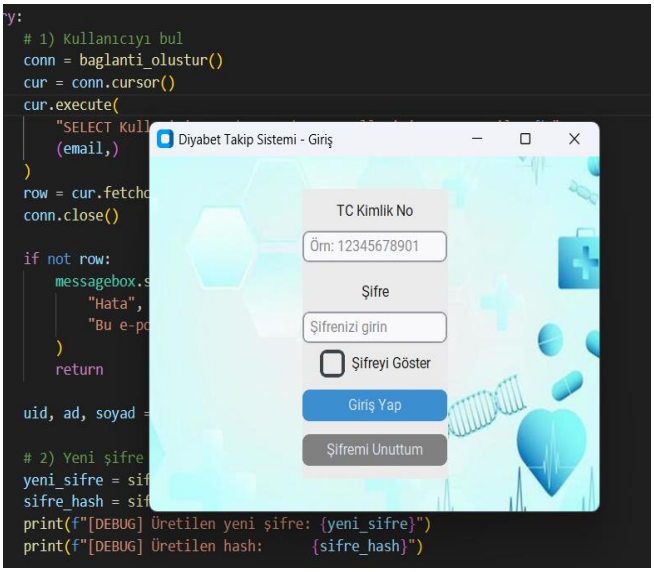
Programlama Labaratuvarı 6.Proje Raporu

Sadık Gölpek
sadikgolpek@gmail.com

Ali Kılınç
aliklnc4104@gmail.com

I.Özet

Bu çalışmada, diyabet hastalarının günlük kan şekeri ölçümlerini takip etmelerini ve doktorlarıyla paylaşmalarını sağlayan bir masaüstü uygulama geliştirilmiştir. Python programlama dili kullanılarak geliştirilen sistemde CustomTkinter ile modern bir grafik kullanıcı arayüzü, MySQL tabanlı ilişkisel bir veritabanı ve SHA-256 algoritması ile şifrelenmiş giriş mekanizması kullanılmıştır. Sistem, hasta ve doktor rollerine göre farklı paneller sunarak Rol Tabanlı Erişim Kontrolü (RBAC) sağlamaktadır. Hastalar; ölçüm verilerini kaydedebilir, diyet ve egzersiz durumlarını girebilir, geçmiş verilerini grafik olarak görüntüleyebilir. Doktorlar ise hastaların verilerini analiz ederek sistem tarafından önerilen karar destek bilgilerine ulaşabilir. Bu yapı, sağlık verilerinin güvenli, düzenli ve açıklanabilir bir biçimde yönetilmesine olanak tanır.

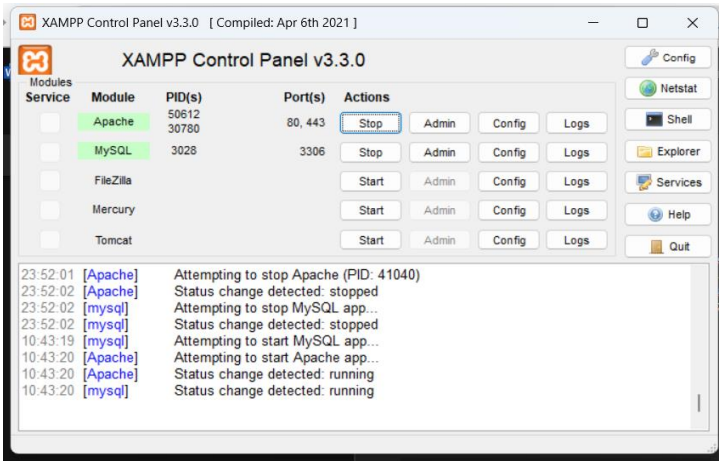


II.Giriş

Bu projede, Sistem, önceki benzer projelerden farklı olarak veritabanı tabanlıdır; bu sayede bilgiler sadece geçici olarak tutulmaz, yapısal olarak saklanır, analiz edilebilir hale gelir ve gerektiğinde kolayca erişilebilir.

Veritabanı kullanımı, kullanıcı bilgileri, ölçüm verileri, diyet ve egzersiz kayıtları gibi tüm verilerin tutarlı ve organize bir şekilde yönetilmesini sağlar. Projede MySQL veritabanı tercih edilmiştir çünkü kullanıcı dostu, açık kaynaklı ve birçok sistemle uyumlu bir çözümdür. Veritabanı işlemleri XAMPP paneli üzerinden gerçekleştirilmiş; bu panel, Apache ve MySQL servislerini yerel olarak çalıştırarak geliştirme sürecini kolaylaştırmıştır.

