T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ

PROJE KONUSU:
WEB ARAYÜZÜNDEN OTOMATİK DERS PROGRAMI OLUŞTURMA

ÖĞRENCİ ADI:
ÖĞRENCİ NUMARASI:
Sema Su YILMAZ - 220502016
Zehra YARDIMCI - 220502038
Senem ADALAN - 220502045

DERS SORUMLUSU:
DR. ÖĞR. ÜYESİ ELİF PINAR HACIBEYOĞLU

TARİH:
7 MAYIS 2025

İÇİNDEKİLER

1. GİRİŞ
1.1 Projenin Amacı

Bu projenin amacı, mühendislik fakültesi bünyesindeki farklı bölümlere ait derslerin haftalık programını otomatik ve manuel olarak oluşturabilecek kullanıcı dostu bir web tabanlı sistem geliştirmektir. Django framework'ü kullanılarak geliştirilen bu sistemde, kullanıcılar farklı rollerle giriş yaparak kendi yetkileri doğrultusunda ders programlarını görüntüleyebilir, düzenleyebilir veya oluşturabilirler. Sistem, veri tabanında tanımlı olan ders, öğretim üyesi ve derslik bilgilerini kullanarak çeşitli kısıtlamaları (derslik kapasitesi, öğretim üyesi uygunluk durumu, ortak ders çakışmaları) dikkate alarak ders programlarını otomatik olarak oluşturmaktadır. Bunun yanında kullanıcılar tarafından manuel düzenlemeye de olanak tanır.

2. GEREKSİNİM ANALİZİ
2.1 Arayüz Gereksinimleri

1. Giriş Ekranı
- Kullanıcı adı ve şifre ile giriş yapılabilecek bir oturum açma arayüzü bulunmalıdır.
- Giriş yapan kullanıcı, rolüne (Koordinatör, Öğretim Üyesi, Öğrenci) göre kendi kontrol paneline yönlendirilmelidir.
- Hatalı girişlerde kullanıcıya uygun uyarı mesajı gösterilmelidir.

2. Rol Bazlı Kontrol Panelleri
- Koordinatör Paneli: Fakülte, bölüm, ders, öğretim üyesi, öğrenci, derslik ve haftalık program gibi tüm sistem bileşenlerini yönetebileceği bir arayüz sunulmalıdır.
- Öğretim Üyesi Paneli: Kendisine atanan ve diğer öğretim üyelerine ait haftalık ders programını görüntüleyebilmelidir. Seçtiği bölüm ve sınıfa ait haftalık ders programını görüntüleyebileceği bir arayüz sunulmalıdır.
- Öğrenci Paneli: Yaptığı ders seçimine göre haftalık ders programını takip edebileceği bir ekran sağlanmalıdır.

3. Ders Programı Görüntüleme Arayüzü
- Haftalık ders programı tablo formatında görselleştirilmelidir.
- Her hücrede ders adı, öğretim üyesi ve derslik bilgileri yer almalıdır.

4. Manuel Program Düzenleme Arayüzü
- Koordinatör, seçilen herhangi bir ders için mevcut programa uygun alternatif ders saatlerini ve derslikleri görüntüleyebilmeli ve programda manuel olarak yerleştirme yapabilmelidir.
- Yapılan değişiklikler kaydedilmelidir.

5. Ders, Öğretim Üyesi, Derslik ve Öğrenci Yönetimi Arayüzü
- Yeni ders, öğretim üyesi, derslik ve öğrenci ekleme bölümleri bulunmalıdır.
- Mevcut kayıtları listeleyen ve düzenleme/silme işlemlerine izin veren tablolar yer almalıdır.

6. Excel Çıktı Butonu
- Oluşturulan ders programı, Excel dosyası olarak indirilebilmelidir.

2.2 Fonksiyonel Gereksinimler

Kullanıcı Girişi ve Yetkilendirme (İG-1)

- Kullanıcılar (Koordinatör, Öğretim Üyesi, Öğrenci) sisteme doğrulama yaparak giriş yapabilmelidir.
- Her kullanıcı sadece yetkili olduğu modüllere erişebilmelidir.

Ders, Öğretim Üyesi, Derslik ve Öğrenci Yönetimi (İG-2)

- Koordinatör, sisteme yeni ders, öğretim üyesi, derslik ve öğrenci ekleyebilmelidir.
- Var olan ders, öğretim üyesi, derslik ve öğrenci bilgilerini güncelleyip silebilmelidir.
- Uygun formatta içe/dışa aktarma yapabilmelidir.

Otomatik Ders Programı Oluşturma (İG-3)

- Sistem öğretim üyesi uygunlukları, derslik kapasiteleri ve çakışma kontrollerine göre haftalık ders programını otomatik oluşturabilmelidir.
- Program oluşturma sonucunda Excel çıktısı üretilebilmelidir.

Manuel Ders Programı Düzenleme (İG-4)

- Koordinatör, otomatik olarak oluşturulan program üzerinde elle değişiklik yapabilmelidir.
- Kullanıcılar, dersin saatini ve sınıfını değiştirebilmelidir.

Ders Programını Görüntüleme (İG-5)

- Öğretim üyeleri, kendilerine ve diğer öğretim üyelerine ait haftalık ders programını görüntüleyebilmelidir.
- Öğretim üyeleri, bölüm ve sınıflara ait haftalık ders programını görüntüleyebilmelidir.
- Öğrenciler, ders seçimi yaparak seçtiği derslere haftalık ders programını görüntüleyebilmelidir.
- Koordinatör, tüm ders programına erişebilmelidir.

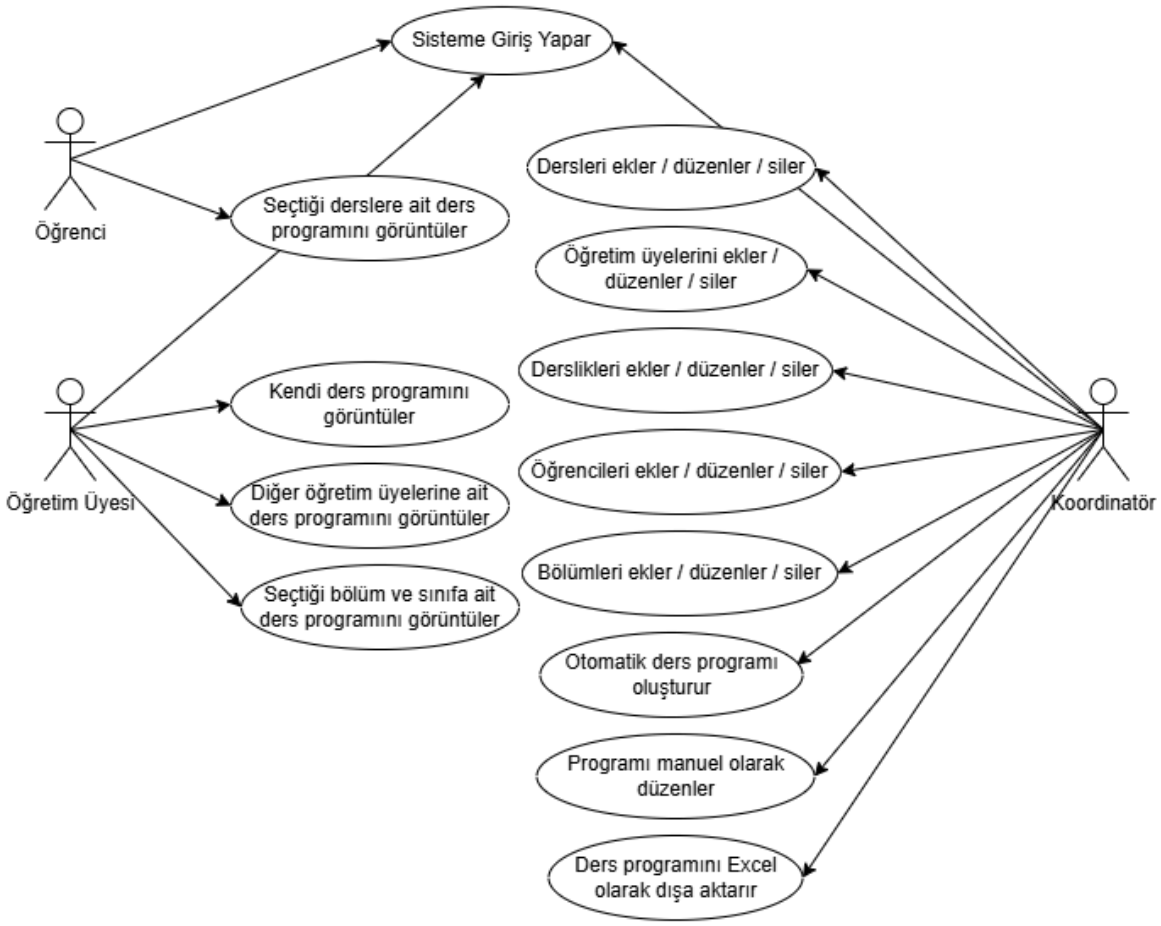
Program Çakışma Kontrolü (İG-6)

- Sistem, program oluştururken aynı zaman diliminde aynı öğretim üyesine birden fazla ders atamamalıdır.
- Aynı dersliğe aynı anda birden fazla ders atanmasına izin verilmemelidir.
- Ortak dersler, tüm ilgili bölümlerin çakışmayacağı şekilde yerleştirilmelidir.
- Online dersler belirlenen saat dilimlerine yerleştirilmelidir.

