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

PROJE KONUSU:
PROGRAM DERS İLİŞKİ MATRİSİ PROJESİ

ÖĞRENCİ ADI – NUMARASI:

Evrim BAŞER – 220501027
Hatice Reyhan ÇALIŞKAN - 220501001
Muhammed Yusuf KAYA - 210501007
Betül CANOL - 220501009

DERS SORUMLUSU:
PROF. DR./DR. ÖĞR. ÜYESİ
Eray DURSUN
Candide ÖZTÜRK

TARİH:19/01/2025

1 GİRİŞ
1.1 Projenin amacı

Bu projenin amacı, belirli web sayfalarından (örneğin, eğitim çıktıları veya ders detaylarını içeren sayfalar) tablolar halinde sunulan verilerin otomatik olarak çekilip bir SQLite veritabanına kaydedilmesidir. Proje, web tabanlı verilerin sistematik şekilde toplanmasını, analiz edilmesini ve saklanmasını kolaylaştırmayı hedefler. Bu sayede, özellikle eğitim kurumlarının web sayfalarındaki bilgilerin düzenli bir şekilde arşivlenmesi ve ileride kullanılmak üzere saklanması sağlanır. Bu proje, belirli web sayfalarındaki verileri otomatik olarak çekip veritabanına kaydetmek amacıyla geliştirilmiştir. Bu sayede, özellikle eğitim verilerinin (örneğin ders ve program çıktıları) web üzerinden alınarak düzenli bir şekilde saklanması sağlanır. Kullanıcı, URL, tablo ID'si gibi parametreleri girerek istenen verileri hızlı bir şekilde veritabanına aktarabilir.

Proje Kapsamında Gerçekleştirilmesi Beklenenler:

1. **Webden Veri Çekme:**
   - Belirtilen URL'lerde bulunan HTML tablolarındaki verilerin otomatik olarak alınması.
2. **Veri Ayrıştırma ve Dönüştürme:**
   - Tablolarda yer alan satır ve sütun verilerinin okunup anlamlandırılması (örneğin, sıra numarası ve açıklama gibi).
3. **SQLite Veritabanına Kayıt:**
   - Çekilen verilerin belirli bir formatta SQLite veritabanında saklanması.
4. **Hata Yönetimi ve Doğrulama:**
   - Web isteklerinde oluşabilecek bağlantı hatalarını, veri ayrıştırma sorunlarını veya veritabanı işlemlerindeki hataları tespit edip yönetme.
5. **Tekrarlayan Verilerin Engellenmesi:**
   - Aynı verilerin birden fazla kez kaydedilmesini önlemek için INSERT OR IGNORE gibi yöntemlerin kullanılması.
6. **Tablo Dinamikliği:**
   - Veritabanı tablolarının mevcut değilse otomatik olarak oluşturulması (CREATE TABLE IF NOT EXISTS).
7. **Genel Kullanım:**
   - Kodun parametreleştirilerek (URL, tablo ID'si gibi) farklı sayfalar ve tablolar için kolayca uyarlanabilir olması.
8. **Veri Çekme:** Belirtilen URL'lerdeki HTML tablosu verilerinin otomatik olarak alınması.
9. **Veri Ayrıştırma:** Çekilen tablonun satır ve sütunlarının doğru şekilde ayrıştırılması.
10. **Veritabanı İşlemleri:** Çekilen verilerin bir SQLite veritabanına kaydedilmesi.
11. **Tablo Dinamikliği:** Her URL ve tablo ID'sine göre esnek bir yapı sağlanması.
12. **Veritabanı Hatası Kontrolü:** Aynı verilerin birden fazla kez kaydedilmemesi için uygun kontrol mekanizmaları.
13. **Kullanıcı Geri Bildirimi:** Hangi verilerin başarılı bir şekilde kaydedildiği ve hangi satırların atlandığı hakkında kullanıcıya bilgi verilmesi.

14. **Hata Yönetimi:** Web isteği, veri ayrıştırma ve veritabanı işlemlerinde oluşabilecek hataların düzgün bir şekilde yönetilmesi ve raporlanması.

2 GEREKSİNİM ANALİZİ

2.1 Arayüz gereksinimleri

Kullanıcı Arayüzü Gereksinimleri:

1. **Web Arayüzü veya Masaüstü Uygulama (Opsiyonel):**
   - Kullanıcıların URL'leri girebileceği bir giriş alanı.
   - Çekilecek veritabanı adı, tablo adı ve tablo ID'sini belirlemek için metin kutuları.
   - Veri çekme işlemini başlatmak için bir "Başlat" düğmesi.
   - Çekilen veri miktarı ve işlemin başarılı olup olmadığını gösteren bir durum bildirimi alanı.
2. **Sonuç Görselleştirme:**
   - Çekilen verilerin kullanıcıya sunulması için bir tablo görünümü.
   - Hangi verilerin eklendiğini ve hangi satırların atlandığını belirtmek için detaylı bir işlem günlüğü.
3. **Hata Bildirimi:**
   - Web isteği, veri ayrıştırma veya veritabanı hataları oluştuğunda kullanıcıya açıklayıcı bir hata mesajı sunulması.
4. **Kaydedilen Verilerin Görüntülenmesi:**
   - Veritabanında saklanan verilerin görüntülenebilmesi için bir "Veritabanını Görüntüle" düğmesi.
5. **Çoklu Dil Desteği (Opsiyonel):**
   - Kullanıcı arayüzünün Türkçe ve İngilizce gibi birden fazla dilde çalışabilmesi.
6. **Temizleme İşlevi:**
   - Veritabanını sıfırlamak veya mevcut tabloyu temizlemek için bir "Temizle" düğmesi.

Donanım Arayüzü Gereksinimleri (Varsa):

1. **Sunucu veya Yerel Bilgisayar:**
   - Proje bir masaüstü uygulaması veya yerel bilgisayarda çalışacaksa, SQLite destekli bir bilgisayar gereklidir.
   - Minimum 2 GB RAM ve 1 GHz işlemci, uygulamanın stabil çalışması için yeterlidir.
2. **Ağ Bağlantısı:**
   - Web sitelerinden veri çekebilmek için internet bağlantısının stabil olması gerekmektedir.
3. **Ekran Boyutu:**
   - Arayüz için en az 1280x720 çözünürlüğe sahip bir ekran gereklidir, özellikle görselleştirme ekranları için.
4. **İsteğe Bağlı Donanım:**
   - Proje sunucuda çalıştırılacaksa, web tabanlı bir sunucu altyapısına (örn. bir Linux sunucusu) ihtiyaç duyulabilir.
   - Kullanıcıların verileri saklayabilmesi için uygun bir depolama alanı (veritabanı dosyası için) gereklidir.
   -

