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

PROJE KONUSU:
HAFTALIK 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:
22 MART 2025

İÇİNDEKİLER

1. GİRİŞ
1.1 Projenin Amacı

Bu proje, mühendislik fakültesine ait dersleri içeren haftalık ders programını otomatik olarak oluşturmayı amaçlamaktadır. Proje kapsamında derslerin çakışmaması, öğretim üyelerinin uygunluk saatlerine göre atamaların yapılması, dersliklerin kapasitesine uygun olarak derslerin yerleştirilmesi ve online derslerin belirlenen saat aralıklarına atanması sağlanacaktır. Öğrenci ve öğretim üyelerinin kolayca erişerek programlarını görebilecekleri dinamik bir sistem sunmayı hedeflemektedir.

2. GEREKSİNİM ANALİZİ
2.1 Fonksiyonel Gereksinimler

Kullanıcı Yönetimi (İG-1)
- Öğretim üyesi ekleme ve çıkarma işlemleri.
- Öğrenci ekleme ve çıkarma işlemleri.

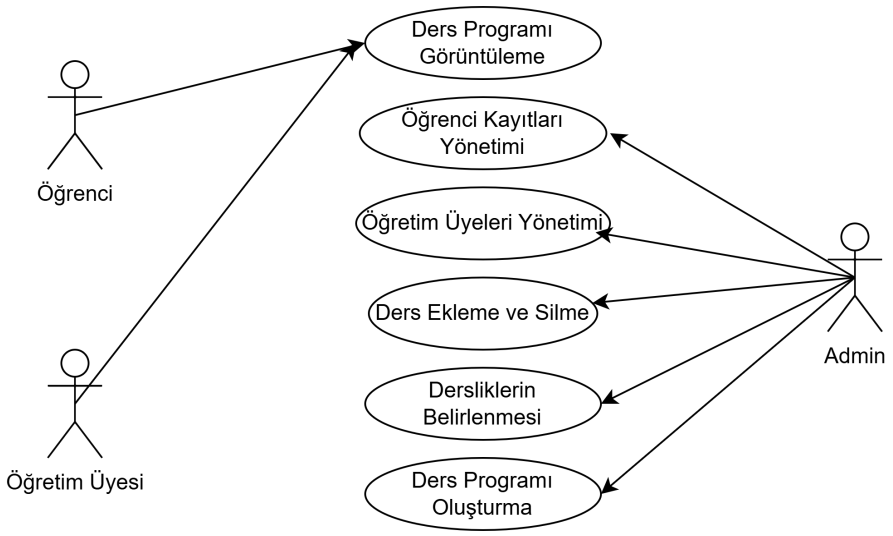
Ders ve Program Yönetimi (İG-2)
- Ders ekleme ve çıkarma işlemleri.
- Ders saatlerinin öğretim üyelerinin uygunluk durumuna göre atanması.
- Ortak derslerin bölümler arasında çakışmadan yönetilmesi.
- Haftalık ders programının otomatik oluşturulması
- Zorunlu ders saatlerinin dikkate alınarak yerleştirme yapılması
- Derslerin online veya yüz yüze olarak belirlenebilmesi

Derslik Yönetimi (İG-3)
- Derslik kapasitesine uygun olarak derslerin yerleştirilmesi.
- Laboratuvar ve normal derslik ayrımı yapılarak uygun atamaların yapılması.

Zamanlama ve Çakışma Kontrolleri (İG-4)
- Aynı dersliğin farklı derslere aynı saatte atanmaması.
- Aynı öğretim üyesinin aynı anda birden fazla ders verememesi.
- Ortak verilen derslerin uygun saatlerde atanması.
- Online derslerin belirlenen saat dilimlerinde yapılması.

Veri Dışa Aktarım (İG-5)
- Excel formatında haftalık ders programı oluşturma.

