metin, simge, sembol, logo, amblem içeren bir resim

Açıklama otomatik olarak oluşturuldu

**ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ**

**MÜHENDİSLİK MİMARLIK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**VERİTABANI YÖNETİM SİSTEMLERİ DERSİ**

**PROJE ADI:** ANAKOD YAZILIM AKADEMİSİ ŞİRKETİ KURS KAYIT, ÖĞRENCİ VE ÖDEME TAKİP SİSTEMİ

**GRUP NO:** 7

**HAZIRLAYANLAR:**

MEVLÜT BERKE YILDIRIM (GRUP KOORDİNATÖRÜ) – 152120211084

SALİH ENES ÜNAL – 152120211061

EREN KALKAN – 152120221000

**ARALIK 2024**

**PROJEDE ORTAK ÇALIŞILACAK PAYDAŞ ŞİRKET BİLGİLERİ**

**tilki, çizgi film, memeli içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**ŞİRKET İSMİ:** Anakod Yazılım Akademisi Teknoloji Limited Şirketi

**ŞİRKET İLETİŞİM BİLGİLERİ**

**ŞİRKET YÖNETİCİSİ:** MUHAMMED TAHA ÖZDEMİR

**EMAIL:** [iletisim@anakodakad​emi.com](mailto:iletisim@anakodakad​emi.com)

**GSM: +**90 0531 492 25 41

**ADRES** : HOŞNUDİYE MAH. 770 SK. CASABA ​MODERN NO: 4 A İÇ KAPI NO: 304 ​TEPEBAŞI/ ESKİŞEHİR

İçindekiler Tablosu

[1.Projenin Amacı 4](#_Toc185275626)

[1.1 Projenin Hedefleri 4](#_Toc185275627)

[1.2 Kullanılan Teknolojiler 4](#_Toc185275628)

[1.3 Yapılan Çalışmalar ve Gelinen Son Nokta 5](#_Toc185275629)

[2.Gereksinimler 6](#_Toc185275630)

[2.1 Gereksinim Analizi Final Versiyon 6](#_Toc185275631)

[3.Veri Yapı Tasarımı 12](#_Toc185275632)

[3.1 ER Diagramı Tasarımı Final Versiyon 12](#_Toc185275633)

[3.2 Veri Yapısı Tasarımı 13](#_Toc185275634)

[4. Sorgular 23](#_Toc185275635)

[5. VTYS Araç Kullanımı 30](#_Toc185275636)

[6. Kullanıcı Arayüzü (User Interface) 40](#_Toc185275637)

[7- Diğer Entegrasyonlar 49](#_Toc185275638)

[8- Çalışma Adam-saat değerleri 55](#_Toc185275639)

[9. Sonuç ve Değerlendirme 56](#_Toc185275640)

# 1.Projenin Amacı

Bu proje, Anakod Yazılım Akademisi eğitim kurumunun öğrenci kayıt, kurs yönetimi ve ödeme takibi süreçlerini dijitalleştirerek daha etkili ve verimli bir yönetim sağlamak amacıyla geliştirilmiştir. Proje, eğitim kurumunun idari yüklerini hafifletmeyi, süreçlerdeki hataları minimize etmeyi ve öğrencilerin ödemelerinin takibini kolaylaştırmayı hedefler.

## Projenin Hedefleri

**Kayıt ve Bilgi Yönetimini Kolaylaştırmak:** Öğrencilerin kurslara kaydı ve yöentilmesi dijital ortamda saklayarak, yöneticilerin ve eğitmenlerin bu verilere kolayca erişmesini sağlamak.

**Ödeme Takibini Sağlamak:** Öğrencilerin ödeme durumlarını takip etmek, dekontları depolamak ve ödeme yapılmadığı durumlarda veliye bilgilendirme mailleri göndermek.

**Veli ve Eğitmen Yönetimi:** Velilerle ve öğrencilere iletişim kurarak ödeme ve öğrenci durumları hakkında bilgilendirme yapılması; eğitmenlerin kurs yönetimini ve öğrencilerin aldıkları derslerin takip edebilmesi.

## Kullanılan Teknolojiler

* Windows Forms – Arayüz
* SQL Server, Azure, T-SQL – SQL Query
* ClosedXML.Excel – Execele okuma, yazma, düzenleme yapmak
* Google Apis Auth- Google API'lerine bağlanmak ve kimlik doğrulama işlemlerini yönetmek (Sosyal Medya Entegrasyonu için kullanıldı)
* OpenXml – exceli XML formatında okuma, yazmak
* Text.RegularExpressions – email ve telefon numarası doğrulamak
* E-posta gönderimi için SMTP (Simple Mail Transfer Protocol) sunucusuyla iletişim kurar.
* Uİ için formsun kendi componentleri kullanıldı.

## 1.3 Yapılan Çalışmalar ve Gelinen Son Nokta

1-Tüm modüllerimiz için formların UI tasarımı tamamlandı.

2-Giriş ekranı için Google ile giriş yapma sağlandı ve Sosyal Medya Login kısmı tamamlandı. Veritabanında kullanıcı kaydı varsa sisteme giriş sağlandı.

3-Şifremi Unuttum Kısmı eklendi. Şifresini unutan kullanıcın mailini yeni şifresi

İletilmektedir.

4-Erişim yetkilendirme kısmı tamamlandı. Muhasebe ve admin kendi yetkisindeki formları açabilecek.

5- Öğrenci Yönetimi, Kurs Eklem, Eğitmen Yönetimi, Eğitmene Kurs Atama, Öğrenciyi Kursa Ekleme, Ödeme Yönetimi, Kurs Önerme, Bilgilendirme Yönetimi ve Kullanıcı Yönetim formları veritabanına bağlandı ve **CRUD** işlemlerini gerçekleştirmektedir.bu işlemleri yaparken (güncelleme, silme, ekleme vs.) Hata ayıklamalar kullanılarak yapıldı.

6- Excele import / export işlevi eklendi.

7-Mail ile bilgilendirme kısmı eklendi bu sayede Velilerle ve öğrencilere iletişim kurarak ödeme ve öğrenci durumları hakkında bilgilendirme yapılması sağlanacak.Gönderilen mailler veritabanında saklanacak.

8- Kurslara level(Levels) bilgisi eklendi. Level 1 , Level 2 , Level 3 şeklinde her kursun kendisine ait seviyelerde oluşturulması sağlanacak . Örneğin Python level 1 , Python level 2 , Python level 3 gibi.

9- Ödeme kısmına ödenecek tutarı taksitlendirme kısmı da eklendi. (Installments , PaymentDetails).Receipts tablosunun işlevi uygulamamıza eklendi pdflerin yolları veritabanına kaydı gerçekleştiriliyor.

10-Kurs Öneri Sistemi UI tasarlandı ve veritabanında gerekli tablolar(Suggestions, StudentSuggestions) oluşturuldu.

11- Store Procedureler Oluşturuldu ve Kurs Önerme için Trigger yazıldı.

12- Öğrenci Alanlarına şirketten Aldığımız geri dönüt doğrultusunda gender alanı ve

Okul alanı ekledik.

# 2.Gereksinimler

**Eklenen Kısımlar Özeti:** Suggestions, StudentSuggestions, PaymentDetails tabloları eklendi, Auditlogs Tablosu çıkartıldı, NotificationTemplate tablosu çıkartıldı.

## 2.1 Gereksinim Analizi Final Versiyon

**Öğrenci Tablosu (Students)**

1. Öğrenciler için bir tablo oluşturulmalıdır.
2. Her öğrenci benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Öğrencilerin isim ve soyisim bilgileri kaydedilmelidir.
4. Öğrencilerin doğum tarihleri kaydedilmelidir.
5. Öğrencilerin iletişim numaraları kaydedilmelidir.
6. Öğrencilerin e-posta adresleri kaydedilmelidir.
7. Öğrencilerin kayıt tarihleri kaydedilmelidir.
8. Öğrencilerin eğitim seviyesi (ilkokul, ortaokul, lise, üniversite) kaydedilmelidir.
9. Öğrencilerin kodlama deneyimi olup olmadığı kaydedilmelidir.
10. Öğrenci Cinsiyet bilgisi kaydedilmelidir.
11. Öğrencinin Okulu Kaydedilmelidir.

**Veli Tablosu (Guardians)**

1. Veliler için bir tablo oluşturulmalıdır.
2. Her veli benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Velilerin isim bilgisi kaydedilmelidir.
4. Velilerin iletişim numaraları kaydedilmelidir.
5. Velilerin kendilerine ait benzersiz telefon numaraları olmalıdır.
6. Velilerin e-postaları benzersiz ve boş değer olmamalıdır.
7. Veliler bir veya birden fazla öğrenciyle ilişkilendirilebilmelidir.

**Kurs Tablosu (Courses)**

1. Kurslar için bir tablo oluşturulmalıdır.
2. Her kurs benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Kurs adı kaydedilmelidir.
4. Kurs tipi kaydedilmelidir (örn. programlama, tasarım).
5. Kurs seviyesi (örn. başlangıç, ileri) kaydedilmelidir.
6. Kursun açıklaması kaydedilmelidir.
7. Kurs süresi kaydedilmelidir.
8. Kursun başlangıç tarihi kaydedilmelidir.
9. Kursun maksimum katılımcı kapasitesi belirlenmelidir.

