

**T.C**

**KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ**

**MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ**

**BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ**

**PROGRAM DERS İLİŞKİ MATRİSİ PROJESİ**

**HAZIRLAYANLAR:**

**Müge ÖZLER Yavuz Selim GÜRSOY 210501004 220501003**

**Emrah ŞAHİN Talha TUNA**

**220502025 220502015**

**DERS SORUMLUSU:**

**DR. ÖĞR. ÜYESİ**

**Elif Pınar HACIBEYOĞLU**

**TARİH:22.11.2024**

**1 GİRİŞ**

**1.1 Projenin Amacı**

Bu projede amaç, **ders çıktıları**, **değerlendirme kriterleri**, **öğrenci not tabloları** ve **program çıktıları** arasındaki ilişkiyi kullanarak **başarı hesaplamasını** otomatikleştiren ve kullanıcı dostu bir arayüz sağlamaktır. Projenin temel hedefleri şunlardır:

* Farklı kaynaklardan (Excel, manuel giriş vb.) girilebilen **ders verileri**, **öğrenci listeleri**, **öğrenme çıktıları** ve **değerlendirme kriterleri** bilgilerinin tek bir arayüz üzerinden yönetilebilmesi.
* Tüm dersler için girilen verilerin **kolay ve hatasız** şekilde işlenmesi; tablo 4 ve tablo 5 gibi çıktıları otomatik oluşturabilecek bir altyapı oluşturmak.
* Kullanıcıya, öğrenci notlarının girişinden değerlendirme kriterlerinin ağırlıklandırılmasına kadar tüm süreci **adım adım yöneten** sezgisel bir arayüz sunmak.
* Kullanıcının her ders için ayrı ayrı veriler girip, sonuçları elde ettikten sonra **yeni bir derse kolaylıkla geçebileceği** bir yapı kurmak.

Bu sayede, hem öğretim elemanlarının hem de eğitim yönetiminde bu verileri kullanacak diğer paydaşların **zamandan tasarruf etmesi** ve **standart, doğrulanabilir sonuçlar** elde etmesi sağlanacaktır.

**2 GEREKSİNİM ANALİZİ**

**2.1 Arayüz Gereksinimleri**

1. Kullanıcı Dostu Arayüz

* Ara yüz, basit ve anlaşılır menülerle oluşturulmalıdır.
* Girilen verilerin doğruluğunu kontrol edecek (örneğin [0, 1] aralığını denetleyen) hata mesajları veya uyarı mekanizmaları bulunmalıdır.
* Arayüzde yönlendirici butonlar (Kaydet, İptal, Düzenle, Geri vb.) net bir şekilde konumlandırılmalıdır.

1. Ders Ekleme ve Öğrenci Listesi Yönetimi

* Minimum 3 ders eklenebilmelidir.
* Ders kodu ve adı (Örn. “YZM 315 – Yazılım Lab I”) kullanıcı dostu menülerle listelenmelidir.
* Öğrenci listesi, sadece “Öğrenci Numarası” bilgisini içeren bir Excel dosyasından yüklenebilmelidir.

1. Program Çıktıları ve Dersin Öğrenme Çıktıları Yönetimi

* Program çıktıları ve dersin öğrenme çıktıları hem Excel’den yüklenebilmelidir hem de manuel olarak girilebilmelidir.
* Giriş sırasında değer aralıkları veya formatlar kontrol edilerek hata önlenmelidir.

1. Değerlendirme Kriterleri ve Ağırlıkları

* Minimum 5 değerlendirme kriteri (Ödev, Proje, Vize, Final, Katılım vb.) bulunmalıdır.
* Kriter seçildiğinde, o kritere ait ağırlık yüzdesi (örneğin Ödev %10) girilir.
* Girilen tüm ağırlıkların toplamının %100 olması sağlanmalı, aksi durumlarda uyarı verilmelidir.

1. Tablo Yükleme/İndirme Alanları

* Tablo 1 ve Tablo 2 (ikisi de [0,1] aralığında değerler içerir) Excel’den yüklenebilir veya manuel girilebilir, aralığı aşan değerlerde uyarı vermelidir.
* Öğrenci notlarını içeren tablo (Not Tablosu), Excel’den yüklenebilir veya manuel girilebilir.
* Tüm tablolarda yapılan değişiklikler “Kaydet” butonu ile onaylanmalıdır.
* Tablo 3, sistemde arka planda hesaplanmalı, arayüzde gösterilmesi zorunlu değildir.

1. Çıktı Oluşturma

* Tüm öğrenciler için tek bir Excel dosyasında Tablo 4 çıktısı oluşturulmalıdır.
* Tüm öğrenciler için tek bir Excel dosyasında Tablo 5 çıktısı oluşturulmalıdır.

1. Reset / Ders Seçim Sayfasına Dön

* Kullanıcı bir ders için tüm verileri girip çıktılarını oluşturduktan sonra, kolayca yeni ders seçimine geçebilmelidir.
  1. **Fonksiyonel Gereksinimler**

1. Ders Ekleme Fonksiyonu

* Kullanıcı, “Ders Kodu” ve “Ders Adı” bilgilerini manuel girerek yeni ders oluşturabilir.
* Sistem, en az 3 farklı dersin eklenmesine izin vermelidir.

1. Öğrenci Listesi Yönetimi Fonksiyonu

* Excel dosyası formatında öğrenci listesi yüklenebilmelidir.
* Öğrenci listesi sadece Öğrenci Numarasını içerebilir.
* Sistem, yüklenen verilerdeki hatalı formatları kontrol ederek uyarı vermelidir.

1. Çıktı Yönetimi (Program Çıktıları ve Ders Öğrenme Çıktıları)

* Program çıktıları ve öğrenme çıktıları Excel’den veya manuel ekrandan yüklenir.
* [0,1] aralığında veya “metin” girişi gibi farklı tipte veri girişlerinde uygun kontroller yapılır.

1. Değerlendirme Kriterleri Yönetimi

* Kullanıcı, sistemde önceden tanımlı kriterlerden (Ödev, Proje, Sunum vb.) seçim yapar ve ilgili ağırlığı (örneğin %10) tanımlar.
* Tüm kriterlerin toplamı %100 olmadığında uyarı verir ve kaydetme işlemine izin vermez.
* Mevcut kriter ağırlıkları düzenlenebilir.

1. Not Girişi ve Hesaplama

* Öğrencilerin notları Excel dosyasından yüklenebilmeli veya manuel girilebilmelidir.
* Kaydet butonuna basıldıktan sonra sistem, verileri güvenli bir biçimde saklayabilmeli ve hesaplama işlemlerini (Tablo 3 ve devamındaki tablolar) arka planda gerçekleştirebilmelidir.

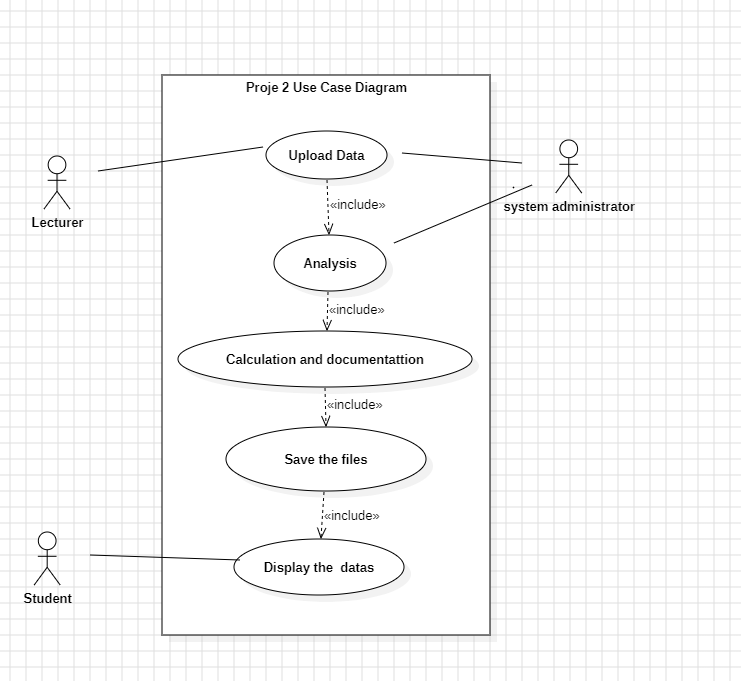
1. Oluşturulacak Tabloların İndirilmesi

* Tüm öğrenciler için tek seferde Tablo 4 ve Tablo 5 Excel dosyası oluşturulabilmeli ve indirilebilmelidir.

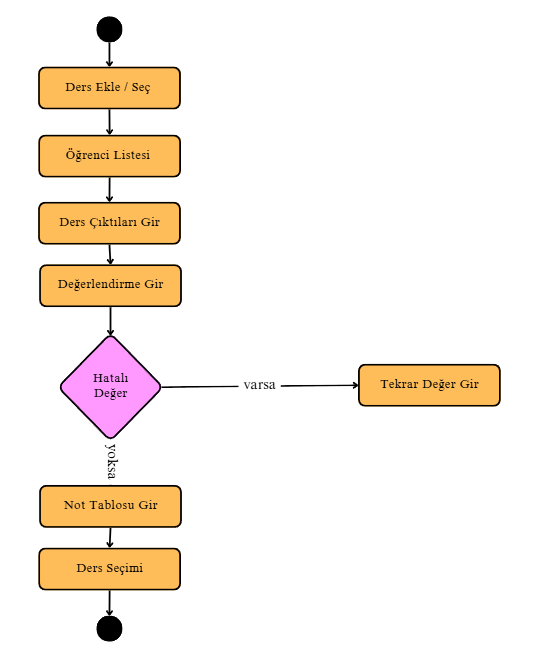
1. Sistemi Sıfırlama

* Kullanıcı yeni bir ders için tekrar aynı adımları takip etmek istediğinde, “Ders Seçimi Sayfası”na geri dönebilmeli, önceki işlemler sıfırlanabilmelidir**.**

**2.3 Use-Case Diyagramı**



* 1. **Aktivite (Akış) Diyagramı**



**3 TASARIM**

**3.1 Mimari Tasarım**

Sistem, üç ana katmandan oluşacak şekilde tasarlanacaktır:

* **Veri Katmanı:**
  + Excel dosyalarından gelen verilerin okunması, doğrulanması ve analiz sonuçlarının kaydedilmesini yönetir.
  + Kullanılacak Teknolojiler: Pandas, OpenPyXL.
* **Uygulama Katmanı:**
  + Veri işleme, hesaplama ve raporlama işlemlerinin gerçekleştirildiği katmandır.
  + Kullanılacak Teknolojiler: Python, Flask/Django (API için).
* **Sunum Katmanı:**
  + Kullanıcı arayüzü, grafiksel raporlar ve veri yükleme/sunma işlemlerinin gerçekleştirildiği katmandır.
  + Kullanılacak Teknolojiler: HTML, CSS, JavaScript (React veya Vue.js)

**3.2 Kullanılacak Teknolojiler**

1. **Programlama Dili:** Python (veri işleme ve analiz için).
2. **Veritabanı:** SQLite veya PostgreSQL (eğer sistem, kalıcı veritabanı desteği gerektirirse).
3. **Kütüphaneler:**
   * Veri işleme: Pandas, NumPy
   * Dosya işlemleri: OpenPyXL, PyPDF2
   * Grafik ve Raporlama: Matplotlib, Seaborn, Plotly
4. **Frontend Teknolojileri:**
   * HTML, CSS (Bootstrap ile)
   * JavaScript Frameworkü: React veya Vue.js
5. **Backend Teknolojileri:** Flask veya Django (API ve veri işleme için).
6. **Test Araçları:**
   * Unit Test: Pytest
   * Entegrasyon Testleri: Postman
7. **Versiyon Kontrolü:** Git ve GitHub.

**4 UYGULAMA**

**4.1 Kodlanan Bileşenlerin Açıklamaları**

## 4.1.1 Kütüphaneler

from PyQt5 import QtWidgets, QtGui: PyQt5, Python için bir GUI (Grafiksel Kullanıcı Arayüzü) kütüphanesidir. Bu kod, Qt'nin widget'larını (örneğin, butonlar, metin kutuları) ve grafik öğelerini (ikonlar, yazı tipleri) içeren modüllerini içe aktarır.

from pathlib import Path: pathlib, Python'un standart kütüphanesinde bulunan bir modüldür ve dosya/dizin yollarını nesne olarak temsil etmeye ve bu yollarla çalışmaya olanak tanır. Path, dosya/dizin işlemlerini yapmak için kullanılır.

from typing import TypeVar: Python'un typing modülü, statik tür ipuçları sağlamak için kullanılır. TypeVar, jenerik tipler tanımlamak için kullanılır. Örneğin, birden fazla yerde aynı tipte veri bekleniyorsa, bunu tanımlamak için TypeVar kullanılabilir.

from PyQt5.QtCore import pyqtSignal: pyqtSignal, PyQt'de özel sinyaller oluşturmak için kullanılır. Sinyaller, bir olay meydana geldiğinde diğer bileşenleri bilgilendirmek için kullanılır (örneğin, bir düğmeye tıklandığında bir işlem başlatmak gibi).

from src.setget import set\_percentage, set\_column\_name, get\_percentage, get\_column\_names: Bu satır, bir src dizini içindeki bir modülden belirli işlevleri içe aktarır. Bu işlevler, yüzde değerlerini veya sütun adlarını ayarlamak/almak gibi işlemler için özel olarak tanımlanmış olabilir.

import subprocess: subprocess modülü, Python'dan harici komutlar veya programlar çalıştırmak için kullanılır. Örneğin, bir komut satırı komutunu Python üzerinden çalıştırabilirsiniz.