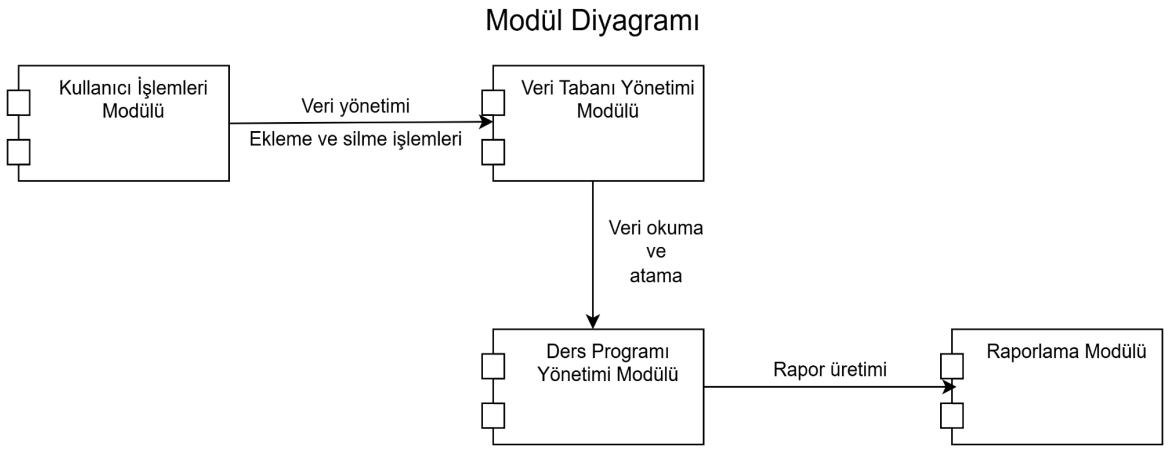
2.2 Use-Case Diyagramı



3. TASARIM

3.1 Mimari Tasarım

- Sunum Katmanı
  - Öğrenci, öğretim üyesi ve yöneticilerin ders programlarını görüntülemesine olanak tanır.
- İşlem Katmanı
  - Ders atama algoritmaları ve program oluşturma süreçlerini yönetir.
  - Öğretim üyelerinin uygunluk durumlarına göre derslerin atanmasını sağlar.
  - Derslerin çakışmasını önleyen algoritmalar geliştirilir.
- Veri Tabanı Katmanı
  - Dersler, öğretim üyeleri, öğrenciler ve derslikler için yapılandırılmış ilişkisel veri tabanını içerir.
  - Tablolar arası ilişkiler ve veri bütünlüğü korunur.



3.2 Kullanılacak Teknolojiler

- Bu projede Python programlama dili kullanılmıştır.
- Pyodbc kütüphanesi veri tabanına bağlanmak için kullanılan bir kütüphanedir. Bu kütüphane sayesinde Python uygulamalarında SQL tabanlı Microsoft SQL Server ile kolayca iletişim kurulabilir.
- Openpyxl kütüphanesi kullanılmıştır. Microsoft Excel dosyalarıyla çalışmak için kullanılan bir kütüphanedir. Bu kütüphane sayesinde Excel dosyaları okunabilir, oluşturulabilir ve düzenlenebilir.
- Veri analizi işlemleri için Pandas kütüphanesi kullanılmıştır.

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.

Ogrenciler Tablosu

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

Dersler Tablosu

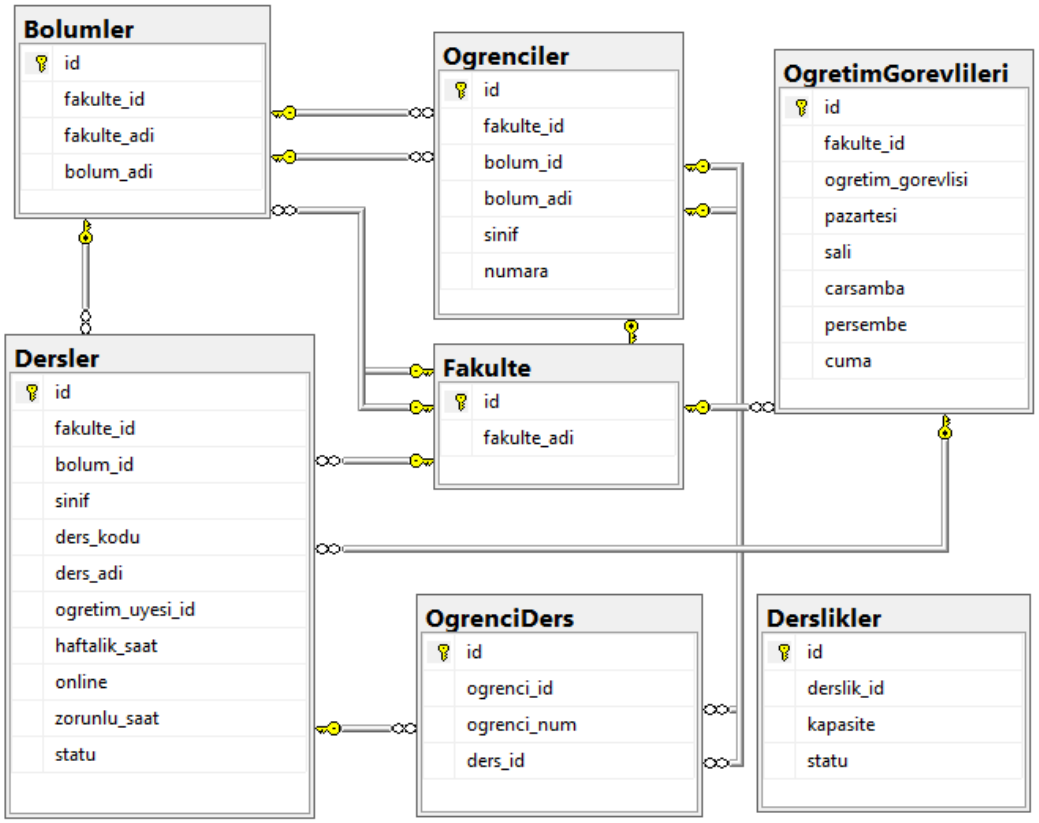
- Ders bilgilerini, dersin bağlı olduğu fakülte ve bölümü saklar.
- Dersin öğretim üyesi, haftalık ders saati, online olup olmadığı gibi detayları içerir.
- Zorunlu saatleri ve dersin statüsünü belirtir.