**Seviye Tablosu (Levels)**

1. Kurs seviyeleri için bir tablo oluşturulmalıdır.
2. Her seviye benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Her seviye, ilgili bir kurs ile ilişkilendirilmelidir.
4. Seviye adı kaydedilmelidir.
5. Seviye açıklaması kaydedilmelidir.
6. Seviyenin minimum gereksinimleri kaydedilmelidir.

**Kayıt Tablosu (Enrollments)**

1. Öğrenci kayıtları için bir tablo oluşturulmalıdır.
2. Her kayıt benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Her kayıt, ilgili bir öğrenciyle ilişkilendirilmelidir.
4. Her kayıt, ilgili bir kursla ilişkilendirilmelidir.
5. Kayıt tarihi kaydedilmelidir.
6. Kayıt durumu (örn. tamamlandı, devam ediyor) kaydedilmelidir.

**Eğitmen Tablosu (Instructors)**

1. Eğitmenler için bir tablo oluşturulmalıdır.
2. Her eğitmen benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Eğitmen isim bilgisi kaydedilmelidir.
4. Eğitmen iletişim numarası benzersiz olmalıdır.
5. Eğitmen e-posta adresi kaydedilmelidir.

**Eğitmen Atama Tablosu (InstructorAssignments)**

1. Eğitmen atamaları için bir tablo oluşturulmalıdır.
2. Her atama benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Her atama, ilgili bir eğitmenle ilişkilendirilmelidir.
4. Her atama, ilgili bir kursla ilişkilendirilmelidir.
5. Atama tarihi kaydedilmelidir.

**Ödeme Tablosu (Payments)**

1. Ödemeler için bir tablo oluşturulmalıdır.
2. Her ödeme benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Her ödeme, ilgili bir öğrenciyle ilişkilendirilmelidir.
4. Ödeme miktarı kaydedilmelidir.
5. Ödeme tarihi kaydedilmelidir.
6. Ödeme durumu (örn. ödendi, ödenmedi) kaydedilmelidir.
7. Ödeme tipi (örn. peşin, taksitli) kaydedilmelidir.
8. Taksitli ödemelerde toplam taksit sayısı ve her taksit tutarı kaydedilmelidir.
9. Şu ana kadar ödenen toplam tutar kaydedilmelidir.
10. Elden yapılan ödeme tarihleri ve tutarları kaydedilmelidir.
11. Ekstra notlar eklenebilmelidir.

**Taksit Tablosu (Installments)**

1. Taksitler için bir tablo oluşturulmalıdır.
2. Her taksit benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Her taksit, ilgili bir ödeme ile ilişkilendirilmelidir.
4. Taksit numarası kaydedilmelidir.
5. Her taksitin ödeme miktarı kaydedilmelidir.
6. Taksit ödeme tarihleri kaydedilmelidir.
7. Taksit ödeme durumu (örn. beklemede, ödendi) kaydedilmelidir.

**Makbuz Tablosu (Receipts)**

1. Makbuzlar için bir tablo oluşturulmalıdır.
2. Her makbuz benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Her makbuz, ilgili bir ödeme ile ilişkilendirilmelidir.
4. Makbuz tarihi kaydedilmelidir.
5. Makbuz PDF dosyasının kaydedileceği dosya yolu tutulmalıdır.

**E-posta Bildirim Tablosu (EmailNotifications)**

1. E-posta bildirimleri için bir tablo oluşturulmalıdır.
2. Her bildirim benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Bildirim, ilgili bir öğrenciyle ilişkilendirilmelidir.
4. Bildirim, ilgili bir veliyle ilişkilendirilmelidir.
5. Bildirimin konusu kaydedilmelidir.
6. Bildirimin içeriği kaydedilmelidir.
7. Gönderim tarihi kaydedilmelidir.
8. Bildirim durumu (örn. beklemede, gönderildi) kaydedilmelidir.
9. Bildirim türü kaydedilmelidir.

**Kullanıcı Tablosu (Users)**

1. Kullanıcılar için bir tablo oluşturulmalıdır.
2. Her kullanıcı benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Kullanıcı adı benzersiz olmalıdır.
4. Kullanıcı şifresi kaydedilmelidir.
5. Kullanıcı e-posta adresi kaydedilmelidir.
6. Kullanıcı tipi, bir kullanıcı türüyle ilişkilendirilmelidir.
7. Kullanıcının oluşturulma tarihi kaydedilmelidir.

**Kullanıcı Türleri Tablosu (UserTypes)**

1. Kullanıcı türleri için bir tablo oluşturulmalıdır.
2. Her kullanıcı türü benzersiz bir kimlik numarası ile tabloya kaydedilmelidir.
3. Kullanıcı türü adı benzersiz olmalıdır.
4. Kullanıcı türünün açıklaması kaydedilmelidir.

**Table Suggestions**

91.Suggestions tablosu oluşturulmalıdır.

92.Her öneri için benzersiz bir kimlik numarası atanmalıdır.

93.Önerilen kursun, daha önce tamamlanmış bir kurs ile ilişkisi tanımlanmalıdır.

94.Tamamlanmış bir kursa dayalı olarak önerilen kurs bilgisi kaydedilmelidir.

95.Önerinin e-posta ile paylaşılıp paylaşılmadığı belirtilmelidir.

**Table StudentSuggestions**

96.StudentSuggestions tablosu oluşturulmalıdır.

97.Her öğrenci önerisi için benzersiz bir kimlik numarası atanmalıdır.

98.Öğrenci önerisi, ilgili bir öğrenciyle ilişkilendirilmelidir.

99.Önerinin ilişkilendirildiği öğrencinin adı kaydedilmelidir.

100.Öneri, öğrencinin tamamladığı bir kurs ile ilişkilendirilmelidir.

101.Öneri, öğrencinin alması tavsiye edilen bir kurs ile ilişkilendirilmelidir.

102.Önerinin, Suggestions tablosundaki bir kayıt ile ilişkisi belirtilmelidir.

103.Önerinin oluşturulma tarihi otomatik olarak kaydedilmelidir.

**Table PaymentDetails**

104.PaymentDetails tablosu oluşturulmalıdır.

105.Her ödeme detayı için benzersiz bir kimlik numarası atanmalıdır.

106.Ödeme detayı, Payments tablosundaki bir kayıt ile ilişkilendirilmelidir.

107.Ödemenin yapıldığı tarih otomatik olarak kaydedilmelidir.

108.Ödemenin toplam tutarı kaydedilmelidir.

109.Ödeme yöntemi (örn. kredi kartı, havale) belirtilmelidir.

110.Ödemeye ilişkin açıklamalar kaydedilmelidir.

# 3.Veri Yapı Tasarımı

## 3.1 ER Diagramı Tasarımı Final Versiyon

metin, ekran görüntüsü, sayı, numara, paralel içeren bir resim

Açıklama otomatik olarak oluşturuldu

## 3.2 Veri Yapısı Tasarımı

**Students Tablosu**

Öğrenci bilgilerini saklar.

**Sütunlar ve Kısıtlamalar**

student\_id

**Kısıtlama:**

**PRIMARY** KEY: Her öğrenci için benzersiz kimlik.

**IDENTITY**: Otomatik artan bir sayı.

level\_id

**Kısıtlama:**

FOREIGN **KEY**: Levels tablosundaki level\_id sütununa referans verir.

first\_name

**Kısıtlama:**

NOT NULL: Öğrenci adı belirtilmek zorunda.

last\_name

**Kısıtlama**:

NOT NULL: Öğrenci soyadı belirtilmek zorunda.

date\_of\_birth

Öğrencinin doğum tarihi.

contact\_number

Öğrencinin telefon numarası.

email

Kısıtlama:

NOT NULL: E-posta adresi belirtilmek zorunda.

enrollment\_date

Kısıtlama:

NOT NULL: Kayıt tarihi belirtilmek zorunda.

level

Kısıtlama:

NOT NULL: Eğitim seviyesi belirtilmek zorunda.

CHECK: "ilkokul", "ortaokul", "lise", "universite" değerlerini alabilir.

coding\_experience

Programlama deneyimi bilgisi (BIT veri tipi).

status

Öğrencinin durumu.

**Guardians Tablosu**

Velilere ait bilgileri saklar.

**Sütunlar ve Kısıtlamalar**

guardian\_id

**Kısıtlama**:

PRIMARY KEY: Her veli için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

student\_id

**Kısıtlama:**

FOREIGN KEY: Students tablosundaki student\_id sütununa referans verir.

ON DELETE CASCADE: Öğrenci kaydı silindiğinde veli kaydı da silinir.

name

**Kısıtlama:**

NOT NULL: Veli adı belirtilmek zorunda.

contact\_number

Veli telefon numarası.

email

**Kısıtlama**:

NOT NULL: E-posta adresi belirtilmek zorunda.

UNIQUE: Her veli için benzersiz olmalı.

**Courses Tablosu**

Kurs bilgilerini saklar.