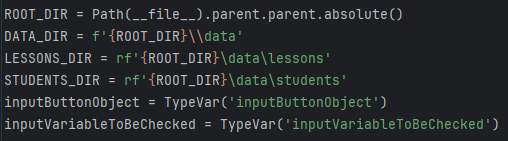
import win32com.client as win32: pywin32 kütüphanesi, Windows üzerinde COM (Component Object Model) arayüzleri ile çalışmayı sağlar. Bu modül, özellikle Excel veya Word gibi Microsoft Office uygulamalarıyla etkileşim için sıkça kullanılır.

import os: os modülü, işletim sistemiyle etkileşim kurmak için kullanılır. Dosya ve dizin işlemleri (oluşturma, silme, kopyalama vb.) için temel işlevler sunar.

import shutil: shutil, dosya ve dizinleri kopyalamak, taşımak ve silmek gibi yüksek düzeyde dosya işlemleri gerçekleştirmek için kullanılır.

import sys: sys modülü, Python'un çalışma zamanı ortamı ile etkileşim kurmaya olanak tanır. Komut satırı argümanlarını almak veya programdan çıkmak gibi işlemler yapılabilir.

import runpy: runpy, başka bir Python betiğini çalıştırmak için kullanılır. Özellikle, bir Python dosyasını (.py) bir modül olarak çalıştırmak için yararlıdır.

ROOT\_DIR = Path(\_\_file\_\_).parent.parent.absolute(): Uygulamanın çalıştığı Python dosyasının kök dizinini (\_\_file\_\_), iki üst dizine çıkarak belirler ve bunun tam yolunu alır. Bu, projedeki diğer dizinlere göreceli yollar oluşturmak için bir temel sağlar.

DATA\_DIR = f'{ROOT\_DIR}\\data': Kök dizin altında data adlı bir alt dizinin yolunu tanımlar. Uygulamanın veri dosyalarını depolamak için kullanılabilir.

LESSONS\_DIR = rf'{ROOT\_DIR}\data\lessons': data dizini altında lessons adlı bir alt dizinin yolunu tanımlar. Bu dizin, dersle ilgili dosyaların saklanacağı bir depo olabilir.

STUDENTS\_DIR = rf'{ROOT\_DIR}\data\students': data dizini altında students adlı bir alt dizinin yolunu tanımlar. Bu dizin, öğrencilerle ilgili dosyaların saklanacağı bir depo olabilir.

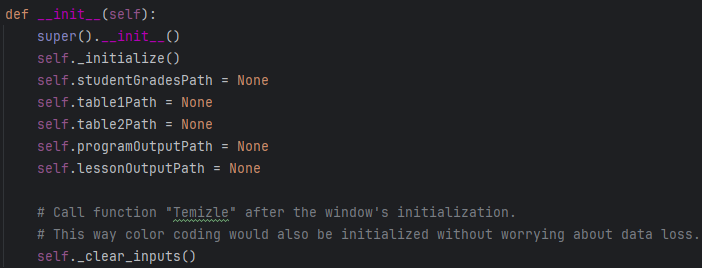
inputButtonObject = TypeVar('inputButtonObject'): Bu, inputButtonObject adında bir jenerik tür değişkeni tanımlar. Farklı yerlerde bir türü belirtmek ve tutarlılığı sağlamak için kullanılır. Örneğin, bu değişken bir buton nesnesinin türü olabilir.

inputVariableToBeChecked = TypeVar('inputVariableToBeChecked'): Bu, inputVariableToBeChecked adında bir jenerik tür değişkeni tanımlar. Kontrol edilecek girdilerle ilgili bir türü temsil edebilir. Farklı işlevlerde aynı türdeki girdilerle çalışmayı kolaylaştırır.

Bu kod, proje dizinlerini tanımlayarak dosya işlemlerini kolaylaştırır ve tip ipuçları ile kodun tutarlılığını artırır.

**4.1.2classCreateLessonWindow(QtWidgets.QWidget):**

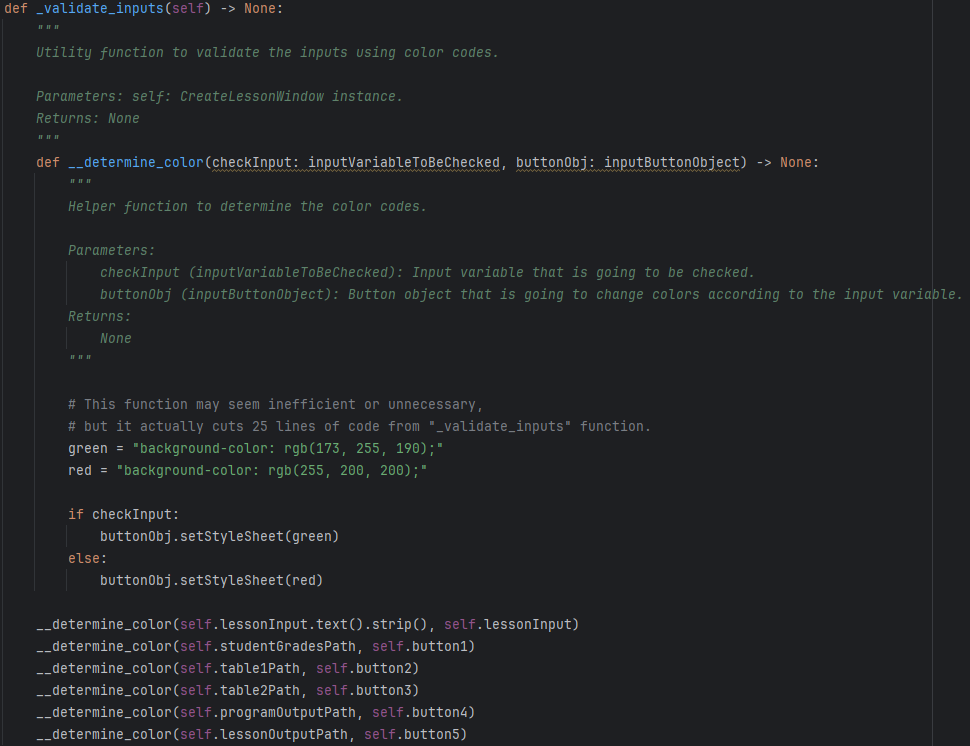
**def \_\_init\_\_(self):**



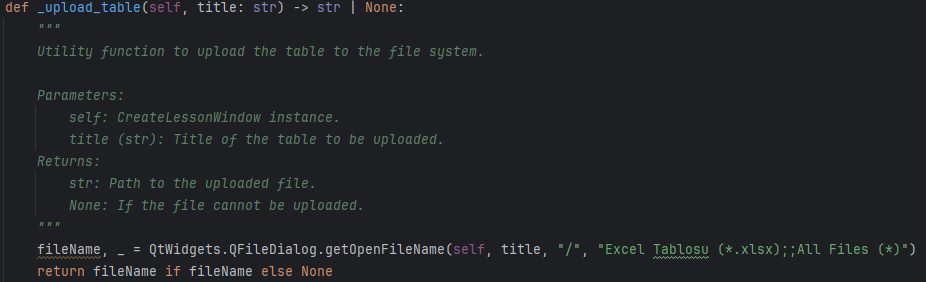
CreateLessonWindow sınıfı, bir ders oluşturma arayüzü için PyQt kullanılarak tanımlanmış bir sınıftır. lessonCreated = pyqtSignal() ifadesi, yeni bir ders oluşturulduğunda diğer bileşenleri bilgilendirmek için kullanılan özel bir PyQt sinyalini tanımlar. \_\_init\_\_ metodu sınıfın yapıcı fonksiyonu olup, super().\_\_init\_\_() ile üst sınıf olan QWidget'in yapıcı fonksiyonunu çağırarak pencerenin temel özelliklerini başlatır. Bu metot içinde self.\_initialize() çağrılarak pencerenin görünümü ve kullanıcı arayüzü bileşenleri ayarlanır. Kullanıcıdan alınacak veriler için self.studentGradesPath, self.table1Path, self.table2Path, self.programOutputPath ve self.lessonOutputPath değişkenleri tanımlanır ve başlangıçta None olarak ayarlanır. Bu değişkenler, ders oluşturma sırasında kullanılacak dosyaların yollarını tutmak için kullanılır. Son olarak, self.\_clear\_inputs() çağrılarak tüm girişler temizlenir ve başlangıç durumunda herhangi bir eski veri bulunmaması sağlanır. Bu işlem, aynı zamanda renk kodlamalarını ve varsayılan durumları sıfırlayarak pencerenin temiz bir başlangıç yapmasını sağlar.

**def \_initialize(self) -> None:**

CreateLessonWindow sınıfının \_initialize adlı bir metodudur ve ders oluşturma penceresinin kullanıcı arayüzünü oluşturur. İlk olarak, kullanıcıdan ders ismini almak için bir etiket (QLabel) ve bir metin kutusu (QLineEdit) ekler, metin kutusuna girilen veriler değiştikçe \_validate\_inputs metodu çağrılır. Ardından, kullanıcıdan grades.xlsx, table1.xlsx, table2.xlsx, program\_output.xlsx ve lesson\_output.xlsx dosyalarını yüklemek için her biri bir etiket (QLabel) ve bir butondan (QPushButton) oluşan beş farklı grup eklenir. Bu butonlara tıklanarak ilgili yükleme fonksiyonları çağrılır (upload\_student\_grades, upload\_table1, upload\_table2, upload\_program\_output, upload\_lesson\_output). Kullanıcı dersi oluşturmak için "Dersi Oluştur" butonuna basabilir, bu buton create\_lesson metodunu çağırır. "Temizle" butonu, tüm girişleri sıfırlayarak \_clear\_inputs metodunu çağırır. "Geri Dön" butonu ise ana menüye dönmek için go\_back\_to\_main\_menu metodunu çalıştırır. Tüm bu bileşenler dikey bir düzen (QVBoxLayout) içine yerleştirilir ve pencerenin genel düzeni oluşturulur. Pencere başlığı "Ders Oluşturma Aracı" olarak ayarlanır ve boyutu 300x400 piksel ile sabitlenir. Bu metot, ders oluşturma işlemi için gerekli olan kullanıcı arayüzünü başlatır ve düzenler.

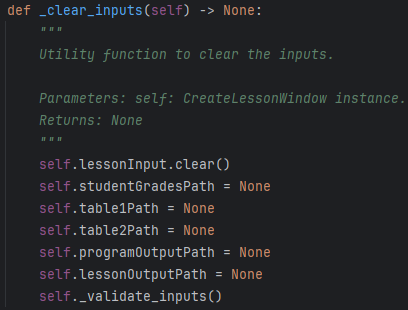
 **def \_validate\_inputs(self) -> None:**

Bu kod, CreateLessonWindow sınıfının \_validate\_inputs adlı bir metodudur ve kullanıcının yaptığı girişleri doğrulayarak bunlara uygun renk kodlaması uygular. Girişlerin doğrulanmasında bir yardımcı fonksiyon olan \_\_determine\_color kullanılır. Bu yardımcı fonksiyon, bir giriş değişkenini (checkInput) kontrol eder ve ilgili buton veya giriş nesnesine (buttonObj) yeşil veya kırmızı bir arka plan rengi uygular. Eğer giriş geçerli ise nesnenin arka plan rengi yeşil (rgb(173, 255, 190)), aksi takdirde kırmızı (rgb(255, 200, 200)) yapılır. \_validate\_inputs metodu, ders adı için metin girişini (self.lessonInput.text().strip()) ve ders oluşturma için gerekli olan grades.xlsx, table1.xlsx, table2.xlsx, program\_output.xlsx ve lesson\_output.xlsx dosyalarının yollarını (self.studentGradesPath, self.table1Path, self.table2Path, self.programOutputPath, self.lessonOutputPath) kontrol eder. Her biri için \_\_determine\_color fonksiyonu çağrılarak, kullanıcıya eksik veya hatalı girişleri görsel olarak göstermek için ilgili giriş bileşenlerinin renkleri değiştirilir. Bu metot, kullanıcı deneyimini iyileştirir ve eksik bilgilerin kolayca fark edilmesini sağlar.