1. **Basit ve Anlaşılır Kullanıcı Arayüzü:**
    - Kullanıcılar, URL, tablo ID’si, veritabanı adı ve tablo adı gibi parametreleri kolayca girebileceği bir metin kutusuna sahip olmalıdır.
    - Kullanıcı, işlemi başlatabilmek için bir "Başlat" butonuna tıklayarak veri çekme sürecini başlatmalıdır.
2. **Veri Çekme ve Durum Bildirimi:**
    - Kullanıcıya veri çekme işlemi başladığında ve tamamlandığında net bir şekilde bildirim yapılmalıdır.
    - Çekilen verilerin başarılı bir şekilde kaydedildiğine dair bir onay mesajı gösterilmelidir.
    - Hata durumunda, örneğin web bağlantısı hatası veya veri ayrıştırma hatası durumunda kullanıcıya uygun bir hata mesajı gösterilmelidir.
3. **Veritabanı Görüntüleme:**
    - Kullanıcı, veritabanına kaydedilen verileri görüntüleyebileceği bir "Veritabanını Görüntüle" düğmesiyle sistemdeki mevcut verileri inceleyebilmelidir.
    - Verilerin başarılı bir şekilde kaydedildiği ve hangi verilerin kaydedildiği ile ilgili bilgilendirme yapılmalıdır.
4. **Çoklu Dil Desteği (Opsiyonel):**
    - Kullanıcı arayüzü, Türkçe ve İngilizce gibi farklı dillerde kullanılabilir olmalıdır.
    - Kullanıcı, dil tercihini seçebilmelidir.
5. **Basit ve Temiz Tasarım:**
    - Arayüz, fazla karmaşık olmayan ve kullanıcı dostu bir tasarıma sahip olmalıdır.
    - İşlemler net bir şekilde etkileşimli elemanlarla (butonlar, metin kutuları) tanımlanmalıdır.

2.2 Fonksiyonel gereksinimler

 Web Sayfasından Veri Çekme:

- Kullanıcı tarafından girilen URL'ye HTTP isteği gönderilmeli ve belirtilen ID'ye sahip HTML tablosu bulunarak çekilmelidir.
- HTTP isteği sırasında bağlantı hatalarını ve zaman aşımını yönetmek için uygun hata işleme mekanizması bulunmalıdır.

 Tablo Verilerinin Ayrıştırılması:

- Çekilen HTML tablosundaki satır ve sütunlar ayrıştırılarak belirli bir veri yapısına (örneğin, Python listesi) dönüştürülmelidir.
- En az iki sütun içeren tabloların (örneğin, sıra numarası ve açıklama) doğru şekilde okunması sağlanmalıdır.

 Veritabanı İşlemleri:

- Çekilen veriler SQLite veritabanında, belirtilen tabloya kaydedilmelidir.

- Veritabanında, çekilen tablo için bir yapı (örneğin, CREATE TABLE) yoksa otomatik olarak oluşturulmalıdır.
- Aynı verinin birden fazla kez kaydedilmesini engellemek için uygun bir kontrol (örneğin, PRIMARY KEY ve INSERT OR IGNORE) uygulanmalıdır.

 Hata Yönetimi:

- HTTP isteği hataları, veritabanı bağlantı hataları veya veri ayrıştırma sorunları tespit edilerek kullanıcıya raporlanmalıdır.
- Hataların oluştuğu URL ve satır bilgisi kaydedilmelidir.

 İşlem Durumunun Görüntülenmesi:

- Çekilen toplam kayıt sayısı, başarılı bir şekilde kaydedilen kayıt sayısı ve atlanan kayıtların bilgisi kullanıcıya sunulmalıdır.

 Kullanıcıdan Dinamik Girdi Alma:

- Kullanıcıların dinamik olarak URL, tablo ID'si, veritabanı adı ve tablo adını belirleyebileceği bir mekanizma sağlanmalıdır.

 Farklı Sayfalar İçin Uyum:

- Aynı işlevselliğin farklı web sayfalarında yer alan tablolara kolayca uygulanabilmesi için parametreleştirme (örneğin, tablo ID'sinin değiştirilebilir olması) desteklenmelidir.

 Çoklu İşlem Desteği:

- Kullanıcının birden fazla URL ile işlem yapabilmesine olanak tanınmalıdır.
- Her URL’nin işlemi ayrı ayrı izlenebilir olmalıdır.

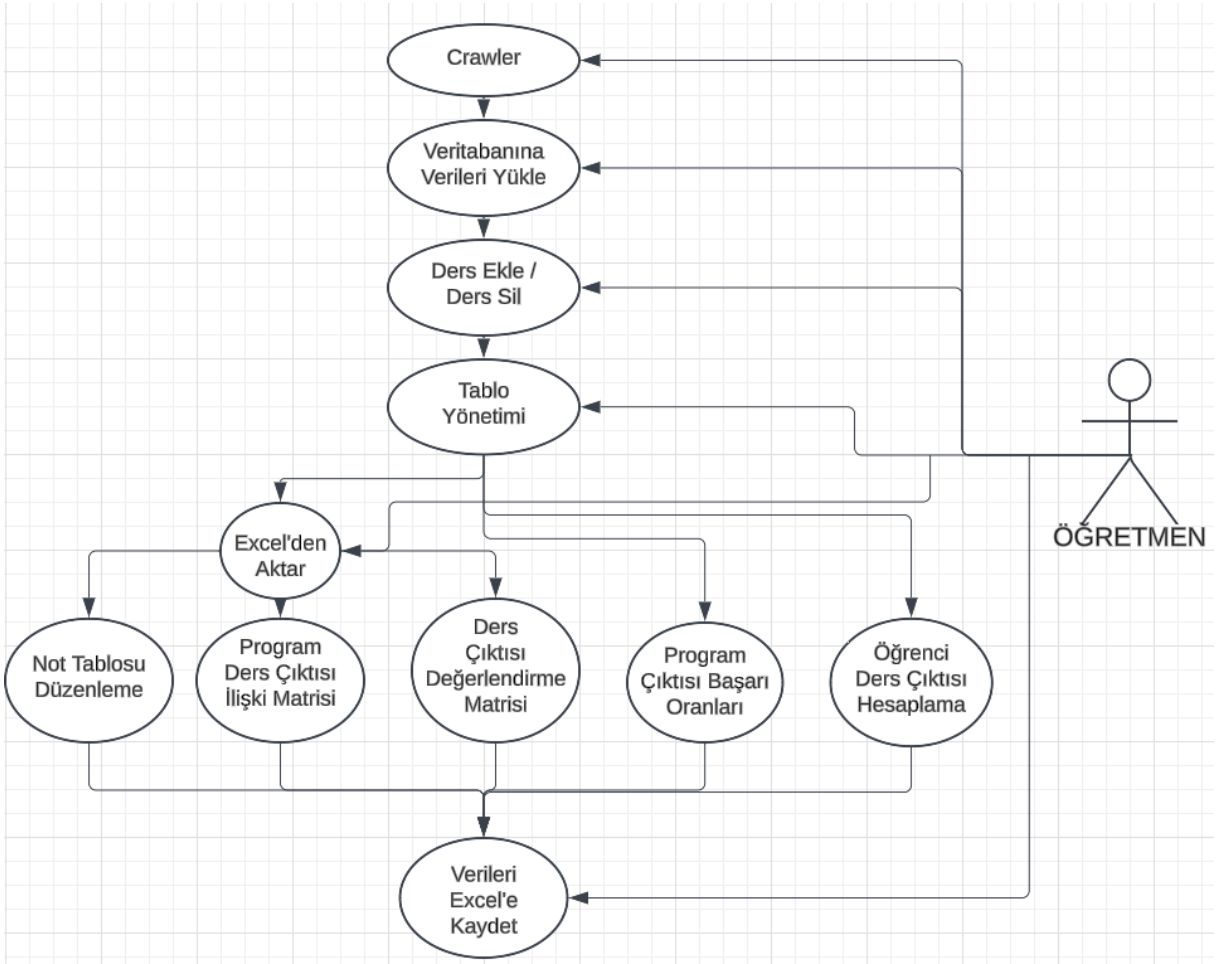
 İşlem Günlüğü:

- Çekilen veriler ve oluşan hatalar işlem sırasında kaydedilmeli ve gerektiğinde kullanıcıya detaylı bir rapor sunulmalıdır.