**Sütunlar ve Kısıtlamalar**

course\_id

**Kısıtlama**:

PRIMARY KEY: Her kurs için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

course\_name

**Kısıtlama**:

NOT NULL: Kurs adı belirtilmek zorunda.

course\_type

**Kısıtlama**:

NOT NULL: Kurs tipi belirtilmek zorunda.

course\_level

**Kısıtlama**:

NOT NULL: Kurs seviyesi belirtilmek zorunda.

description

Kursun açıklaması.

duration

Kurs süresi (gün olarak).

course\_Date

**Kısıtlama:**

NOT NULL: Kurs tarihi belirtilmek zorunda.

MaxCapacity

**Kısıtlama**:

NOT NULL: Maksimum kapasite belirtilmek zorunda.

**Levels Tablosu**

Kurs seviyelerini saklar.

Sütunlar ve Kısıtlamalar

level\_id

**Kısıtlama**:

PRIMARY KEY: Her seviye için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

course\_id

Kısıtlama:

FOREIGN KEY: Courses tablosundaki course\_id sütununa referans verir.

ON DELETE CASCADE: Kurs kaydı silindiğinde bu seviyeler de silinir.

level\_name

**Kısıtlama:**

NOT NULL: Seviye adı belirtilmek zorunda.

level\_description: Seviye açıklaması.

min\_requirements :Seviye için minimum gereksinimler.

**Enrollments Tablosu**

Öğrencilerin kurs kayıt bilgilerini saklar.

**Sütunlar ve Kısıtlamalar**

enrollment\_id

**Kısıtlama**:

PRIMARY KEY: Her kayıt için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

student\_id

**Kısıtlama**: FOREIGN KEY: Students tablosundaki student\_id sütununa referans verir.

ON DELETE CASCADE: Öğrenci kaydı silindiğinde bu kayıt da silinir.

course\_id

**Kısıtlama**:

FOREIGN KEY: Courses tablosundaki course\_id sütununa referans verir.

ON DELETE CASCADE: Kurs kaydı silindiğinde bu kayıt da silinir.

enrollment\_date

**Kısıtlama**:

NOT NULL: Kayıt tarihi belirtilmek zorunda.

completion\_status

Kursun tamamlanma durumu.

**Instructors Tablosu**

Eğitmen bilgilerini saklar.

**Sütunlar ve Kısıtlamalar**

instructor\_id

**Kısıtlama**:

PRIMARY KEY: Her eğitmen için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

name

**Kısıtlama**:

NOT NULL: Eğitmen adı belirtilmek zorunda.

contact\_number

**Kısıtlama**:

UNIQUE: Eğitmen telefon numarası benzersiz olmalı.

email

Eğitmen e-posta adresi.

**InstructorAssignments Tablosu**

Eğitmen atama bilgilerini saklar.

**Sütunlar ve Kısıtlamalar**

assignment\_id

**Kısıtlama**:

PRIMARY KEY: Her atama için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

instructor\_id

**Kısıtlama**:

FOREIGN KEY: Instructors tablosundaki instructor\_id sütununa referans verir.

ON DELETE CASCADE: Eğitmen silindiğinde bu atama da silinir.

course\_id

**Kısıtlama**:

FOREIGN KEY: Courses tablosundaki course\_id sütununa referans verir.

ON DELETE CASCADE: Kurs silindiğinde bu atama da silinir.

assignment\_date

**Kısıtlama**:

NOT NULL: Atama tarihi belirtilmek zorunda.

**Payments Tablosu**

Öğrenci ödemelerini saklar.

Sütunlar ve Kısıtlamalar

payment\_id

**Kısıtlama**:

PRIMARY KEY: Her ödeme için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

student\_id

**Kısıtlama**:

FOREIGN KEY: Students tablosundaki student\_id sütununa referans verir.

ON DELETE CASCADE: Öğrenci kaydı silindiğinde ödeme kaydı da silinir.

amount

**Kısıtlama**:

NOT NULL: Ödeme miktarı belirtilmek zorunda.

payment\_date

Ödemenin yapıldığı tarih.

payment\_status

**Kısıtlama**:

CHECK: "Ödenmedi", "Ödendi", "Takip Edilecek" değerlerini alabilir.

payment\_type

**Kısıtlama**:

CHECK: "Peşin", "Taksitli", "Elden" değerlerini alabilir.

cash\_payment\_date

Elden yapılan ödemenin tarihi.

cash\_amount

Elden yapılan ödeme miktarı.

Notes

Ekstra notlar.

installment\_count

Taksit sayısı.

installment\_amount

Her taksit için ödenecek miktar.

total\_paid

Kısıtlama:

DEFAULT 0: Şu ana kadar ödenen toplam miktar.

**Installments Tablosu**

Taksit bilgilerini saklar.

**Sütunlar ve Kısıtlamalar**

installment\_id

**Kısıtlama**:

PRIMARY KEY: Her taksit için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

payment\_id

**Kısıtlama**:

FOREIGN KEY: Payments tablosundaki payment\_id sütununa referans verir.

ON DELETE CASCADE: Ödeme kaydı silindiğinde taksit kaydı da silinir.

installment\_number

Taksit numarası.

amount

Her bir taksitin tutarı.

due\_date

Taksit ödeme tarihi.

payment\_status

**Kısıtlama**:

CHECK: "Beklemede", "Ödendi" değerlerini alabilir.

**Receipts Tablosu**

Ödemelerle ilişkili dekont bilgilerini saklar.

**Sütunlar ve Kısıtlamalar**

receipt\_id

**Kısıtlama**:

PRIMARY KEY: Her dekont için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

payment\_id

**Kısıtlama**:

FOREIGN KEY: Payments tablosundaki payment\_id sütununa referans verir.

ON DELETE CASCADE: Ödeme kaydı silindiğinde dekont kaydı da silinir.

receipt\_date

**Kısıtlama**:

NOT NULL: Dekont tarihi belirtilmek zorunda.

file\_path

Dekont dosyasının yolunu saklar.

**EmailNotifications Tablosu**

E-posta bildirimlerini saklar.

Sütunlar ve **Kısıtlamalar**

email\_notification\_id

**Kısıtlama**:

PRIMARY KEY: Her bildirim için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

student\_id

**Kısıtlama**:

FOREIGN KEY: Students tablosundaki student\_id sütununa referans verir.

ON DELETE CASCADE: Öğrenci kaydı silindiğinde bildirim kaydı da silinir.

guardian\_id

**Kısıtlama**:

FOREIGN KEY: Guardians tablosundaki guardian\_id sütununa referans verir.

ON DELETE SET NULL: Veli kaydı silindiğinde bu alan NULL olur.

subject

**Kısıtlama**:

NOT NULL: Bildirim konusu belirtilmek zorunda.

body

**Kısıtlama**:

NOT NULL: Bildirim içeriği belirtilmek zorunda.

send\_date

**Kısıtlama**:

NOT NULL: Gönderim tarihi belirtilmek zorunda.

status

**Kısıtlama**:

DEFAULT "Beklemede": Bildirim durumu varsayılan olarak beklemede olur.

notification\_type

Bildirim tipi bilgisi.

**Users Tablosu**

Sistem kullanıcı bilgilerini saklar.

**Sütunlar ve Kısıtlamalar**

user\_id

**Kısıtlama**:

PRIMARY KEY: Her kullanıcı için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

username

**Kısıtlama**:

NOT NULL: Kullanıcı adı belirtilmek zorunda.

UNIQUE: Her kullanıcı için benzersiz olmalı.

password

**Kısıtlama**:

NOT NULL: Şifre belirtilmek zorunda.

user\_type\_id

**Kısıtlama**:

FOREIGN KEY: UserTypes tablosundaki user\_type\_id sütununa referans verir.

ON DELETE SET NULL: Kullanıcı tipi kaydı silindiğinde bu alan NULL olur.

email

Kullanıcı e-posta adresi.

created\_at

**Kısıtlama**:

DEFAULT GETDATE(): Kullanıcı oluşturma tarihi varsayılan olarak sistem tarihidir.

**UserTypes Tablosu**

Kullanıcı türlerini saklar.

**Sütunlar** **ve** **Kısıtlamalar**

user\_type\_id

**Kısıtlama**:

PRIMARY KEY: Her kullanıcı türü için benzersiz kimlik.

IDENTITY: Otomatik artan bir sayı.

user\_type\_name

**Kısıtlama**:

NOT NULL: Kullanıcı türü adı belirtilmek zorunda.

**PaymentDetails Tablosu**

Ödemelere ait detaylı bilgileri saklar.

**Sütunlar ve Kısıtlamalar**

* **payment\_detail\_id**
  + **Kısıtlama:**
    - PRIMARY KEY: Her ödeme detayı için benzersiz bir kimlik.
    - IDENTITY: Otomatik artan bir sayı.
* **payment\_id**
  + **Kısıtlama:**
    - FOREIGN KEY: Payments tablosundaki payment\_id sütununa referans verir.
    - Ödeme detayının hangi ödeme kaydına ait olduğunu belirtir.
* **payment\_date**
  + **Kısıtlama:**
    - DEFAULT: Varsayılan olarak mevcut tarih (GETDATE()) atanır.
    - Ödemenin yapıldığı tarih.
* **paid\_amount**
  + Ödenen tutar.
* **payment\_method**
  + Ödeme yöntemi. Örneğin: Nakit, kredi kartı.
* **description**
  + İsteğe bağlı bir açıklama içerir.