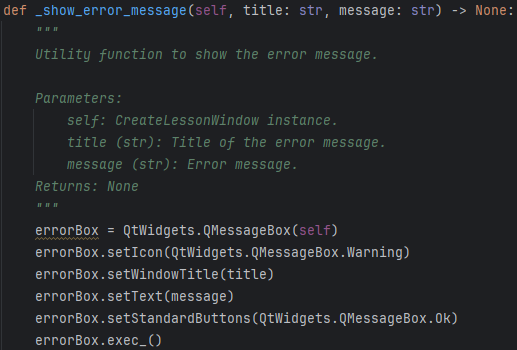
 **def \_upload\_table(self, title: str) -> str | None:**

Bu kod, CreateLessonWindow sınıfının \_upload\_table adlı bir yardımcı metodudur ve kullanıcıdan bir tablo dosyasını seçmesini sağlayarak seçilen dosyanın yolunu döndürür. Metot, title adlı bir parametre alır ve bu parametre, dosya seçme penceresinin başlığını belirler. Metot, QtWidgets.QFileDialog.getOpenFileName fonksiyonunu kullanarak bir dosya seçme penceresi açar. Bu pencere, varsayılan olarak "Excel Tablosu (\*.xlsx)" filtreleme seçeneği ile açılır, ancak tüm dosyalar da seçilebilir. Kullanıcı bir dosya seçip onayladığında, seçilen dosyanın tam yolu fileName değişkeninde saklanır. Eğer kullanıcı dosya seçmeden pencereyi kapatırsa, fileName boş olur. Metot, dosya seçilmişse fileName'i, seçilmemişse None döndürür. Bu metot, dosya yükleme işlemini kolaylaştırır ve uygulamanın farklı yerlerinde tekrar kullanılabilir.

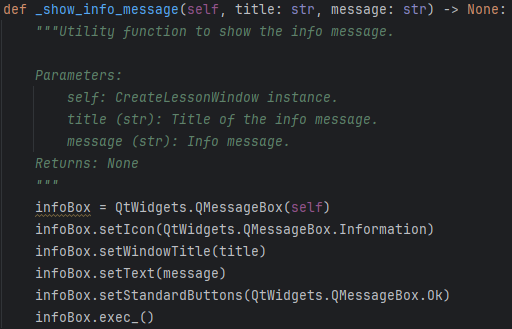
**def \_clear\_inputs(self) -> None:**

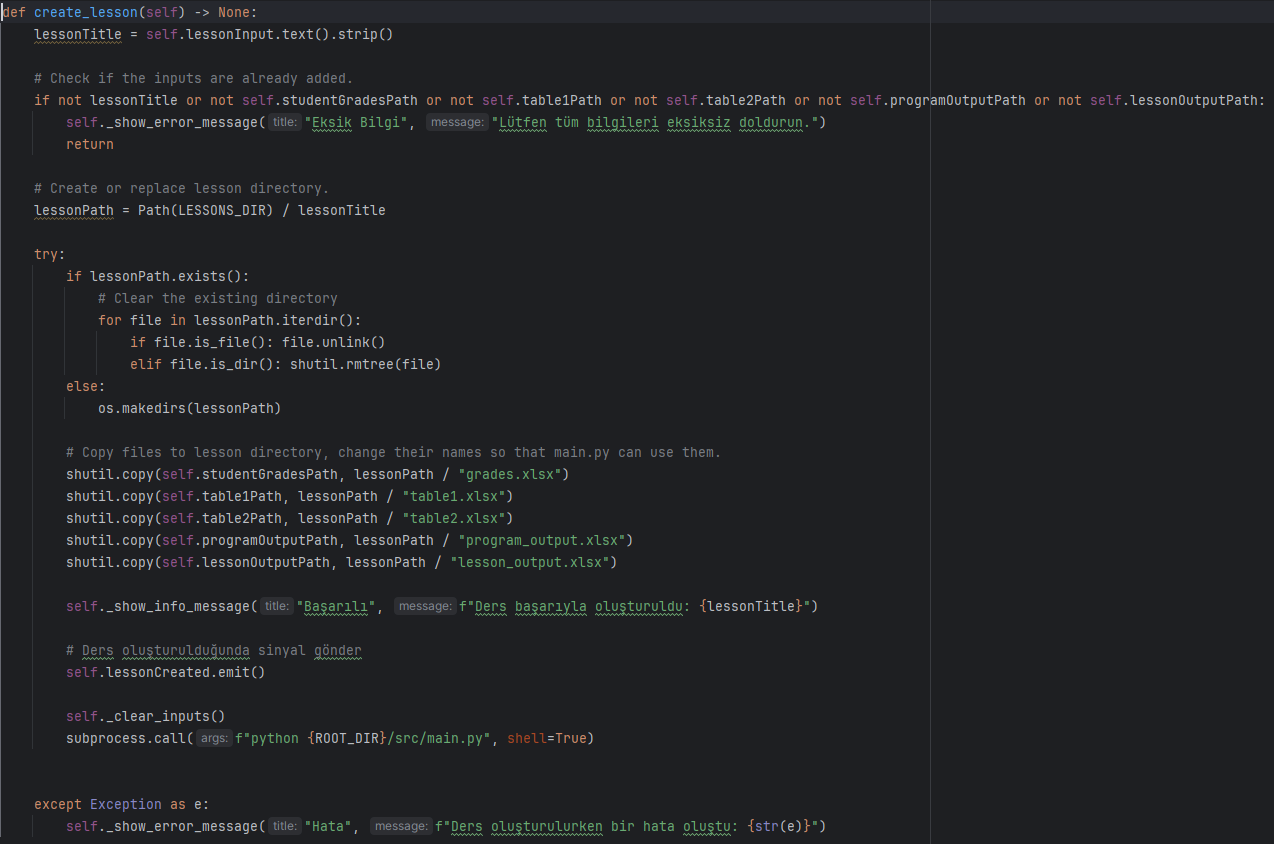
Bu kod, CreateLessonWindow sınıfının \_clear\_inputs adlı bir metodudur ve ders oluşturma penceresindeki tüm girişleri temizleyerek başlangıç durumuna döndürür. İlk olarak, self.lessonInput.clear() ifadesi ile ders ismini almak için kullanılan metin kutusu temizlenir. Ardından, dosya yollarını tutan self.studentGradesPath, self.table1Path, self.table2Path, self.programOutputPath ve self.lessonOutputPath değişkenleri None olarak sıfırlanır, böylece daha önce seçilen dosyalarla ilgili bilgiler silinir. Son olarak, \_validate\_inputs metodu çağrılarak tüm girişlerin durumları tekrar doğrulanır ve gerekli renk kodlamaları uygulanır. Bu metot, kullanıcı arayüzünü temiz bir hale getirir ve yeni bir ders oluşturma işlemi için hazırlar.

**def \_show\_error\_message(self, title: str, message: str) -> None:**

Bu kod, CreateLessonWindow sınıfının \_show\_error\_message adlı bir metodudur ve kullanıcıya hata mesajlarını göstermek için kullanılır. Metot, title ve message olmak üzere iki parametre alır. title, hata mesajının başlığını belirlerken, message hata içeriğini tanımlar. İlk olarak, bir hata mesajı kutusu oluşturmak için QtWidgets.QMessageBox sınıfı kullanılır. Mesaj kutusunun simgesi, bir uyarı ikonu (QtWidgets.QMessageBox.Warning) olarak ayarlanır. Daha sonra, mesaj kutusunun başlığı setWindowTitle(title) ile belirlenir ve hata içeriği setText(message) ile eklenir. Mesaj kutusuna bir "Tamam" düğmesi eklenir (setStandardButtons(QtWidgets.QMessageBox.Ok)) ve exec\_() metodu çağrılarak mesaj kutusu ekrana getirilir. Bu metot, kullanıcılara kolayca anlaşılabilir hata bildirimleri sağlamak için tasarlanmıştır.

**def \_show\_info\_message(self, title: str, message: str) -> None:**

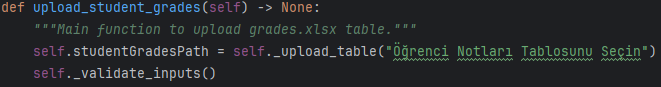
Bu kod, CreateLessonWindow sınıfının \_show\_info\_message adlı bir metodudur ve kullanıcıya bilgi mesajları göstermek için kullanılır. Metot, title ve message olmak üzere iki parametre alır. title, bilgi mesajının başlığını belirlerken, message bilgi içeriğini tanımlar. İlk olarak, bir bilgi mesajı kutusu oluşturmak için QtWidgets.QMessageBox sınıfı kullanılır. Mesaj kutusunun simgesi, bir bilgi ikonu (QtWidgets.QMessageBox.Information) olarak ayarlanır. Daha sonra, mesaj kutusunun başlığı setWindowTitle(title) ile belirlenir ve bilgi içeriği setText(message) ile eklenir. Mesaj kutusuna bir "Tamam" düğmesi eklenir (setStandardButtons(QtWidgets.QMessageBox.Ok)) ve exec\_() metodu çağrılarak mesaj kutusu ekrana getirilir. Bu metot, kullanıcılara başarı, tamamlanma veya genel bilgilendirme gibi mesajlar sağlamak için tasarlanmıştır.

**def create\_lesson(self) -> None:**

Bu kod, CreateLessonWindow sınıfının create\_lesson adlı ana fonksiyonudur ve bir ders oluşturma işlemini gerçekleştirir. İlk olarak, kullanıcı tarafından girilen ders adı lessonInput üzerinden alınır ve baştaki ve sondaki boşluklar kaldırılarak lessonTitle değişkenine atanır. Ardından, gerekli girişlerin (ders adı ve dosya yolları) tamamlanıp tamamlanmadığını kontrol eder. Eğer herhangi bir bilgi eksikse, \_show\_error\_message metodu çağrılarak kullanıcıya eksik bilgi mesajı gösterilir ve işlem sonlandırılır.

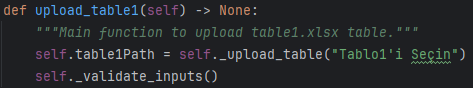
Ders oluşturma işlemi için, dersin saklanacağı dizin yolu lessonPath değişkenine atanır. Eğer bu dizin zaten mevcutsa, içeriği temizlenir; dosyalar silinir, alt dizinler varsa bunlar da kaldırılır. Eğer dizin mevcut değilse, os.makedirs ile yeni bir dizin oluşturulur. Kullanıcının seçtiği dosyalar (notlar, tablolar ve program çıktıları), bu dizine kopyalanır ve isimleri ana programın (main.py) bu dosyaları kullanabileceği şekilde değiştirilir (grades.xlsx, table1.xlsx, table2.xlsx, program\_output.xlsx, lesson\_output.xlsx).

Ders başarıyla oluşturulduğunda, \_show\_info\_message metodu ile kullanıcıya başarı mesajı gösterilir ve lessonCreated.emit() ile bir sinyal gönderilir, böylece diğer bileşenler bu olayı algılayabilir. Son olarak, \_clear\_inputs metodu çağrılarak girişler temizlenir ve subprocess.call ile ana program (main.py) çalıştırılır. Eğer herhangi bir hata oluşursa, hata mesajı \_show\_error\_message ile kullanıcıya iletilir. Bu metot, ders oluşturma işlemini baştan sona düzenleyen, kullanıcıya geri bildirim sağlayan ve hata durumlarını ele alan bir işlem akışını yönetir.

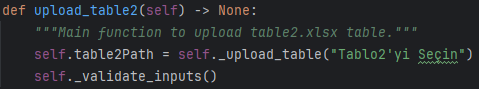
 **def upload\_student\_grades(self) -> None:**

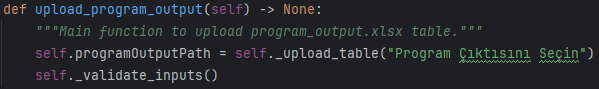
Bu kod, CreateLessonWindow sınıfının upload\_student\_grades adlı bir metodudur ve grades.xlsx dosyasını yükleme işlemini gerçekleştirir. Kullanıcıdan bir öğrenci notları tablosu seçmesini ister. İlk olarak, \_upload\_table metodunu çağırır ve pencere başlığını "Öğrenci Notları Tablosunu Seçin" olarak belirler. Bu metot, kullanıcı bir dosya seçtiğinde seçilen dosyanın tam yolunu self.studentGradesPath değişkenine atar. Daha sonra, \_validate\_inputs metodu çağrılarak girişlerin doğruluğu kontrol edilir ve eksiklik varsa görsel olarak belirtilir. Bu metot, kullanıcıdan doğru dosyanın yüklenmesini sağlar ve ders oluşturma sürecinin bir adımını tamamlar.