OgrenciDers Tablosu

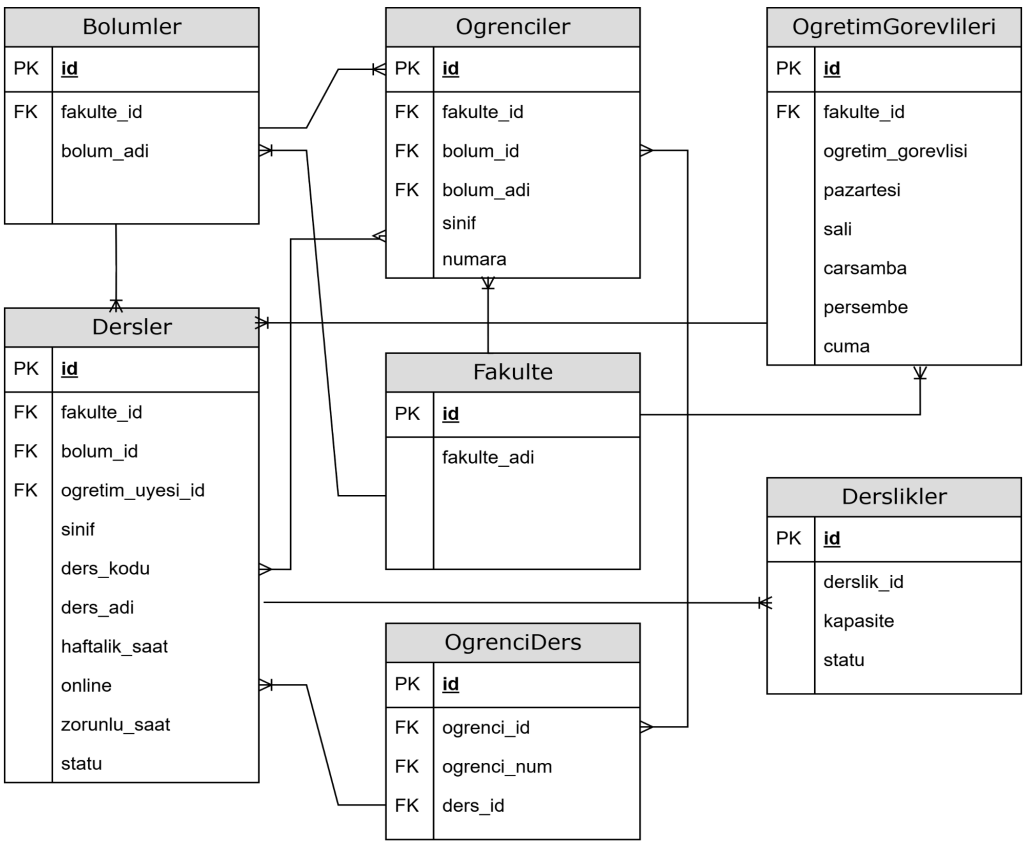
- Öğrencilerin aldığı dersleri ilişkilendirir.
- Öğrenci kimliği ve ders kimliği ile bağlantılıdır.

Derslikler Tablosu

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



İlişkisel Veri Tabanı Gösterimi



ER Diyagramı

4. UYGULAMA

4.1 Kodlanan Bileşenlerin Açıklamaları

- **get_connection(database=None)**: SQL Server'a bağlantı kurar. Belirtilen veri tabanına bağlanır.
- **database_exists()**: 'DersProgramiDB' adlı veri tabanının olup olmadığını kontrol eder.
- **create_database()**: Eğer 'DersProgramiDB' yoksa oluşturur.
- **table_exists(table_name, conn)**: Belirtilen tablonun veri tabanında olup olmadığını kontrol eder.
- **create_tables()**: Gerekli tabloları ('Fakulte', 'Bolumler', 'OgretimGorevlileri', 'Ogrenciler', 'OgrenciDers','Derslikler') oluşturur.
- **insert_faculties_from_excel(students_file)**: Excel dosyasından fakülteleri okuyup 'Fakulte' tablosuna ekler.
- **insert_departments_from_excel(students_file)**: Excel'den bölümleri alıp 'Bolumler' tablosuna ekler.
- **insert_students_from_excel(students_file)**: Excel'deki öğrencileri 'Ogrenciler' tablosuna ekler.
- **insert_faculty_members_from_excel(faculty_members_file)**: Öğretim üyelerini Excel'den okuyup 'OgretimGorevlileri' tablosuna ekler.
- **insert_classrooms_from_excel(classroom_file)**: Derslikleri Excel'den alarak 'Derslikler' tablosuna ekler.
- **insert_courses_from_excel(courses_file)**: Ders bilgilerini Excel'den alıp 'Dersler' tablosuna kaydeder.
- **insert_students_courses_from_excel(students_file)**: Öğrenci-ders ilişkisini Excel'den alarak 'OgrenciDers' tablosuna ekler.
- **add_faculty()**: Kullanıcıdan fakülte adını alıp 'Fakulte' tablosuna ekler.
- **delete_faculty()**: Kullanıcıdan fakülte adı alıp 'Fakulte' tablosundan siler.
- **add_department()**: Kullanıcıdan bölüm ve fakülte adını alıp 'Bolumler' tablosuna ekler.
- **delete_department()**: Kullanıcıdan bölüm ve fakülte adı alıp 'Bolumler' tablosundan siler.
- **add_instructor()**: Kullanıcıdan öğretim görevlisi bilgilerini alıp 'OgretimGorevlileri' tablosuna ekler.
- **delete_instructor()**: Kullanıcıdan öğretim görevlisi adını alıp 'OgretimGorevlileri' tablosundan siler.
- **add_student()**: Kullanıcıdan öğrenci bilgilerini alıp 'Ogrenciler' tablosuna ekler.
- **delete_student()**: Kullanıcıdan öğrenci numarasını alıp 'Ogrenciler' tablosundan siler.
- **add_classroom()**: Kullanıcıdan derslik bilgilerini alıp 'Derslikler' tablosuna ekler.
- **delete_classroom()**: Kullanıcıdan derslik ID'sini alıp 'Derslikler' tablosundan siler.
- **add_course()**: Kullanıcıdan ders bilgilerini alıp 'Dersler' tablosuna ekler.
- **delete_course()**: Kullanıcıdan ders kodunu alıp 'Dersler' tablosundan siler.
- **add_student_course()**: Kullanıcıdan öğrenci numarası ve ders kodunu alarak 'OgrenciDers' tablosuna ekler.
- **delete_student_course()**: Kullanıcıdan öğrenci numarası ve ders kodunu alarak 'OgrenciDers' tablosundan siler.
- **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.
- **main():** 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.
- **menu():** Kullanıcıya fakülte, bölüm, öğretim üyesi, öğrenci, derslik ve ders yönetimi işlemlerini içeren bir menü sunar. Kullanıcının seçimine göre ilgili ekleme veya silme fonksiyonlarını çağırarak veri tabanı işlemlerini gerçekleştirir. Kullanıcı çıkış yapmak istediğinde programı sonlandırır.