**Suggestions Tablosu**

Tamamlanan bir kursa göre önerilen kurs bilgilerini saklar.

**Sütunlar ve Kısıtlamalar**

* **id**
  + **Kısıtlama:**
    - PRIMARY KEY: Her öneri için benzersiz bir kimlik.
    - IDENTITY: Otomatik artan bir sayı.
* **finished\_course\_id**
  + **Kısıtlama:**
    - FOREIGN KEY: Courses tablosundaki course\_id sütununa referans verir.
    - Önerinin hangi tamamlanan kursa ait olduğunu belirtir.
* **suggested\_course\_id**
  + **Kısıtlama:**
    - FOREIGN KEY: Courses tablosundaki course\_id sütununa referans verir.
    - Hangi kursun önerildiğini belirtir.
* **mail\_sent**
  + **Kısıtlama:**
    - DEFAULT: Varsayılan olarak 0 (false) atanır.
    - Öneri mailinin gönderilip gönderilmediğini belirtir.
* **created\_at**
  + **Kısıtlama:**
    - DEFAULT: Varsayılan olarak mevcut tarih (GETDATE()) atanır.
    - Önerinin oluşturulduğu zaman.

**StudentSuggestions Tablosu**

Öğrencilere yönelik bireysel kurs önerilerini saklar.

**Sütunlar ve Kısıtlamalar**

* **id**
  + **Kısıtlama:**
    - PRIMARY KEY: Her öneri kaydı için benzersiz bir kimlik.
    - IDENTITY: Otomatik artan bir sayı.
* **student\_id**
  + **Kısıtlama:**
    - FOREIGN KEY: Students tablosundaki student\_id sütununa referans verir.
    - Hangi öğrenciye öneride bulunulduğunu belirtir.
* **student\_name**
  + **Kısıtlama:**
    - NOT NULL: Öğrenci adı belirtilmek zorunda.
    - Görsel amaçlarla kullanılır.
* **finished\_course\_id**
  + **Kısıtlama:**
    - FOREIGN KEY: Courses tablosundaki course\_id sütununa referans verir.
    - Öğrencinin tamamladığı kursu belirtir.
* **suggested\_course\_id**
  + **Kısıtlama:**
    - FOREIGN KEY: Courses tablosundaki course\_id sütununa referans verir.
    - Önerilen kursu belirtir.
* **suggestion\_id**
  + **Kısıtlama:**
    - FOREIGN KEY: Suggestions tablosundaki id sütununa referans verir.
    - Önerinin bağlı olduğu genel öneriyi belirtir.
* **created\_at**
  + **Kısıtlama:**
    - DEFAULT: Varsayılan olarak mevcut tarih (GETDATE()) atanır.
    - Önerinin oluşturulduğu zaman.

# 4. Sorgular

Projemizde sorgular, Microsoft SQL Server kullanılarak gerçekleştirildi. Veritabanına bağlanmak ve sorguları çalıştırmak için bir Connection String tanımladık. SQL Server üzerindeki sorgular, genellikle T-SQL (Transact-SQL) kullanılarak yazıldı. Projede veri erişimi ve yönetimi için aşağıdaki adımları izledik:

1. Connection String Kullanımı:

* Uygulamamız, SQL Server ile bağlantıyı sağlamak için bir Connection String yapılandırması kullandı. Bu yapılandırma, sunucu adresi, veritabanı adı, kimlik doğrulama bilgileri gibi gerekli detayları içeriyordu.

2. T-SQL Sorguları:

* Veritabanı işlemleri, T-SQL kullanılarak gerçekleştirildi. Bu sorgular; tablo oluşturma, veri ekleme (INSERT), güncelleme (UPDATE), silme (DELETE) ve veri sorgulama (SELECT) işlemlerini içerdi.
* Karmaşık işlemler için saklı yordamların (Stored Procedure) icerisinde JOIN ve CASE gibi islemler kullanildi.

1. **Birden Fazla Kursa Kayıtlı Öğrenciler**

metin, yazı tipi, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Tanım**

Bu sorgu, öğrencilere ait kurs kayıt bilgilerini listeleyerek, her öğrencinin toplam kaç kursa kayıtlı olduğunu döndürür. Sorgu, yalnızca birden fazla kursa kayıtlı olan öğrencileri içerecek şekilde filtrelenmiştir.

**Kullanım**

Bu sorgu herhangi bir parametre almaz. Çalıştırıldığında, birden fazla kursa kayıt yaptırmış öğrencilerin isimlerini ve toplam kurs sayılarını döndürür. Eğitim kurumları, öğrencilerin kurs dağılımını analiz etmek veya özel kampanyalar düzenlemek için bu sorguyu kullanabilir.

**Açıklamalar**

1. **INNER JOIN**:  
   Students tablosu ile Enrollments tablosu, student\_id sütunu üzerinden birleştirilmiştir. Bu işlem, öğrencilerin hangi kurslara kayıtlı olduğunu belirlemek için yapılmıştır.
2. **CONCAT**:  
   Öğrencinin adı ve soyadı, first\_name + ' ' + last\_name ifadesiyle birleştirilmiştir. Bu, sonuçların daha okunabilir bir şekilde gösterilmesini sağlar.
3. **COUNT**:  
   COUNT(e.course\_id) ifadesi, her bir öğrencinin kayıtlı olduğu kursların toplam sayısını hesaplar.
4. **GROUP BY**:  
   Öğrencilerin ad ve soyadlarına göre gruplama yapılmıştır. Böylece her öğrenci için bir satır döndürülür ve kurs sayıları hesaplanabilir.
5. **HAVING**:  
   HAVING COUNT(e.course\_id) > 1 ifadesiyle yalnızca birden fazla kursa kayıtlı öğrenciler filtrelenmiştir. Tek kursa kayıtlı öğrenciler sorgu sonucuna dahil edilmez.

2. **Öğrencinin tüm kurslara olan toplam borcunu görmek ve detayları (ödenen kısmı, toplam borç, mail bilgisi) almak için sorgu** metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Tanım**

Bu sorgu, öğrencilerin ödeme bilgilerini ve velileriyle ilgili detayları listeleyen bir analiz sorgusudur. Her öğrenci için toplam borç, ödenen tutar ve kalan borç bilgileri gösterilir. Sorgu yalnızca borcu kalan öğrencileri sonuçlara dahil eder.

**Kullanım**

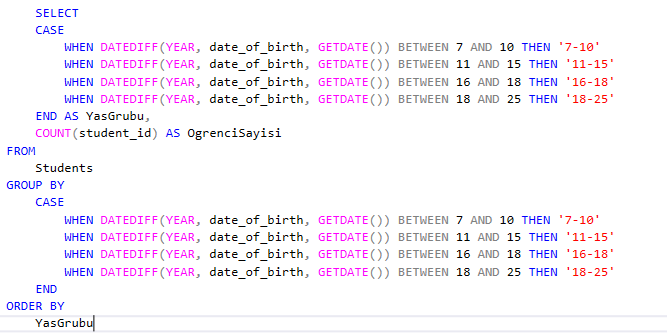
Bu sorgu herhangi bir parametre almaz. Çalıştırıldığında:

* Öğrenci adları, e-posta bilgileri,
* Velilerin isimleri ve iletişim bilgileri,
* Toplam borç, ödenen tutar ve kalan borç gibi finansal verileri döndürür.  
  Eğitim kurumları, öğrenci borç durumlarını ve veli iletişim bilgilerini hızlı bir şekilde görmek için bu sorguyu kullanabilir.

**Açıklamalar**

1. **LEFT JOIN**:
   * Students tablosu ile Payments ve Guardians tabloları birleştirilmiştir:
     + Payments tablosu, öğrencinin ödeme bilgilerini,
     + Guardians tablosu, velilerin adını ve telefon numarasını sağlar.
   * LEFT JOIN, ödeme kaydı ya da veli bilgisi olmayan öğrencilerin de listelenmesini sağlar.
2. **SUM**:
   * SUM(p.amount): Her öğrenci için toplam borcu hesaplar.
   * SUM(p.total\_paid): Öğrenci tarafından ödenmiş toplam tutarı hesaplar.
   * (SUM(p.amount) - SUM(p.total\_paid)): Kalan borç, toplam borçtan ödenen tutarın çıkarılmasıyla hesaplanır.
3. **GROUP BY**:
   * GROUP BY ile öğrenciler ve velilere ait benzersiz bilgiler gruplandırılır.
   * Bu sayede, bir öğrenciye ait ödeme bilgileri tek bir satırda gösterilir.
4. **HAVING**:
   * HAVING SUM(p.amount) > SUM(p.total\_paid) ile yalnızca borcu kalan öğrenciler filtrelenir.

**3. Öğrenci yaş dağılımı**



**Tanım**

Bu sorgu, öğrencilerin yaş aralıklarına göre gruplandırılmış istatistiklerini döndürür. Öğrenciler dört yaş grubuna ayrılır ve her yaş grubundaki öğrenci sayısı liste halinde sunulur.