**def upload\_table1(self) -> None:**

Bu kod, CreateLessonWindow sınıfının upload\_table1 adlı bir metodudur ve table1.xlsx dosyasını yükleme işlemini gerçekleştirir. Kullanıcıdan birinci tabloyu seçmesini ister. İlk olarak, \_upload\_table metodunu çağırır ve dosya seçme penceresinin başlığını "Tablo1'i Seçin" olarak belirler. Kullanıcı bir dosya seçtiğinde, seçilen dosyanın tam yolu self.table1Path değişkenine atanır. Daha sonra \_validate\_inputs metodu çağrılarak girişlerin doğruluğu kontrol edilir ve eksiklik varsa görsel olarak belirtilir. Bu metot, kullanıcıdan doğru birinci tablonun yüklenmesini sağlar ve ders oluşturma sürecinin bir adımını tamamlar.

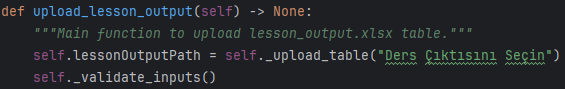
**def upload\_table2(self) -> None:**

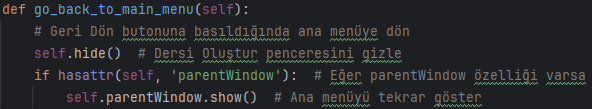
Bu kod, CreateLessonWindow sınıfının upload\_table2 adlı bir metodudur ve table2.xlsx dosyasını yükleme işlemini gerçekleştirir. Kullanıcıdan ikinci tabloyu seçmesini ister. İlk olarak, \_upload\_table metodunu çağırır ve dosya seçme penceresinin başlığını "Tablo2'yi Seçin" olarak belirler. Kullanıcı bir dosya seçtiğinde, seçilen dosyanın tam yolu self.table2Path değişkenine atanır. Daha sonra \_validate\_inputs metodu çağrılarak girişlerin doğruluğu kontrol edilir ve eksiklik varsa görsel olarak belirtilir. Bu metot, kullanıcıdan doğru ikinci tablonun yüklenmesini sağlar ve ders oluşturma sürecinin bir adımını tamamlar.

 **def upload\_program\_output(self) -> None:**

Bu kod, CreateLessonWindow sınıfının upload\_program\_output adlı bir metodudur ve program\_output.xlsx dosyasını yükleme işlemini gerçekleştirir. Kullanıcıdan program çıktısı tablosunu seçmesini ister. İlk olarak, \_upload\_table metodunu çağırır ve dosya seçme penceresinin başlığını "Program Çıktısını Seçin" olarak belirler. Kullanıcı bir dosya seçtiğinde, seçilen dosyanın tam yolu self.programOutputPath değişkenine atanır. Daha sonra \_validate\_inputs metodu çağrılarak girişlerin doğruluğu kontrol edilir ve eksiklik varsa görsel olarak belirtilir. Bu metot, kullanıcıdan doğru program çıktısı dosyasının yüklenmesini sağlar ve ders oluşturma sürecinin bir adımını tamamlar.

**def upload\_lesson\_output(self) -> None:**

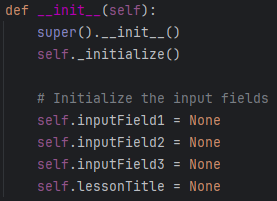
Bu kod, CreateLessonWindow sınıfının upload\_lesson\_output adlı bir metodudur ve lesson\_output.xlsx dosyasını yükleme işlemini gerçekleştirir. Kullanıcıdan ders çıktısı tablosunu seçmesini ister. İlk olarak, \_upload\_table metodunu çağırır ve dosya seçme penceresinin başlığını "Ders Çıktısını Seçin" olarak belirler. Kullanıcı bir dosya seçtiğinde, seçilen dosyanın tam yolu self.lessonOutputPath değişkenine atanır. Daha sonra \_validate\_inputs metodu çağrılarak girişlerin doğruluğu kontrol edilir ve eksiklik varsa görsel olarak belirtilir. Bu metot, kullanıcıdan doğru ders çıktısı dosyasının yüklenmesini sağlar ve ders oluşturma sürecinin bir adımını tamamlar.

 **def go\_back\_to\_main\_menu(self):**

Bu kod, CreateLessonWindow sınıfının go\_back\_to\_main\_menu adlı bir metodudur ve kullanıcı "Geri Dön" butonuna bastığında ana menüye dönmeyi sağlar. İlk olarak, self.hide() ile mevcut "Ders Oluştur" penceresi gizlenir. Ardından, hasattr(self, 'parentWindow') kontrolü ile bu pencerenin parentWindow adında bir üst pencere referansı olup olmadığı doğrulanır. Eğer bu referans varsa, self.parentWindow.show() çağrılarak ana menü penceresi tekrar görünür hale getirilir. Bu metot, kullanıcı deneyimini kolaylaştırmak için pencereler arasında geçiş yapılmasını sağlar.

**4.1.3** **class EditLessonWindow(QtWidgets.QWidget):**

**def \_\_init\_\_(self):**



EditLessonWindow adında bir sınıf tanımlar ve bu sınıf, "Dersi Düzenle" sekmesi için bir kullanıcı arayüzü sağlar. QtWidgets.QWidget sınıfından türetilen bu sınıf, ders düzenleme işlemlerini gerçekleştirmek üzere tasarlanmıştır.

Sınıfın açıklamasında, temel nitelikleri ve işlevleri tanımlanmıştır:

* self.lessonTitle: Seçilen dersin başlığını tutar.
* self.inputField1: İlk metin giriş alanını temsil eder.
* self.inputField2: İkinci metin giriş alanını temsil eder.
* self.inputField3: Üçüncü metin giriş alanını temsil eder.

Sınıfın yardımcı işlevleri şunlardır:

* \_initialize: Sınıfın kullanıcı arayüzünü ve diğer temel özelliklerini başlatır.
* \_clear\_inputs: Tüm giriş alanlarını temizler ve başlangıç durumuna getirir.
* \_show\_error\_message: Kullanıcıya hata mesajı gösterir.
* \_show\_info\_message: Kullanıcıya bilgi mesajı gösterir.

Ana işlevleri ise şunlardır:

* load\_lesson: Seçilen dersi yükler ve verilerini ilgili alanlara doldurur.
* save\_changes: Kullanıcının yaptığı değişiklikleri kaydeder.
* go\_back\_to\_main\_menu: "Geri Dön" işlemini gerçekleştirerek ana menüye döner.

\_\_init\_\_ metodu, sınıfın yapıcı fonksiyonudur ve üst sınıfın yapıcısını çağırarak (super().\_\_init\_\_()) temel özellikleri başlatır. Daha sonra \_initialize fonksiyonunu çağırarak kullanıcı arayüzü bileşenlerini ve gerekli ayarları yapılandırır. Sınıfın inputField1, inputField2, inputField3 ve lessonTitle nitelikleri başlangıçta None olarak ayarlanır ve daha sonra gerekli verilerle doldurulmak üzere hazır bırakılır. Bu sınıf, kullanıcıların mevcut dersleri düzenlemesini sağlamak için tasarlanmıştır.

**def \_initialize(self) -> None**:

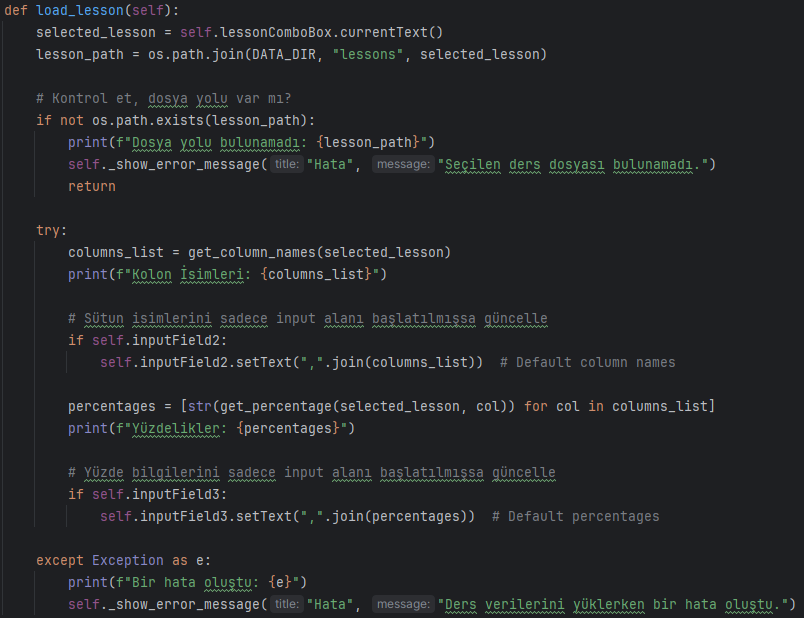
Bu kod, EditLessonWindow sınıfının \_initialize adlı bir metodudur ve "Dersi Düzenle" penceresinin kullanıcı arayüzünü oluşturur ve düzenler. İlk olarak, kullanıcının düzenlemek istediği dersi seçmesi için bir etiket (QLabel) ve bir açılır menü (QComboBox) eklenir. Açılır menüye, LESSONS\_DIR dizininde bulunan mevcut derslerin listesi yüklenir ve bir ders seçildiğinde load\_lesson metodu çağrılır.

Daha sonra, düzenleme yapılacak dersin özelliklerini değiştirmek için üç giriş alanı eklenir: Yeni Ders Adı için bir metin kutusu (QLineEdit), Yeni Kolon İsimleri için bir metin kutusu ve Yeni Yüzdelikler için bir metin kutusu. Bu giriş alanları kullanıcı tarafından doldurulmak üzere hazırlanır.

Kullanıcının yaptığı değişiklikleri kaydetmesi, giriş alanlarını temizlemesi veya ana menüye geri dönmesi için sırasıyla "Kaydet", "Temizle" ve "Geri Dön" adında üç buton eklenir. "Kaydet" butonu save\_changes metodunu, "Temizle" butonu clear\_inputs metodunu ve "Geri Dön" butonu go\_back\_to\_main\_menu metodunu çalıştırır.

Tüm bu bileşenler dikey bir düzen (QVBoxLayout) içine yerleştirilir ve pencere düzeni oluşturulur. Pencerenin başlığı "Ders Düzenle" olarak ayarlanır ve boyutu 300x350 piksel ile sabitlenir. Bu metot, kullanıcıların bir dersi seçerek özelliklerini düzenleyebileceği basit ve düzenli bir kullanıcı arayüzü oluşturur.

**def load\_lesson(self):**

Bu kod, EditLessonWindow sınıfının load\_lesson adlı bir metodudur ve kullanıcının seçtiği dersin bilgilerini yükleyerek düzenleme alanlarına doldurur. İlk olarak, kullanıcı tarafından seçilen ders adı lessonComboBox üzerinden alınır ve lesson\_path değişkenine dersin dosya yolu atanır. Daha sonra, os.path.exists ile bu dosya yolunun varlığı kontrol edilir. Eğer dosya yolu bulunamazsa, bir hata mesajı (\_show\_error\_message) gösterilir ve işlem sonlandırılır.

Dosya yolu mevcutsa, get\_column\_names fonksiyonu çağrılarak seçilen dersin sütun adları alınır ve columns\_list değişkenine atanır. Bu sütun adları konsola yazdırılır ve inputField2 giriş alanına virgülle ayrılmış şekilde yerleştirilir (eğer inputField2 giriş alanı başlatılmışsa). Daha sonra, her sütun için get\_percentage fonksiyonu çağrılarak sütunlara ait yüzde değerleri alınır ve percentages listesine atanır. Bu yüzde değerleri de konsola yazdırılır ve inputField3 giriş alanına virgülle ayrılmış şekilde yerleştirilir (eğer inputField3 giriş alanı başlatılmışsa).

Eğer işlem sırasında bir hata oluşursa, hata mesajı konsola yazdırılır ve \_show\_error\_message metodu çağrılarak kullanıcıya bir hata mesajı gösterilir. Bu metot, seçilen dersin bilgilerini düzenleme penceresine yüklemek için kullanılır ve giriş alanlarını dersin mevcut bilgileriyle günceller.

**def save\_changes(self) -> None:**

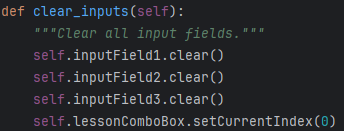
Bu kod, EditLessonWindow sınıfının save\_changes adlı bir metodudur ve kullanıcı tarafından yapılan değişiklikleri kaydederek mevcut dersin bilgilerini günceller. İlk olarak, lessonComboBox üzerinden seçilen ders adı alınır ve selected\_lesson değişkenine atanır.