Excel Desteği (İG-7)

- Sistem, oluşturulan ders programını Excel formatında dışa aktarabilmelidir.
- Gerekirse dışarıdan alınan Excel dosyası üzerinden veri girişi yapılabilmelidir.

2.3 Use-Case Diyagramı

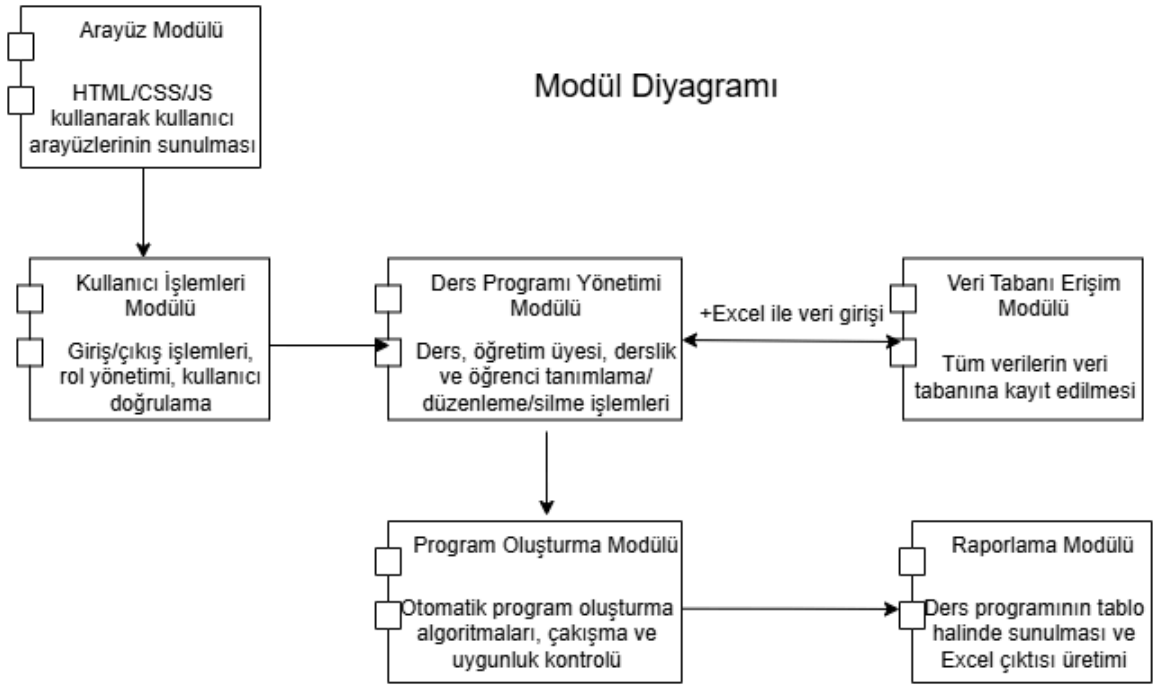


3. TASARIM

3.1 Mimari Tasarım

- Katmanlı Mimari Yapı
  - Proje, Django'nun sağladığı MTV (Model-Template-View) mimari desenine göre yapılandırılmıştır.
    - Model: Veri tabanı tablolarını ve ilişkilerini temsil eder.
    - Template: Kullanıcılara gösterilen HTML arayüzlerini içerir.
    - View: İş mantığını içerir, modelden veri alır ve uygun template'e aktarır.
- Veri Tabanı Katmanı
  - Django'nun ORM yapısı kullanılarak tüm veri tabanı işlemleri Python sınıfları üzerinden gerçekleştirilir.
  - Microsoft SQL Server ilişkisel bir veri tabanı ile entegredir.
- Otomatik Ders Programı Algoritması
  - Belirli kısıtlamalara göre (derslik çakışmaları, öğretim üyesi uygunluğu vb.) çalışan özel bir algoritma ile haftalık ders yerleşimi yapılır.
- Manuel Düzenleme Mekanizması
  - Koordinatör programı elle değiştirebilir.
  - Değişiklikler oluşturulan ders programına yansıtılır.

- Excel Entegrasyonu
  - Openpyxl kütüphanesi ile programın Excel formatında dışa aktarımı sağlanır.
  - Dışarıdan alınan Excel dosyasından veri girişi yapılabilir.
- Arayüz Yapısı
  - Django'nun templates klasöründe HTML sayfaları yer alır.
  - Her rol için ayrı arayüzler ve kullanıcı deneyimi tasarlanmıştır.



3.2 Kullanılacak Teknolojiler

Proje, Python programlama dili kullanılarak Django web uygulaması ile geliştirilmiştir. Veri tabanı olarak Microsoft SQL Server tercih edilmiş ve Django'nun SQL Server desteği için mssql-django ve pyodbc kütüphaneleri kullanılmıştır. Uygulama geliştirme sürecinde veri tabanı işlemleri için python manage.py makemigrations ve migrate komutları ile şema oluşturulmuş, createsuperuser komutu ile admin paneli tanımlanmıştır. Ayrıca ders programı verilerinin dışa aktarımı için django-import-export ve Excel desteği için openpyxl kütüphaneleri projeye dahil edilmiştir. Bu yapı, kullanıcı dostu arayüz, veri yönetimi ve raporlama özellikleriyle desteklenmiştir.

3.3 Veri Tabanı Tasarımı

DersProgramiDB Veri Tabanı

- Projenin tüm verilerini saklayan ana veri tabanıdır.
- Fakulte, Bolumler, OgretimGorevlileri, Ogrenciler, Dersler, OgrenciDers ve Derslikler olmak üzere 7 tablo oluşturulmuştur.

Fakulte Tablosu

- Üniversitedeki fakülteleri saklar.
- Fakülte adı ve benzersiz fakülte kimliği içerir.

Bolumler Tablosu

- Fakültelere bağlı bölümleri içerir.
- Hangi fakülteye ait olduğunu belirten fakülte kimliği ile ilişkilidir.

OgretimGorevlileri Tablosu

- Öğretim üyelerinin bilgilerini tutar.
- Hangi fakülteye ait olduğunu belirten fakülte kimliği ile ilişkilidir.
- Haftalık uygunluk saatleri kayıt edilir.
- Kullanıcı adı ve şifre içerir.

Ogrenciler Tablosu

- Öğrencilerin şifre bilgilerini, öğrenci numarası ve sınıf bilgilerini içerir.
- Hangi fakülteye ve bölüme ait olduğunu belirten kimlikler ile ilişkilidir.

Dersler Tablosu

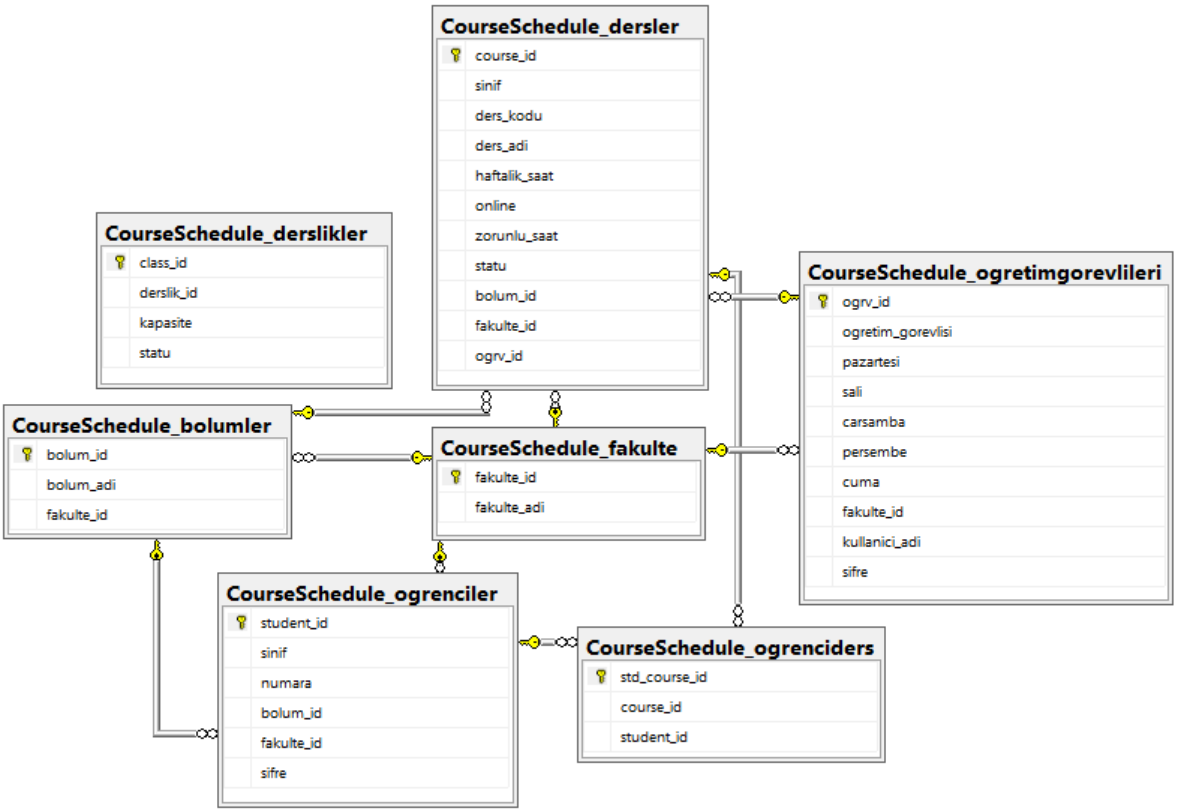
- Ders bilgilerini, dersin bağlı olduğu fakülte ve bölümü saklar.
- Dersin öğretim üyesi, haftalık ders saati, online olup olmadığı gibi detayları içerir.
- Zorunlu saatleri ve dersin statüsünü belirtir.