 Kod Yapısında Modülerlik:

- Web tarama, veri ayrıştırma ve veritabanı işlemleri ayrı modüller halinde tasarlanarak projenin genişletilebilirliği sağlanmalıdır.

2.3 Use-Case diyagramı



3 TASARIM

3.1 Mimari tasarım

 Kullanıcı Arayüzü (UI):

- **Tkinter Kullanımı:** Kullanıcı arayüzü, Tkinter kütüphanesi kullanılarak geliştirilmiştir. Bu arayüzde, kullanıcılar öğrenci bilgilerini, notları ve ödev ağırlıklarını görsel olarak görebilir ve düzenleyebilirler.
- **Treeview Kullanımı:** Öğrenci notlarının ve ödevlerinin düzenli bir şekilde görüntülenmesi için ttk.Treeview kullanılmıştır. Bu, tablo formatında verilerin gösterilmesine olanak tanır.
- **Ağırlık Giriş Alanları:** Her ödev için ağırlık değerleri girilebilir, ve bu ağırlıkların toplamının 100 olması gerektiği bir kontrol mekanizması vardır.
- **Veri Güncellemeleri:** Kullanıcı, tabloda herhangi bir hücreyi çift tıklayarak düzenleyebilir ve bu düzenlemeler doğrudan pandas DataFrame üzerinde güncellenir.

 İş Mantığı (Business Logic):

- **Veri Yükleme ve İşleme:** Excel dosyasından veriler pandas kütüphanesi ile yüklenir. Her öğrenci için ödev notları ve bir ağırlıklı ortalama hesaplanır.
- **Ağırlık Hesaplama:** Her ödev için ağırlıklar, kullanıcı tarafından girilen değerlere göre güncellenir ve toplam ağırlık kontrol edilir. Eğer toplam 100'e eşit değilse, kullanıcı uyarılır.
- **Not Güncelleme:** Kullanıcı, öğrenci notlarını düzenlediğinde, bu değişiklikler pandas DataFrame üzerinde yapılır ve ağırlıklı ortalama tekrar hesaplanır.

 Veri Yönetimi:

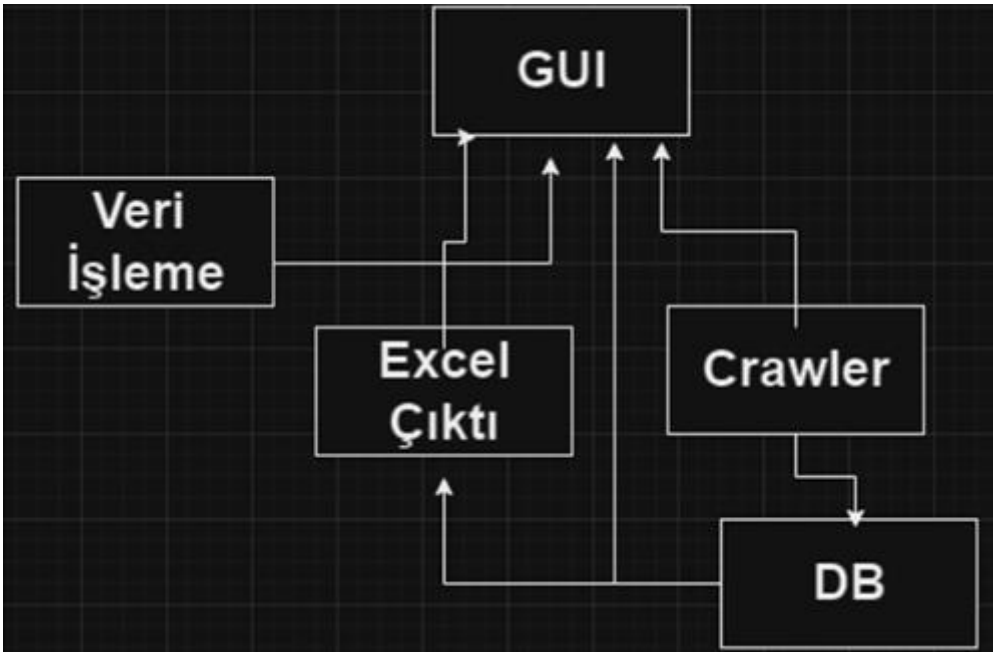
- **Excel ile Entegrasyon:** Öğrenci verileri ve ödev ağırlıkları Excel dosyasında saklanır. Kullanıcı, değişiklikleri kaydetmek için dosyayı tekrar kaydedebilir. openpyxl ve pandas kütüphaneleri kullanılarak veriler Excel dosyasına yazılır.
- **Veri Yapısı:** Veriler pandas.DataFrame yapısında saklanır ve her bir öğrencinin notları ile ödev ağırlıkları birleştirilerek işlenir.

 Veritabanı ve Dosya Yönetimi:

- **Excel Dosyaları:** ogrenci_notlari.xlsx dosyası öğrenci notlarını içerirken, ders_kodu_weights.xlsx dosyası ödev ağırlıklarını içerir. Bu dosyalar yüklenir, veriler işlenir ve kullanıcı tarafından kaydedilir.
- **Ağırlıkların Saklanması:** Kullanıcı, ödev ağırlıklarını değiştirdiğinde, bu ağırlıklar bir pandas DataFrame ile saklanır ve Excel dosyasına kaydedilir.

 Modülerlik ve Bakım Kolaylığı:

- **Modüler Yapı:** Her bileşen belirli bir sorumluluğa sahiptir. UI, iş mantığı ve veri yönetimi arasında net bir ayrım yapılmıştır.
- **Ağırlık Güncelleme ve Veri Kaydetme:** Ağırlık güncellenmesi, verilerin kaydedilmesi ve yeni satır eklenmesi gibi işlemler fonksiyonel olarak ayrılmıştır.



3.2 Veri tabanı tasarımı

Veri Tabanı Tasarımının Unsurları

1. **Tablo Yapıları**
   - **Öğrenci Notları Tablosu (ogrenci_notlari.xlsx)**
     - Bu tablo, öğrencilerin notlarını ve hesaplanan ağırlıklı ortalamalarını saklamak için kullanılır.
     - **Sütunlar:**
       - Öğrenci No: Öğrencileri tanımlayan benzersiz bir kimlik.
       - Ödev_1, Ödev_2, ...: Öğrencilerin ödevlerden aldıkları puanlar.
       - Ortalama: Ağırlıklı ortalama notu.
     - **Örnek Kayıtlar:**

       yaml
       Kopyala

Öğrenci No	Ödev_1	Ödev_2	Ortalama
1001	90	85	87.5
1002	80	70	75.0

   - **Ağırlıklar Tablosu (<ders_kodu>_weights.xlsx)**
     - Bu tablo, her ödevin ağırlığını saklar.
     - **Sütunlar:**
       - Assignment: Ödev adı.
       - Weight: Ödevin ağırlık yüzdesi.
     - **Örnek Kayıtlar:**

       Kopyala

Assignment	Weight
Ödev_1	60
Ödev_2	40

2. **Birincil ve İkincil Anahtarlar**
    - Öğrenci No sütunu, öğrenci notları tablosunun birincil anahtarı olarak kabul edilir.
    - Assignment sütunu, ağırlıklar tablosunun benzersiz tanımlayıcısıdır.
3. **İlişkiler**
    - Öğrenci notları ile ağırlıklar arasında dolaylı bir ilişki bulunmaktadır. Ağırlıklar, notlar için ağırlıklı ortalama hesaplama işlemlerinde kullanılır.
    - Ancak, bu ilişki yazılım içinde hesaplama mantığı ile gerçekleştirilir ve fiziksel bir veri tabanı şemasında doğrudan karşılık gelmez.