**Kullanım**

Bu sorgu herhangi bir parametre almaz. Çalıştırıldığında:

* Öğrenciler, yaşlarına göre belirlenen gruplara atanır.
* Her grup için kaç öğrencinin o yaş aralığında olduğu hesaplanır ve sıralı bir liste döndürülür.

Bu sorgu, bir okul veya akademinin yaşa dayalı öğrenci dağılımını analiz etmek için kullanılabilir.

**Açıklamalar**

1. **CASE IFADESİ**:
   * CASE ifadesi, öğrencilerin date\_of\_birth (doğum tarihi) sütununu kullanarak yaş hesaplaması yapar.
   * DATEDIFF (YEAR, date\_of\_birth, GETDATE()): Doğum tarihinden bugüne kadar geçen yılı hesaplar.
   * Yaş aralıkları şu şekilde gruplandırılmıştır:
     + 7-10 yaş grubu: 7 ila 10 yaşındaki öğrenciler.
     + 11-15 yaş grubu: 11 ila 15 yaşındaki öğrenciler.
     + 16-18 yaş grubu: 16 ila 18 yaşındaki öğrenciler.
     + 18-25 yaş grubu: 18 ila 25 yaşındaki öğrenciler.
2. **GROUP BY**:
   * CASE ifadesi kullanılarak oluşturulan yaş grubu, GROUP BY ile gruplandırılır.
   * Aynı yaş grubundaki öğrenciler bir araya getirilerek toplam öğrenci sayısı hesaplanır.
3. **COUNT**:
   * Her yaş grubu için öğrenci sayısı COUNT(student\_id) ile belirlenir.
4. **ORDER BY**:
   * Yaş grupları, YasGrubu sütununa göre artan sırada sıralanır.

4-**Her eğitmenin yönettiği kurslar ve öğrenci sayısını analiz eden sorgu**

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Tanım**

Bu sorgu, eğitmenlerin kurs ve öğrenci dağılımını analiz eder. Her eğitmen için:

* Kaç kursa atanmış olduğu (KursSayisi)
* Kurslarında toplam kaç öğrencinin kayıtlı olduğu (ToplamOgrenciSayisi)  
  bilgilerini döndürür.

**Kullanım**

Bu sorgu, eğitmenlerin performansını ve kurslara katkısını görmek için kullanılabilir. Çalıştırıldığında:

* Eğitmen adları (Egitmen), kurs sayıları ve toplam öğrenci sayıları bir liste olarak döner.
* LEFT JOIN yapısı sayesinde, herhangi bir kursa atanmasa bile eğitmen bilgileri sonuçta yer alır.

**Açıklamalar**

1. **LEFT JOIN**:
   * Instructors ve InstructorAssignments tablosu LEFT JOIN ile birleştirilmiştir. Bu, her eğitmeni ve onun atanmış olduğu kursları listeler.
   * InstructorAssignments ve Enrollments tablosu da LEFT JOIN ile birleştirilmiştir. Bu, her kursa kayıtlı öğrencileri içerir.
   * Bu yapılar sayesinde, atanmış kursu veya öğrencisi olmayan eğitmenler de sonuçta görünür.
2. **COUNT**:
   * COUNT(ia.course\_id): Her eğitmenin atanmış olduğu kurs sayısını hesaplar.
3. **CASE WHEN**:
   * SUM(CASE WHEN e.student\_id IS NOT NULL THEN 1 ELSE 0 END): Öğrencisi olan her kurs için, kayıtlı öğrenci sayısını toplar. Eğer bir kursta öğrenci yoksa 0 olarak kabul edilir.
4. **GROUP BY**:
   * i.name: Eğitmen adına göre gruplandırılı

# 5. VTYS Araç Kullanımı

**Kurs Öneri Trigger Tetikleyicisi**

**metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

Bu trigger, **Enrollments** tablosunda bir **INSERT** işlemi gerçekleştirildiğinde tetiklenir. Amaç, yeni eklenen bir kaydın kursu tamamlamış bir öğrenci için önerilen kurslarla ilişkisini kontrol etmek ve **StudentSuggestions** tablosuna uygun kayıtları eklemektir.

**Mantık ve İşleyiş**

1. **Trigger'ın Oluşturulması**
   * Eğer belirtilen isimde bir trigger daha önce oluşturulmamışsa (IF OBJECT\_ID('{triggerName}', 'TR') IS NULL), yeni bir trigger oluşturulur.
   * Trigger ismi dinamik olarak ({triggerName}) atanır.
2. **INSERT İşlemi Üzerine Çalışma**
   * **Enrollments** tablosuna yeni bir kayıt eklendiğinde tetiklenir.
3. **Yeni Eklenen Verilerin Alınması**
   * inserted sanal tablosu kullanılarak, yeni eklenen student\_id ve course\_id değerleri alınır.
4. **Koşul Kontrolü**
   * Eğer öğrenci, eklenen kursu tamamladı durumunda (completion\_status = 'tamamlandi') kayıtlı ise, işlem devam eder.
   * **StudentSuggestions Tablosuna Kayıt Eklenmesi**
   * Yeni öğrenci ve tamamlanan kursla ilişkili önerilen kurslar için, Suggestions tablosuna bakılarak kayıtlar eklenir.

**Tetikleyici: trg\_DeleteRelatedEnrollments**

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Amaç:**

Bu tetikleyici, Courses tablosundan bir kayıt silindiğinde, o kursa ilişkin tüm Enrollments (kayıtlar) tablosundaki kayıtları da otomatik olarak siler. Bu, veritabanında ilişkili verilerin tutarlılığını sağlamak için kullanılır.

**Tetikleyicinin Çalışma Şekli:**

1. **Tetikleyici Türü:** AFTER DELETE
   * Bu tetikleyici, Courses tablosunda bir kayıt silindiğinde **silme işleminden sonra** çalışır. Yani, kurs kaydının silinmesinden sonra ilişkili Enrollments kayıtları silinir.
2. **Tetikleyicinin İçeriği:**
   * **DELETE FROM Enrollments**: Enrollments tablosundan, silinen kursla ilişkili olan tüm kayıtları siler.
   * WHERE course\_id IN (SELECT course\_id FROM deleted):
     + deleted: Bu sanal tablo, silinen kurs kaydını temsil eder. Tetikleyici çalıştırıldığında, deleted tablosunda silinen kursun bilgileri bulunur.
     + SELECT course\_id FROM deleted: deleted tablosundaki silinen kursun course\_id değerini alır.
     + WHERE course\_id IN (...): Enrollments tablosundaki kurs kayıtları, silinen kursun course\_id ile eşleşiyorsa silinir.
3. **Mesaj Yazdırma:**
   * **PRINT 'Silinen kursa ait tüm kayıtlar temizlendi.'**: Bu satır, tetikleyici çalıştığında ekrana bilgilendirme mesajı yazdırır. Ancak, bu sadece SQL Server Management Studio gibi araçlarda görünür. Uygulama üzerinden erişilemez.

**Tetikleyicinin Faydaları:**

* Bu tetikleyici, Courses tablosundan bir kurs silindiğinde, o kursa ilişkin tüm öğrenci kayıtlarının (enrollments) veritabanında kalmamasını sağlar. Böylece, silinen bir kursa ait veriler tutarsızlık yaratmadan temizlenir.
* Veritabanı tutarlılığı sağlanır çünkü silinen bir kursa ait tüm ilişkili kayıtlar (Enrollments) otomatik olarak silinir.

**sp\_OgrenciListele Prosedürü**

metin, ekran görüntüsü, yazılım, web sayfası içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Tanım:** sp\_OgrenciListele prosedürü, Students ve Guardians (veli) tablolarını birleştirerek, öğrenci bilgilerini listeleyen bir prosedürdür. Bu prosedür, öğrenciye ait temel bilgileri ve veli bilgilerini getirir.

**Kullanım:** Bu prosedür herhangi bir parametre almaz. Sadece çalıştırıldığında tüm öğrencilerin bilgilerini döndürür.

Örnek Kullanım: EXEC sp\_OgrenciListele;  
  
  
**Açıklamalar:**

* LEFT JOIN ile Students tablosu ile Guardians tablosu birleştirilmiştir. Bu birleştirme, her öğrencinin velisi ile ilgili bilgileri almayı sağlar.
* Öğrenci bilgileri arasında öğrenci ID'si, adı, soyadı, iletişim bilgileri, cinsiyet, doğum tarihi, okul adı, seviye ve durumu gibi bilgiler yer almaktadır.
* Ayrıca, öğrencinin kodlama geçmişi, veli adı ve iletişim bilgileri de seçilen alanlar arasındadır.
* CASE ifadesi ile coding\_experience sütunundaki değere göre öğrenciye ait "Kodlama Geçmişi" bilgisi 'Evet' ya da 'Hayır' olarak döndürülür.

**sp\_KullanıcılarıListele Saklı yordamı**

**metin, elektronik donanım, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Amaç:**Bu prosedür, tüm kullanıcıların bilgilerini, kullanıcı tipleriyle ilişkilendirerek listeler.