OgrenciDers Tablosu

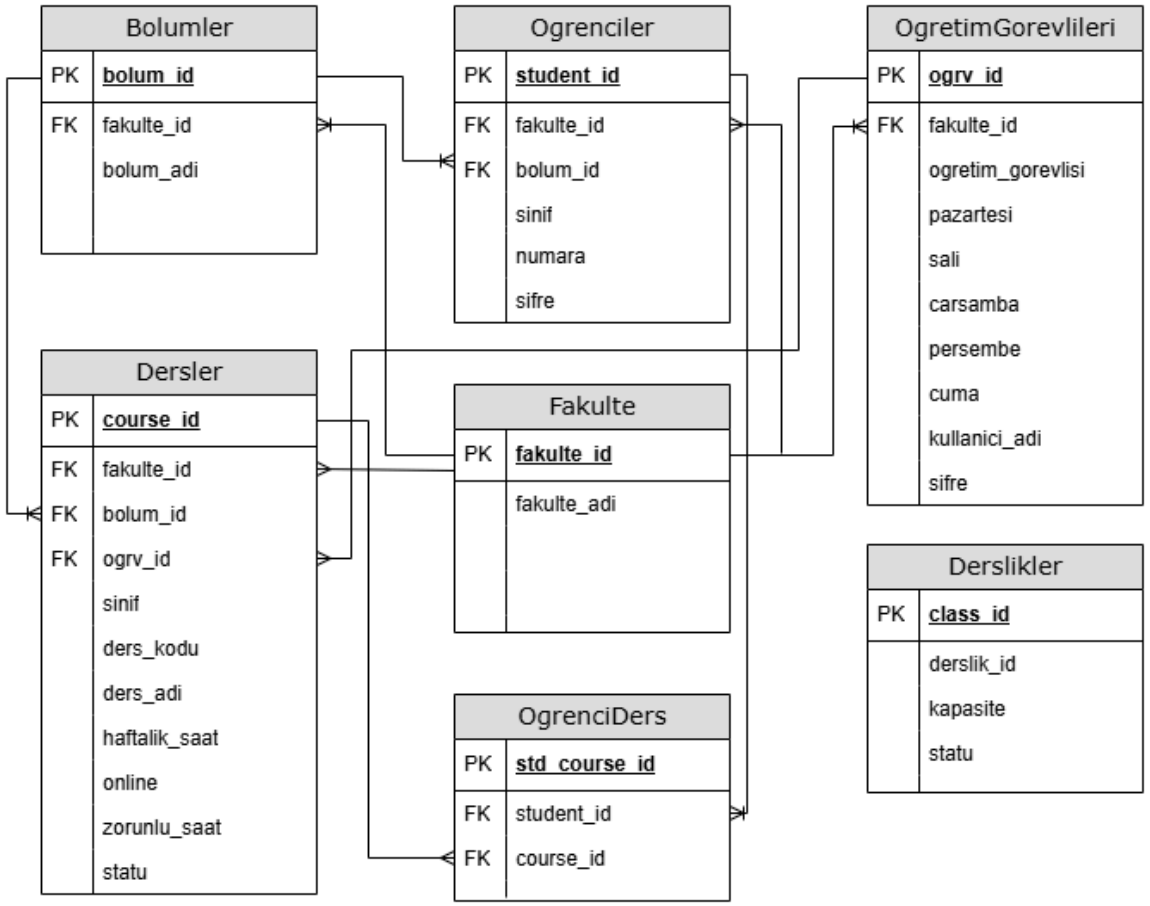
- Öğrencilerin aldığı dersleri ilişkilendirir.
- Öğrenci kimliği ve ders kimliği ile bağlantılıdır.

Derslikler Tablosu

- Derslerin yapılacağı sınıfları ve laboratuvarları içerir.
- Derslik kapasitesini ve sınıfın türünü (normal/laboratuvar) belirtir.



İlişkisel Veri Tabanı Gösterimi

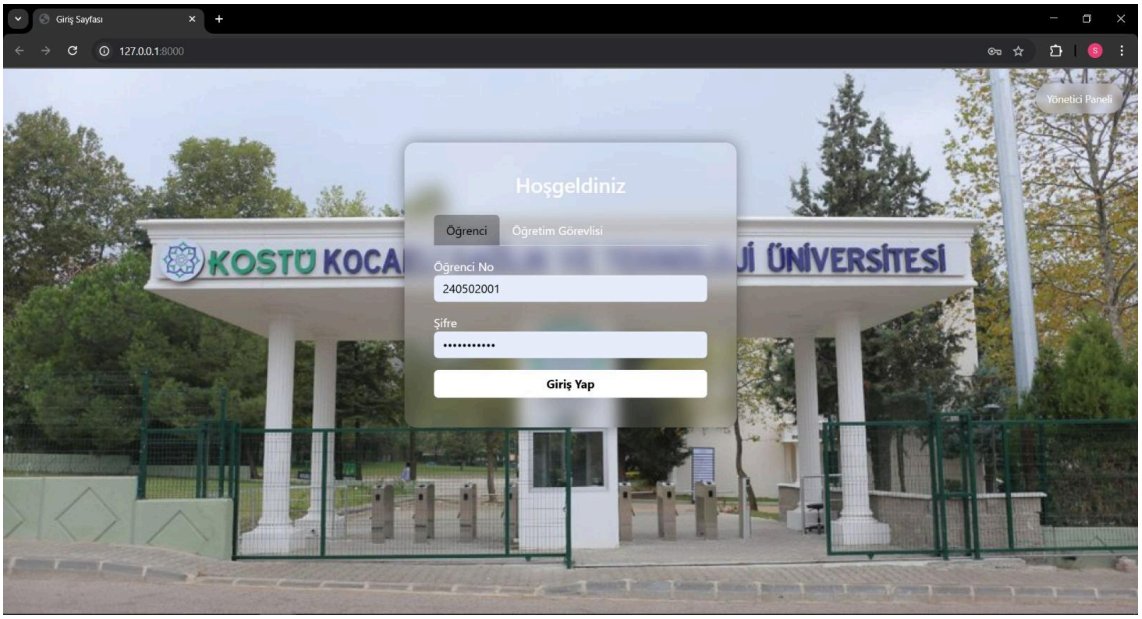


ER Diyagramı

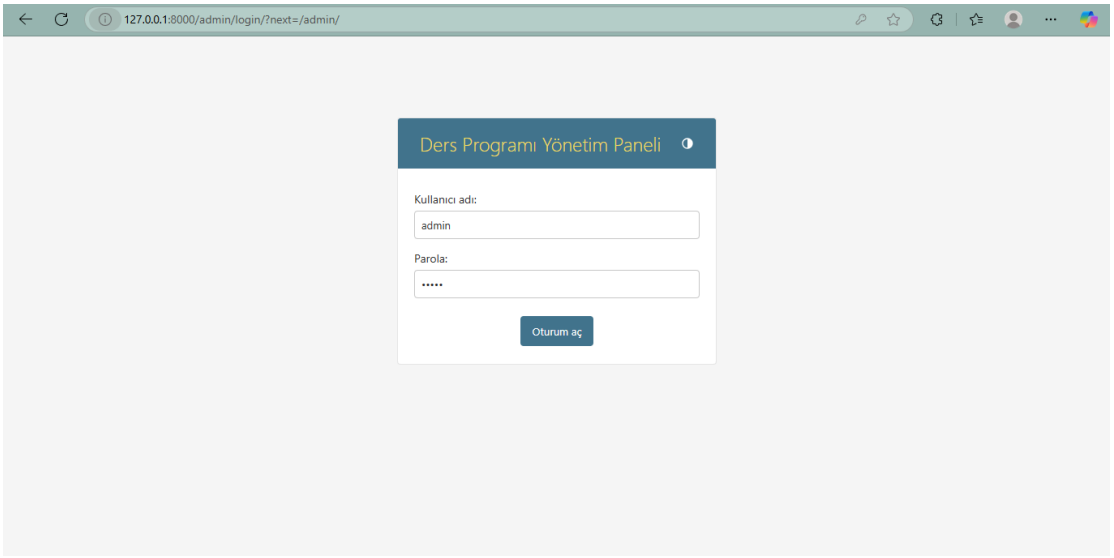
3.4 Kullanıcı Arayüzü Tasarımı

Projenin kullanıcı arayüzü, HTML, CSS ve JavaScript kullanılarak kullanıcı dostu ve işlevsel olacak şekilde tasarlanmıştır. Django'nun yerleşik template sistemi ile dinamik sayfa içerikleri oluşturulmuş, sayfa geçişleri ve veri formları bu yapı üzerinden yönetilmiştir.