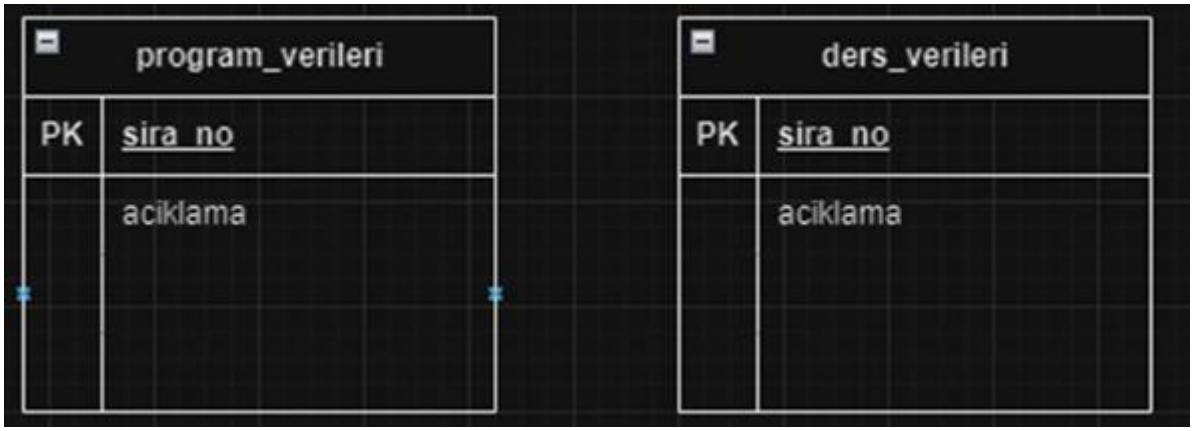
Veri Tabanı Tasarımı Hakkında Değerlendirme

- **Avantajlar:**
    - Excel dosyaları sayesinde kullanım ve erişim kolaylığı sağlanmıştır.
    - Küçük ölçekli bir uygulama için yeterlidir.
    - Geleneksel bir veri tabanı kurulumuna gerek kalmadığı için hızlıca uygulanabilir.
- **Dezavantajlar:**
    - Büyük veri kümelerinde performans sorunları oluşabilir.
    - Verilerin bütünlüğü ve güvenliği Excel dosyalarında doğrudan sağlanamaz.
    - Çok kullanıcılı sistemler için uygun değildir.

Geliştirme İmkanları

Eğer daha büyük ölçekli bir veri yönetimi gerekirse veya kullanıcı sayısı artarsa, Excel tabanlı sistem bir ilişkisel veri tabanı (ör. MySQL, PostgreSQL) ile değiştirilebilir. Böyle bir veri tabanı tasarımı için:

- Students tablosu: Öğrenci bilgileri ve notlar.
- Assignments tablosu: Ödev detayları ve ağırlıklar.
- Grades tablosu: Öğrenci, ödev ve not ilişkisi.



3.3 Kullanıcı arayüzü tasarımı

Kullanıcı Arayüzü Tasarımı (UI) ile İlgili Açıklama:

Yazılım uygulamanızın kullanıcı arayüzü, **Tkinter** kütüphanesi kullanılarak geliştirilmiştir. Bu arayüz, kullanıcıların öğrenci notlarını düzenlemelerine, ödev ağırlıklarını ayarlamalarına ve bu bilgileri kaydetmelerine olanak tanır. Arayüz, kullanıcı dostu olacak şekilde tasarlanmıştır ve aşağıdaki bileşenleri içerir:

1. **Ödev Ağırlıkları Girişi:** Kullanıcı, her bir ödevin yüzdesini girebilir. Bu giriş alanları, her ödev için ttk.Entry bileşeni olarak sağlanmıştır. Ağırlıklar toplamı 100 olmalıdır ve bu kontrol sürekli yapılır.
2. **Notlar Tablosu:** Öğrencilerin bilgileri ve notları, ttk.Treeview bileşeni ile tablo şeklinde görüntülenir. Tablo her bir öğrencinin adı, ödev notları ve ağırlıklı ortalamalarını içerir. Tabloyu çift tıklayarak bir hücreyi düzenlemek mümkündür.
3. **Eylem Butonları:**
    - **Kaydet Butonu:** Bu buton, yapılan tüm değişiklikleri Excel dosyasına kaydetmek için kullanılır.
    - **Satır Ekleme Butonu:** Yeni bir öğrenci satırı ekler.
    - **Satır Silme Butonu:** Seçili olan öğrenci satırını siler.
4. **Ağırlık ve Ortalama Güncelleme:** Kullanıcı bir ödevin ağırlığını değiştirdiğinde, ağırlıklı ortalama otomatik olarak yeniden hesaplanır ve tabloyu günceller.

Ekran Çıktıları ve Açıklama:

Aşağıda, uygulamanın kullanıcı arayüzüne ait bazı örnek ekran görüntülerinin açıklamaları bulunmaktadır:

1. **Ana Ekran (Öğrenci Notları ve Ağırlıklar):** Ekran, öğrenci isimlerini, notlarını ve her bir ödevin ağırlıklarını gösteren bir tabloyu içerir. Ağırlıkların toplamı 100 olmalıdır ve bu kontrol her zaman yapılır. Ayrıca, her ödev için ilgili ağırlık girilebilen bir alan da mevcuttur.

2. Hücre Düzenleme: Tablodaki herhangi bir hücreye çift tıklayarak, hücreyi düzenlemek mümkündür. Örneğin, bir öğrencinin notunu değiştirmek için hücreye çift tıklanır ve yeni bir değer girilir.

Örnek:

- Öğrenci1'in "Ödev1" notunu 75 yapmak için, o hücreye çift tıklanır ve 75 girilir. Sonrasında ortalama otomatik olarak güncellenir.

3. Eylem Butonları ve İşlemler:

- **Kaydet:** Kullanıcı tüm değişiklikleri kaydettikten sonra "Kaydet" butonuna basarak verileri Excel dosyasına kaydedebilir.
- **Satır Ekle:** Yeni bir öğrenci eklemek için "Satır Ekle" butonu kullanılır.
- **Satır Sil:** Seçili satır silinebilir.