Metot üç temel değişiklik türünü ele alır:

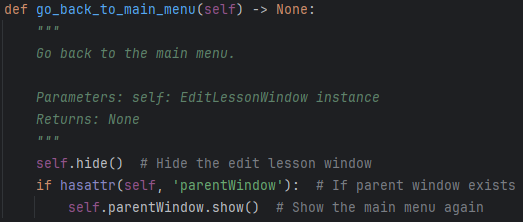
1. Ders Adı Değişikliği: Kullanıcı yeni bir ders adı girdiyse (inputField1.text()), bu adın mevcut dersler arasında olup olmadığı kontrol edilir. Eğer zaten var olan bir ad seçilmişse, \_show\_error\_message ile kullanıcıya bir hata mesajı gösterilir. Yeni ad mevcut değilse, dersin dizin adı os.rename kullanılarak güncellenir ve işlem hakkında bir bilgi mesajı konsola yazdırılır.
2. Sütun İsimleri Değişikliği: Kullanıcı yeni sütun isimleri girdiyse (inputField2.text()), bu isimler virgülle ayrılmış bir listeye dönüştürülür. Ardından, mevcut sütun isimlerinin sayısı ile yeni sütun isimlerinin sayısı karşılaştırılır. Eğer sayılar eşleşmiyorsa, kullanıcıya bir hata mesajı gösterilir. Eğer eşleşiyorsa, set\_column\_name fonksiyonu çağrılarak her sütunun adı güncellenir ve işlem hakkında bir bilgi mesajı konsola yazdırılır.
3. Yüzde Değerleri Değişikliği: Kullanıcı yeni yüzde değerleri girdiyse (inputField3.text()), bu değerler virgülle ayrılmış bir listeye dönüştürülüp tamsayıya çevrilir. Ardından, mevcut sütunların sayısı ile yeni yüzde değerlerinin sayısı karşılaştırılır. Eğer sayılar eşleşmiyorsa, kullanıcıya bir hata mesajı gösterilir. Eğer eşleşiyorsa, set\_percentage fonksiyonu çağrılarak her sütunun yüzde değeri güncellenir ve işlem hakkında bir bilgi mesajı konsola yazdırılır.

Eğer herhangi bir işlem sırasında bir hata oluşursa, except bloğu çalışır ve hata konsola yazdırılarak kullanıcıya \_show\_error\_message ile bir hata mesajı gösterilir. Başarılı bir şekilde değişiklikler kaydedildiğinde \_show\_success\_message metodu çağrılır ve kullanıcıya bir başarı mesajı gösterilir. Bu metot, ders düzenleme işlemlerini yönetir ve kullanıcıdan gelen değişiklikleri doğru bir şekilde uygulayarak kaydeder.

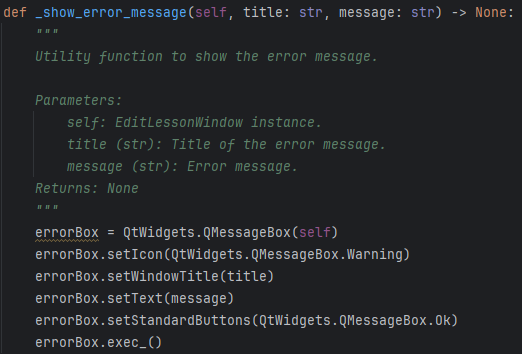
**def clear\_inputs(self):**

Bu kod, EditLessonWindow sınıfının clear\_inputs adlı bir metodudur ve "Dersi Düzenle" penceresindeki tüm giriş alanlarını temizleyerek başlangıç durumuna döndürür. İlk olarak, self.inputField1.clear(), self.inputField2.clear() ve self.inputField3.clear() ifadeleriyle metin giriş alanlarının içeriği silinir. Ardından, self.lessonComboBox.setCurrentIndex(0) ile ders seçme açılır menüsü ilk seçeneğe (genellikle varsayılan veya ilk ders) ayarlanır. Bu metot, kullanıcıya giriş alanlarını hızlıca sıfırlama imkanı sağlayarak yeni bir işlem için pencerede temiz bir başlangıç yapılmasını sağlar.

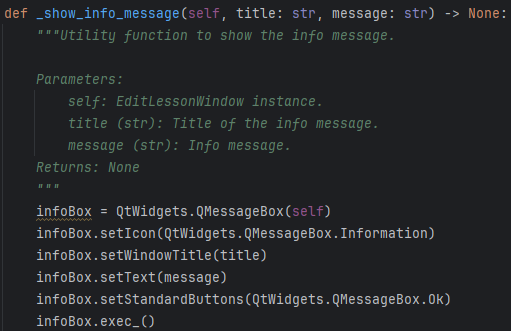
**def go\_back\_to\_main\_menu(self) -> None:**

Bu kod, EditLessonWindow sınıfının go\_back\_to\_main\_menu adlı bir metodudur ve kullanıcı "Geri Dön" işlemi yapmak istediğinde ana menüye dönüşü sağlar. İlk olarak, self.hide() ile mevcut "Dersi Düzenle" penceresi gizlenir. Daha sonra, hasattr(self, 'parentWindow') ile bu pencerenin bir üst pencereye (parentWindow) referansı olup olmadığı kontrol edilir. Eğer bu referans mevcutsa, self.parentWindow.show() ile ana menü penceresi tekrar görünür hale getirilir. Bu metot, kullanıcıların düzenleme penceresinden çıkıp ana menüye kolayca dönebilmelerini sağlar ve pencere geçişlerini yönetir.

**def \_show\_error\_message(self, title: str, message: str) -> None:**

Bu kod, EditLessonWindow sınıfının \_show\_error\_message adlı bir metodudur ve kullanıcıya hata mesajlarını göstermek için kullanılır. Metot, iki parametre alır: title, hata mesajının başlığını belirler, ve message, hata içeriğini tanımlar. İlk olarak, bir hata mesajı kutusu oluşturmak için QtWidgets.QMessageBox sınıfı kullanılır. Mesaj kutusunun simgesi bir uyarı ikonu (QtWidgets.QMessageBox.Warning) olarak ayarlanır. Daha sonra, mesaj kutusunun başlığı setWindowTitle(title) ile belirlenir ve hata içeriği setText(message) ile eklenir. Kullanıcıya bir "Tamam" düğmesi sunulur (setStandardButtons(QtWidgets.QMessageBox.Ok)) ve exec\_() metodu çağrılarak mesaj kutusu ekrana getirilir. Bu metot, uygulamada bir hata meydana geldiğinde kullanıcıyı bilgilendirmek ve sorunun ne olduğunu anlamalarına yardımcı olmak için tasarlanmıştır.

**def \_show\_info\_message(self, title: str, message: str) -> None:**

Bu kod, EditLessonWindow sınıfının \_show\_info\_message adlı bir metodudur ve kullanıcıya bilgi mesajları göstermek için kullanılır. Metot, iki parametre alır: title, bilgi mesajının başlığını belirler, ve message, bilgi içeriğini tanımlar. İlk olarak, bir bilgi mesajı kutusu oluşturmak için QtWidgets.QMessageBox sınıfı kullanılır. Mesaj kutusunun simgesi bir bilgi ikonu (QtWidgets.QMessageBox.Information) olarak ayarlanır. Daha sonra, mesaj kutusunun başlığı setWindowTitle(title) ile belirlenir ve bilgi içeriği setText(message) ile eklenir. Kullanıcıya bir "Tamam" düğmesi sunulur (setStandardButtons(QtWidgets.QMessageBox.Ok)) ve exec\_() metodu çağrılarak mesaj kutusu ekrana getirilir. Bu metot, kullanıcıya başarı, bilgilendirme veya işlem tamamlandığında olumlu geri bildirim vermek için tasarlanmıştır.

**4.1.4** **class MainWindow(QtWidgets.QMainWindow):**

**def \_\_init\_\_(self):**

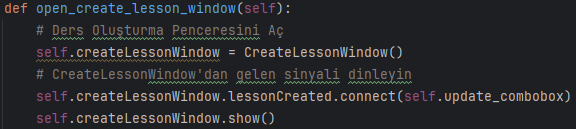
Bu fonksiyon, MainWindow adlı bir sınıf tanımlar ve bu sınıf, uygulamanın ana menüsünü oluşturur. PyQt'nin QMainWindow sınıfını miras alarak ana pencere özelliklerini sağlar. İlk olarak, pencerenin başlığı setWindowTitle("Ana Menü") ile "Ana Menü" olarak ayarlanır ve boyutu setFixedSize(300, 350) ile sabitlenir.

Menü çubuğu oluşturularak (menuBar = self.menuBar()), "Ders İşlemleri" adında bir menü eklenir. Bu menüye, "Ders Oluştur" ve "Dersi Düzenle" adlı iki seçenek eklenir. "Ders Oluştur" seçeneğine tıklandığında open\_create\_lesson\_window, "Dersi Düzenle" seçeneğine tıklandığında open\_edit\_lesson\_window metodu çağrılır.

Pencerenin merkezi bir widget'ı tanımlanır ve bir dikey düzen (QVBoxLayout) eklenir. Kullanıcıdan bir ders seçmesini istemek için "Dersi seçin:" yazılı bir etiket (QLabel) ve bir açılır menü (QComboBox) eklenir. Açılır menüye LESSONS\_DIR dizinindeki mevcut dersler yüklenir. Kullanıcı bir ders seçtiğinde, on\_combobox\_selection metodu çağrılır.

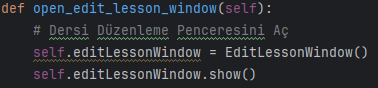
Kullanıcının belirli tabloları açmasına olanak tanıyan bir dizi buton eklenir. Bu butonlar arasında "Program Çıktı Tablosu", "Ders Çıktı Tablosu", "Öğrenci Notları Tablosu" ve tablo1'den tablo5'e kadar uzanan tablolar yer alır. Her bir buton, tıklandığında ilgili tabloyu açan open\_table metodunu çağırır. Kullanıcı dostu bir düzen sağlamak için butonlardan önce bir boşluk bırakılır ve "Açmak istediğiniz tabloyu seçin:" yazılı bir etiket eklenir.

Son olarak, tüm bu bileşenler merkezi widget'a eklenir ve ana pencerenin düzeni tamamlanır. Bu sınıf, kullanıcıların ders ve tablo seçimlerini yapabilecekleri, ders oluşturma ve düzenleme seçeneklerine kolayca erişebilecekleri bir ana menü oluşturur.

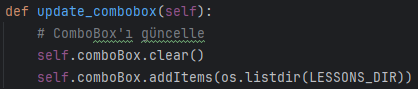
**def open\_create\_lesson\_window(self):**

Bu kod, MainWindow sınıfının open\_create\_lesson\_window adlı bir metodudur ve "Ders Oluştur" penceresini açar. İlk olarak, CreateLessonWindow sınıfından bir pencere nesnesi oluşturulur ve bu nesne self.createLessonWindow değişkenine atanır. Daha sonra, CreateLessonWindow sınıfında tanımlı lessonCreated sinyali dinlenir ve bu sinyal tetiklendiğinde update\_combobox metodu çağrılır. Bu, yeni bir ders oluşturulduğunda açılır menünün (QComboBox) güncellenmesini sağlar. Son olarak, self.createLessonWindow.show() ifadesi ile "Ders Oluştur" penceresi ekrana getirilir. Bu metot, kullanıcıların yeni bir ders oluşturma arayüzüne erişmelerini sağlar ve yeni ders eklendiğinde ana menüdeki ders listesinin otomatik olarak güncellenmesini sağlar.

**def open\_edit\_lesson\_window(self):**

Bu kod, MainWindow sınıfının open\_edit\_lesson\_window adlı bir metodudur ve "Dersi Düzenle" penceresini açar. İlk olarak, EditLessonWindow sınıfından bir pencere nesnesi oluşturulur ve bu nesne self.editLessonWindow değişkenine atanır. Daha sonra, self.editLessonWindow.show() ifadesi ile "Dersi Düzenle" penceresi ekrana getirilir. Bu metot, kullanıcıların mevcut dersleri düzenlemek için özel bir arayüze erişim sağlamasına olanak tanır.

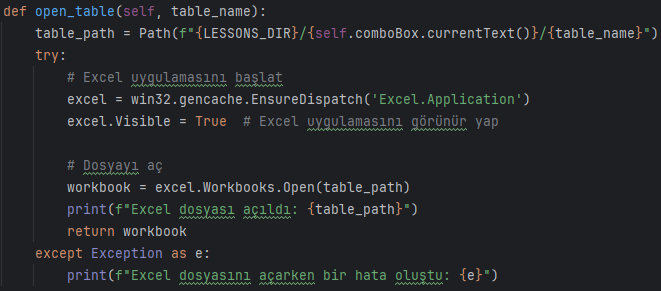
**def update\_combobox(self):**

Bu kod, MainWindow sınıfının update\_combobox adlı bir metodudur ve açılır menüdeki (QComboBox) ders listesini günceller. İlk olarak, self.comboBox.clear() ile açılır menüdeki mevcut tüm öğeler temizlenir. Daha sonra, os.listdir(LESSONS\_DIR) kullanılarak LESSONS\_DIR dizinindeki tüm dersler listelenir ve self.comboBox.addItems() metodu ile açılır menüye eklenir. Bu metot, yeni bir ders oluşturulduğunda veya mevcut derslerde bir değişiklik yapıldığında açılır menünün güncellenmesini sağlar, böylece kullanıcı her zaman en güncel ders listesini görür.