**Sütunlar ve Kısıtlamalar:**

* **Users Tablosu:**
  + **user\_id:** Kullanıcı kimliği.
  + **username:** Kullanıcı adı.
  + **email:** Kullanıcı e-posta adresi.
  + **user\_type\_id:** Kullanıcı tipi kimliği (**UserTypes** tablosuna bağlanır).
* **UserTypes Tablosu:**
  + **user\_type\_id:** Kullanıcı tipi kimliği.
  + **user\_type\_name:** Kullanıcı tipi adı (ör. "Admin", "Standart Kullanıcı").

**Mantık:**

* Kullanıcıların kimlik, kullanıcı adı ve e-posta bilgileri listelenir.
* Kullanıcı tipleri varsa, kullanıcı tipinin adı da sonuçlara dahil edilir.
* Kullanıcı tipi bilgisi eksikse (LEFT JOIN), NULL döner.

**sp\_AddStudentAndGuardian Saklı yordamı**

**metin, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Amaç:**Yeni bir öğrenci ve veli bilgilerini aynı anda ekler.

**Parametreler:**

1. **Öğrenci Bilgileri:**
   * @first\_name, @last\_name, @date\_of\_birth, @contact\_number, @email, @gender, @level, @school, @coding\_experience, @status, @enrollment\_date
2. **Veli Bilgileri:**
   * @guardian\_name, @guardian\_contact, @guardian\_email

**Mantık:**

1. Öğrenci bilgileri **Students** tablosuna eklenir.
   * Öğrenci ID'si, otomatik artışlı birincil anahtardır ve SCOPE\_IDENTITY() ile alınır.
2. Veli bilgileri, eklenen öğrencinin ID'siyle birlikte **Guardians** tablosuna eklenir.
3. Tüm işlem bir **TRANSACTION** içinde gerçekleştirilir:
   * Eğer bir hata olursa, işlemin tamamı geri alınır (**ROLLBACK TRANSACTION**).

Başarılı olursa, tüm veriler kaydedilir (**COMMIT TRANSACTION**).

**sp\_UpdateStudentAndGuardian Saklı yordamı**

**metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Amaç:**Bir öğrenci ve velisinin mevcut bilgilerini günceller.

**Sütunlar ve Kısıtlamalar**

* **Students Tablosu:**
  + Güncellenecek sütunlar: first\_name, last\_name, date\_of\_birth, contact\_number, email, gender, level, school, coding\_experience, status.
* **Guardians Tablosu:**
  + Güncellenecek sütunlar: name, contact\_number, email.

**Mantık:**

1. Öğrencinin ID'si ile öğrencinin temel bilgileri güncellenir.
2. Aynı ID'ye sahip veli bilgileri güncellenir.
3. Tüm işlem bir **TRANSACTION** ile gerçekleştirilir:
   * Başarılı olursa işlemler kaydedilir (**COMMIT TRANSACTION**).

* Hata durumunda, tüm işlemler geri alınır (**ROLLBACK TRANSACTION**).

**sp\_SearchStudents Saklı yordamı**

**metin, ekran görüntüsü, yazılım, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Amaç:**Belirli bir metne göre öğrenci ve veliler arasında arama yapar.

**Sütunlar ve Kısıtlamalar**

* **Students ve Guardians Tablosu:**
  + Öğrenci bilgileri ve veli bilgileri arasında verilen arama metnine uyan kayıtlar döndürülür.
  + Arama kriteri: first\_name veya last\_name içinde geçen metinler.

**Mantık:**

* Giriş parametresi (@searchText) ile ad veya soyadında belirtilen metni içeren kayıtlar sorgulanır.
* Sorgu, öğrenci ve veli bilgilerini birleştirerek (LEFT JOIN) tam bir sonuç döndürür.

**sp\_AddInstructor Saklı yordamı**

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Amaç:**Yeni bir eğitmen ekler.

**Sütunlar ve Kısıtlamalar**

* **Instructors Tablosu:**
  + Eklenen sütunlar: name, contact\_number, email.

**Mantık:**

* Eğitmenin adı, telefon numarası ve e-posta adresi tabloya eklenir.

**sp\_GetInstructors Saklı yordamı**

**metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Amaç:**Tüm eğitmenlerin bilgilerini listeler.

**Sütunlar ve Kısıtlamalar**

* **Instructors Tablosu:**
  + Döndürülen sütunlar: name, contact\_number, email.

**Mantık:**

* Eğitmen bilgileri tablo adlarıyla birlikte sorgulanır ve döndürülür.

**sp\_AddCourse Saklı yordamı**

**metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Amaç:**Yeni bir kurs ekler.

**Sütunlar ve Kısıtlamalar**

* **Courses Tablosu:**
  + Eklenen sütunlar: course\_name, course\_type, course\_level, description, level\_id, duration, course\_date, MaxCapacity.

**Mantık:**

* Kullanıcıdan gelen kurs detayları **Courses** tablosuna eklenir.
* Parametreler:
  + @course\_name: Kursun adı.
  + @course\_type: Kurs türü.
  + @course\_level: Kurs seviyesi.
  + @description: Kurs açıklaması.
  + @level\_id: Seviyeyi belirleyen ID.
  + @duration: Kurs süresi (ör. saat cinsinden).
  + @course\_date: Kursun başlangıç tarihi.
  + @max\_capacity: Maksimum katılımcı sayısı.

**sp\_UpdateCourse Saklı yordamı**

**metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Amaç:**Mevcut bir kursun bilgilerini günceller.

**Sütunlar ve Kısıtlamalar**

* **Courses Tablosu:**
  + Güncellenecek sütunlar: course\_name, course\_type, course\_level, description, level\_id, duration, course\_date, MaxCapacity.

**Mantık:**

* Kullanıcıdan alınan parametreler doğrultusunda belirtilen kurs (@course\_id) güncellenir.
  + Parametreler: @course\_id: Güncellenmesi gereken kursun ID'si.

**deleteEmailNotifications Saklı Yordamı**

metin, elektronik donanım, ekran görüntüsü, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Prosedürün Açıklaması:**

1. **Parametre:**
   * @EmailNotificationId: Silinecek olan e-posta bildirim kaydının ID'si.
2. **DELETE İşlemi:**
   * EmailNotifications tablosunda belirtilen email\_notification\_id'ye sahip kaydın silinmesi sağlanır.
3. **Hata Yönetimi:**
   * TRY-CATCH bloğu kullanılarak, bir hata meydana gelirse hata numarası ve mesajı döndürülür.
4. **Başarılı İşlem Durumu:**
   * Silme işlemi başarılı olursa, başarı mesajı döndürülür.

**Try-Catch Bloklarının Kullanım Amacı, Tanımı ve Uygulama**

**Kullanım Amacı**

try-catch blokları, hataların yönetilmesi ve programın kesintisiz çalışmaya devam etmesi için kullanılır. Bu bloklar, hata oluşabilecek kodların bulunduğu alanları (try) ve bu hataların nasıl ele alınacağını (catch) belirler. Kullanıcı deneyimini olumsuz etkilemeden hataların düzgün bir şekilde raporlanmasını sağlar.

metin, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Tanımı**

* **try bloğu**: Hata alması olası olan kod bloğunu içerir. Bu blok, hata meydana gelirse kontrolü catch bloğuna geçirir.
* **catch bloğu**: try bloğunda oluşan hataları yakalar ve bunlarla ilgili gerekli işlemleri yapar. Bu sayede hata mesajları kullanıcıya bildirilebilir ve uygulama çökmeden devam edebilir.

**Örnek Açıklaması:**

1. **Başlangıç Kontrolleri**:
   * İlk olarak, dataGridView1.SelectedRows.Count ile bir satırın seçilip seçilmediği kontrol edilir. Eğer seçilmemişse, bir hata mesajı gösterilir.
   * E-posta formatı kontrol edilir. Eğer geçersizse, kullanıcıya uygun bir hata mesajı verilir.
2. **Veritabanı İşlemleri**:
   * Veritabanına bağlanmak için SqlConnection kullanılır.
   * SQL komutu çalıştırılmadan önce, gerekli parametreler cmd.Parameters.AddWithValue ile eklenir. Bu parametreler, kullanıcının güncellemek istediği bilgilerle doldurulur.
3. **Hata Yönetimi (Try-Catch)**:
   * Eğer conn.Open() veya cmd.ExecuteNonQuery() gibi işlemler sırasında bir hata oluşursa, bu hata catch bloğunda yakalanır ve ekrana "Hata: [hata mesajı]" şeklinde gösterilir.
   * Hata oluşmazsa, kullanıcıya başarı mesajı gösterilir ve liste güncellenir.

metin, ekran görüntüsü, yazılım, işletim sistemi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Tanımı**

* **try bloğu**: Veritabanına bağlantı açılır, öğrencinin ID'si alınır ve bir stored procedure çalıştırılır. Öğrenci ve veli bilgilerini silmek için gerekli SQL komutları burada yer alır.
* **catch bloğu**: Hata oluşması durumunda, hata mesajı kullanıcıya gösterilir ve işlem durdurulur.
* **finally bloğu**: Bağlantı durumu kontrol edilir ve açık olan bağlantı kapatılır.

**Örnek Açıklaması:**

1. **Veritabanı Bağlantısı ve Hata Yönetimi**:
   * baglanti.Open() ile veritabanı bağlantısı açılır.
   * Seçilen satırdaki öğrenci ID'si alınır ve parametre olarak stored procedure'e gönderilir.
2. **Hata Yönetimi (Try-Catch-Finally)**:
   * Hata oluşursa, catch bloğu devreye girer ve kullanıcıya hata mesajı gösterilir.