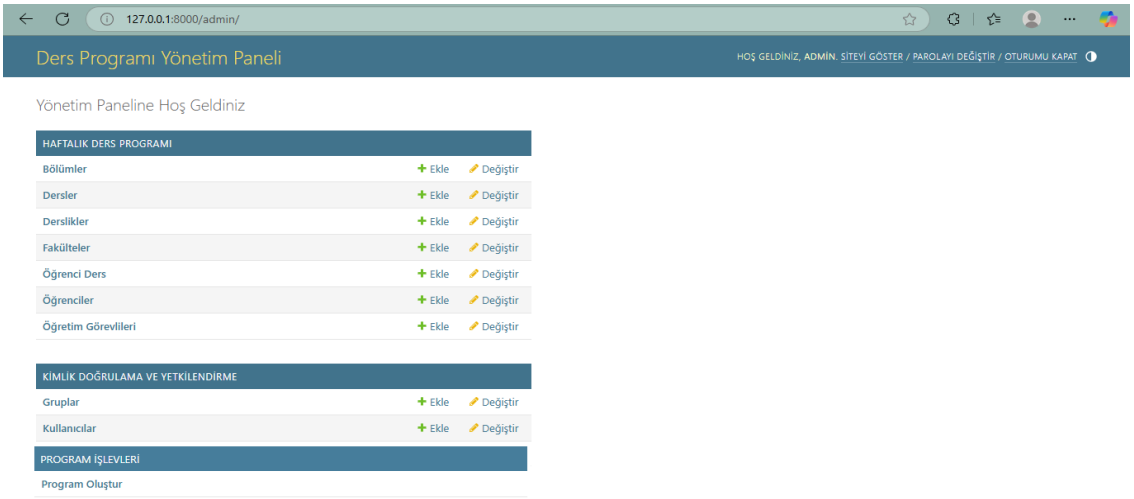
- **Giriş Sayfası**: Kullanıcı adı ve şifre ile kimlik doğrulamanın yapıldığı sade bir form ekranıdır.
- **Yönetici Paneli**: Koordinatörler fakülte, bölüm, ders, öğretim üyesi, derslik, öğrenci ve kullanıcı bilgilerini görüntüleyip düzenleyebileceği ana sayfaya yönlendirilir.
- **Ders Programı Yönetimi**: Kullanıcılar, sistemdeki fakülte, bölüm, ders, öğretim üyesi, derslik, öğrenci ve kullanıcı bilgilerini ekleyebilir, güncelleyebilir veya silebilir.
- **Program Oluşturma Sayfası**: Kullanıcılar, tanımlı veriler doğrultusunda otomatik ders programı oluşturabilir. Oluşturulan ders programı tablo halinde kullanıcıya sunulur ve Excel çıktısı alınabilir.
- **Program Düzenleme Sayfası**:
- **Öğrenci Sayfası:** Öğrenciler, ders seçimi yaparak seçtiği derslere ait ders programını görüntüleyebilir.
- **Öğretim Görevlisi Sayfası:** Öğretim görevlileri, kendilerine ve seçtiği öğretim üyesine ait ders programını görüntüleyebilir. Seçtiği bölüm ve sınıfa ait haftalık ders programını görüntüleyebilir.



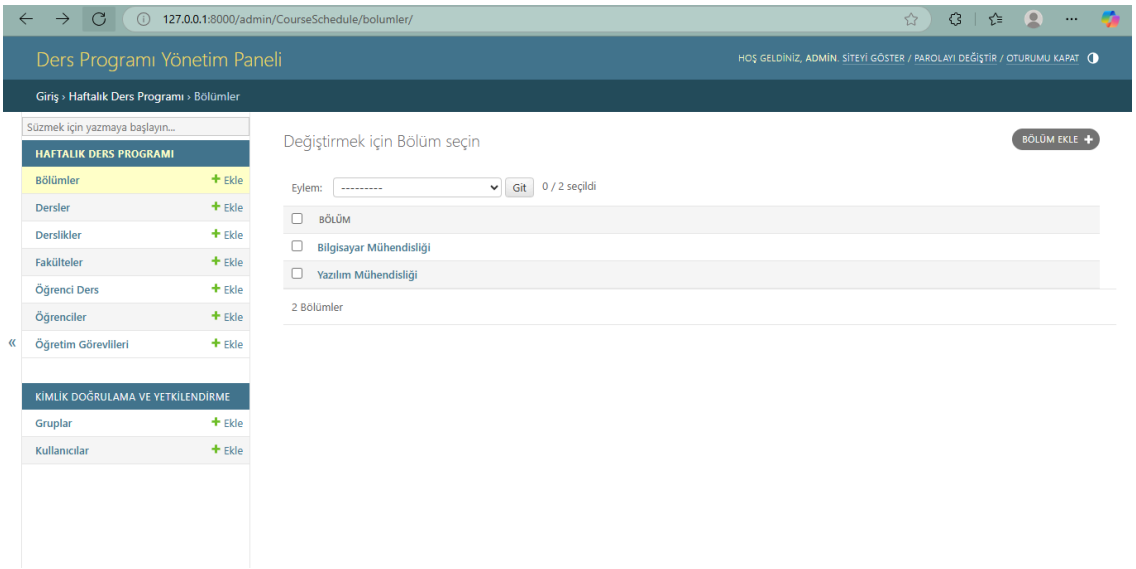
Görsel 3.4.1 Giriş Sayfası



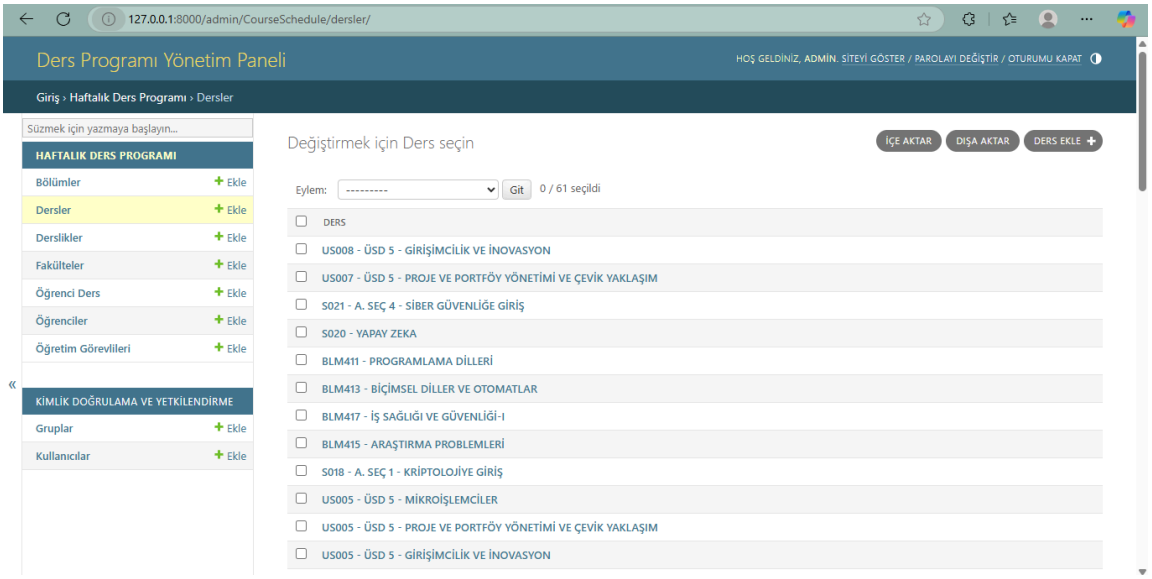
Görsel 3.4.2 Yönetici Paneli Giriş Sayfası



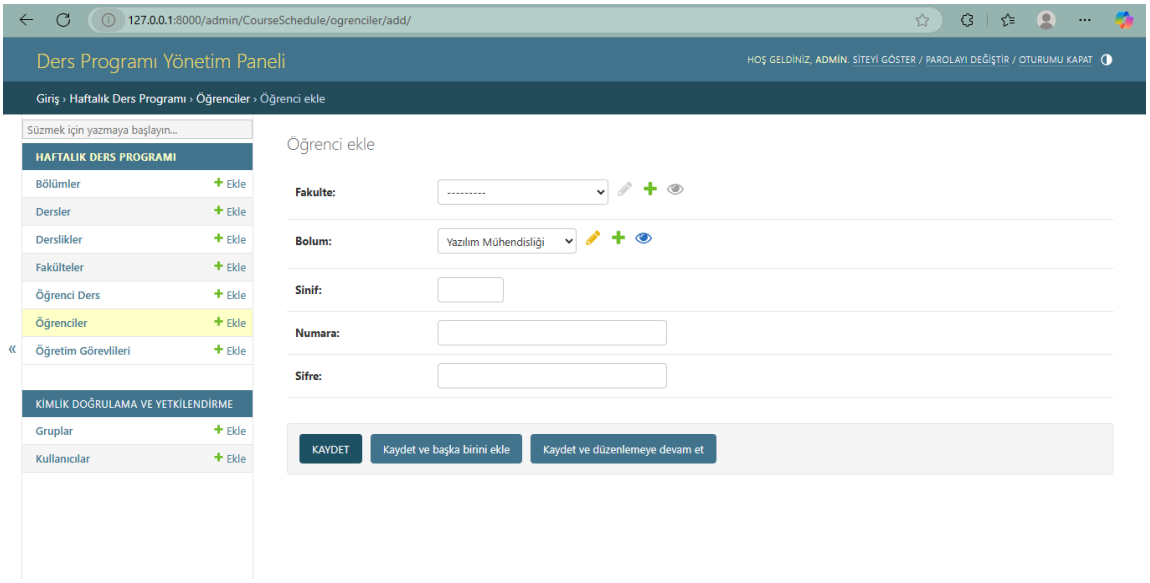
Görsel 3.4.3 Yönetici Paneli Ana Sayfası