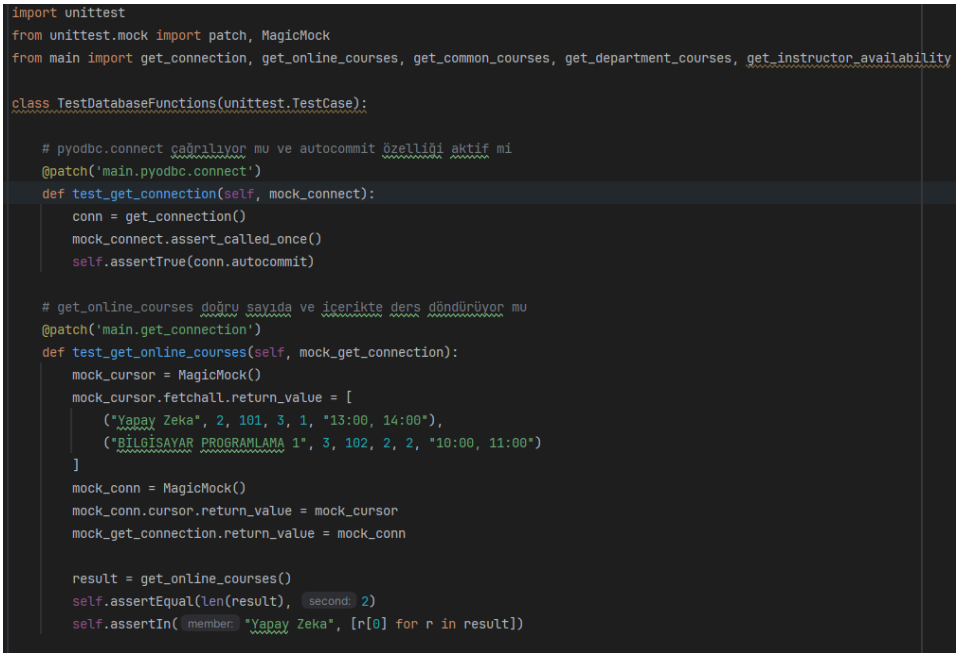
4.2 Görev Dağılımı
Tüm aşamalar birlikte gerçekleştirilmiştir.

4.3 Karşılaşılan Zorluklar ve Çözüm Yöntemleri
Projeyi gerçeklerken birçok sorunla karşılaştık. Bunlardan bir tanesi excel tablolarını uygun biçimde veri tabanına aktarmaktı. Proje isterlerini göz önüne alarak veri tabanı dizaynı yaptık ve excel formatındaki verilerin uygun biçimde veri tabanı tablolarına kayıt edilmesini sağladık.

Ders programına dersleri atarken de bazı sorunlarla karşılaştık. Bir ders atanırken tüm saatleri peş peşe atanmayabiliyordu. Bu problemi çözmek amacıyla öğretim görevlisinin uygunluk saati de göz önüne alınarak en iyi bloğun belirlenmesi yöntemini kullandık. Dersleri bloklar halinde atayarak programın doğru biçimde oluşturulmasını sağladık.