Veritabanının kullanılması, hem hasta hem de doktor tarafında girişlerin kaybolmamasını, geçmişe yönelik analiz yapılabilmesini ve çok kullanıcılı bir sistemin sağlıklı şekilde yönetilmesini mümkün kılmıştır. Bu yönüyle proje, sadece işlevsel değil, aynı zamanda sürdürülebilir ve genişletilebilir bir yapıya sahiptir.



III.Yöntem

3.1 Veritabanı Tasarımı

Sistemin temelini oluşturan veritabanı yapısı, ilk aşamada MySQL kullanılarak oluşturulmuştur. XAMPP paneli üzerinden çalıştırılan MySQL servisi aracılığıyla tablolar tanımlanmış ve veri yapıları inşa edilmiştir. Kullanıcı, doktor, hasta, ölçüm, diyet-egzersiz ve belirti gibi farklı kavramlar için ayrı tablolar tanımlanmış; bunlar arasında **primary key** ve **foreign key** ilişkileri kurularak tutarlı ve ilişkisel bir yapı sağlanmıştır.

Veritabanı, **3NF (Üçüncü Normal Form)** düzeyinde normalizasyon kurallarına uygun olarak tasarlanmıştır. Böylece veri tekrarının önüne geçilmiş, her tablo yalnızca kendi sorumluluğundaki veriyi taşımış ve veri bütünlüğü garanti altına alınmıştır.

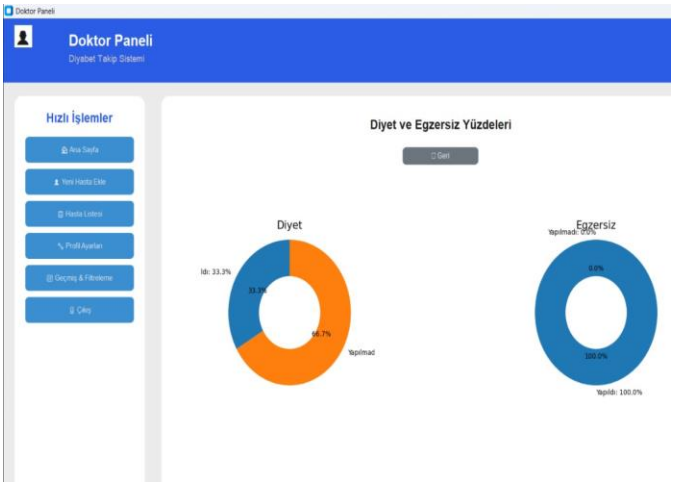
3.2 Uygulama Geliştirme ve Arayüz Tasarımı

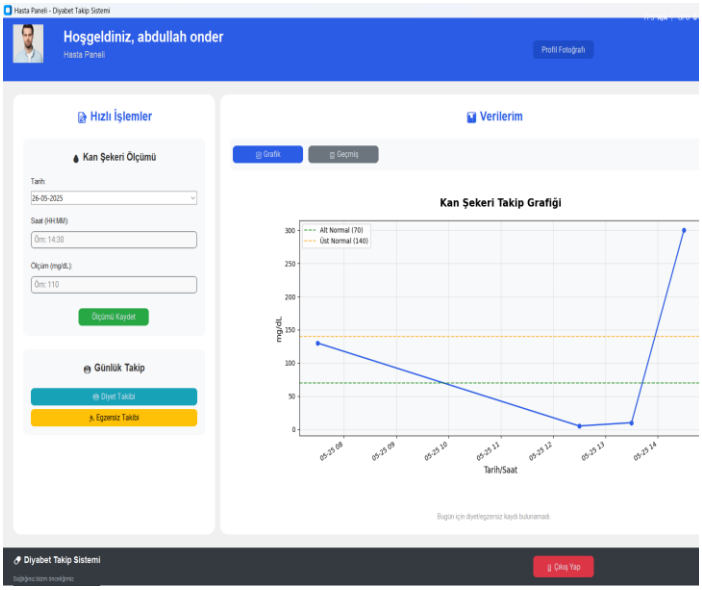
Veritabanı yapısı tamamlandıktan sonra sistem arayüzü, Python programlama dili kullanılarak geliştirilmiştir. Görsel tasarımda `CustomTkinter` kütüphanesi tercih edilmiştir ve bu sayede modern,

okunabilir ve kullanıcı dostu bir arayüz elde edilmiştir.