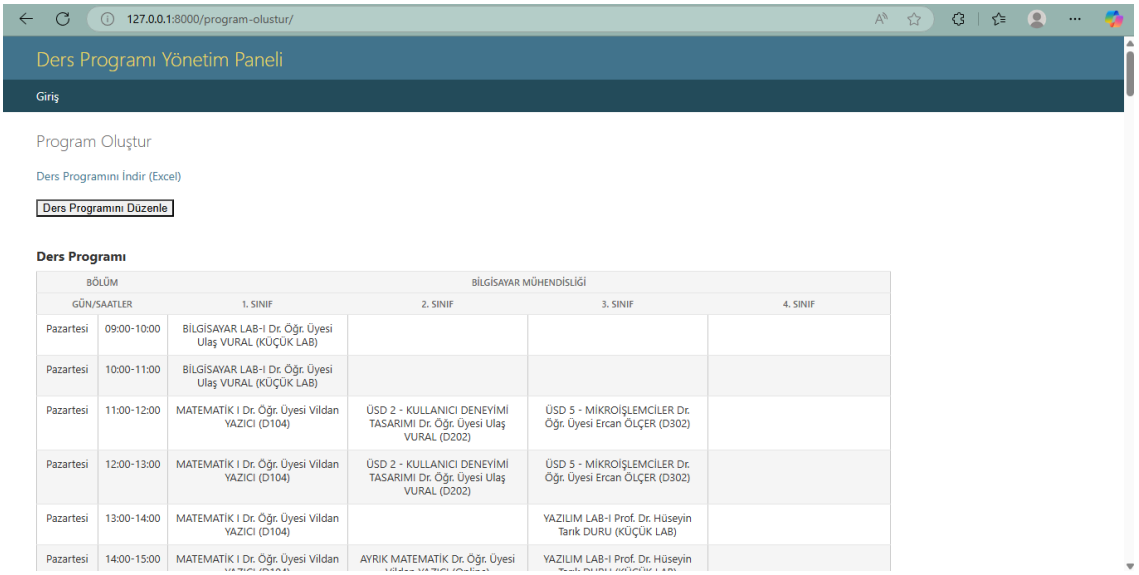
Görsel 3.4.4 Yönetici Paneli Bölümler Sayfası



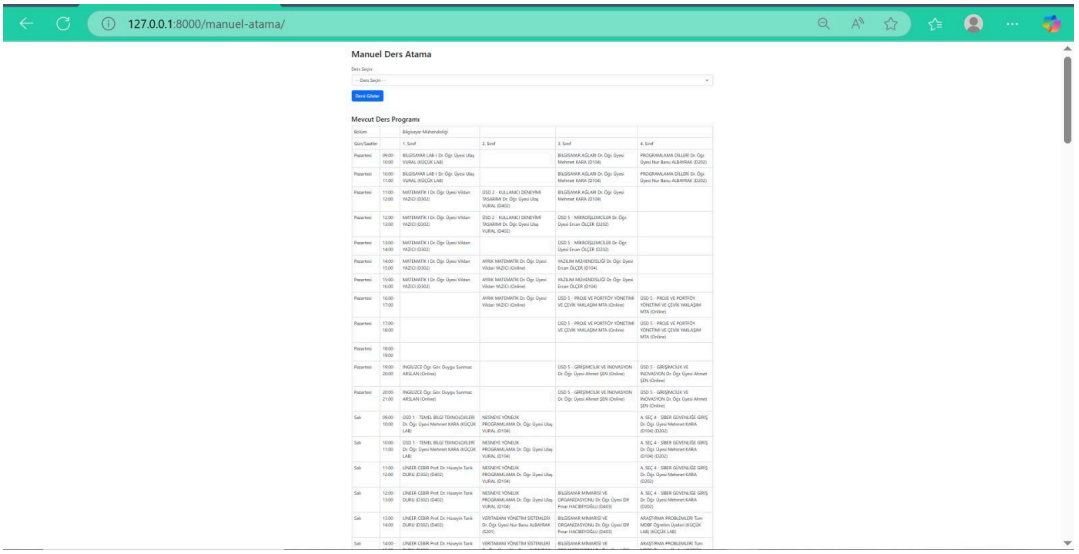
Görsel 3.4.5 Yönetici Paneli Dersler Sayfası



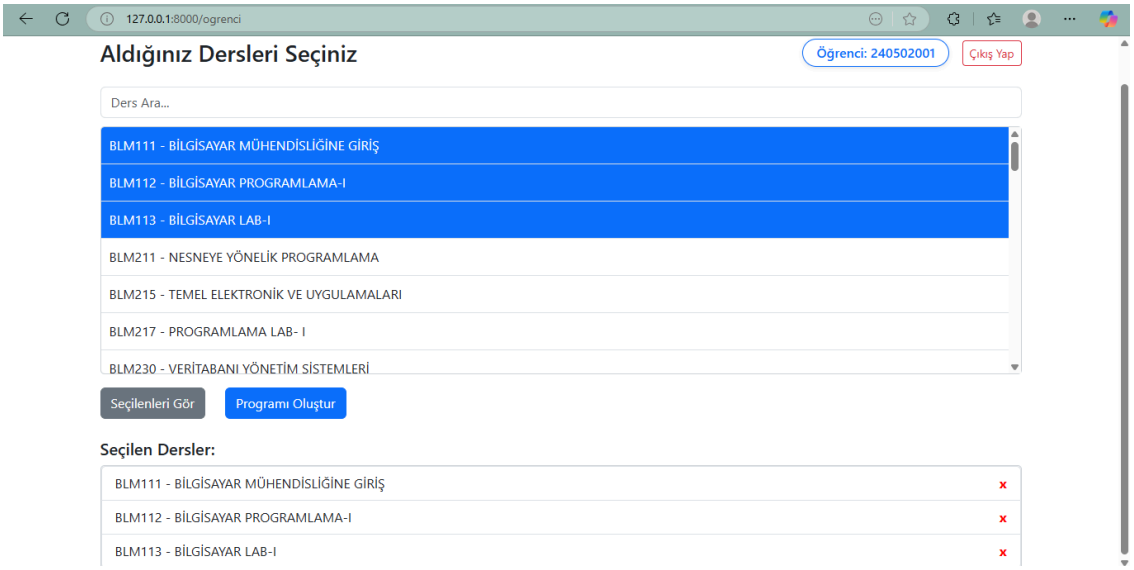
Görsel 3.4.6 Yönetici Paneli Öğrenci Ekle Sayfası



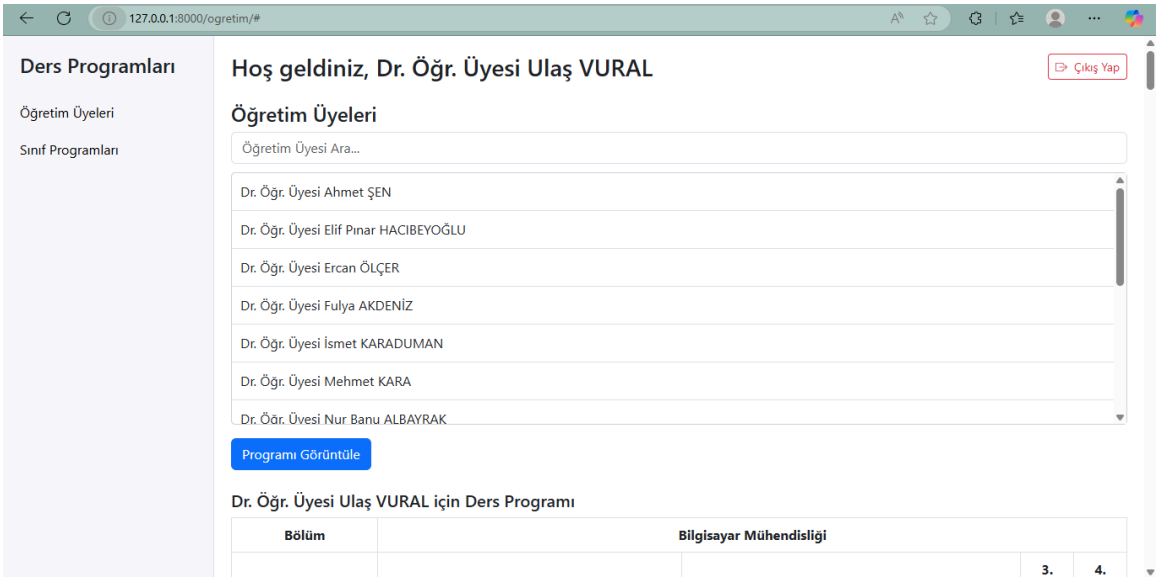
Görsel 3.4.7 Yönetici Paneli Program Oluştur Sayfası