4.4 Proje İsterlerine Göre Eksik Yönler
Projede gerçeklenmesi beklenen görevlerin tümü tamamlanmıştır.

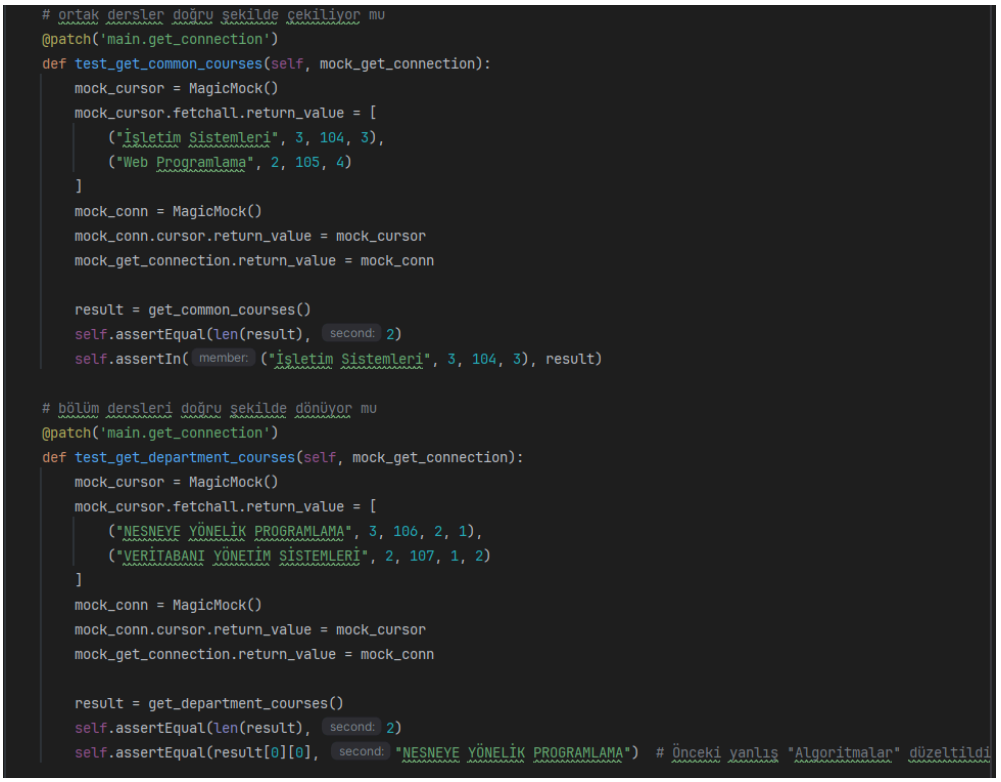
5. TEST VE DOĞRULAMA
5.1 Yazılımın Test Süreci
Bu birim testi, veri tabanı işlemleriyle ilgili çeşitli fonksiyonların doğru çalışıp çalışmadığını test etmek amacıyla yazılmıştır. Testler veri tabanı bağlantısı oluşturma, online ve ortak dersleri çekme, bölüme özel derslerin alınması ve öğretim üyelerinin uygunluk durumlarının kontrolünü kapsamaktadır. Her bir test, ilgili fonksiyonun doğru SQL sorgularını çalıştırıp çalıştırmadığını ve beklenen çıktıyı verip vermediğini kontrol eder.



```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

class TestDatabaseFunctions(unittest.TestCase):

    # pyodbc.connect çağrılıyor mu ve autocommit özelliği aktif mi
    @patch('main.pyodbc.connect')
    def test_get_connection(self, mock_connect):
        conn = get_connection()
        mock_connect.assert_called_once()
        self.assertTrue(conn.autocommit)

    # get_online_courses doğru sayıda ve içerikte ders döndürüyor mu
    @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), second: 2)
        self.assertIn( member: "Yapay Zeka", [r[0] for r in result])
```

test_get_connection testi, get_connection fonksiyonunun doğru şekilde çalışıp çalışmadığını kontrol eder. pyodbc.connect fonksiyonu @patch ile taklit edilir. Test bağlantının oluşturulduğunu ve autocommit özelliğinin aktif olduğunu doğrular.

test_get_online_courses testi, get_online_courses fonksiyonunun doğru sayıda ve içerikte online ders bilgisini veri tabanından çekip çekmediğini test eder. MagicMock file veri tabanı sorgusu simüle edilir ve dönen veriler içinde belirli bir dersin bulunup bulunmadığı kontrol edilir.



```
    # ortak dersler doğru şekilde çekiliyor mu
    @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),  second: 2)
        self.assertIn( member: ("İşletim Sistemleri", 3, 104, 3), result)

    # bölüm dersleri doğru şekilde dönüyor mu
    @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_conn = MagicMock()
        mock_conn.cursor.return_value = mock_cursor
        mock_get_connection.return_value = mock_conn

        result = get_department_courses()
        self.assertEqual(len(result),  second: 2)
        self.assertEqual(result[0][0],  second: "NESNEYE YÖNELİK PROGRAMLAMA")  # Önceki yanlış "Algoritmalar" düzeltildi
```

test_get_common_courses testi, get_common_courses fonksiyonunun ortak dersleri doğru şekilde alıp almadığını kontrol eder. Ortak ders listesi mock verilerle test edilir ve dönen listenin beklenen dersleri içerdiği doğrulanır.

test_get_department_courses testi, get_department_courses fonksiyonunun bölüme özel dersleri doğru şekilde çekip çekmediğini kontrol eder. Dönüş değerinin sayısı ve içeriği test edilir.