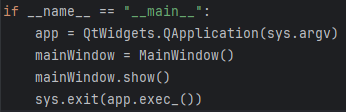
**def on\_combobox\_selection(self, index):**

Bu kod, MainWindow sınıfının on\_combobox\_selection adlı bir metodudur ve açılır menüde (QComboBox) bir seçim yapıldığında tetiklenir. Metot, açılır menüde seçilen öğenin metnini alır ve selectedItem değişkenine atar. index parametresi, seçilen öğenin sırasını temsil eder, ancak bu kodda kullanılmamaktadır. Bu metot, kullanıcının hangi dersi seçtiğini belirlemek ve bu seçime göre işlemler yapmak için bir temel sağlar.

**def open\_table(self, table\_name):**

Bu kod, MainWindow sınıfının open\_table adlı bir metodudur ve kullanıcı tarafından belirtilen bir tablo dosyasını açmak için tasarlanmıştır. İlk olarak, LESSONS\_DIR dizinindeki mevcut derslerden seçilen dersin adı (self.comboBox.currentText()) ve açılacak tablo dosyasının adı (table\_name) birleştirilerek tam dosya yolu table\_path değişkenine atanır. Daha sonra bir try bloğu içinde Microsoft Excel uygulaması başlatılır (win32.gencache.EnsureDispatch('Excel.Application')) ve excel.Visible = True ifadesiyle Excel uygulamasının kullanıcıya görünür hale gelmesi sağlanır. excel.Workbooks.Open(table\_path) metodu ile belirtilen tablo dosyası açılır ve açılan dosyanın workbook nesnesi olarak döndürülmesi sağlanır. Eğer dosya başarıyla açılırsa, dosya yolunun konsola yazdırıldığı bir bilgi mesajı gösterilir.

Eğer işlem sırasında bir hata oluşursa, except bloğu çalışır ve hatanın ayrıntıları konsola yazdırılır. Bu metot, kullanıcıların seçtikleri bir dersin tablolarını Excel'de görüntülemelerine olanak tanır ve tablo dosyalarının düzgün bir şekilde açılmasını sağlar.

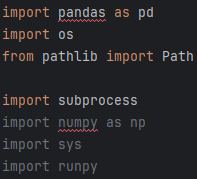


Bu kod, Python uygulamasının giriş noktasını tanımlar ve PyQt ile oluşturulmuş bir grafiksel kullanıcı arayüzünü çalıştırır. İşlevselliği şu şekilde açıklanabilir:

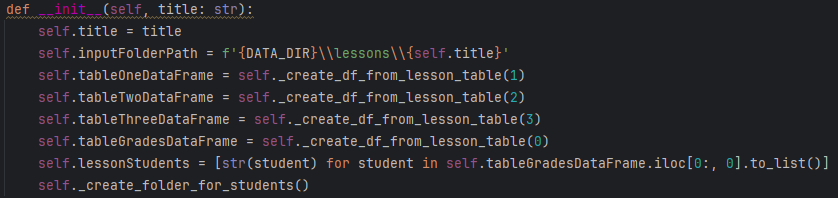
1. if \_\_name\_\_ == "\_\_main\_\_":: Bu satır, bu Python dosyasının doğrudan çalıştırılıp çalıştırılmadığını kontrol eder. Eğer dosya doğrudan çalıştırılıyorsa (\_\_name\_\_ == "\_\_main\_\_"), kod bloğu çalıştırılır; başka bir modül tarafından içe aktarılmışsa çalıştırılmaz.
2. app = QtWidgets.QApplication(sys.argv): PyQt uygulaması başlatılır. sys.argv, komut satırından alınan argümanları PyQt uygulamasına iletmek için kullanılır. PyQt uygulaması, kullanıcı arayüzü olaylarını işlemek için bir uygulama döngüsü oluşturur.
3. mainWindow = MainWindow(): Daha önce tanımlanmış olan MainWindow sınıfından bir nesne oluşturulur. Bu, uygulamanın ana penceresini temsil eder.
4. mainWindow.show(): Ana pencere ekranda görünür hale getirilir.
5. sys.exit(app.exec\_()): PyQt uygulaması döngüsünü başlatır (app.exec\_()), bu da kullanıcı arayüzünün çalışmasını sağlar. Uygulama kapanana kadar bu döngü çalışır. sys.exit() ile uygulama düzgün bir şekilde sonlandırılır ve çıkış kodu işletim sistemine iletilir.

**4.1.5 Main Dosyası**

**Kütüphaneler:**

Bu kod parçasında çeşitli işlemler için kullanılan önemli kütüphaneler içe aktarılıyor. import pandas as pd veri analizi ve veri manipülasyonu için kullanılan bir kütüphanedir ve tablo benzeri veri yapılarını işlemek için kullanılır. import os işletim sistemiyle etkileşim kurmayı sağlar, dosya ve dizin işlemleri için kullanılır. from pathlib import Path modern bir dosya ve dizin yolu işleme kütüphanesi olup platform bağımsız ve okunabilir dosya yolları oluşturmak için tercih edilir. import subprocess dış komutları çalıştırmak ve başka bir Python betiği veya komut satırı işlemini yürütmek için kullanılır. import numpy as np sayısal hesaplamalar için kullanılan bir kütüphanedir, çok boyutlu diziler ve matematiksel işlemler için kullanılır. import sys Python yorumlayıcısı ve çalışma ortamıyla ilgili bilgiler sağlayarak komut satırı argümanlarını almak veya çıkış kodlarını ayarlamak gibi işlemlere imkan tanır. import runpy Python modüllerini dinamik olarak çalıştırmak için kullanılır. Bu kütüphaneler birlikte veri işleme, dosya sistemi manipülasyonu, dış komut çağrıları, çalışma ortamı kontrolü ve modül çalıştırma gibi geniş bir yelpazede işlemleri gerçekleştirebilir.

**class Lesson:**

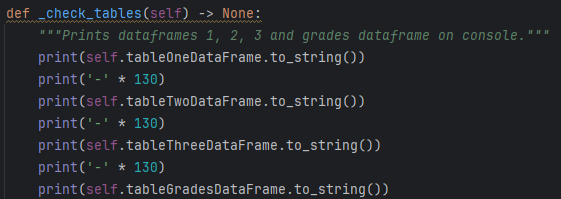
**def \_\_init\_\_(self, title: str):**

Bu metot, Lesson sınıfının kurucusudur (\_\_init\_\_ metodu). Bir dersin başlatılması ve dersle ilgili tüm temel bilgilerin yüklenmesi işlemini gerçekleştirir. İlk olarak, dersin başlığı olan title parametresini alır ve bu başlığı sınıfın self.title özelliğine atar. Daha sonra, dersle ilgili veri dosyalarının bulunduğu klasör yolunu oluşturur ve bunu self.inputFolderPath özelliğine kaydeder. Ardından, dersle ilgili dört farklı tabloyu oluşturmak için \_create\_df\_from\_lesson\_table fonksiyonunu sırasıyla çağırır ve bu tabloları self.tableOneDataFrame, self.tableTwoDataFrame, self.tableThreeDataFrame ve self.tableGradesDataFrame özelliklerine atar. Öğrenci bilgilerini grades tablosundan alır ve dersin kayıtlı öğrencilerini bir listeye dönüştürerek self.lessonStudents özelliğine kaydeder. Son olarak, öğrenciler için gerekli klasörleri oluşturmak amacıyla \_create\_folder\_for\_students metodunu çağırır. Bu işlem, dersle ilgili tüm verilerin yüklenmesini ve düzenlenmesini otomatik hale getirir.

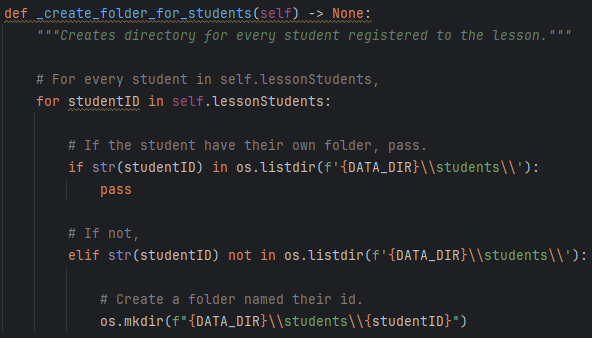
**def\_create\_df\_from\_lesson\_table(self,tableNum:int)->pd.DataFrame:**

Bu fonksiyon, bir ders tablosundan veri alıp bir pandas DataFrame nesnesi oluşturur ve işlemler yaparak gerekirse güncellenmiş tabloyu Excel dosyasına yazar. tableNum parametresi ile hangi tablonun işleneceği belirlenir: 1, 2, 3 numaralı tablolar ya da 0 numaralı notlar tablosu. tableNum == 1 durumunda, tablo 1 okunur ve değerlerin 0 ile 1 arasında olup olmadığını kontrol eder. Ardından, bir "İlişki Değeri" sütunu hesaplanır ve bu sütun tabloya eklenir. tableNum == 2 için tablo 2 okunur, en az 3 sütun olduğundan emin olunur ve ağırlıkların toplamının 100 olup olmadığı kontrol edilir. Sonrasında "TOPLAM" sütunu eklenir veya güncellenir. tableNum == 3 için tablo 2'den alınan veriler ağırlıklı değerlere göre yeniden hesaplanır, anahtar sütunlar manuel olarak düzenlenir ve sonuçta "TOPLAM" ve "Ders Çıktı" sütunları eklenir. tableNum == 0 için notlar tablosu işlenir, "ORT" sütunu yoksa ağırlıklı ortalamalar hesaplanarak eklenir. Tüm durumlarda "Unnamed" sütun başlıkları temizlenir ve tablo düzenlenir. Güncellenmiş tablo belirtilen Excel dosyasına kaydedilir ve son hali bir DataFrame olarak döndürülür.

**def \_check\_tables(self) -> None:**

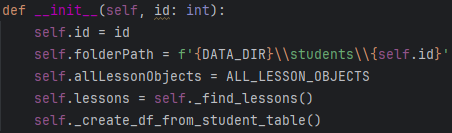
Bu fonksiyon, ilgili ders için oluşturulmuş olan dört tabloyu (1, 2, 3 numaralı tablolar ve notlar tablosu) konsola yazdırır. İlk olarak tableOneDataFrame, ardından sırasıyla tableTwoDataFrame, tableThreeDataFrame, ve tableGradesDataFrame DataFrame nesnelerinin tam içeriğini konsolda görüntüler. Tablolar arasında daha net bir ayrım yapmak için 130 karakter uzunluğunda çizgiyle bir ayırıcı eklenir. Bu yöntem, tüm tabloları hızlıca kontrol etmek veya hata ayıklama sırasında tabloların mevcut durumunu görmek için kullanılır.

**def \_create\_folder\_for\_students(self) -> None:**

Bu fonksiyon, derse kayıtlı olan her öğrenci için birer klasör oluşturur. İlk olarak, self.lessonStudents listesindeki her bir öğrenci kimliği (studentID) üzerinde döngüye girer. Eğer öğrencinin kimliğiyle eşleşen bir klasör, {DATA\_DIR}\students\ dizininde zaten mevcutsa, bu durum atlanır. Ancak, böyle bir klasör mevcut değilse, öğrencinin kimliğiyle adlandırılmış yeni bir klasör oluşturulur. Bu işlem, her öğrencinin kendine ait bir dizin yapısına sahip olmasını sağlar ve dersle ilgili bireysel verilerin düzenli bir şekilde saklanmasına olanak tanır.

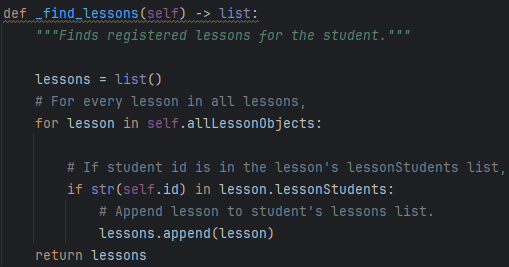
**class Student:**

Bu sınıf, bir öğrenciyi temsil eder ve bir öğrencinin kimliğine göre ders kayıtlarını ve ilişkili verileri işler. \_\_init\_\_ metodu, bir öğrenci nesnesi oluşturulduğunda çağrılır ve aşağıdaki işlemleri gerçekleştirir:

 **def \_\_init\_\_(self, id: int):**

Öğrencinin kimliği (id) alınır ve bu kimlik self.id özelliğine atanır. Daha sonra, öğrencinin verilerinin saklanacağı klasör yolu oluşturulur ve self.folderPath özelliğine atanır. Sistemde mevcut olan tüm ders nesneleri, ALL\_LESSON\_OBJECTS küresel değişkeninden alınarak self.allLessonObjects özelliğine atanır. Öğrencinin kayıtlı olduğu dersler, \_find\_lessons metodu çağrılarak bulunur ve self.lessons özelliğine kaydedilir. Son olarak, öğrencinin derslerle ilgili kişisel tablolarını oluşturmak ve bu verileri bir Excel dosyasına yazmak için \_create\_df\_from\_student\_table metodu çağrılır. Bu işlem, her öğrencinin derslerle ilgili bireysel verilerinin oluşturulmasını ve düzenlenmesini sağlar.