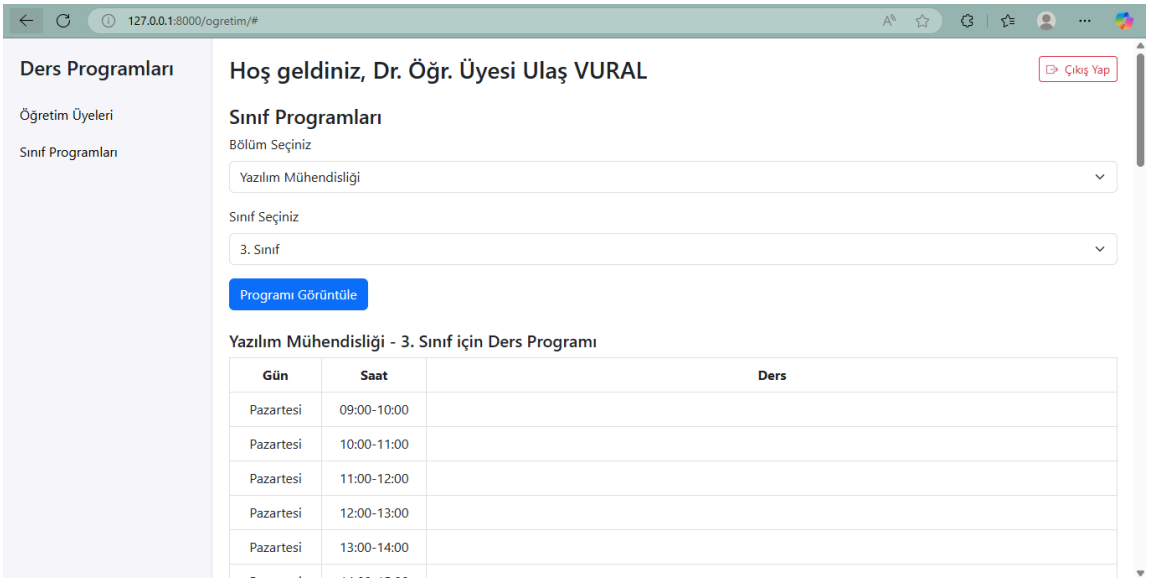
Görsel 3.4.8 Yönetici Paneli Manuel Atama Sayfası



Görsel 3.4.9 Öğrenci Sayfası



Görsel 3.4.10 Öğretim Üyeleri Sayfası



Görsel 3.4.11 Öğretim Üyeleri Sınıf Programları Sayfası

4. UYGULAMA

4.1 Kodlanan Bileşenlerin Açıklamaları

- **get_connection(database=None)**: SQL Server'a bağlantı kurar. Belirtilen veri tabanına bağlanır.
- **Excel tasarımı oluşturma:** Ders programı için Excel dosyası oluşturur ve biçimlendirir. Günleri, saat dilimlerini ve sınıf başlıklarını ekleyerek Bilgisayar ve Yazılım Mühendisliği bölümleri için ayrı ders çizelgeleri hazırlar. Hücreleri uygun renklerle doldurur ve kenarlıklar ekleyerek düzenli bir görünüm sağlar.
- **get_online_courses()**: Sadece online dersleri getirir.
- **get_common_courses()**: Ortak dersleri belirleyip liste olarak döndürür.
- **expand_time_range(start_time, end_time)**: Verilen başlangıç ve bitiş saatleri arasındaki tüm saat aralıklarını oluşturur.
- **get_instructor_availability()**: Öğretim üyelerinin uygunluk saatlerini alır.
- **sort_instructors_by_availability(instructor_availability)**: Öğretim üyelerini uygunluk durumuna göre sıralar.
- **convert_times_to_slots(instructor_availability, time_slots)**: Saatleri uygun zaman dilimlerine dönüştürür.
- **get_department_courses()**: Bölüme özel dersleri getirir.
- **convert_mandatory_time(mandatory_time)**: 'Zorunlu saat' verisini uygun formatta saat dilimlerine çevirir.
- **get_instructor_name(instructor_id)**: Verilen öğretim üyesi ID'sine karşılık gelen adı döndürür.
- **assign_courses_to_schedule(online_courses, time_slots)**: Online dersleri uygun zaman dilimlerine yerleştirir. Derslerin kaç saat süreceğini ve hangi saatlerde yerleştirileceğini belirler ve programı Excel dosyasına kaydeder.
- **assign_common_courses(common_courses, instructor_availability, time_slots)**: Ortak dersleri öğretim üyelerinin uygunluk saatlerine göre en uygun zaman dilimlerine yerleştirir. Aynı dersi alan farklı bölümlerin programlarının çakışmamasını sağlar. Güncellenen ders programını Excel'e kaydeder.
- **assign_department_courses(department_courses, instructor_availability,**

time_slots): Bölümlere özel dersleri öğretim üyelerinin müsait olduğu saatlerde en uygun zaman dilimlerine atar. Derslerin haftalık saatlerine ve bölümlerin önceliklerine dikkat ederek yerleştirme yapar. Güncellenmiş programı Excel dosyasına kaydeder.

- **get_classrooms():** Tüm dersliklerin ID, kapasite ve statü bilgilerini getirir.
- **get_student_count_for_course(course_id):** Belirtilen dersin kaç öğrenci tarafından alındığını döndürür.
- **get_course_id(course_name):** Ders adını kullanarak ilgili dersin ID'sini getirir.
- **get_online_status(course_id):** Belirtilen dersin online olup olmadığını kontrol eder.
- **read_courses_from_excel(filename="Ders_Programi.xlsx"):** Kaydedilmiş Excel dosyasını okuyarak ders programını bir sözlük olarak döndürür.
- **get_course_status(course_id):** Verilen dersin statüsünü ('LAB' veya 'NORMAL') döndürür.
- **get_course_duration(course_id):** Belirtilen dersin haftalık toplam saatini getirir.
- **assign_classrooms_to_courses(schedule, time_slots):** Ders programındaki her derse, öğrenci sayısına ve dersin statüsüne uygun bir derslik atar. Online dersleri atlayarak sadece yüz yüze dersler için uygun sınıfları belirler. Dersliklerin çakışmaması için ders süresi boyunca doluluk durumlarını kontrol eder.
- **generate_schedule_excel():** Online, ortak ve bölüme özel dersleri veri tabanından çekerek öğretim üyelerinin uygunluk saatlerini alır. Tüm dersleri uygun zaman dilimlerine ve dersliklere atayarak güncellenmiş programı Excel'e kaydeder. Derslerin atanması sırasında hata kontrolü yaparak eksik veya çakışan dersleri bildirir.
- **models.py:** Django modelinde, fakülteler, bölümler, öğretim görevlileri, öğrenciler, dersler, derslikler ve öğrencilerin aldığı derslerle ilgili veri tabanı yapılarını oluşturur. Modeller arasındaki ilişkiler doğru şekilde tanımlanarak her modelin işlevselliği ve yönetim panelindeki görüntülenme biçimleri özelleştirilmiştir.
- **resources.py:** Django'nun import_export kütüphanesini kullanarak veri tabanı modellerine dışa aktarım ve içe aktarım işlemleri için resource sınıfları oluşturur. Her model için özel alanlar tanımlanmış ve bu alanların dışa aktarılacak başlıklarıyla ilişkilendirilmiştir. Ayrıca CustomDersWidgetWithBolum widget'ı ile öğrencilerin aldıkları derslerin, ders adı ve bölüm ID'sine göre ilişkilendirilmesi sağlanır. Bu yapı Excel gibi dışa aktarım dosyaları üzerinden veri import/export işlemlerinin düzgün çalışmasını sağlar.
- **admin.py:** Django'nun admin panelinde, veri tabanı modelleri için import_export kütüphanesini kullanarak dışa ve içe aktarım işlevselliği ekler. Her model için özel admin sınıfları oluşturulmuş ve ImportExportModelAdmin sınıfıyla entegre edilmiştir. Dersler, Derslikler, Ogrenciler, OgretimGorevlileri, ve OgrenciDers modelleri, ilgili resource sınıflarıyla dışa ve içe aktarılabilir hale getirilmiştir. Admin paneli başlıkları özelleştirilmiş ve her model için listede görünen alanlar belirlenmiştir.
- **views.py:** Bu kod, Django'da bir ders atama sistemini yönetir. Kullanıcı bir ders seçtiğinde sistem seçilen dersin öğretim görevlisinin ders programını kontrol eder, Excel dosyasındaki mevcut saatleri analiz eder ve herhangi bir çakışma yoksa dersin uygun saatlerdeki alternatiflerini sunar. Ayrıca "Bilgisayar Mühendisliği" veya "Yazılım Mühendisliği" için ders ataması yapılabilir ve dersin silinmesi gibi işlemler de yapılır. Çakışma durumunda ise uygun saatler listeden çıkarılır ve en uygun alternatifler kullanıcıya sunulur.
- **urls.py:** Bu Django URL yapılandırma dosyasında, çeşitli sayfalara ve işlevlere yönlendiren URL yolları tanımlanmıştır. path() fonksiyonu, her bir URL'yi belirli bir view fonksiyonuyla ilişkilendirmiştir.
- **index.html:** Django kullanarak öğrenci ve öğretim görevlileri için ayrı ayrı giriş yapılabilen sekmeli bir giriş ekranı oluşturulmuştur. Arka planda SQL Server veri

tabanı bağlantısı, admin paneli yönetimi ve import-export entegrasyonu ile veri yönetimi sağlanmış ayrıca kullanıcı dostu ve şık bir arayüz tasarlanmıştır.