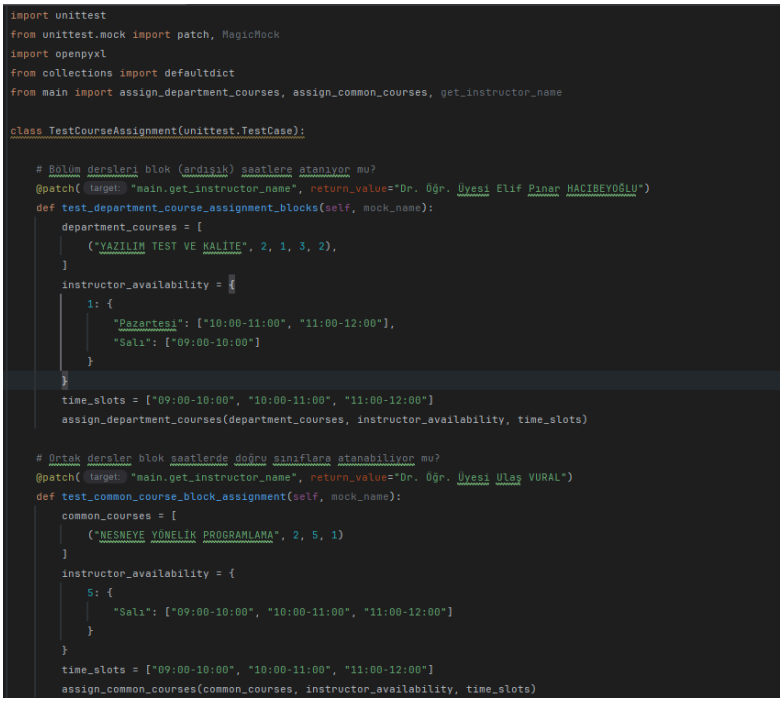
```python
    # eğitmen uygunluk verileri doğru şekilde çekilip ayrıştırılıyor mu
    @patch('main.get_connection')
    def test_get_instructor_availability(self, mock_get_connection):
        mock_cursor = MagicMock()
        mock_cursor.fetchall.return_value = [
            (101, "Dr. Öğr. Üyesi İsmet 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( member: 101, result)
        self.assertIn( member: "Pazartesi", result[101])
        self.assertTrue(isinstance(result[101]["Pazartesi"], list))

if __name__ == '__main__':
    unittest.main()
```

test_get_instructor_availability testi, get_instructor_availability fonksiyonunun öğretim üyelerinin uygunluk saatlerini veri tabanından çekip bunları doğru bir formatta ayrıştırıp ayrıştırmadığını test eder. Testte öğretim üyesinin verilerinin çekilip gün bazlı saat aralıkları içeren sözlük yapısına dönüştüğü kontrol edilir.

```python
def test_cell_format_is_course_instructor_room(self):
    """
    Hücre içeriği şu formatta olmalı:
    'Ders Adı\nEğitmen Adı\n(Derslik)'
    """
    ws = self.ws

    bm_rows = range(3, 63)   # Bilgisayar Mühendisliği
    bm_cols = [3, 4, 5, 6]
    sw_rows = range(67, 127)  # Yazılım Mühendisliği
    sw_cols = [3, 4, 5]

    invalid_cells = []

    def is_valid_format(value):
        lines = value.strip().split('\n')
        return (
            len(lines) == 3 and
            lines[2].startswith("(") and lines[2].endswith(")")
        )

    # Bilgisayar Müh. hücreleri kontrol et
    for row in bm_rows:
        for col in bm_cols:
            value = ws.cell(row=row, column=col).value
            if value and not is_valid_format(value):
                invalid_cells.append((row, col, value))

    # Yazılım Müh. hücreleri kontrol et
    for row in sw_rows:
        for col in sw_cols:
            value = ws.cell(row=row, column=col).value
            if value and not is_valid_format(value):
                invalid_cells.append((row, col, value))

    # Eğer bozuk hücre varsa test başarısız olur
    self.assertFalse(invalid_cells, msg: f"Formatı bozuk hücre(ler) bulundu:\n{invalid_cells}")
```

Bu birim testi, oluşturulan Excel ders programı dosyasının (Ders_Programi.xlsx) biçimsel ve içeriksel bütünlüğünü denetlemeye yönelik test senaryolarını kapsar. Testler hücre içeriklerinin önceden belirlenmiş formatlara uygunluğunu ve her iki bölüm (Bilgisayar Mühendisliği ile Yazılım Mühendisliği) için en az bir dersin tabloya işlenmiş olduğunu kontrol eder.

