

**T.C**

**KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ**

**MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ**

**BİLGİSAYAR/YAZILIM MÜHENDİSLİĞİ**

**PROJE KONUSU:**

**HAZIRLAYANLAR:**

**Müge ÖZLER Yavuz Selim GÜRSOY Emrah ŞAHİN Talha**

**210501004 220501003**

**DERS SORUMLUSU:**

**DR. ÖĞR. ÜYESİ**

**Elif Pınar HACIBEYOĞLU**

**TARİH:22.11.2024**

**1 GİRİŞ**

**1.1 Projenin Amacı**

Bu projenin amacı, eğitim kurumları için öğrenci ve ders yönetimi yazılımı geliştirmektir. Yazılım, öğretmenlerin ve yöneticilerin ders bilgilerini, öğrenci kayıtlarını ve notları etkili bir şekilde yönetmelerine olanak tanıyacaktır. Kullanıcılar, ders bilgilerini girebilir, güncelleyebilir, notları kaydedebilir ve öğrenci başarılarını hesaplayabilir. Proje, eğitim kurumlarının yönetim süreçlerini kolaylaştırmayı hedeflemektedir.

**2 GEREKSİNİM ANALİZİ**

**2.1 Arayüz Gereksinimleri**

Kullanıcı Dostu Arayüz: Yazılım, kullanıcıların kolayca etkileşimde bulunabileceği, sezgisel bir arayüze sahip olmalıdır.

Ders Ekleme ve Güncelleme Formları: Kullanıcıların ders bilgilerini girmesi ve güncellemesi için formlar.

Öğrenci Ekleme ve Güncelleme Formları: Öğrenci bilgilerini girmek ve güncellemek için formlar.

Not Girişi ve Başarı Hesaplama Tabloları: Kullanıcıların notları girebileceği ve başarı oranlarını görebileceği tablolar.

Raporlama ve Grafiksel Gösterimler: Kullanıcıların başarı oranlarını ve ders performansını grafiksel olarak görüntüleyebilmesi.

**2.2 Fonksiyonel Gereksinimler**

Ders Yönetimi: Kullanıcıların ders bilgilerini ekleyebilmesi, güncelleyebilmesi ve silebilmesi.

Öğrenci Yönetimi: Kullanıcıların öğrenci bilgilerini ekleyebilmesi, güncelleyebilmesi ve silebilmesi.

Not Girişi: Kullanıcıların ders notlarını girebilmesi ve güncelleyebilmesi.

Başarı Hesaplama: Kullanıcıların öğrenci başarı oranlarını hesaplayabilmesi.

Veri Kaydetme: Tabloların Excel formatında kaydedilmesi.

Raporlama: Öğrenci ve ders bilgilerine göre raporların oluşturulması.

**2.3 Use-Case Diyagramı**

Kullanıcı: Ders ekleme, öğrenci ekleme, not girişi, başarı hesaplama.

Sistem: Veritabanı ile etkileşim, tablo oluşturma, dosya kaydetme.

**3 TASARIM**

**3.1 Mimari Tasarım**

Yazılım, Model-View-Controller (MVC) mimarisi kullanılarak tasarlanmıştır. Bu mimari, uygulamanın bileşenlerini ayırarak daha modüler ve yönetilebilir hale getirir.

Model: Ders ve öğrenci bilgilerini yöneten sınıflar (Lesson, Student).

View: Kullanıcı arayüzü bileşenleri.

Controller: Kullanıcı etkileşimlerini yöneten ve model ile view arasında köprü görevi gören fonksiyonlar.

**3.2 Kullanılacak Teknolojiler**

Python: Yazılımın ana programlama dili.

Pandas: Veri analizi ve tablo yönetimi için.

OpenPyXL: Excel dosyaları ile etkileşim için.

unittest: Yazılımın test edilmesi için.

**4 UYGULAMA**

**4.1 Kodlanan Bileşenlerin Açıklamaları**

**4.1.1 Lesson Sınıfı**

**Tanım:** Lesson sınıfı, ders bilgilerini tutar ve dersle ilgili tabloları oluşturur.

**Özellikler:**

title: Dersin başlığını tutar.

inputFolderPath: Dersin dosya yolunu belirtir.

tableOneDataFrame, tableTwoDataFrame,tableThreeDataFrame, tableGradesDataFrame: Dersle ilgili tabloları tutan DataFrame nesneleri.

lessonStudents: Derse kayıtlı öğrencilerin listesini tutar.