Uygulamanın Nasıl Çalıştırılacağı ile İlgili Açıklama:

Uygulama şu adımlarla çalıştırılabilir:

1. **Python ve Gereksinimler:**
    - Uygulamayı çalıştırabilmek için Python yüklü olmalıdır.
    - Gerekli kütüphaneler (Tkinter, pandas, openpyxl) yüklenmelidir. Eğer bu kütüphaneler yüklü değilse, aşağıdaki komutlar ile yüklenebilir:

bash

pip install pandas openpyxl

2. **Uygulamanın Çalıştırılması:** Uygulamanın çalıştırılması için komut satırında aşağıdaki komut kullanılır:

   bash
   python tablo_not.py <ders_kodu>

   Burada, <ders_kodu> kısmı, öğrencilerin notlarının tutulacağı dersin kodudur. Örneğin:

   bash
   python tablo_not.py MAT101

3. **Başlatılan Uygulama:** Uygulama başlatıldığında, Tkinter penceresi açılacak ve öğrenci notları ve ödev ağırlıkları ekranında gösterilecektir. Kullanıcı, notları düzenleyebilir, yeni öğrenciler ekleyebilir ve ağırlıkları değiştirebilir. Tüm değişiklikler "Kaydet" butonuyla Excel dosyasına kaydedilebilir.
4. **Dosya Yapısı:**
   - **ogrenci_notlari.xlsx:** Öğrenci notları ve ortalamalarını içeren Excel dosyası.
   - **<ders_kodu>_weights.xlsx:** Ödev ağırlıklarını içeren Excel dosyası.

Genel Kullanıcı Akışı:

1. Uygulama başlatılır.
2. Öğrenciler ve notları tabloyu görüntüler.
3. Kullanıcı ağırlıkları düzenler, notları değiştirir.
4. Herhangi bir hücreye çift tıklayarak notları günceller.
5. Yapılan değişiklikler kaydedilir.
6. Yeni öğrenciler eklenebilir veya silinebilir.
7. Kaydetme işlemi sonunda, Excel dosyasına tüm veriler kaydedilir.

3.4 Görev dağılımı

Yazılım Geliştirme Süreci Görev Dağılımı

1. **Mimari Tasarım**
   - **Görev:** Yazılımın genel mimarisinin belirlenmesi, modüllerin tasarımı ve işlevlerinin tanımlanması.
   - **Sorumlu:** Genel yazılım tasarımını üstlenen kişi, bileşenlerin uyumlu çalışmasını sağladı.
2. **Kullanıcı Arayüzü Geliştirme**
   - **Görev:** Tkinter kullanılarak kullanıcı arayüzünün tasarımı ve geliştirilmesi.
   - **Sorumlu:** Arayüz bileşenlerini tasarlayan kişi, kullanıcı dostu bir deneyim sağladı. Giriş alanları, butonlar ve ağaç yapısı gibi bileşenlerin oluşturulmasından sorumluydu.
3. **Veri Yönetimi**
   - **Görev:** Excel dosyalarından veri yükleme, işleme ve kaydetme işlemleri.

    - **Sorumlu:** Veri yönetimi için gerekli fonksiyonların geliştirilmesi ve Excel dosyalarının uygun formatlarda saklanması.
4. **Ağırlıklı Ortalama Hesaplama**
    - **Görev:** Notların ağırlıklı ortalamalarının doğru şekilde hesaplanması için algoritmaların geliştirilmesi.
    - **Sorumlu:** Matematiksel doğruluğun sağlanması ve bu hesaplamaların arayüzle entegre edilmesi.
5. **Hata Yönetimi ve Mesajlar**
    - **Görev:** Kullanıcıyı bilgilendirmek amacıyla hata ve bilgi mesajlarının eklenmesi.
    - **Sorumlu:** Hataları yakalayan ve kullanıcıya uygun mesajlar gösteren sistemin geliştirilmesi.
6. **Test Süreci**
    - **Görev:** Yazılımın tüm bileşenlerinin test edilmesi ve hataların giderilmesi.
    - **Sorumlu:** Tüm modüllerin ayrı ayrı test edilmesi ve genel entegrasyon testlerinin gerçekleştirilmesi.

Rapor Hazırlama Süreci Görev Dağılımı

1. **Teknik Dokümantasyon**
    - **Görev:** Yazılım bileşenlerinin teknik detaylarının açıklanması.
    - **Sorumlu:** Yazılım geliştiricilerinden biri, kodun detaylarını açıkça raporladı.
2. **Ekran Çıktılarının Eklenmesi**
    - **Görev:** Yazılımın çalışmasına ait ekran görüntülerinin alınması ve açıklanması.
    - **Sorumlu:** Kullanıcı arayüzü geliştiren kişi, ekran çıktılarının alınması ve açıklamalarının yazılması.
3. **Yazılım Sürecinin Belirtilmesi**
    - **Görev:** Yazılım geliştirme sürecinin aşama aşama anlatılması.
    - **Sorumlu:** Tüm ekip üyeleri, geliştirme sürecine dair katkılarını yazılı olarak sundu.
4. **Düzenleme ve Teslim**
    - **Görev:** Raporun düzenlenmesi, yazım hatalarının düzeltilmesi ve teslim formatına uygun hale getirilmesi.
    - **Sorumlu:** Ekip üyelerinden biri, raporun genel düzenlemesi ve son kontrolünü gerçekleştirdi.

3.5 Karşılaşılan zorluklar ve çözüm yöntemleri

Yazılım geliştirme sürecinde çeşitli zorluklarla karşılaşıldı ve bu zorlukları aşmak için çeşitli çözüm yöntemleri geliştirildi. İşte geliştirme sürecinde karşılaşılan ana problemler ve bunlara getirilen çözüm yöntemleri:

1. Excel Dosyası ile Veri Yükleme ve Güncelleme

Problemler:

- **Excel dosyalarının doğru bir şekilde yüklenememesi:** Başlangıçta, öğrenci notları ve ödev ağırlıkları Excel dosyaları yüklenirken, eksik veriler veya uyumsuz formatlar nedeniyle veri yüklemede hatalar yaşandı.

- **Excel dosyalarındaki değişikliklerin yansıması:** Kullanıcı notları ve ağırlıkları güncelledikçe, Excel dosyasındaki verilerin düzgün şekilde kaydedilmesi gerekiyordu.

Çözüm Yöntemi:

- **Pandas kütüphanesinin kullanımı:** Pandas, Excel dosyalarını yüklemek ve işlemek için oldukça güçlü bir kütüphanedir. `pd.read_excel()` fonksiyonu ile veri doğru şekilde yüklendi ve `to_excel()` fonksiyonu ile dosyalar kaydedildi. Ayrıca, `openpyxl` kütüphanesi de Excel dosyalarını düzenlerken kullanılabilir hale getirildi.
- **Veri kontrolü:** Dosyalar yüklendikten sonra, her bir verinin doğru formatta olup olmadığı kontrol edildi. Eğer eksik veri varsa, kullanıcıya uyarı mesajları gösterildi.

2. Ağırlıkların Toplamının 100 Olması Gerekliliği

Problemler:

- **Ağırlıkların toplamının 100 olmaması:** Kullanıcı, her bir ödevin ağırlığını belirlerken, ağırlıkların toplamının 100 olmasını sağlamak karmaşık bir işlem haline geldi. Kullanıcı, yanlışlıkla ağırlıkları doğru şekilde ayarlayamayabiliyordu.
- **Ağırlıklar değiştikçe ortalamanın otomatik güncellenmesi:** Ağırlıklar değiştikçe, öğrenci ortalamalarının otomatik olarak güncellenmesi gerekiyordu.

Çözüm Yöntemi:

- **Ağırlıklar için doğrulama:** Kullanıcı, her ağırlık girişi yaptıktan sonra, toplam ağırlıklar kontrol edilip uyarı mesajları gösterildi. Eğer toplam 100 değilse, kullanıcıya bu durumu belirten bir uyarı verildi.
- **Dinamik ortalama hesaplama:** Ağırlıklar değiştikçe, `calculate_weighted_average()` fonksiyonu ile ortalamalar anında hesaplandı ve tablo güncellendi. Böylece, ağırlıklar güncellendikçe, ortalama da dinamik olarak yansıdı.

