ANKARA ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BLM4531

AĞ TABANLI TEKNOLOJİLER VE UYGULAMALARI

HASTANE RANDEVU SİSTEMİ

Zeliha Kıyak

22290878

GitHub : zelihakiyak

VİDEO LİNKİ: https://youtu.be/_JQHUPqAkGk

1. PROJE ÖZETİ VE AMACI

Proje, üç farklı kullanıcı rolü (Admin, Doktor, Hasta) üzerine kurgulanmıştır. Her rolün kendine özgü yetki ve iş akışları bulunmaktadır. Sistemin temel amacı, randevu çakışmalarını önlemek, doktor verimliliğini artırmak ve hastalara 7/24 randevu alma imkanı sunmaktır. Backend tarafında ASP.NET Core MVC mimarisi kullanılmıştır.

2. VERİTABANI MİMARİSİ VE İLİŞKİLER

Sistem, ilişkisel bir veritabanı yapısı üzerine kurulmuştur. Veritabanı şeması, verinin tutarlılığını sağlamak için tasarlanmıştır. Veriler Code First ile migration edilmiştir. MSSQL tercih edilmiştir. Tablolar arası ilişkilerden birkaçı şu şekildedir:

Departments - Doctors (1:N): Bir poliklinikte birden fazla doktor görev yapabilir. Doctors tablosu, DepartmentId üzerinden bu ilişkiyi tutar.

Doctors - Appointments (1:N): Bir doktorun gün içinde birçok randevusu olabilir. Randevular, DoctorId ile doktorlara bağlanır.

Patients - Appointments (1:N): Bir hasta birden fazla randevu alabilir. PatientId ile ilişki kurulur.

Admins: Sistemin altyapısını yöneten, diğer tablolardan bağımsız yetki tablosudur.

3. YAZILIM MİMARİSİ

Projenin "Clean Architecture" (Temiz Mimari) prensiplerine uygun olması hedeflenmiştir. Swagger arayüzü için API klasörü üçünde ApiControllerlar oluşturulmuştur. Dependency Injection kullanarak sınıflar arası bağımlılık azaltıldı ve Single Responsibility prensibiyle her katmanın görevini net bir şekilde ayrıldı. Bu sayede kod çok daha profesyonel bir hale geldi

3.1. ViewModel Kullanımı

Domain modellerini (veritabanı sınıflarını) doğrudan arayüze göndermek yerine, sadece gerekli verileri içeren ViewModel sınıfları kullanılmıştır. Bu sayede şifre hash'leri gibi hassas veriler arayüze sızmaz ve model doğrulama işlemleri daha güvenli yapılır.

3.2. Service Katmanı

İş mantığı (Business Logic) Controller içerisinden çıkarılarak AppointmentService gibi servis sınıflarına taşınmıştır. Örneğin; boş randevu saatlerinin hesaplanması veya hafta sonu kısıtlaması bu katmanda yönetilir.

4. KULLANICI ROLLERİ VE FONKSİYONLAR

4.1. Admin Modülü

Admin departman ve doktor yönetimi kısmından önce departman ekleyip daha sonrasında doktor kaydı oluşturur. Departman silme işlemi silinecek olan departmana bağlı doktor yoksa gerçekleşecek şekilde düzenlenmiştir. Hasta yönetimi kısmında kullanıcıların kaydını silebilir. Tüm randevuları görüntüleme yetkisine sahiptir.

4.2. Hasta Modülü

Randevu al bölümünden seçtiği bölüme ait doktorları görüntüleyebilir. Seçtiği doktorun randevu almak istediği tarihteki müsaitlik durumuna göre randevu saatleri görüntülenir. Randevularım kısmından mevcut randevularını iptal edebilir. Geçmiş randevularını görüntüleyebilir.

Hafta sonları randevu alımı engellenmiştir. Ayrıca geçmişe dönük veya mesai saati dışındaki randevular filtrelenir.

4.3. Doktor Modülü

Gelecek randevular kısmında günün randevularıyla birlikte gelecekteki randevuları görüntüler. Aynı zamanda geçmişte kedisinden randevu almış hastaları da görüntüleyebilmektedir. Hastalarım bölümümden kendisinden randevu almış tüm hastaların bilgisini görüntüleyebilir.

4.4 Hesap İşlemleri

Şifrelerin BCrypt ile hash'lenmesi sürecinde hashleme ve doğrulama (verify) metodolojileri uygulanmıştır ve veritabanı güvenliği sağlanmıştır.

Tüm kullanıcılara şifre değiştirme özelliği eklenmiştir.

Rol bazlı giriş yetkilendirmesi yapılmıştır.

5. SONUÇ

Bu çalışma, modern yazılım geliştirme standartlarına uygun, güvenli ve ölçeklenebilir bir sistemin nasıl inşa edileceğini göstermektedir. Proje boyunca katmanlı mimari yönetimi, veritabanı ilişkilerinin kod tarafında doğru kurgulanması ve kullanıcı rollerine göre özelleştirilmiş arayüz tasarımları öğrenilen en kritik kazanımlar olmuştur.