**Metotlar:**

\_\_init\_\_(self, title: str): Sınıfın başlatıcısıdır. Ders başlığını alır ve ilgili tabloları oluşturur.

\_create\_df\_from\_lesson\_table(self, tableNum: int): Belirtilen tablo numarasına göre ilgili tabloyu okur ve gerekli işlemleri yaparak bir DataFrame oluşturur.

Tablo 1: Değerlerin 0 ile 1 arasında olduğunu kontrol eder ve 'İlişki Değeri' sütununu hesaplar.

Tablo 2: En az 3 sütun ve not ağırlıklarının toplamının 100 olduğunu kontrol eder.

Tablo 3: Ağırlıklı değerlendirme hesaplamaları yapar ve 'Ders Çıktı' sütununu ekler.

Notlar Tablosu (Tablo 0): Öğrenci notlarını okur ve 'ORT' sütununu hesaplar.

\_check\_tables(self): Tabloları konsola yazdırır.

\_create\_folder\_for\_students(self): Her öğrenci için bir klasör oluşturur.

**4.1.2 Student Sınıfı**

**Tanım:** Student sınıfı, öğrenci bilgilerini tutar ve öğrencinin kayıtlı olduğu dersleri bulur.

**Özellikler:**

id: Öğrencinin kimlik numarasını tutar.

folderPath: Öğrencinin klasör yolunu belirtir.

allLessonObjects: Tüm ders nesnelerini tutar.

lessons: Öğrencinin kayıtlı olduğu dersleri tutar.

**Metotlar:**

\_\_init\_\_(self, id: int): Sınıfın başlatıcısıdır. Öğrenci kimliğini alır ve ilgili dersleri bulur.

\_find\_lessons(self): Öğrencinin kayıtlı olduğu dersleri bulur.

\_create\_df\_from\_student\_table(self): Öğrencinin dersleri için tabloları oluşturur ve dosyaya kaydeder**.**

**4.1.3 LessonDataMutator Sınıfı**

**Tanım:** LessonDataMutator sınıfı, Lesson nesneleri için tablo güncelleme fonksiyonlarını içerir.

**Metotlar:**

set\_table\_one(self, new\_table: pd.DataFrame): Tablonun tamamını günceller.

set\_table\_one\_column(self, column\_name: str, new\_column: pd.Series): Belirli bir sütunu günceller.

set\_table\_one\_row(self, row\_index: int, new\_row: pd.Series): Belirli bir satırı günceller.

set\_table\_one\_value(self, column\_name: str, row\_index: int, new\_value): Belirli bir hücreyi günceller.

**4.1.4 StudentDataMutator Sınıfı**

**Tanım:** StudentDataMutator sınıfı, Student nesneleri için tablo güncelleme fonksiyonlarını içerir.

**Metotlar:**

set\_table4(self, lesson\_title: str, new\_table: pd.DataFrame): Belirli bir ders için tablo 4'ü günceller.

set\_table5(self, lesson\_title: str, new\_table: pd.DataFrame): Belirli bir ders için tablo 5'i günceller.

**4.2 Görev Dağılımı**

**Geliştirici 1:**

Lesson ve Student sınıflarının geliştirilmesi.

Tabloların oluşturulması ve güncellenmesi.

**Geliştirici 2:**

Veri analizi ve tablo yönetimi için fonksiyonların yazılması.

Kullanıcı arayüzü bileşenlerinin tasarımı.

**Geliştirici 3:**

Test senaryolarının yazılması ve test süreçlerinin yürütülmesi.

Yazılımın dokümantasyonunun hazırlanması.

**5 TEST VE DOĞRULAMA**

**5.1 Yazılımın Test Süreci**

Yazılım, unittest modülü kullanılarak test edilmiştir. Her bir sınıf ve fonksiyon için aşağıdaki testler gerçekleştirilmiştir:

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.

**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