Kullanıcılar sisteme giriş yaptıktan sonra rollerine göre yönlendirilir. Hasta ve doktor için farklı arayüzler tasarlanmıştır. Hasta panelinde; ölçüm girişi, dönem kontrolü, ölçüm geçerlilik analizi, diyet-egzersiz kaydı, geçmiş verilerin listelenmesi ve grafiksel görselleştirme gibi modüller yer almaktadır. Doktor paneli ise kendi hastalarının verilerini izleyebilmekte, sistemin sunduğu kurallara göre hastalar için öneriler görüntüleyebilmektedir.

Uygulama ayrıca mail gönderme özelliğine de sahiptir. “Şifremi unuttum” işlemi sırasında sistem, ilgili e-posta adresine yeni oluşturulan geçici bir şifreyi SMTP protokolü ile iletir. Bu özellik, kullanıcı deneyimini artırmakla birlikte hesap güvenliğini de desteklemektedir.





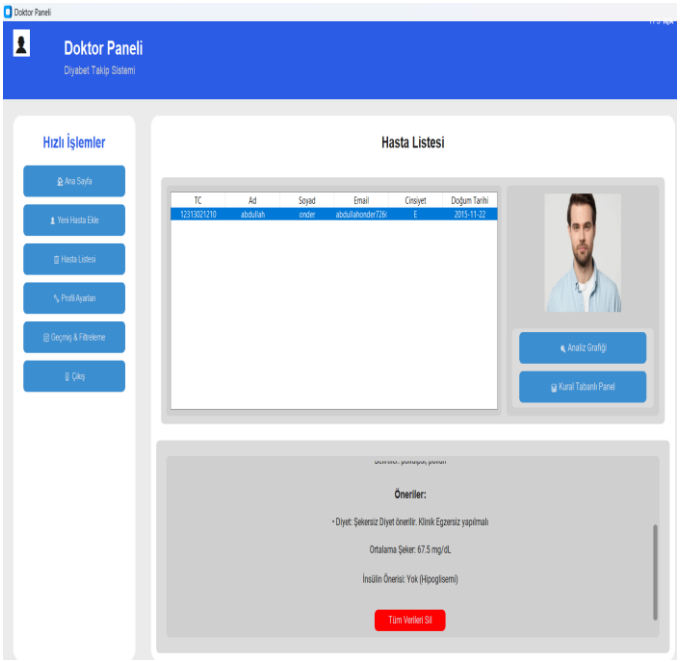
3.3 Rol Tabanlı Erişim ve Güvenlik

Sistem, çok kullanıcılı yapıya uygun olarak Rol Tabanlı Erişim Kontrolü (RBAC) prensibine göre geliştirilmiştir. Her kullanıcı sisteme giriş yaptıktan sonra rolü (doktor veya hasta) belirlenir ve sadece bu role ait ekranlara yönlendirilir. Böylece doktorlar yalnızca kendi hastalarını yönetebilirken, hastalar sadece kendilerine ait verileri görüntüleyebilir ve giriş yapabilir. Bu yapı sayesinde veri gizliliği ve kullanıcı ayrımı net bir şekilde sağlanmıştır.

Sistem ilk kurulum aşamasında doktor kullanıcıları doğrudan veritabanına kayıt edilmiştir. Bu kayıtlar sayesinde hasta ekleme ve hasta yönetim işlemleri ilgili doktorlarla ilişkilendirilebilmiştir. Yeni hasta kaydı yapıldığında, sisteme giriş yapan doktorun kimliği üzerinden veritabanına hasta ile ilişkilendirme yapılır.

Giriş işlemleri sırasında kullanıcı kimlik doğrulaması yapılır. Şifreler sistemde açık biçimde saklanmaz. Bunun yerine, SHA-256 algoritması kullanılarak hash’lenir ve yalnızca hash karşılaştırması yapılır. Bu yöntem, veritabanı sızıntılarına karşı önemli bir güvenlik katmanı sağlar.

