

# Öğrenci Otomasyon Sistemi

**Amaç** Bu proje, stajyer adayların basit bir **öğrenci otomasyon sistemi** geliştirmelerini hedefler. Proje, temel CRUD işlemleri, kullanıcı yönetimi ve basit raporlamadan oluşacaktır. Adaylardan yazılım geliştirme disiplinlerini gösterebilmeleri, dokümantasyon ve versiyon kontrolü ile temiz kodlama alışkanlıklarını yansıtmaları beklenmektedir.

**Beklenen Çıktı:** Çalışır durumda bir fullstack uygulama (Backend + Frontend), GitHub repository'si ve kısa README.

---

## Kullanılacak Teknolojiler

- **Backend:** .NET 9
  - **ORM:** Entity Framework Core
  - **Frontend:** İstediğiniz frameworkü kullanabilirsiniz (React, Angular, Vue, Blazor vb.). Tercihimiz Blazor'dur.
  - **Veritabanı:** PostgreSQL
  - **Versiyon Kontrol:** GitHub
- 

## Proje Gereksinimleri

### 1. Kullanıcı Yönetimi

- Giriş (Login) ve kayıt (Register).
- Roller: Admin, Teacher, Student.

### 2. Öğrenci İşlemleri (CRUD)

- Admin ve öğretmen; öğrenci ekleyebilir, güncelleyebilir, listeleyebilir.
- Öğrenci kendi bilgilerini görüntüleyebilir.

### 3. Öğretmen İşlemleri (CRUD)

- Admin öğretmen ekleyebilir, güncelleyebilir, listeleyebilir.

### 4. Ders Yönetimi (CRUD)

- Admin ders oluşturabilir.
- Öğretmen kendi derslerini görebilir, durumlarını (ders başladı, ders bitti vb.) güncelleyebilir.
- Öğretmen, öğrencilerini yorumlayabilir.
- Öğretmen, derse öğrenci ekleyebilir, silebilir.

### 5. Not ve Devamsızlık

- Öğretmen, öğrencilerine ders bazında not ekleyebilir.
- Öğrenciler notlarını görebilir.
- Devamsızlık kaydı tutulabilir.

## 6. Frontend Sayfaları

- Login / Register ekranı.
  - Dashboard (rol bazlı içerik).
  - Öğrenci listesi ve öğrenci detay.
  - Öğretmen listesi.
  - Ders listesi.
  - Not/Devamsızlık sayfası.
- 

## Zorunlu Alanlar

- GitHub repository'si (public veya private, erişim paylaşılmalı).
  - Backend ve frontend ayrı klasörlerde olmalı.
  - README.md dosyası içinde:
    - Proje açıklaması.
    - Kurulum ve çalıştırma adımları.
    - Kullanıcı test bilgileri (ör. test admin, test öğrenci).
    - Yapılan bonus görevler
- 

## Notlar

- Güvenlik için parolalar hashlenmelidir.
  - Kodlar clean code prensiplerine uygun yazılmalıdır.
  - **Projeyi tamamen bitirmeniz değil, 11.09.2025 23:59 tarihine kadar kaydettiğiniz ilerleme önceliklidir. (Projeyi tüm isterleriyle tamamlanmanız ekstra puan olacaktır.)**
- 

## Bonus Görevler

- Projeyi Publish Etmek
  - Backend'i Azure, AWS veya Heroku'ya; frontend'i Netlify, Vercel veya GitHub Pages'e deploy etmek.
- Unit Test / Integration Test Yazmak
  - Önemli endpointler için xUnit, NUnit veya Jest kullanarak test eklemek.
- Docker Desteği
  - Backend ve veritabanını Docker Compose ile ayağa kaldırmak.
- Swagger / API Dokümantasyonu
  - Swagger/OpenAPI eklemek.
- CI/CD Entegrasyonu
  - GitHub Actions veya GitLab CI ile otomatik build/test pipeline kurmak.

- Doküman
    - Yapılan projeye ait doküman hazırlanması
  - Frontend iyileştirmeleri
    - Basit grafikler eklemek (ör. not ortalaması için Chart.js veya Recharts).
    - UI/UX düzenlemeleri (dark mode, responsive tasarım).
  - Ek Özellikler
    - Şifre resetleme (email simülasyonu ile).
    - Öğrenciye not ortalaması hesaplama ve raporlama ekranı.
    - Öğretmene “kendi derslerine ait öğrenci listesi” filtresi.
    - Kodlar clean code prensiplerine uygun yazılmalıdır.
  - Ek Bonuslar
    - Kullanıcı arama ve filtreleme özelliği eklemek.
    - Raporlama için CSV veya PDF export fonksiyonu eklemek.
    - Tema veya renk seçenekleri ekleyerek frontend özelleştirmesi.
    - Basit bildirim sistemi (örn. ders notu girildiğinde öğrenciye alert).
    - Kod kalite analizi (SonarCloud veya benzeri araçlarla).
- 

**Beklenti:** Basit ama uçtan uca çalışan bir öğrenci otomasyon sistemi teslim edilmelidir.