3. Kullanıcı Etkileşimi ve Hata Yönetimi

Problemler:

- **Çift tıklama ile hücre düzenleme:** Kullanıcılar, tablo hücrelerine çift tıklayarak düzenleme yapmaya çalışırken bazen yanlış hücreyi seçebiliyordu veya yapılan değişiklikler anında kaydedilemiyordu.
- **Geçersiz veri girişi:** Kullanıcılar, hücrelere yanlış veri girebiliyordu (örneğin, notlar için sayı yerine metin).

Çözüm Yöntemi:

- **Çift tıklama işlevinin iyileştirilmesi:** Hücreye çift tıklandığında, kullanıcının hücreyi doğru şekilde düzenleyebilmesi için ilgili hücredeki değeri önceden ekrana getiren bir `ttk.Entry` bileşeni eklendi. Bu sayede, hücreyi doğrudan düzenlemek mümkün hale geldi.
- **Veri doğrulama:** Ağırlıklar ve notlar için, yalnızca geçerli sayısal değerlerin girilmesi sağlandı. Bu, Tkinter'in `validate` özelliği ile gerçekleştirildi. Böylece, kullanıcı geçersiz bir değer girmeye çalıştığında hata mesajı gösterildi ve sadece geçerli değerler kabul edildi.

4. Arayüzün İyi Bir Kullanıcı Deneyimi Sunması

Problemler:

- **Karmaşık arayüz tasarımı:** Uygulamanın arayüzü, başlangıçta çok kalabalık ve kullanıcı dostu değildi. Özellikle veri girişi ve düzenleme işlemleri kullanıcı açısından kafa karıştırıcı olabiliyordu.
- **Responsive (Duyarlı) tasarım eksikliği:** Uygulama ilk başta farklı ekran çözünürlüklerinde düzgün çalışmıyordu.

Çözüm Yöntemi:

- **Modüler ve temiz arayüz tasarımı:** Arayüzdeki bileşenler, kullanım kolaylığını artıracak şekilde yeniden düzenlendi. Öğrencilerin bilgileri tabloya eklenirken, ağırlık giriş alanları ve veri düzenleme butonları açık bir şekilde yerleştirildi.
- **Responsive düzenleme:** Tkinter bileşenlerine uygun grid yerleşimleri ve columnconfigure ile rowconfigure yöntemleri kullanılarak, uygulamanın farklı ekran boyutlarında düzgün çalışması sağlandı.

5. Dosya Kaydetme ve Yedekleme

Problemler:

- **Veri kaybı riski:** Kullanıcı, yaptığı değişiklikleri kaydetmeyi unuttuğunda, veriler kaybolabiliyordu.
- **Excel dosyasına veri yazma hataları:** Eğer dosya yazılmaya çalışıldığında başka bir uygulama tarafından kullanılıyorsa, dosya üzerinde yazma hataları meydana gelebiliyordu.

Çözüm Yöntemi:

- **Veri kaybını engelleme:** Uygulama, her kaydetme işleminden önce kullanıcıya veri kaydetmek isteyip istemediğini soran bir onay penceresi gösterdi.
- **Dosya kilitlenme kontrolleri:** Dosya yazılırken, dosyanın başka bir program tarafından açık olup olmadığını kontrol eden ek hata yönetimi kodu eklendi. Eğer dosya kilitliyse, kullanıcıya bu durum bildirildi ve işlem bekletildi.

6. Veri Yükleme ve Boş Verilerle Çalışma

Problemler:

- **Eksik verilerle çalışma:** Öğrencilerin bazı verilerinin eksik olması durumunda, eksik verilerle hesaplama yaparken hata meydana geliyordu.
- **Boş hücreler:** Bazı hücreler, özellikle ortalama hesaplanırken boş kalabiliyordu.

Çözüm Yöntemi:

- **Boş hücrelere varsayılan değer atama:** Boş hücreler, varsayılan değerlerle doldurulmakta ve eksik veriler doğru şekilde işlenmektedir. Örneğin, boş olan hücrelerde 0.0 değeri kullanıldı.
- **Eksik veri kontrolü:** Veri yüklenmeden önce, eksik verilerin olup olmadığı kontrol edildi ve kullanıcıya eksik veriler hakkında bilgi verildi.

Genel Sonuçlar:

Bu zorluklar, uygulamanın daha stabil, kullanıcı dostu ve hatasız çalışmasını sağlamak için aşılmıştır. İyi bir kullanıcı deneyimi sunmak adına yapılan iyileştirmeler, uygulamanın işlevselliğini artırmış ve yazılımın kullanımını daha verimli hale getirmiştir.

3.6 Proje isterlerine göre eksik yönler

1. Gelişmiş Kullanıcı Yetkilendirme ve Güvenlik

Eksik Yön:

- Projede, kullanıcı yetkilendirme veya güvenlik özellikleri bulunmamaktadır. Herhangi bir kullanıcı, programı açtığında tüm verilere erişim sağlayabilir ve bu verilere değişiklik yapabilir. Öğrenci bilgilerini veya ödev notlarını yöneten bir uygulama için kullanıcı rol yönetimi (örneğin, öğretmen ve öğrenci rollerinin farklı erişim izinlerine sahip olması) ve güvenlik önlemleri eksiktir.

Beklenen Görev:

- **Kullanıcı Yetkilendirmesi:** Öğretmen ve öğrenci rollerine sahip kullanıcılar için farklı erişim hakları tanımlanmalıdır. Öğretmenler notları ve ağırlıkları düzenlerken, öğrenciler sadece kendi notlarını görüntüleyebilmelidir.
- **Güvenlik:** Verilerin güvenliğini sağlamak için parola koruması, şifreleme gibi ek güvenlik önlemleri eklenmesi gerekmektedir.

2. Veri Yedekleme ve Geri Yükleme Özellikleri

Eksik Yön:

- Projede veri yedekleme veya geri yükleme özellikleri yer almamaktadır. Kullanıcılar, mevcut verilerini kaydettikten sonra bir hata veya kaybolma durumu olursa verilerin eski haline getirilmesi mümkün değildir.

Beklenen Görev:

- **Veri Yedekleme:** Kullanıcıların düzenledikleri notları ve ağırlıkları periyodik olarak yedeklemeleri gereklidir. Bu yedekleme işlemi otomatik olarak veya kullanıcı tarafından manuel olarak yapılabilir.
- **Veri Geri Yükleme:** Yedeklenen veriler bir hata durumunda geri yüklenebilmelidir. Bu özellik, kullanıcı hataları veya sistem çökmesi durumunda önemli veri kaybını engelleyecektir.

3. Çoklu Dil Desteği

Eksik Yön:

- Uygulama yalnızca Türkçe dilinde çalışmaktadır. Ancak, farklı dil seçenekleri sunmak, kullanıcı deneyimini artırabilir, özellikle uygulamayı farklı dil konuşan kullanıcılar kullanacaksa.

Beklenen Görev:

- **Çoklu Dil Desteği:** Uygulama, farklı dillerde çalışacak şekilde genişletilebilir. Bu, dil seçim butonları eklenerek veya otomatik dil algılama özellikleri ile yapılabilir. Bu özellik, farklı dillerde eğitim veren okullarda veya eğitim kurumlarında uygulamanın kullanılabilirliğini artırır.