İleri seviye güvenlik önlemleri kapsamında, sistemde e-posta tabanlı şifre sıfırlama özelliği de bulunmaktadır. Kullanıcı e-posta adresini girerek sistemden otomatik olarak rastgele bir geçici şifre alabilir. Bu işlem SMTP üzerinden gerçekleştirilir ve şifre güvenli bir biçimde iletilir.



IV. Deneysel Sonuçlar

Projenin geliştirme sürecinde XAMPP paneli ile birlikte kullanılan MySQL veritabanı, sistemin performansı, veri yönetimi ve anlaşılabilirliği açısından oldukça verimli olmuştur. Verilerin yerel olarak, yani doğrudan geliştiricinin bilgisayarında saklanması sayesinde fiziksel veri katmanı daha net bir şekilde gözlemlenebilmiştir. Bu yapı, veritabanı derslerinde teorik olarak öğrendiğimiz katmanları pratikte deneyimlememizi sağladı.

XAMPP, Apache ve MySQL servislerini tek panel üzerinden kontrol edebilmemizi sağlayarak geliştirme sürecini oldukça kolaylaştırdı. Özellikle veritabanı oluşturma, tablo ilişkilerini görme, sorgu yazma ve test etme işlemleri `phpMyAdmin` arayüzü sayesinde daha anlaşılır hale geldi. Bu sayede yalnızca yazılım geliştirmeye odaklanmak yerine, veri katmanının nasıl çalıştığını ve nasıl optimize edilebileceğini de

deneyimleme fırsatı bulduk.

Veritabanında yapılan sorgular sırasında otomatik olarak **indeksleme (indexing)** yapısının devreye girdiğini gözlemledik. Veriye erişim hızlarının artması ve `primary key`, `foreign key` yapılarının arka planda veri okuma/yazma performansını nasıl etkilediğini anlama şansımız oldu. Bu durum bize aslında veritabanı motorlarının iki ana görevi olduğunu öğretti: veri yazma (INSERT, UPDATE) ve veri okuma (SELECT). Geliştirme süreci boyunca her iki katmanın nasıl optimize edilebileceğini deneyimledik.

Başlangıçta MySQL ile PostgreSQL arasında kararsız kalmıştık. Ancak XAMPP panelinin MySQL'i yerel ve kullanıcı dostu biçimde sunması, phpMyAdmin ile doğrudan kontrol imkanı sağlaması nedeniyle MySQL tercih edilmiştir. Bu sayede uygulama ve veritabanı bütünlüğünü daha hızlı kurabildik.

Ayrıca süreç içerisinde şunu da fark ettik: **Veritabanı ile proje geliştirirken son derece dikkatli olunmalıdır.** Yanlış bir `DELETE` sorgusu, eksik bir ilişkilendirme veya yanlış `foreign key` kullanımı ciddi veri kayıplarına yol açabilir. Bu nedenle sistemin mantıksal tasarımı yapılmadan doğrudan kodlamaya başlanmamalı, önce veritabanı yapısı doğru ve anlamlı şekilde inşa edilmelidir. Bu yaklaşım, projenin sürdürülebilirliği açısından oldukça kritiktir.

V.Sonuçlar

Bu proje, hem teknik olarak işlevsel bir diyabet takip sistemi ortaya koymuş hem de geliştiriciler olarak bizim için önemli bir öğrenme süreci olmuştur. Özellikle veritabanı temelli bir sistem geliştirmenin ne kadar kapsamlı, dikkat gerektiren ve disiplinli bir süreç olduğunu bu projede deneyimledik.

Projeye başlamadan önce veritabanı yapıları, normalizasyon, foreign key ilişkileri ve SQL sorgularının ne kadar hayati bir rol oynadığını teorik olarak biliyorduk. Ancak bu projeyle birlikte, bu kavramların gerçek bir sistem içinde nasıl çalıştığını doğrudan gözlemleme ve uygulama şansımız oldu. Geliştirme süreci boyunca verilerin bütünlüğünü korumanın, doğru ilişkileri kurmanın, erişim kontrollerini sağlamanın ne kadar önemli olduğunu yaşayarak öğrendik.

Sonuç olarak bu proje, yalnızca bir ders ödevi olmaktan çok, veritabanı destekli sistem tasarımı, kullanıcı arayüzü geliştirme ve güvenlik konularında bizi ileriye taşıyan kapsamlı bir deneyim alanı olmuştur.