**def \_find\_lessons(self) -> list:**

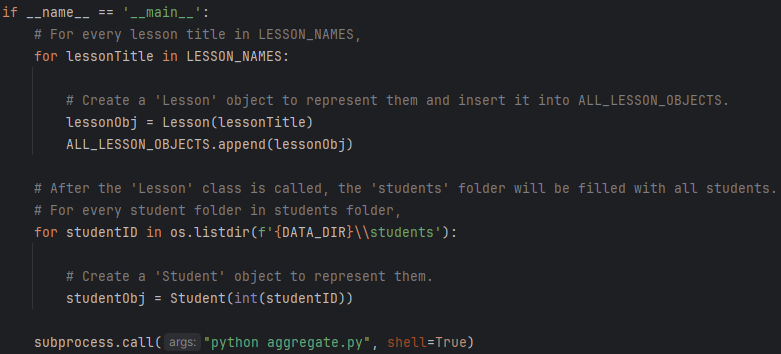
Bu fonksiyon, bir öğrencinin kayıtlı olduğu dersleri bulur ve bu derslerin listesini döndürür. İlk olarak, boş bir liste olan lessons oluşturulur. Daha sonra, tüm mevcut ders nesneleri (self.allLessonObjects) üzerinde döngü yapılır. Her bir ders nesnesinde, öğrencinin kimliği (self.id) dersin öğrenci listesi (lesson.lessonStudents) içinde kontrol edilir. Eğer öğrenci kimliği bu listede bulunuyorsa, o ders nesnesi lessons listesine eklenir. Döngü tamamlandığında, öğrencinin kayıtlı olduğu tüm derslerin yer aldığı lessons listesi geri döndürülür. Bu işlem, bir öğrencinin hangi derslere kayıtlı olduğunu tespit etmek için kullanılır.

**def \_create\_df\_from\_student\_table(self) -> None:**

Bu fonksiyon, bir öğrenciye kayıtlı dersler için verileri işler, tablolar oluşturur ve bu verileri Excel dosyalarına yazar. Öğrencinin ders listesi üzerinde döngüye girerek her ders için işlem yapar. Öncelikle, dersin öğrencinin dizininde bir klasör olup olmadığını kontrol eder, yoksa yeni bir klasör oluşturur. Ardından, dersle ilgili **tableThreeDataFrame** (Tablo 3) ve **tableGradesDataFrame** (Notlar tablosu) DataFrame'lerini alır.

**Tablo 4 oluşturulurken**, Tablo 3'ün verileri yeniden hesaplanır, "Ders Çıktı" sütunu eklenir ve her öğrencinin notlarına göre ağırlıklı toplamlar hesaplanarak "TOPLAM", "MAX" ve "Başarı" sütunları oluşturulur. Elde edilen Tablo 4, öğrencinin klasörüne **table4.xlsx** olarak yazılır.

**Tablo 5 oluşturulurken**, Tablo 1'den alınan veriler işlenir. Tablo 4'ün "Başarı" sütunuyla çarpılarak öğrencinin başarı oranlarını temsil eden yeni bir tablo oluşturulur. "Prg Çıktı" sütunu eklenir, toplam başarı hesaplanır ve maksimum başarı ile karşılaştırılarak "Başarı Oranı" sütunu eklenir. Tablo 5, öğrencinin klasörüne **table5.xlsx** olarak kaydedilir.



Bu kod, programın ana yürütme noktasıdır ve dersler ile öğrencilerin işlenmesini otomatikleştirir. İlk olarak, **LESSON\_NAMES** listesindeki her ders başlığı üzerinde döngü yapılır. Her başlık için bir **Lesson** nesnesi oluşturulur ve bu nesne **ALL\_LESSON\_OBJECTS** listesine eklenir. Bu işlem sırasında, her dersle ilişkili tablolar yüklenir ve gerekli öğrenci klasörleri oluşturulur.

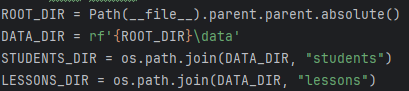
Daha sonra, **students** klasöründeki her öğrenci klasörü üzerinde döngü yapılır. Her öğrenci kimliği (**studentID**) için bir **Student** nesnesi oluşturulur. Bu işlem sırasında, öğrencinin dersleri tespit edilir, bireysel tablolar oluşturulur ve bu tablolar **table4.xlsx** ve **table5.xlsx** olarak öğrencinin klasörüne kaydedilir.

Son olarak, **subprocess.call("python aggregate.py", shell=True)** komutu kullanılarak, dış bir Python betiği olan **aggregate.py** çalıştırılır. Bu, önceki işlemlerden elde edilen verileri daha ileri düzeyde işlemek veya raporlamak için kullanılan bir adımdır.

**4.1.6 Aggregate Dosyası**

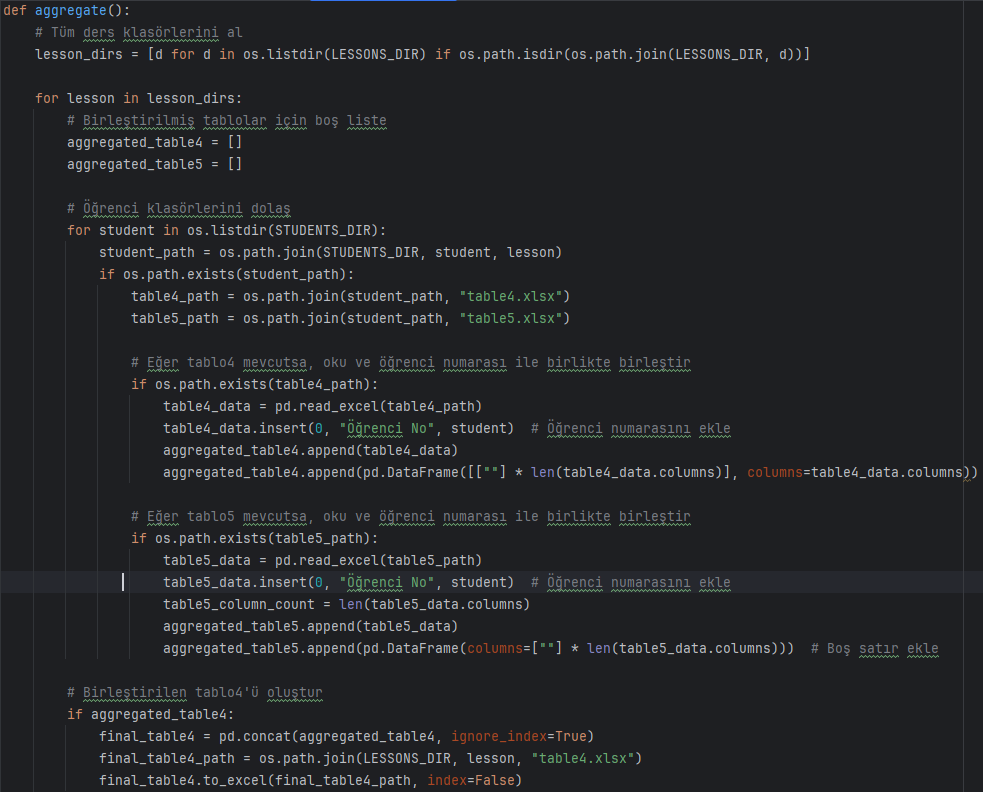
**Kütüphaneler:**

Kod, işletim sistemi ile ilgili işlemleri gerçekleştirmek (os), tablo verilerini okumak ve manipüle etmek (pandas), ve platformdan bağımsız olarak dosya yollarını yönetmek (pathlib) için gerekli olan modülleri içe aktarır. os modülü, dosya ve dizin yollarını birleştirme, listeleme ve kontrol etme işlemlerini sağlarken; pandas modülü Excel tablolarını okuma, birleştirme ve yazma işlemlerini gerçekleştirir. Path sınıfı ise projenin kök dizinini belirlemek ve bu dizin içinde alt dizinleri tanımlamak için kullanılmıştır. Bu kombinasyon, veri işleme ve dosya yönetimi iş akışını kolaylaştırır.

Kod, projenin kök dizinini belirler ve bu kök dizine dayalı olarak "data", "students" ve "lessons" klasörlerinin tam yollarını tanımlar. ROOT\_DIR ile proje kök dizini, DATA\_DIR ile kökteki "data" klasörü, STUDENTS\_DIR ile "data" altındaki "students" klasörü ve LESSONS\_DIR ile "data" altındaki "lessons" klasörünün yolları platformdan bağımsız bir şekilde hazırlanır. Bu yapı, proje dizinindeki verilerin organize bir şekilde erişilmesini sağlar.

**def aggregate():**

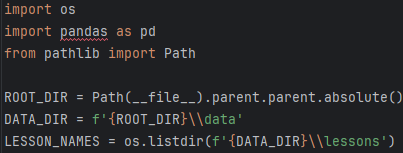
Bu kod, öğrenci klasörlerinden ders bazında Excel dosyalarını toplayarak birleştirir ve sonuçları ilgili ders klasörlerine kaydeder. İlk olarak LESSONS\_DIR içindeki ders klasörlerini listeler ve her ders için döngü başlatır. Her ders için iki liste (aggregated\_table4 ve aggregated\_table5) oluşturularak sırasıyla table4.xlsx ve table5.xlsx dosyalarını birleştirmek için hazırlanır. Ardından STUDENTS\_DIR içindeki her öğrenci klasörüne giderek, ilgili derse ait klasörün varlığı kontrol edilir. Eğer table4.xlsx mevcutsa dosya okunur, öğrenci numarası eklenir ve listeye dahil edilir, ardından görsel düzen için boş bir satır eklenir. Benzer şekilde, table5.xlsx dosyası varsa aynı işlemler uygulanır. Tüm öğrenciler için işlem tamamlandıktan sonra aggregated\_table4 doluysa bu liste pd.concat ile birleştirilerek ilgili ders klasörüne table4.xlsx olarak kaydedilir. aggregated\_table5 için yeni bir DataFrame oluşturulur, her tablo bu DataFrame ile birleştirilir ve sonuç yine ders klasörüne table5.xlsx olarak kaydedilir. Kod, doğrudan çalıştırıldığında aggregate() fonksiyonunu tetikleyerek tüm bu işlemleri gerçekleştirir.

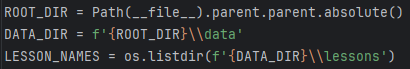


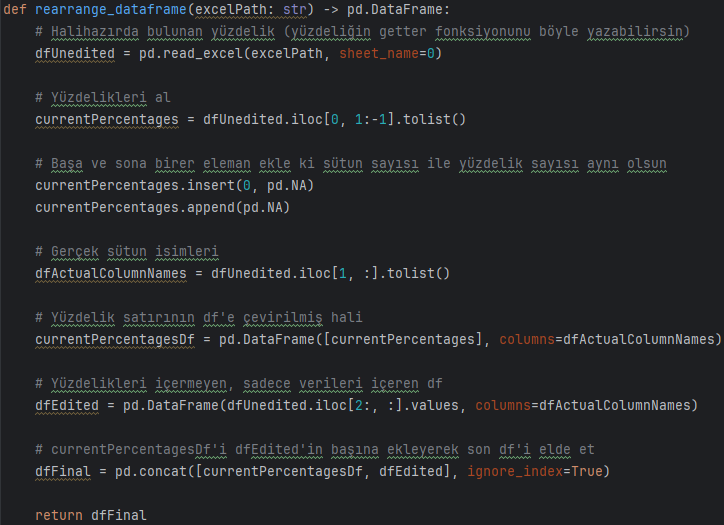


**4.1.7 Setget Dosyası**

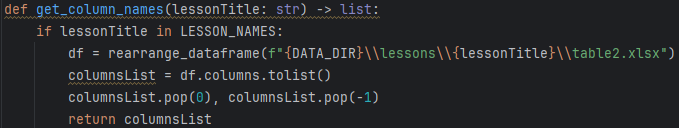
**Kütüphaneler:**

Bu kod parçası, Python'da dosya ve veri işlemleri için gerekli modülleri içeri aktarır. import os işletim sistemiyle ilgili işlemler yapmak için kullanılır, örneğin dosya ve dizinleri listeleme, oluşturma, taşıma veya silme gibi işlemleri sağlar. import pandas as pd veri manipülasyonu ve analizi için kullanılan popüler bir kütüphanedir, özellikle Excel ve CSV gibi dosyaları okuma-yazma işlemleri ve DataFrame yapısıyla veri işleme için kullanılır. from pathlib import Path platformdan bağımsız dosya ve dizin yollarını yönetmek için kullanılan bir modüldür, klasör ve dosya yollarını oluşturmayı, düzenlemeyi ve çözümlemeyi kolaylaştırır. Bu üç modül bir arada dosya yollarının dinamik yönetimi, veri setleriyle çalışma ve işletim sistemiyle etkileşim için temel araçları sağlar.