test_cell_format_is_course_instructor_room, testi hücre içeriklerinin "Ders Adı\nÖğretim Görevlisi Adı\n(Derslik)" formatına uygun olup olmadığını kontrol eder.



```python
import unittest
from unittest.mock import patch, MagicMock
import openpyxl
from collections import defaultdict
from main import assign_department_courses, assign_common_courses, get_instructor_name

class TestCourseAssignment(unittest.TestCase):

    # Bölüm dersleri blok (ardışık) saatlere atanıyor mu?
    @patch( target: "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)

    # Ortak dersler blok saatlerde doğru sınıflara atanabiliyor mu?
    @patch( target: "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, 5, 1)
        ]
        instructor_availability = {
            5: {
                "Salı": ["09:00-10:00", "10:00-11:00", "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)
```

test_department_course_assignment_blocks testi, assign_department_courses'ın bölüm derslerini blok (ardışık) saat dilimlerine doğru şekilde yerleştirip yerleştirmediğini kontrol eder. Öğretim Görevlisi uygunluklarına göre ardışık saatlerin seçilip seçilmediği doğrulanır.

test_common_course_block_assignment testi, ortak derslerin tüm sınıflara aynı anda blok olarak atanıp atanmadığını kontrol eder. Öğretim Görevlisi uygunluk saatleri dikkate alınarak assign_common_courses fonksiyonu üzerinden doğrulama yapılır.

```python
    # Online dersler, zorunlu olarak belirlenen saatlere atanmış mı?
    def test_online_courses_assigned_to_mandatory_slots(self):
        wb = openpyxl.load_workbook("Ders_Programi.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
                            time_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"
                            ][time_index]
                            self.assertIn(slot, mandatory_slots,
                                msg: f" {course_name} dersi zorunlu olmayan bir saate atanmış: {slot}")
```

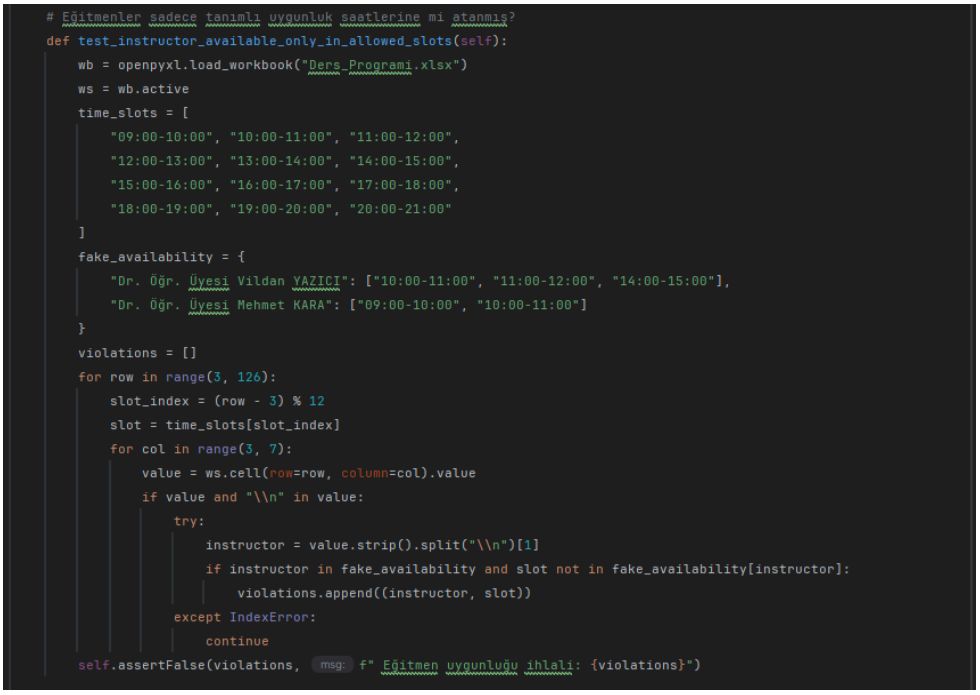
test_online_courses_assigned_to_mandatory_slots testi, online olarak belirlenmiş derslerin yalnızca tanımlı zorunlu saat aralıklarında programa yerleştirilip yerleştirilmediğini kontrol eder. Excel dosyasındaki hücre içerikleriyle karşılaştırma yapılarak saat uyumu doğrulanır.

```python
# Aynı eğitmen aynı saatte birden fazla sınıfa atanmış mı (çakışma var mı)?
def test_instructor_conflict(self):
    wb = openpyxl.load_workbook("Ders_Programi.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)
```

test_instructor_conflict testi, aynı öğretim üyesinin aynı zaman diliminde birden fazla sınıfta görev alıp almadığını kontrol eder.

