Videoda projenin çalışan demosu gösterilmeli ve kodun önemli kısımları (Örn: Identity kurulumu, Repository yapısı, EF Core ilişkileri, zorlu bir algoritma) açıklanmalıdır.
- Video, **YouTube**'a **"Liste Dışı" (Unlisted)** olarak yüklenmelidir. (Herkese açık veya özel olmamalıdır).
- Video linki, OBIS'e yüklenecek olan .txt dosyasına eklenecektir.

6. Değerlendirme Kriterleri

- **Gereksinimlerin Karşılanması (%30):** Seçilen projenin tüm temel işlevlerinin (CRUD,

roller vb.) eksiksiz ve doğru çalışması.
- **Mimari ve Kod Kalitesi (%20):** Repository Pattern/Service Layer kullanımı, Dependency Injection, View Model kullanımı, temiz ve okunabilir kod (Clean Code) prensipleri.
- **Güvenlik (%15):** Authentication (giriş/kayıt) ve Authorization (rol yönetimi) mekanizmalarının doğru ve güvenli bir şekilde uygulanması.
- **Git Kullanımı ve Commit Geçmişi (%10):**
  - Projenin ilk günden itibaren düzenli olarak commitlenmesi.
  - Commit mesajlarının anlamlı ve açıklayıcı olması.
  - Projenin tek bir commit ile yüklenmemiş olması.
- **Video Sunumu (%10):** Projenin açık, net ve kurallara uygun (süre, yüz görünürlüğü, ses) bir şekilde sunulması. Kodun ve uygulamanın anlaşıldığını göstermesi.
- **EF Core Kullanımı (%10):** Code-First, Migrations ve ilişkisel veri modelinin doğru kurgulanması.
- **Kullanıcı Arayüzü (UI/UX) (%5):** Temiz, anlaşılır ve Bootstrap (veya benzeri) ile tasarlanmış kullanılabilir bir arayüz.

7. Bonus Puanlama

- Zorunlu gereksinimlerin üzerine çıkarak projesine ek ve yaratıcı özellikler (Örn: Raporlama için bir dashboard oluşturma, AJAX ile dinamik sayfa güncellemeleri, zengin metin editörü (TinyMCE vb.) entegrasyonu, üçüncü parti bir API kullanımı vb.) ekleyen öğrencilere değerlendirme notuna ek olarak **bonus puan** verilecektir.
- Eklediğiniz bonus özellikleri video sunumunuzda ve README.md dosyanızda belirtmeniz gerekmektedir.

8. Teslimat

Teslimat iki aşamadan oluşmaktadır: GitHub ve OBIS.

1. GitHub Classroom (Zorlu):

- Tüm proje kodları GitHub üzerinde barındırılmalıdır.
- Her öğrenci, aşağıdaki GitHub Classroom linkini kullanarak kendi özel (private) repository'sini oluşturmalıdır:
  https://classroom.github.com/a/EJW4--AP
- Proje geliştirme süreci bu repository üzerinden yürütülecektir.
- Projenin ilk günden itibaren düzenli aralıklarla **commit** yapılması zorunludur.
- Repository'nin ana dizininde README.md dosyası bulunmalı ve projeyi çalıştırmak için gerekli talimatları (Örn: update-database komutu) ve varsa *bonus özellikleri* içermelidir.

2. Öğrenci Bilgi Sistemi (OBIS):

- OBIS'teki ilgili ödev teslim bölümüne, adınızın OgrenciNo.txt (Örn: B201210001.txt) olduğu bir .txt dosyası yüklenecektir.
- Bu .txt dosyası **sadece 3 satır** içermelidir:
  - **1. Satır:** Öğrenci Numarası Ad Soyad (Örn: B201210001 Ali Yılmaz)

- **2. Satır:** Projenin GitHub linki (Örn: https://github.com/web-prog-dersi/proje-ali-yilmaz)
- **3. Satır:** YouTube video anlatım linki (Liste dışı olmalıdır)

ÖNEMLİ UYARI:
Videonun yanlış yüklenmesi, linkin hatalı paylaşılması (liste dışı olmaması, özel olması vb.) veya başka bir sebeple görüntülenememesi durumunda mazeret kabul edilmeyecektir. Aynı şekilde, GitHub repository'sinin erişilebilir olmaması veya kodların paylaşılmaması durumunda proje değerlendirmeye alınmayacaktır.

Son Teslim Tarihi: 21.12.2025 23:59

(Bu tarihteki son GitHub commit'iniz ve OBIS yüklemeniz değerlendirmeye alınacaktır.)

Başarılar dilerim.