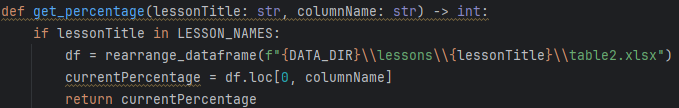
Bu kod parçası dosya yollarını dinamik olarak oluşturur ve bir klasörün içeriğini listeler. ROOT\_DIR = Path(\_\_file\_\_).parent.parent.absolute() çalıştırılan dosyanın iki üst dizinini temsil eden mutlak yolu belirler. DATA\_DIR = f'{ROOT\_DIR}\\data' üst dizine bağlı olarak "data" klasörünün tam yolunu oluşturur. LESSON\_NAMES = os.listdir(f'{DATA\_DIR}\\lessons') ise "data" klasöründeki "lessons" alt klasörünün içindeki tüm dosya ve klasörlerin isimlerini bir liste olarak alır. Bu yapı, dosya yollarını dinamik ve platformdan bağımsız bir şekilde oluşturmak ve hedef klasörün içeriğine erişmek için kullanılır.

 **def rearrange\_dataframe(excelPath: str) -> pd.DataFrame:**

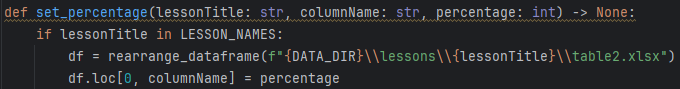
Bu fonksiyon bir Excel dosyasını okuyarak düzenlenmiş bir DataFrame döndürür. dfUnedited = pd.read\_excel(excelPath, sheet\_name=0) Excel dosyasını ilk sayfasından okuyarak ham DataFrame oluşturur. currentPercentages = dfUnedited.iloc[0, 1:-1].tolist() ilk satırdaki yüzdelik verilerini alır ve listeye dönüştürür. currentPercentages.insert(0, pd.NA) ve currentPercentages.append(pd.NA) yüzdeliklerin başına ve sonuna birer boş değer ekler, böylece sütun sayısıyla eşleşir. dfActualColumnNames = dfUnedited.iloc[1, :].tolist() ikinci satırdaki sütun isimlerini alır. currentPercentagesDf = pd.DataFrame([currentPercentages], columns=dfActualColumnNames) yüzdelik verilerinden tek satırlı bir DataFrame oluşturur. dfEdited = pd.DataFrame(dfUnedited.iloc[2:, :].values, columns=dfActualColumnNames) yüzdelik veriler hariç diğer verilerle bir DataFrame oluşturur. dfFinal = pd.concat([currentPercentagesDf, dfEdited], ignore\_index=True) yüzdelik DataFrame'i ve veri DataFrame'ini birleştirerek nihai DataFrame'i oluşturur ve bu DataFrame döndürülür.

 **def get\_column\_names(lessonTitle: str) -> list:**

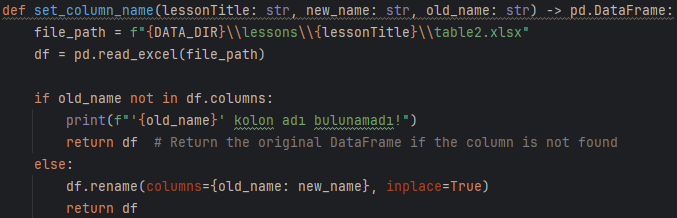
Bu fonksiyon, belirtilen dersin adını kullanarak ilgili Excel dosyasındaki sütun isimlerini döndürür. if lessonTitle in LESSON\_NAMES ifadesi, verilen ders adının LESSON\_NAMES listesinde olup olmadığını kontrol eder. Eğer varsa, df = rearrange\_dataframe(f"{DATA\_DIR}\\lessons\\{lessonTitle}\\table2.xlsx") ile ilgili Excel dosyasını düzenlenmiş bir DataFrame olarak okur. Ardından, columnsList = df.columns.tolist() ile DataFrame'in sütun isimlerini bir listeye dönüştürür. columnsList.pop(0) ve columnsList.pop(-1) ifadeleri listenin ilk ve son elemanını kaldırarak bu sütunları çıkarır. Son olarak, kalan sütun isimlerinin listesi columnsList döndürülür.

 **def get\_percentage(lessonTitle: str, columnName: str) -> int:**

Bu fonksiyon, belirtilen ders ve sütun adına göre Excel dosyasındaki yüzdelik değerini döndürür. if lessonTitle in LESSON\_NAMES ifadesi, verilen ders adının LESSON\_NAMES listesinde olup olmadığını kontrol eder. Eğer ders adı bulunursa, df = rearrange\_dataframe(f"{DATA\_DIR}\\lessons\\{lessonTitle}\\table2.xlsx") ile ilgili Excel dosyasını düzenlenmiş bir DataFrame olarak okur. Ardından, currentPercentage = df.loc[0, columnName] ifadesiyle belirtilen sütunun ilk satırındaki (yüzdelik) değeri alır ve currentPercentage değişkenine atar. Bu değer fonksiyonun sonucunda döndürülür.

 **def set\_percentage(lessonTitle: str, columnName: str, percentage: int) -> None:**

Bu fonksiyon, belirtilen ders ve sütun adına göre Excel dosyasındaki yüzdelik değerini günceller. if lessonTitle in LESSON\_NAMES ifadesi, verilen ders adının LESSON\_NAMES listesinde olup olmadığını kontrol eder. Eğer ders adı bulunursa, df = rearrange\_dataframe(f"{DATA\_DIR}\\lessons\\{lessonTitle}\\table2.xlsx") ile ilgili Excel dosyasını düzenlenmiş bir DataFrame olarak okur. Ardından, df.loc[0, columnName] = percentage ifadesiyle belirtilen sütunun ilk satırındaki yüzdelik değeri verilen percentage değeriyle günceller. Bu işlem sadece DataFrame üzerinde değişiklik yapar, ancak güncellenmiş veriyi dosyaya yazmaz.

 **def set\_column\_name(lessonTitle: str, new\_name: str, old\_name: str) -> pd.DataFrame:**

Bu fonksiyon, belirtilen dersin Excel dosyasındaki bir sütun adını değiştirir. file\_path = f"{DATA\_DIR}\\lessons\\{lessonTitle}\\table2.xlsx" ifadesiyle dersin Excel dosyasının tam yolu oluşturulur ve df = pd.read\_excel(file\_path) ile dosya bir DataFrame olarak okunur. if old\_name not in df.columns ifadesi eski sütun adının DataFrame'de olup olmadığını kontrol eder. Eğer bulunamazsa, print(f"'{old\_name}' kolon adı bulunamadı!") ile bir mesaj yazdırır ve orijinal DataFrame'i döndürür. Eğer eski sütun adı mevcutsa, df.rename(columns={old\_name: new\_name}, inplace=True) ifadesi eski sütun adını yeni sütun adıyla değiştirir ve düzenlenmiş DataFrame döndürülür. Bu işlem dosyaya yazma işlemini içermez, sadece DataFrame üzerinde değişiklik yapar.

 **def write\_to\_excel(lessonTitle: str, df:pd.DataFrame) -> None:**

Bu kod, bir DataFrame'i ilgili Excel dosyasına yazar ve ana programda sütun adını değiştirip güncellenmiş veriyi kaydeder. file\_path = f"{DATA\_DIR}\\lessons\\{lessonTitle}\\table2.xlsx" ile dersin Excel dosyasının tam yolu oluşturulur. df.reset\_index(drop=True, inplace=True) ifadesi, DataFrame'in indekslerini sıfırlayarak eski indekslerin etkisini ortadan kaldırır. Ardından, df.to\_excel(file\_path, index=False) ile düzenlenmiş DataFrame belirtilen dosya yoluna indeksler hariç yazılır. Ana programda önce updated\_df = set\_column\_name("BLM001", "Öd1", "Öd4") ile "BLM001" dersinin Excel dosyasındaki "Öd4" sütun adı "Öd1" olarak değiştirilir. Daha sonra write\_to\_excel("BLM001", updated\_df) ile bu değişiklikler ilgili Excel dosyasına kaydedilir. Kod, "tablo 2"nin değerlendirme kriterlerini güncelleyip Excel dosyasına yazmak amacıyla kullanılır.

**4.2 Görev Dağılımı**

**Geliştirici Emrah Şahin:**

* Lesson ve Student sınıflarının geliştirilmesi.
* Tablo1 ve Tablo2 alımından sonra Tablo3 oluşturulması ve güncellenmesi.
* **Setter/Getter fonksiyonların oluşturulması ve düzenlenmesi**
* **Ders önizleme menüsü**

**Geliştirici Yavuz Selim Gürsoy:**

* Tablo4 ve Tablo5 oluşturulması ve gerekli yerlerde tablo değişiklikleri yapması.
* Bir sonraki proje aşaması için kullanıcı arayüzü kolaylıklarını sağlama.
* **Ders oluşturma ve Ders düzenleme menüleri**

**Geliştirici Talha Tuna:**

* Proje planı
* Excel okuma ve yazdırma
* **Setter/Getter fonksiyonlarının oluşturulması ve düzenlenmesi**
* **Aggregate fonksiyonunun oluşturulması**

**Geliştirici Müge Özler:**

* Test senaryolarının yazılması ve test süreçlerinin yürütülmesi.
* Yazılımın dokümantasyonunun hazırlanması.
* **Arayüzün el ile testi, eski testlerin düzenlenmesi**
* **Arayüzün planlanması ve backend ile kaynaştırılması**

**5 TEST VE DOĞRULAMA**

**5.1 Yazılımın Test Süreci**

Yazılım, unittest modülü kullanılarak test edilmiştir. Her bir sınıf ve fonksiyon için aşağıdaki testler gerçekleştirilmiştir **(**Testler arayüz geliştirme esnasında setter/getter fonksiyonlarıyla çakıştığı için kaldırılmıştır.):

**Lesson Sınıfı Testleri:**

* Başlatma, tablo oluşturma, öğrenci klasörlerinin oluşturulması.
* Tabloların doğru bir şekilde güncellenip güncellenmediği.

**Student Sınıfı Testleri:**

* Başlatma, ders bulma, öğrenci tablolarının oluşturulması.
* Öğrenci bilgileri ile ders bilgileri arasındaki ilişki.

**Fonksiyonel Testler:**

* Kullanıcı etkileşimleri ve beklenen sonuçların doğrulanması.
* Not girişi ve başarı hesaplama işlemlerinin doğruluğu.
* Test süreci, yazılımın güvenilirliğini ve işlevselliğini sağlamak için düzenli olarak yapılacaktır. Her yeni özellik eklendiğinde veya değişiklik yapıldığında testlerin güncellenmesi ve çalıştırılması gerekecektir.
* **Arayüz testleri proje çalışanları tarafından el ile gerçekleştirilmiştir. Arayüz için herhangi bir test fonksiyon dosyası yazılmamıştır.**

**5.2 Hata Yönetimi**

Yazılım, kullanıcı hatalarını ve sistem hatalarını yönetmek için uygun hata mesajları ve istisna yönetimi mekanizmaları içermektedir. Kullanıcıların karşılaşabileceği yaygın hatalar için bilgilendirici mesajlar sağlanacaktır.

**6 SONUÇ**

Bu rapor, yazılım projesinin kapsamını, gereksinimlerini, tasarımını, uygulamasını ve test sürecini detaylı bir şekilde açıklamaktadır. Proje, eğitim kurumlarının yönetim süreçlerini daha verimli hale getirmeyi hedeflemektedir. Geliştirilen yazılım, kullanıcıların ders ve öğrenci bilgilerini etkili bir şekilde yönetmelerine olanak tanırken, aynı zamanda başarı oranlarını hesaplayarak eğitim süreçlerine katkıda bulunmaktadır. Yazılımın gelecekteki sürümlerinde, kullanıcı geri bildirimleri doğrultusunda yeni özellikler eklenmesi ve mevcut işlevlerin iyileştirilmesi planlanmaktadır

**7 GITHUB**

* [Müge Özler](https://github.com/Yavuz-Selim-Gursoy/Ders-Programi)
* [Emrah Şahin](https://github.com/Yavuz-Selim-Gursoy/Ders-Programi)
* [Yavuz Selim Gürsoy](https://github.com/Yavuz-Selim-Gursoy/Ders-Programi)
* [Talha Tuna](https://github.com/Yavuz-Selim-Gursoy/Ders-Programi)