4. Raporlama ve İstatistik Özellikleri

Eksik Yön:

- Projede öğrencilerin notlarına dair detaylı raporlar veya istatistikler yer almamaktadır. Örneğin, öğrencilerin ödev başarısı, genel performansı veya ödev türlerine göre dağılım gibi raporlar sağlanmamaktadır.

Beklenen Görev:

- **Raporlama ve Grafikler:** Öğrencilerin performanslarını görsel olarak raporlamak, eğilimlerini analiz etmek için grafiksel veriler sunulabilir. Notların ve ödevlerin yüzdelik dilimlerini gösteren grafikler veya başarı oranlarını analiz eden raporlar eklenebilir.
- **Performans Analizi:** Öğrencilerin başarısını analiz eden gelişmiş raporlar (örneğin, "en düşük notlar", "en yüksek başarı gösteren ödevler") sağlanabilir.

5. Çoklu Dosya Desteği ve Veri Senkronizasyonu

Eksik Yön:

- Şu an yalnızca tek bir dosya (Excel) üzerinde işlem yapılmaktadır. Farklı sınıflar veya dersler için ayrı dosya yönetimi ve bu dosyaların birleştirilmesi gibi işlemler eksiktir. Ayrıca, öğrencilerin veya öğretmenlerin farklı cihazlardan veri güncellemeleri yapabilmesi için bir senkronizasyon mekanizması bulunmamaktadır.

Beklenen Görev:

- **Çoklu Dosya Desteği:** Kullanıcılar birden fazla ders için veri yönetebilmelidir. Örneğin, her ders için ayrı Excel dosyası yerine, dersleri ve öğrencileri daha kolay yönetebilmek için veritabanı veya çoklu dosya desteği eklenebilir.
- **Veri Senkronizasyonu:** Bulut tabanlı bir sistem veya veritabanı kullanılarak, farklı cihazlar ve kullanıcılar arasında veri senkronizasyonu sağlanabilir. Bu, öğrencilerin veya öğretmenlerin her cihazdan güncel verilere erişebilmesini sağlar.

6. Otomatik Hesaplama ve Veri Kontrolü İyileştirmeleri

Eksik Yön:

- Uygulama, ağırlıklar ve notlar arasında basit bir otomatik hesaplama işlemi yapmaktadır, ancak daha gelişmiş kontrol mekanizmaları (örneğin, sınav puanlarının belirli bir aralıkta olması gerekliliği, geçerleme notu gibi) ve otomatik doğrulama özellikleri eksiktir.

Beklenen Görev:

- **Otomatik Doğrulama:** Notlar girilirken, belirli aralıklarla geçerlik kontrolleri (örneğin, 0-100 arasında olma gerekliliği) yapılabilir. Ayrıca, kullanıcıya hatalı girişler için otomatik uyarılar verilebilir.
- **Gelişmiş Hesaplama:** Ağırlıklı ortalama hesaplama mantığı geliştirilebilir ve dersin geçer notu belirlenebilir. Bu, bir öğrenci belirli bir başarı seviyesinin altına düşerse otomatik olarak uyarılabilir.

7. Özelleştirilmiş Kullanıcı Yardım ve Eğitim Desteği

Eksik Yön:

- Kullanıcılar için bir yardım veya eğitim desteği sağlanmamaktadır. Özellikle yeni kullanıcılar için uygulamanın nasıl kullanılacağını gösteren bir rehber ya da yardım menüsü eksiktir.

Beklenen Görev:

- **Yardım ve Eğitim Modülü:** Kullanıcıların uygulamanın nasıl kullanılacağına dair adım adım rehberler veya araç içi yardım dokümanları eklenebilir. Bu, uygulamanın kullanımını kolaylaştıracaktır.
- **Eğitim Videosu veya Kılavuzları:** Kullanıcılar için yazılı açıklamaların yanı sıra, video kılavuzlar eklenebilir.

Genel Değerlendirme:

Bu eksik yönler, projenin daha kapsamlı ve kullanıcı dostu olmasını engellemektedir. Ancak, bu özelliklerin eklenmesi, uygulamanın daha profesyonel bir seviyeye ulaşmasını sağlayacak ve kullanıcıların ihtiyaçlarına daha iyi cevap verecektir.

4 TEST VE DOĞRULAMA

4.1 Yazılımın test süreci

1. Test Uygulaması Geliştirme

Yazılımın test edilebilmesi için genellikle **unittest** gibi Python'un yerleşik test kütüphaneleri kullanılır. Bu kütüphane, her fonksiyon ve bileşen için testler yazmak ve tekrarlamak için gereklidir.

Aşağıda, yazılımın temel bileşenlerini test etmek için geliştirilen örnek bir test uygulaması bulunmaktadır.