- **programolustur.html:** Django admin paneli altında bir ders programı sayfası oluşturur. Kullanıcıya ders programını indirme ve önizleme imkanı sunar, tablo yapısı dinamik olarak excel_tablo verisine göre oluşturulur ve belirli satırlarda özel başlıklar gösterilir. Ayrıca tablo düzeni ve stilleri sade ve okunabilir bir şekilde özelleştirilmiştir.
- **ogrenci.html:** Bu sayfa, öğrencilerin alacakları dersleri listeden seçmesini ve seçimlerine göre kişisel ders programlarını görüntülemesini sağlar. Seçilen dersler oturumda saklanır, öğrenciler dersleri filtreleyebilir ve görebilir ayrıca program tablo olarak gösterilir.
- **ogretim.html:** Bu sayfada, öğretim görevlileri ve sınıflar için ders programlarını görüntüleme imkanı sunulmuştur. Kullanıcılar öğretim üyesi seçerek ya da bölüm ve sınıf belirleyerek ilgili ders programını tablolar halinde görebilir, ayrıca filtreleme ve seçme işlemleriyle kullanım kolaylığı sağlanmıştır.
- **manuel_atama.html:** Bu HTML sayfası, manuel ders atama işlemini gerçekleştiren bir Django şablonudur. Sayfada kullanıcıların ders seçmesi için bir dropdown menü ve seçilen dersin mevcut ders programını görüntüleyen bir tablo bulunur. Ayrıca, dersin silinmesi veya Bilgisayar Mühendisliği ve Yazılım Mühendisliği bölümleri için uygun alternatif saatlerin seçilerek atama yapılması mümkündür. Alternatif saatler için her gün ve saat için derslik seçimi ve onay kutuları sağlanır. Atama işlemleri, form submit edilerek yapılır ve her işlemde kullanıcıya başarılı bir mesaj gösterilir.

4.2 Görev Dağılımı

Tüm aşamalar birlikte gerçekleştirilmiştir.

4.3 Karşılaşılan Zorluklar ve Çözüm Yöntemleri

Projeyi gerçekleştirirken çeşitli zorluklarla karşılaştık. Bu zorluklardan ilki, Django modülünün alt yapısını kavramak oldu. Hangi işlemlerin hangi dosyalar aracılığıyla gerçekleştirildiğini anlamak önemli ölçüde zamanımızı aldı. Bu süreci daha verimli yönetebilmek adına çeşitli eğitim videoları izledik ve kapsamlı araştırmalar yaptık. Bu çalışmalar sonucunda Django'nun yapısını ve işleyiş mantığını kavrayarak sistemi daha etkin bir şekilde kullanabilir hale geldik.

Program koordinatörü sayfasında manuel atama işlemlerini yaparken birçok sorun ile karşılaştık. Gerekli kısıtları her bir aşamada kontrol ederek manuel atama işlemlerini gerçekleştirdik.

4.4 Proje İsterlerine Göre Eksik Yönler

Projede gerçeklenmesi beklenen görevlerin tümü tamamlanmıştır.

5. TEST VE DOĞRULAMA
5.1 Yazılımın Test Süreci

Bu test, Django projesinde yer alan index sayfasının öğrenci ve öğretim görevlisi giriş işlemlerini doğru şekilde gerçekleştirdiğini kontrol etmek amacıyla geliştirilmiştir. Test kapsamında, hem geçerli hem de geçersiz kullanıcı bilgileriyle yapılan giriş denemeleri ayrı ayrı test edilmiştir. Doğru bilgilerle giriş yapan kullanıcıların ilgili sayfalara yönlendirilip oturum bilgilerinin kaydedildiği, hatalı girişlerde ise uygun hata mesajlarının döndüğü doğrulanmıştır.

```python
from django.test import TestCase
from django.urls import reverse
from CourseSchedule.models import Ogrenciler, OgretimGorevlileri

class IndexViewLoginTests(TestCase):
    def setUp(self):
        self.student = Ogrenciler.objects.create(numara='240502001', sifre='11111111111')
        self.instructor = OgretimGorevlileri.objects.create(
            kullanici_adi='ulas.vural', sifre='20000000005', ogretim_gorevlisi='Dr. Öğr. Üyesi Ulaş VURAL'
        )

    def test_student_login_success(self):
        response = self.client.post(reverse('index'), {
            'student_username': '240502001',
            'student_password': '11111111111'
        })
        self.assertRedirects(response, reverse('ogrenci_sayfa'))
        self.assertEqual(self.client.session['student_no'], '11111111111')

    def test_student_login_fail(self):
        response = self.client.post(reverse('index'), {
            'student_username': '999999',
            'student_password': '000000'
        })
        self.assertContains(response, 'Hatalı öğrenci girişi')

    def test_instructor_login_success(self):
        response = self.client.post(reverse('index'), {
            'instructor_username': 'ulas.vural',
            'instructor_password': '20000000005'
        })
        self.assertRedirects(response, reverse('ogretim_sayfa'))
        self.assertEqual(self.client.session['ogretmen_adi'], 'Dr. Öğr. Üyesi Ulaş VURAL')

    def test_instructor_login_fail(self):
        response = self.client.post(reverse('index'), {
            'instructor_username': '999999',
            'instructor_password': '000000'
        })
        self.assertContains(response, 'Hatalı öğretim görevlisi girişi')
```

Bu test, proje kapsamında öğrenci ve öğretim görevlisi sayfalarının erişilebilirliğini, kullanıcı oturumu ile ilişkilendirilmiş ders seçimi işlemlerinin doğruluğunu ve çıkış işlevinin düzgün çalıştığını kontrol etmek amacıyla geliştirilmiştir. Geliştirilen test sınıfı ile oturum açma simülasyonu yapılmış, session verileri kullanılarak view fonksiyonlarının doğru yanıt döndürdüğü ve ilgili template'lerin yüklendiği doğrulanmıştır.

```python
from django.test import TestCase, Client
from django.urls import reverse
from CourseSchedule.models import Ogrenciler, OgretimGorevlileri

class ViewFunctionTests(TestCase):
    def setUp(self):
        self.client = Client()
        self.student = Ogrenciler.objects.create(numara='240502001', sifre='11111111111')
        self.teacher = OgretimGorevlileri.objects.create(
            kullanici_adi='ulas.vural', sifre='20000000005', ogretim_gorevlisi='Dr. Öğr. Üyesi Ulaş VURAL'
        )

    def test_ogrenci_sayfa_get(self):
        session = self.client.session
        session['student_no'] = self.student.numara
        session['secilen_dersler'] = ['YZM115 - BİLGİSAYAR LAB-I']
        session.save()
        response = self.client.get(reverse('ogrenci_sayfa'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'CourseSchedule/ogrenci.html')

    def test_ogretim_sayfa_get(self):
        session = self.client.session
        session['ogretmen_adi'] = self.teacher.ogretim_gorevlisi
        session.save()
        response = self.client.get(reverse('ogretim_sayfa'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'CourseSchedule/ogretim_sayfa.html')

    def test_ders_secimi_kaydet_post(self):
        response = self.client.post(
            reverse('ders_secimi_kaydet'),
            content_type='application/json',
            data='{"dersler": ["YZM115 - BİLGİSAYAR LAB-I"]}'
        )
        self.assertEqual(response.status_code, 200)
        self.assertJSONEqual(response.content, {'status': 'LAB'})

    def test_cikis_yap(self):
        session = self.client.session
        session['student_no'] = '240502001'
        session.save()
        response = self.client.get(reverse('cikis_yap'))
        self.assertRedirects(response, reverse('index'))
        self.assertNotIn('student_no', self.client.session)
```

Bu test, otomatik ders programı oluşturan program_olustur_view fonksiyonunun doğru şekilde çalışıp çalışmadığını kontrol etmek amacıyla hazırlanmıştır. Test kapsamında; fonksiyonun doğru template'i yüklediği, excel dosyasının belirtilen konumda oluşturulduğu ve içerdiği verilerin boş olmadığı doğrulanmıştır. Böylece hem görsel çıktıların kullanıcıya düzgün sunulması hem de arka planda dosya üretiminin hatasız gerçekleşmesi güvence altına alınmıştır.

```python
from django.test import TestCase, Client
from django.urls import reverse
from django.conf import settings
import os
import openpyxl

class ProgramOlusturTestCase(TestCase):
    def setUp(self):
        self.client = Client()

    def test_program_olustur_view_status_code(self):
        response = self.client.get(reverse('program_olustur'))
        self.assertEqual(response.status_code, 200)
        self.assertTemplateUsed(response, 'CourseSchedule/programolustur.html')

    def test_excel_file_created(self):
        response = self.client.get(reverse('program_olustur'))
        dosya_adi = "Ders_Programi.xlsx"
        dosya_yolu = os.path.join(settings.MEDIA_ROOT, dosya_adi)
        self.assertTrue(os.path.exists(dosya_yolu))

    def test_excel_content_not_empty(self):
        response = self.client.get(reverse('program_olustur'))
        dosya_yolu = os.path.join(settings.MEDIA_ROOT, "Ders_Programi.xlsx")
        workbook = openpyxl.load_workbook(dosya_yolu)
        sheet = workbook.active
        rows = list(sheet.iter_rows(values_only=True))
        self.assertGreater(len(rows), 5)
```