# 6. Kullanıcı Arayüzü (User Interface)

metin, ekran görüntüsü, logo, marka içeren bir resim

Açıklama otomatik olarak oluşturuldu

Login sayfasında database’de yer alan kullanıcılar kontrol edilerek giriş, Google ile giriş, şifremi unuttum kısımları yer almaktadır.   
  
A screenshot of a black screen

Description automatically generatedGoogle ile giriş entegrasyonundan örnek görüntü.   
  
metin, ekran görüntüsü, diyagram, yazılım içeren bir resim

Açıklama otomatik olarak oluşturuldu

Ana ekran sayfasında kullanıcıların ilk görmek isteyeceği çeşitli bilgiler, Instagram ve Web Sitesine kolay erişim tuşu ve veri yedeklemek için tüm Excel dosyalarının export edildiği bir tuş yer almaktadır.  
  
metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Excel tuşunun tüm database’i export etmesine bir örnek görsel.  
A screenshot of a computer

Description automatically generated

Kullanıcı yönetim sayfası sadece admin’in erişimiyle düzenlenmektedir ve kullanıcının türüne göre kullanıcılar atanmaktadır, veritabanına kaydedilmektedir.  
  
A screenshot of a computer

Description automatically generated  
A screenshot of a computer

Description automatically generated  
Öğrenci yönetim sayfasında, veritabanındaki öğrenciler ve onların velileri için ekleme silme güncelleme işlemleri, filtreleme tuşları yer almaktadır. Ayrıca bu bilgiler için ayrı Excel’e export ve Excel’den import işlemi gerçekleştirilebilmektedir.  
A screenshot of a computer

Description automatically generated

Eğitmen yönetim sayfasında, akademide yer alan eğitmenler için ekleme silme güncelleme işlemleri gerçekleştirilmektedir.  
  
A screenshot of a computer

Description automatically generated  
A screenshot of a computer

Description automatically generated  
A computer screen shot of a computer

Description automatically generated  
Kurs yönetim panelinde, yeni kurs oluşturulması, eğitmenlerin ve öğrencilerin kurslara atanması, ayrıca öğrenciler kurslara atanırken ödeme bilgilerinin alınması işlemleri gerçekleştirilmektedir.  
  
A screenshot of a computer

Description automatically generated  
  
A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated  
  
Ödeme yönetim panelinde, ödeme yapmamış öğrencilerin listelenmesi, ödeme yapan öğrencilerin ödeme kayıtlarının listelenmesi gerçekleştirilmektedir. Ayrıca ödeme yapmamış öğrencilere ödeme eklenerek listeden otomatik silinmesi, kalan tutar taksit sayısı gibi bilgilerin takibi gerçekleştirilmektedir. Buna ek olarak dekontlu ödeme yapan öğrencilerin dekont pdf’leri tutularak eski dekontlar görüntülenmektedir. Uygulama üzerinden dosya yoluna tıklanarak dekontların kendisine erişilebilmektedir.  
  
  
  
metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu  
A screenshot of a computer

Description automatically generated  
  
Bilgilendirme yönetim panelinde, sisteme kayıtlı öğrenci ve velilere mail gönderimi otomatik olarak sağlanmaktadır. Bu işlem kolaylaştırılması için öğrenci ve veliye mail gönderimi ayrılmıştır. Tek tuşla tüm öğrencilere veya velilere veya ikisine birden mail atılması mümkündür. Ayrıca eski emaillerin kayıtları da veritabanında saklanmaktadır.

metin, ekran görüntüsü, yazılım, bilgisayar simgesi içeren bir resim

Açıklama otomatik olarak oluşturuldu

Kurs Önerme sisteminde Sol tarafında yer alan Datagridview içinde oluşturduğumuz öneriler yer almaktadır. Bitirilecek kursa önerilen kurs ataması yapılır, combobox içinde seçili öneriye uyan bütün öğrenciler öğrenci numaraları ile listelenmektedir

# 7- Diğer Entegrasyonlar

**GOOGLE LOGİN ENTEGRASYONU**

Google ile giriş yapılabilmesi için Google API kullanılmıştır. Bu işlem aşağıdaki adımları içerir:

**1.1 Google API Credentials Oluşturma**

* Google Cloud Console'da bir proje oluşturulur.
* "OAuth 2.0 Client ID" oluşturularak **Client ID** ve **Client Secret** alınır.
* Gerekli yetkilendirme kapsamları ("email", "profile") belirtilir.

**1.2 Yetkilendirme Kodları**

* **GoogleWebAuthorizationBroker.AuthorizeAsync** metodu kullanılarak Google hesabına erişim sağlanır.
* Kullanıcı giriş yaptıktan sonra, **IdToken** kullanılarak doğrulama yapılır.
* **GoogleJsonWebSignature.ValidateAsync** metodu ile kullanıcının kimlik bilgileri alınır:
  + Kullanıcının e-posta adresi (payload.Email)
  + Adı ve soyadı (payload.Name)
* **Kodun Çalışma Mantığı**
* **Yetkilendirme ve Kullanıcı Bilgileri Alma**
* Aşağıdaki kod bloğu, kullanıcının Google hesabı ile giriş yapmasını sağlar:

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Bu kod, kullanıcının kimlik bilgilerini doğrular ve aşağıdaki verileri alır:**

Kullanıcı e-posta adresi (Email)

Kullanıcının adı ve profil bilgileri

**Veritabanında Kullanıcı Kontrolü**

Kullanıcı giriş yaptıktan sonra, alınan e-posta adresi ile uygulamanın veritabanında kayıtlı bir kullanıcı olup olmadığı kontrol edilir:

* **GetUserFromDatabase** metodu, kullanıcının veritabanında kayıtlı olup olmadığını kontrol eder.
* Eğer kullanıcı kayıtlıysa, oturum açılır. Kayıtlı değilse, kullanıcıya kayıt olması gerektiği bilgisi verilir.

**2.3 Yeni Kullanıcı Ekleme (Opsiyonel)**

Eğer kullanıcının e-posta adresi veritabanında bulunmuyorsa, bir kayıt ekranına yönlendirilerek yeni kullanıcı eklenebilir.

**3. Önemli Güvenlik Önlemleri**

* **Client Secret ve API Key Güvenliği:**
  + clientSecret gibi kritik bilgilerin kod içinde saklanmaması gerekir. Bu tür bilgiler genellikle bir **config dosyasında** veya çevre değişkenlerinde tutulmalıdır.
* **Token Doğrulama:**
  + Kullanıcı giriş yaptıktan sonra Google tarafından dönen **IdToken**, kesinlikle doğrulanmalıdır. Bu işlem, GoogleJsonWebSignature.ValidateAsync metodu ile yapılır.
* **SSL Kullanımı:**
  + Uygulamanın internet üzerinden çalıştığı durumlarda, verilerin güvenli bir şekilde aktarılması için SSL kullanımı zorunludur.

**4. Google Login İş Akışı**

1. Kullanıcı, Google ile giriş yapmak için butona tıklar.
2. Google hesabı ile giriş yapması için bir tarayıcı penceresi açılır.
3. Kullanıcı giriş yaptıktan sonra Google, bir **IdToken** döndürür.
4. Uygulama, bu **IdToken**'ı doğrular ve kullanıcının e-posta adresini alır.
5. Veritabanında kullanıcı kontrolü yapılır:
   * Eğer kullanıcı kayıtlıysa, giriş yapılır.
   * Eğer kullanıcı kayıtlı değilse, kayıt olması için yönlendirilir.

**Email Entegrasyonu**

**Mail Entegrasyonu Kullanım Amaçları ve Açıklamalar**

Gönderilen kod parçasında, bir yazılım uygulaması içinde **öğrenciler ve velilerle iletişim kurmak** amacıyla email entegrasyonu gerçekleştirilmiştir. Bu entegrasyon, uygulamanın **öğrenci ve veli yönetimi**, **bilgilendirme mesajlarının iletimi**, ve **veri takibi** gibi süreçlerini kolaylaştırmayı hedefler.

**Kullanım Amaçları**

1. **Bilgilendirme Mesajları Gönderme:**
   * Öğrenciler ve velilere duyurular, hatırlatmalar veya özel mesajlar gönderilebilir.
   * Kullanıcılar, mesajları manuel olarak yazıp, seçilen öğrenci veya velilere ulaştırabilir.
2. **E-posta Verilerinin Takibi:**
   * Gönderilen maillerin konu, içerik ve gönderim tarihine göre **veritabanında saklanması** sağlanır.
   * Hangi öğrenci veya veliye mail gönderildiği kaydedilir.
3. **Toplu ve Seçimli Gönderim:**
   * Kullanıcı, belirli kişilere veya tüm öğrencilere/velilere mail gönderimi yapabilir.
   * **Seçim yapılabilen bir arayüz** ile kitle yönetimi kolaylaştırılmıştır.
4. **Kayıtlı Verilerin Kullanımı:**
   * Öğrencilerin ve velilerin e-posta adresleri veritabanından alınarak entegre edilir.