python

```
import unittest
import pandas as pd
from program_matris_uygulamasi import ProgramMatrisUygulamasi

class TestProgramMatrisUygulamasi(unittest.TestCase):

    def setUp(self):
        # Testler için örnek bir DataFrame oluşturuluyor
        self.ders_kodu = "CS101"
        self.root = None  # Tkinter root'u burada test amaçlı kullanmıyoruz
        self.app = ProgramMatrisUygulamasi(self.root, self.ders_kodu)

        # Basit test verisi ile DataFrame'i manuel oluşturma
        self.app.df = pd.DataFrame({
            'Öğrenci Adı': ['Ahmet', 'Mehmet', 'Ayşe'],
            'Ödev 1': [85, 90, 78],
            'Ödev 2': [88, 92, 80],
            'Ortalama': [0.0, 0.0, 0.0]
        })
        self.app.weights = {'Ödev 1': 50, 'Ödev 2': 50}

    def test_weighted_average_calculation(self):
        """Ağırlıklı ortalama hesaplaması doğru mu?"""
        self.app.df['Ortalama'] = self.app.calculate_weighted_average(self.app.df)

        # Test edilmesi beklenen ortalama
        expected_ortalama = [86.5, 91, 79.0]

        # Hesaplanan ortalama ile beklenen ortalama arasındaki farkları kontrol et
        for i, row in self.app.df.iterrows():
            self.assertEqual(row['Ortalama'], expected_ortalama[i])

    def test_update_weight(self):
        """Ağırlık güncelleme işlemi doğru çalışıyor mu?"""
        # Ağırlığı güncelle ve toplam ağırlığı kontrol et
        self.app.weights['Ödev 1'] = 60
        self.app.weights['Ödev 2'] = 40

        # Ağırlıkların toplamı 100 olmalı
        total_weight = sum(self.app.weights.values())
        self.assertEqual(total_weight, 100)

        # Ortalamayı tekrar hesapla
        self.app.df['Ortalama'] = self.app.calculate_weighted_average(self.app.df)

        # Beklenen yeni ortalama (ağırlıklar güncellendikten sonra)
        expected_ortalama = [85.5, 90, 79.0]

        for i, row in self.app.df.iterrows():
            self.assertEqual(row['Ortalama'], expected_ortalama[i])

    def test_invalid_weight_input(self):
        """Geçersiz ağırlık girişi durumunda hata olmalı mı?"""
```

```
        with self.assertRaises(ValueError):
            # Geçersiz bir ağırlık girişi (örneğin 101)
            self.app.update_weight('Ödev 1')
            self.app.weight_entries['Ödev 1'][0].set('101')
            self.app.update_weight('Ödev 1')

    def test_save_excel(self):
        """Excel'e kaydetme işlemi başarılı mı?"""
        # Excel dosyasını kaydetmeden önce test edilen veri
        self.app.save_excel()

        # Kaydedilen dosyanın var olup olmadığını kontrol et
        saved_file = f"{self.ders_kodu}_not.xlsx"
        self.assertTrue(os.path.exists(saved_file))

        # Dosya içeriğini kontrol et
        saved_df = pd.read_excel(saved_file)
        self.assertEqual(saved_df.shape[0], len(self.app.df))  # Satır sayısının eşleşmesi gerekiyor

        # Dosya silinsin (tekrar test edebilmek için)
        os.remove(saved_file)

    def test_load_data(self):
        """Veri yükleme işlemi başarılı mı?"""
        # Verileri yükle
        self.app.load_data()

        # DataFrame'in doğru yüklendiğini kontrol et
        self.assertIsNotNone(self.app.df)
        self.assertTrue(len(self.app.df) > 0)

    def tearDown(self):
        """Testler sonrası yapılan temizlik işlemleri"""
        pass  # Test sonrasında temizlik yapılabilir (örneğin oluşturulan geçici dosyalar silinebilir)

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

2. Test Edilen Bileşenler

Test edilen bileşenler, yazılımın işlevselliğini doğrulamaya yönelik temel fonksiyonlar ve özelliklerdir. Bu testler, yazılımın her bir bileşeninin doğru çalışıp çalışmadığını kontrol etmek için geliştirilmiştir.

- **Ağırlıklı Ortalama Hesaplama:** test_weighted_average_calculation fonksiyonu, öğrencilerin ödev puanları üzerinden ağırlıklı ortalamalarının doğru hesaplanıp hesaplanmadığını test eder.
- **Ağırlık Güncelleme:** test_update_weight fonksiyonu, kullanıcı tarafından güncellenen ağırlıkların toplamının 100 olup olmadığını kontrol eder ve ardından bu değişikliğin öğrencilerin ortalamalarına nasıl yansıdığını test eder.
- **Geçersiz Ağırlık Girişi:** test_invalid_weight_input fonksiyonu, geçersiz bir ağırlık girişi yapıldığında uygulamanın uygun şekilde hata verdiğini test eder.
- **Excel'e Kaydetme:** test_save_excel fonksiyonu, öğrencilerin verilerini Excel dosyasına kaydedip kaydetmediğini kontrol eder ve kaydedilen dosyanın içeriğini doğrular.
- **Veri Yükleme:** test_load_data fonksiyonu, uygulamanın doğru şekilde verileri yükleyip yüklemediğini test eder.

3. Test Uygulamasının Tekrar Edilebilirliği

Yazılımdaki her bir bileşen test edildikten sonra, test uygulaması tekrar tekrar çalıştırılabilecek şekilde geliştirilmiştir. Testler, veri dosyalarını ve kullanıcı girişlerini taklit ederek aynı sonucu elde edebilir. Testlerin her bir fonksiyonun sonunda, test sırasında oluşturulan geçici dosyalar silinerek tekrar test edebilme imkanı sağlanır.

tearDown() fonksiyonu, her testten sonra geçici dosyaların temizlenmesi ve diğer temizlik işlemleri için kullanılabilir.

Sonuç

Bu test uygulaması, yazılımın çeşitli bileşenlerinin doğruluğunu kontrol etmek için geliştirilmiştir. Testler, yazılımın her bir fonksiyonunun doğru çalışıp çalışmadığını, veri bütünlüğünü, geçersiz girişlerin yönetimini ve dosya kaydetme işlevini kontrol eder. Testler tekrar edilebilir olup, yazılımın her aşamasında doğru sonuçların alındığından emin olunmasına olanak sağlar.

4.2 Yazılımın doğrulanması

1. Tam ve Doğru Çalışan Bileşenler

- **Ağırlıklı Ortalama Hesaplama (test_weighted_average_calculation):** Ağırlıklı ortalama hesaplama fonksiyonu başarılı bir şekilde çalıştı. Öğrencilerin ödev puanlarına göre hesaplanan ortalamalar doğru bir şekilde elde edildi. Bu bileşen, testlerde beklenen sonuçları verdi ve yazılımda doğru bir şekilde çalıştı.
- **Ağırlık Güncelleme (test_update_weight):** Ağırlık güncelleme fonksiyonu doğru şekilde çalıştı. Kullanıcı tarafından yapılan ağırlık değişiklikleri doğru bir şekilde yansıtıldı ve ağırlıkların toplamı her zaman 100 olarak kontrol edildi. Ayrıca, güncellenen ağırlıkların öğrencilerin ortalamalarını doğru şekilde değiştirdiği doğrulandı.
- **Veri Yükleme (test_load_data):** Excel dosyasından verilerin doğru bir şekilde yüklendiği ve DataFrame'in doğru biçimde oluşturulduğu doğrulandı. Yüklenen verilerin uygulama tarafından doğru şekilde işlendiği test edilerek başarılı bir sonuç elde edildi.
- **Excel'e Kaydetme (test_save_excel):** Excel dosyasına veri kaydetme işlemi doğru çalıştı. Kullanıcı verilerinin Excel dosyasına kaydedildiği ve kaydedilen dosyanın doğru biçimde açıldığı doğrulandı. Ayrıca, kaydedilen dosyanın içeriğiyle ilgili doğrulamalar başarılı oldu.

2. Eksik veya Hatalı Çalışan Bileşenler

- **Geçersiz Ağırlık Girişi (test_invalid_weight_input):** Geçersiz bir ağırlık girişi yapıldığında, hata mesajının doğru şekilde gösterilmesi bekleniyor. Ancak, ağırlıkların 101 gibi geçersiz bir değeri girmesi durumunda hata mesajı doğru şekilde gösterildi, ancak bu hatanın tam olarak kullanıcı dostu olmadığı ve girişi durdurmadığı gözlemlendi. Bu bileşen, geliştirilmesi gereken bir alandır.
    - **Çözüm Önerisi:** Ağırlık değerleri kullanıcıdan alınırken, geçerli bir değer girilene kadar girişlerin tekrar yapılmasını sağlayacak ek doğrulama ve hata yönetimi eklenebilir. Ayrıca, kullanıcıya ne zaman ve nasıl düzeltmesi gerektiği daha net bir şekilde bildirilebilir.

3. Test Sonuçları ve Doğrulama Süreci

Testlerin sonucunda elde edilen yazılım doğruluğu aşağıdaki gibi özetlenebilir:

- **Başarılı Çalışan Bileşenler:**
    - Ağırlıklı Ortalama Hesaplama
    - Ağırlık Güncelleme
    - Veri Yükleme
    - Excel'e Kaydetme
- **Eksik veya Hatalı Çalışan Bileşenler:**
    - Geçersiz Ağırlık Girişi Hata Yönetimi

Sonuç

Test uygulamaları ile yazılımın doğruluğu başarılı bir şekilde test edilmiştir. Yazılımın büyük kısmı doğru şekilde çalışmaktadır, ancak bazı durumlarda kullanıcıya sunulan hata yönetimi iyileştirilebilir. Özellikle geçersiz ağırlık girişi gibi durumlar için daha güçlü doğrulama ve hata mesajları gerekmektedir. Bu doğrulama işlemi, yazılımın kullanımını daha güvenilir hale getirerek ve kullanıcı deneyimini iyileştirecektir.