Bu test kodu, bir ders atama ve programlama sistemine yönelik çeşitli fonksiyonları doğrulayan bir dizi birim testi içermektedir. İlk olarak, veritabanı bağlantı fonksiyonu ve ders bilgilerini getiren fonksiyonlar (online dersler, ortak dersler, bölüm dersleri, öğretim görevlisi uygunlukları) test edilir. Ayrıca, ders atama işlemlerinin doğru bir şekilde bloklar halinde yapıldığını ve öğretim görevlisi uygunluklarının ihlal edilmediğini kontrol eden testler bulunur. Excel dosyasındaki ders programı üzerinden de derslerin doğru saatlerde atanıp atanmadığı, öğretim görevlisi çakışmalarının olup olmadığı ve ortak derslerin senkronize şekilde yerleştirilip yerleştirilmediği gibi kuralların doğruluğu test edilir. Testler, verilerin doğru şekilde işlendiğinden emin olmak için açıkça tanımlanmış kurallar ve koşullar altında yapılır.

```python
import unittest
from unittest.mock import patch, MagicMock
from main import (
    get_connection, get_online_courses, get_common_courses,
    get_department_courses, get_instructor_availability,
    assign_department_courses, assign_common_courses, get_instructor_name
)
import openpyxl
from collections import defaultdict

class TestDatabaseFunctions(unittest.TestCase):

    @patch('main.pyodbc.connect')
    def test_get_connection(self, mock_connect):
        conn = get_connection()
        mock_connect.assert_called_once()
        self.assertTrue(conn.autocommit)

    @patch('main.get_connection')
    def test_get_online_courses(self, mock_get_connection):
        mock_cursor = MagicMock()
        mock_cursor.fetchall.return_value = [
            ("Yapay Zeka", 2, 101, 3, 1, "13:00, 14:00"),
            ("BİLGİSAYAR PROGRAMLAMA 1", 3, 102, 2, 2, "10:00, 11:00")
        ]
        mock_conn = MagicMock()
        mock_conn.cursor.return_value = mock_cursor
        mock_get_connection.return_value = mock_conn

        result = get_online_courses()
        self.assertEqual(len(result), 2)
        self.assertIn("Yapay Zeka", [r[0] for r in result])

    @patch('main.get_connection')
    def test_get_common_courses(self, mock_get_connection):
        mock_cursor = MagicMock()
        mock_cursor.fetchall.return_value = [
            ("İşletim Sistemleri", 3, 104, 3),
            ("Web Programlama", 2, 105, 4)
        ]
        mock_conn = MagicMock()
        mock_conn.cursor.return_value = mock_cursor
        mock_get_connection.return_value = mock_conn

        result = get_common_courses()
        self.assertEqual(len(result), 2)
        self.assertIn(("İşletim Sistemleri", 3, 104, 3), result)

    @patch('main.get_connection')
    def test_get_department_courses(self, mock_get_connection):
        mock_cursor = MagicMock()
        mock_cursor.fetchall.return_value = [
            ("NESNEYE YÖNELİK PROGRAMLAMA", 3, 106, 2, 1),
            ("VERİTABANI YÖNETİM SİSTEMLERİ", 2, 107, 1, 2)
```

```
        mock_get_connection.return_value = mock_conn

        result = get_department_courses()
        self.assertEqual(len(result), 2)

    @patch('main.get_connection')
    def test_get_instructor_availability(self, mock_get_connection):
        mock_cursor = MagicMock()
        mock_cursor.fetchall.return_value = [
            (101, "Dr. Öğr. Üyesi İsmet KARADUMAN", "09:00, 10:00, 11:00", "10:00, 11:00", "", "", "", "")
        ]
        mock_conn = MagicMock()
        mock_conn.cursor.return_value = mock_cursor
        mock_get_connection.return_value = mock_conn

        result = get_instructor_availability()
        self.assertIn(101, result)
        self.assertIn("Pazartesi", result[101])
        self.assertTrue(isinstance(result[101]["Pazartesi"], list))


class TestCourseAssignment(unittest.TestCase):

    @patch("main.get_instructor_name", return_value="Dr. Öğr. Üyesi Elif Pınar HACIBEYOĞLU")
    def test_department_course_assignment_blocks(self, mock_name):
        department_courses = [("YAZILIM TEST VE KALİTE", 2, 1, 3, 2)]
        instructor_availability = {
            1: {
                "Pazartesi": ["10:00-11:00", "11:00-12:00"],
                "Salı": ["09:00-10:00"]
            }
        }
        time_slots = ["09:00-10:00", "10:00-11:00", "11:00-12:00"]
        assign_department_courses(department_courses, instructor_availability, time_slots)

    @patch("main.get_instructor_name", return_value="Dr. Öğr. Üyesi Ulaş VURAL")
    def test_common_course_block_assignment(self, mock_name):
        common_courses = [("NESNEYE YÖNELİK PROGRAMLAMA", 2, 4, 5, 1)]
        instructor_availability = {
            1: {
                "Pazartesi": ["09:00-10:00", "11:00-12:00"]
            }
        }
        time_slots = ["09:00-10:00", "10:00-11:00", "11:00-12:00"]
        assign_common_courses(common_courses, instructor_availability, time_slots)


class TestExcelScheduleRules(unittest.TestCase):

    def test_online_courses_assigned_to_mandatory_slots(self):
        wb = openpyxl.load_workbook("Ders_Programı.xlsx")
        ws = wb.active
        online_course_slots = {
            "AYRIK MATEMATİK": ["14:00-17:00"],
            "İNGİLİZCE": ["19:00-21:00"]
        }
        for row in ws.iter_rows(min_row=3, max_row=126, min_col=3, max_col=6):
            for cell in row:
                if cell.value:
                    for course_name, mandatory_slots in online_course_slots.items():
                        if course_name in cell.value:
                            row_num = cell.row
                            index = (row_num - 3) % 12
                            slot = [
                                "09:00-10:00", "10:00-11:00", "11:00-12:00",
                                "12:00-13:00", "13:00-14:00", "14:00-15:00",
                                "15:00-16:00", "16:00-17:00", "17:00-18:00",
                                "18:00-19:00", "19:00-20:00", "20:00-21:00"
                            ][index]
                            self.assertIn(slot, mandatory_slots, msg=f"{course_name} dersi zorunlu olmayan bir saatte atanmış: {slot}")

    def test_instructor_conflict(self):
        wb = openpyxl.load_workbook("Ders_Programı.xlsx")
        ws = wb.active
        for row in range(3, 126):
            time_index = (row - 3) % 12
            time_slot = [
                "09:00-10:00", "10:00-11:00", "11:00-12:00",
                "12:00-13:00", "13:00-14:00", "14:00-15:00",
                "15:00-16:00", "16:00-17:00", "17:00-18:00",
                "18:00-19:00", "19:00-20:00", "20:00-21:00"
            ][time_index]
            instructors_in_slot = set()
            for col in range(3, 7):
                value = ws.cell(row=row, column=col).value
                if value:
                    lines = value.strip().split("\n")
                    if len(lines) >= 2:
                        instructor = lines[1]
                        self.assertNotIn(instructor, instructors_in_slot, msg=f"{time_slot} saatinde {instructor} birden fazla sınıfta görünüyor!")
                        instructors_in_slot.add(instructor)
```

5.2 Yazılımın Doğrulanması

- **Fonksiyonel Doğrulama**: Bu testler, yazılımın belirlenen fonksiyonel gereksinimlere uygun olarak çalışıp çalışmadığını denetlemektedir. Ders atama süreçleri, öğretim üyelerinin uygunluk durumları, derslik kapasiteleri ve ortak derslerin eş zamanlı ataması gibi kritik fonksiyonlar test edilmiştir.

- **Ders Atama Testleri:** Bölüm ve ortak derslerin belirli kurallara uygun şekilde zaman çizelgesine atanıp atanmadığı kontrol edilmiştir. Derslerin blok saatlere yerleştirilmesi, online derslerin yalnızca zorunlu saat aralıklarında planlanması, dersliklerin sınıf mevcuduna uygun kapasitede olması gibi durumlar için test senaryoları uygulanmıştır.

- **Veri Tabanı Testleri:** Veri tabanı bağlantısının başarılı bir şekilde kurulup kurulmadığı, doğru sorgularla online, ortak ve bölüm derslerinin çekilip çekilmediği, öğretim üyelerinin uygunluk bilgilerinin doğru bir biçimde alınıp alınmadığı test edilmiştir.

6. GİTHUB BAĞLANTILARI

https://github.com/SemaSuYILMAZ

http://github.com/Zehrayardimci

https://github.com/senemadalan