**Kodun İşlevleri**

1. **SMTP Kullanarak E-posta Gönderimi:**
   * SendEmail metodu, Gmail SMTP hizmeti kullanılarak e-posta gönderir.
   * Kullanıcı adı ve şifre güvenliği sağlanmalıdır.
2. **Veritabanı Entegrasyonu:**
   * Gönderilen mailler EmailNotifications tablosuna kaydedilir.
   * Mail gönderiminde öğrenci veya veli kimlikleri e-posta adresine göre eşleştirilir.
3. **DataGridView ile Seçim:**
   * Kullanıcı, arayüzde bir liste üzerinden öğrenci ve velileri seçerek e-posta gönderebilir.
   * Seçilen satırlardan e-posta adresleri alınır ve mail gönderiminde kullanılır.
4. **UI (Kullanıcı Arayüzü):**
   * TextBox ile e-posta konusu ve içeriği manuel girilir.
   * Butonlar ile öğrencilere, velilere veya herkese mail gönderimi yapılabilir.

metin, ekran görüntüsü, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**E-posta Eşleştirme:**

* GetStudentIdByEmail ve GetGuardianIdByEmail metotları ile e-posta adreslerinden kimlik bilgileri bulunur.

**Uygulama Kullanımı**

1. Arayüzde yer alan DataGridView üzerinden öğrenci ve veli seçimi yapılır.
2. Konu ve içerik girildikten sonra ilgili butona basılarak mail gönderilir:
   * **Tüm öğrencilere:** Öğrenci e-posta adreslerini alır.
   * **Tüm velilere:** Veli e-posta adreslerini alır.
   * **Seçilenlere:** DataGridView’de seçilen satırlardaki adreslere gönderim yapar.
3. Gönderilen mailler, veritabanında saklanır ve gerektiğinde geçmişe dönük kontrol sağlanır.

**EXCEL İMPORT / EXPORT ENTEGRASYONU**

**Excel Dışa Aktarma (Export)**

metin, ekran görüntüsü, yazılım, multimedya yazılımı içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Açıklama:**

1. **Veritabanı Bağlantısı**: İlk olarak, SQL Server veritabanınıza bağlanmak için bir bağlantı dizesi belirlenir. Burada, Data Source ve Initial Catalog parametreleri, SQL Server bağlantı ayarlarını içerir.
2. **Veri Çekme**: SQL sorgusu ile veritabanından Students tablosundan veri çekilir. Bu veriler bir DataTable nesnesinde saklanır.
3. **Excel Dosyası Oluşturma**: ClosedXML kütüphanesi kullanarak bir yeni XLWorkbook oluşturulur ve veriler bu workbook'a bir çalışmaya eklenir (örnekte "Sheet1" olarak adlandırılır).
4. **Kayıtlı Bir Konumda Kaydetme**: Bir SaveFileDialog kullanılarak kullanıcıya bir dosya kaydetme yeri belirleme imkanı sunulur. Kullanıcı Excel dosyasının kaydedileceği konumu seçer ve kaydet düğmesine basarsa, workbook bu konumda saklanır ve bir başarı mesajı gösterilir.
5. **Hata Yönetimi**: Bir hata durumunda, hata mesajı gösterilir.

**IMPORT EXCEL AÇIKLAMA:**

metin, ekran görüntüsü, yazılım, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Açıklama:**

1. **Dosya Varlığı Kontrolü**: Öncelikle, belirtilen dosyanın var olup olmadığını kontrol edilir. Eğer dosya yoksa, işlem durdurulur.
2. **Workbook Yükleme**: Dosya başarıyla bulunursa, ClosedXML kütüphanesi ile Excel dosyası yüklenir.
3. **Çalışma Sayfası Seçimi**: Varsayılan olarak birinci sayfada bulunan veriler çekilir.
4. **Veri Aktarımı**: Her bir satır için Excel'den veriler okunur. Tarih formatları doğru bir şekilde parse edilir. Gerekli tüm hücre verileri string olarak alınır ve uygun SQL komutları ile SQL Server veritabanına kaydedilir.
5. **Öğrenci ve Veli Verileri Ekleme**: Her öğrenci için öğrenci bilgileri SQL Server'a insert edilir ve varsa veli detayları ayrı bir SQL insert komutu ile Guardians tablosuna eklenir.
6. **Hata Yönetimi**: Her işlemde hata olursa, uygun hata mesajı konsola yazdırılır ve işlem devam eder.

# 8- Çalışma Adam-saat değerleri

metin, ekran görüntüsü, diyagram, daire içeren bir resim

Açıklama otomatik olarak oluşturuldu

# 9. Sonuç ve Değerlendirme

**Projenin genel değerlendirmesi:** İlk aşamada uygulamanın gereksinimleri ve veri gereksinimleri tespit edilmiştir. Sıfırdan, ekip üyeleri tarafından veritabanı yapısı tasarlanmıştır. Frontend ve Backend geliştirmeye başlayana kadar olan süreçte gereksinimlerimiz ve veritabanı yapısı değişikliklere uğramıştır. Projeyi geliştirmeye başladıktan sonra, veritabanında çeşitli hatalar ve eksiklikler fark edilmiş ve düzeltilmiştir. Paydaş şirketin bizden isterlerinin çoğu gerçekleştirilmiştir. Projenin test edilmesi konusunda yoğun vakit harcayamadığımız için projenin tüm fonksiyonlarının sorunsuz çalışması taahhüt edilmemektedir. Projeyi geliştirirken, veritabanının yönetilmesi için yazılması gereken sorgular, triggerlar, saklı yordamları kullanma ve yazma konusunda fazlasıyla ekip üyelerine katkısı olduğu fark edilmiştir. Ayrıca projeyi geliştirirken çeşitli entegrasyonların veritabanı ile uyumlu çalışması için neler gerektiği öğrenilmiştir.

**Uygulamanın tam entegre tamamlanmasının değerlendirilmesi:** İlk başta kurs bitirmiş öğrencilerin tutulması gereken tablo kısmı, projeyi geliştirirken göz ardı edilmiştir. Sosyal medya (google) ile login, excel'e import export entegrasyonları başarıyla tamamlanmıştır. İlk başta planan SMS bilgilendirme kısmı, SMS gönderebilmek için ücretli denemeler yapılması gerektirdiği için kolay bir şekilde sms'e dönebilecek şekilde mail gönderme sistemine dönülmüştür. Ödeme ve mail gönderme kısımlarının önemli olduğu göz önüne alınmış ve başarıyla tamamlanmıştır.

**Uygulamanın alanda kullanılabilirlik oranı:** Herhangi bir akademi için, bu uygulamanın kullanılabilirlik oranı yüksektir. Öğrenci, veli, eğitmen, kurslar gibi verilerin detaylarıyla tutulabilmesi, muhasebe ve mail bilgilendirme kısımlarının uygulama üzerinden takip edilebilmesi herhangi bir akademi için kağıt üstünden yapılacak işleri büyük oranda dijitale dökmektedir.

**Vakit olsaydı planlanan çalışmalar:** Detaylı bir testing yapamadığımız için tüm fonksiyonların sorunsuz çalışması için detaylı testler gerçekleştirilecektir. Mail gönderme yerine SMS gönderme kısmına dönülmesi denenecektir. Ayrıca eski öğrencilerin de veritabanında tutulması eklenecektir. Kullanıcı arayüzü iyileştirilecektir. Trigger ve stored procedure kullanımı daha sıklaştırılarak performans iyileştirmesi yapılacaktır. Ayrıca (Varsa) güvenlik açıkları kapatılacaktır. Şirkette henüz kullanabileceğimiz bir server olmadığı için uygulama localde çalışmaktadır. Bunun server oluşturulması halinde localde değil de birden fazla kullanıcının aynı anda işlem yapabileceği ortam sağlanacaktır.

|  |  |
| --- | --- |
| **Özdeğerlendirme Başlığı** | **% xx**  **(0-100)** |
| 1-Proje başındaki amaç-hedeflerde değişme oranı | 20 |
| 2- Mevcut veriyapısı (tasarım, oluşturulan tablo vb) nin proje ister ve uygulama ihtiyaçlarını karşılama oranı | 90 |
| 3- Mevcut Karmaşık sorguların tasarım ve tam entegre uygulamaya alınma oranı; | 85 |
| 4- Saklı Yordam, Hata Ayıklama **ve** Tetikleyicilerin tam entegre uygulamaya alınma oranı; | 75 |
| 5- Arayüzlerin tamamlanması ve tam entegre uygulamaya alınma oranı | 95 |
| 6- Diğer Entegrasyonların (sosyal medya, e-mail/gsm, export/import) tam entegre uygulamaya alınma oranı | 100 |
| 7- Projenin tam entegre tamamlanma oranı | 90 |
| 8-Uygulamanın alanda kullanılabilirlik oranı | 95 |
| 9- Sunumun; genel akışı, teknik içerik aktarım yöntemleri, grup çalışanlarının kendi içinde dengeli şekilde konu anlatımı, zamanı etkin kullanım vb açısından değerlendirmesi | 80 |
| 10- Proje raporunun proje çalışmalarını yansıtma oranı | 75 |