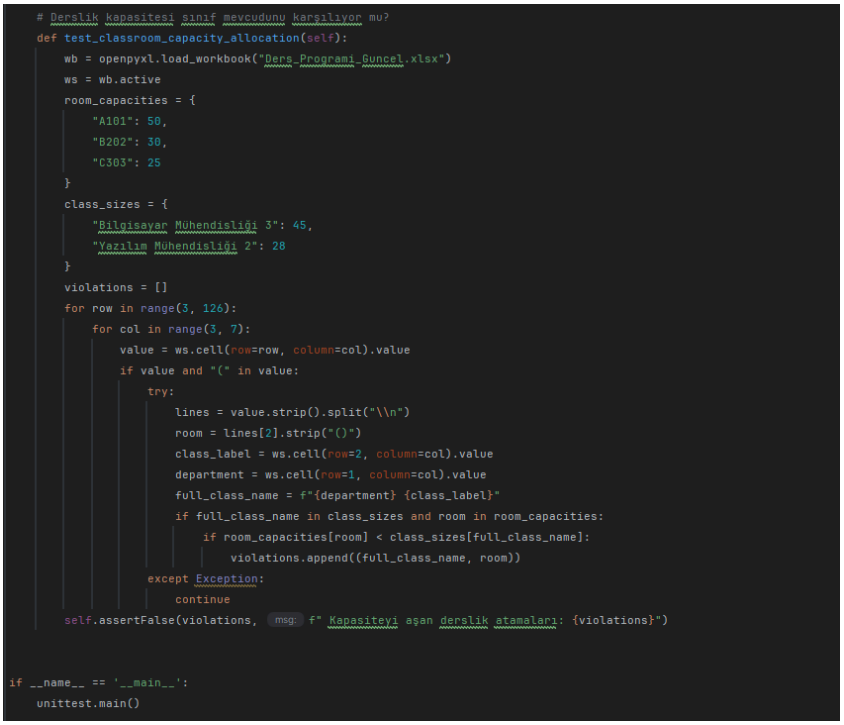
```python
# Ortak dersler tüm sınıflara aynı zaman diliminde mi atanmış?
def test_common_course_synchronized_across_classes(self):
    wb = openpyxl.load_workbook("Ders_Programi.xlsx")
    ws = wb.active
    common_course_slots = defaultdict(list)
    for row in range(3, 126):
        for col in range(3, 7):
            value = ws.cell(row=row, column=col).value
            if value:
                course = value.split("\\n")[0]
                common_course_slots[course].append((row, col))
    desynchronized = []
    for course, positions in common_course_slots.items():
        rows = {pos[0] for pos in positions}
        if len(rows) > 1 and len(positions) > 1:
            desynchronized.append(course)
    self.assertFalse(desynchronized, msg: f" Ortak dersler senkronize değil: {desynchronized}")
```

test_common_course_synchronized_across_classes testi, Ortak derslerin, tüm sınıflara aynı zaman diliminde yazılıp yazılmadığını test eder. Ders isimlerine göre hücrelerin satır konumları karşılaştırılarak kontrol sağlanır.



```python
    # Eğitmenler sadece tanımlı uygunluk saatlerine mi atanmış?
    def test_instructor_available_only_in_allowed_slots(self):
        wb = openpyxl.load_workbook("Ders_Programi.xlsx")
        ws = wb.active
        time_slots = [
            "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"
        ]
        fake_availability = {
            "Dr. Öğr. Üyesi Vildan YAZICI": ["10:00-11:00", "11:00-12:00", "14:00-15:00"],
            "Dr. Öğr. Üyesi Mehmet KARA": ["09:00-10:00", "10:00-11:00"]
        }
        violations = []
        for row in range(3, 126):
            slot_index = (row - 3) % 12
            slot = time_slots[slot_index]
            for col in range(3, 7):
                value = ws.cell(row=row, column=col).value
                if value and "\\n" in value:
                    try:
                        instructor = value.strip().split("\\n")[1]
                        if instructor in fake_availability and slot not in fake_availability[instructor]:
                            violations.append((instructor, slot))
                    except IndexError:
                        continue
        self.assertFalse(violations, msg: f" Eğitmen uygunluğu ihlali: {violations}")
```

test_instructor_available_only_in_allowed_slots testi, öğretim üyelerinin yalnızca önceden tanımlanmış uygunluk saatlerinde programa atanıp atanmadığını kontrol eder.



```python
    # Derslik kapasitesi sınıf mevcudunu karşılıyor mu?
    def test_classroom_capacity_allocation(self):
        wb = openpyxl.load_workbook("Ders_Programi_Guncel.xlsx")
        ws = wb.active
        room_capacities = {
            "A101": 50,
            "B202": 30,
            "C303": 25
        }
        class_sizes = {
            "Bilgisayar Mühendisliği 3": 45,
            "Yazılım Mühendisliği 2": 28
        }
        violations = []
        for row in range(3, 126):
            for col in range(3, 7):
                value = ws.cell(row=row, column=col).value
                if value and "(" in value:
                    try:
                        lines = value.strip().split("\\n")
                        room = lines[2].strip("()")
                        class_label = ws.cell(row=2, column=col).value
                        department = ws.cell(row=1, column=col).value
                        full_class_name = f"{department} {class_label}"
                        if full_class_name in class_sizes and room in room_capacities:
                            if room_capacities[room] < class_sizes[full_class_name]:
                                violations.append((full_class_name, room))
                    except Exception:
                        continue
        self.assertFalse(violations, msg: f" Kapasiteyi aşan derslik atamaları: {violations}")


if __name__ == '__main__':
    unittest.main()
```

test_classroom_capacity_allocation testi, ders programında atanan dersliklerin ilgili sınıfın öğrenci mevcudunu karşılayacak kapasiteye sahip olup olmadığını kontrol eder. Excel dosyasındaki her hücrede belirtilen derslik bilgisi sınıf adıyla eşleştirilerek sınıfın büyüklüğü ile karşılaştırılır.

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