VI.Algoritmanın Kaba Kodu

Kan şekeri seviyesi ve belirtilere göre öneri üretme

```
FONKSİYON KuralTabanliOneri(seviye, belirtiler):

  EĞER seviye BOŞ VEYA belirtiler BOŞ İSE

    DÖN "⚠ Gerekli veriler eksik."


  b = belirtiler (hepsi küçük harfe çevrilmiş)


  EĞER seviye >= 180:

    EĞER {"yaraların yavaş iyileşmesi", "polifaji", "polidipsi"} Tüm b'de VARSA:

      DÖN "Şekersiz Diyet önerilir. Klinik Egzersiz yapılmalı."

    EĞER {"yaraların yavaş iyileşmesi", "kilo kaybı"} VARSA:

      DÖN "Şekersiz Diyet önerilir. Yürüyüş yapılmalı."


  EĞER 110 < seviye < 180:

    EĞER {"bulanık görme", "nöropati", "yorgunluk"} VARSA:

      DÖN "Az Şekerli Diyet önerilir. Yürüyüş yapılmalı."

    EĞER {"poliüri", "polidipsi"} VARSA:

      DÖN "Şekersiz Diyet önerilir. Klinik Egzersiz yapılmalı."

    EĞER {"bulanık görme", "nöropati"} VARSA:

      DÖN "Az Şekerli Diyet önerilir. Klinik Egzersiz yapılmalı."
```

EĞER 70 <= seviye <= 110:

EĞER {"yorgunluk", "kilo kaybı"} VARSA:

DÖN "Az Şekerli Diyet önerilir. Yürüyüş yapılmalı."

EĞER {"polifaji", "polidipsi"} VARSA:

DÖN "Dengeli Beslenme önerilir. Yürüyüş yapılmalı."

EĞER seviye < 70:

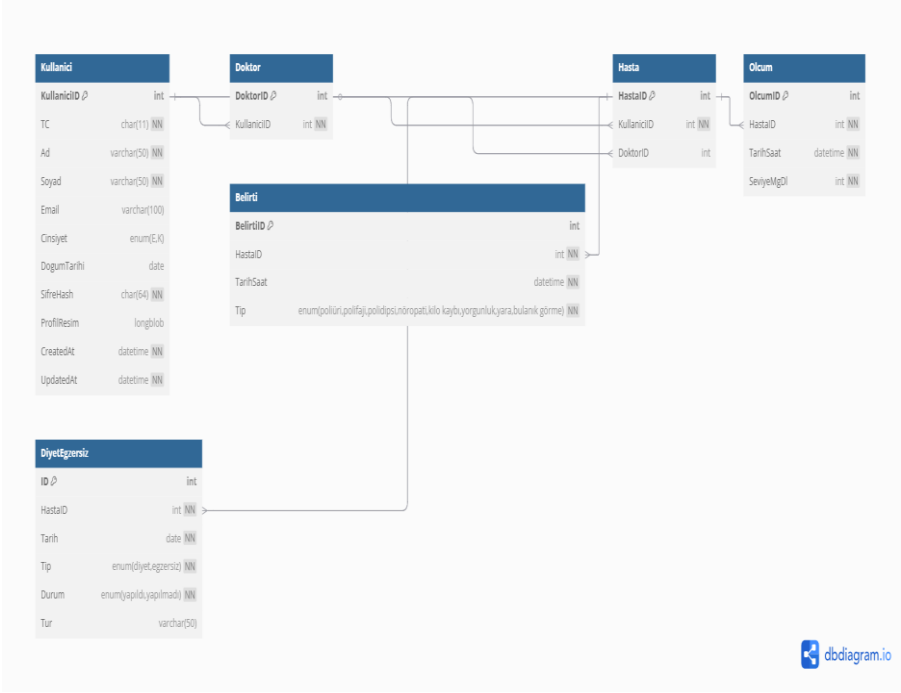
EĞER {"nöropati", "polifaji", "yorgunluk"} VARSA:

DÖN "Dengeli Beslenme önerilir. Egzersiz yapılmamalı."

DÖN "⚠ Bu belirtiler ve seviye için öneri bulunamadı."

VII.SQL Şeması ve ER Diyagramı

Aşağıda yer alan görsel, geliştirilen diyabet takip sisteminin **SQL tabanlı tablo yapısını** göstermektedir. Bu şema, her tablonun sütunlarını, veri tiplerini ve tablolar arası ilişkileri teknik düzeyde sunmaktadır. Görsel `dbdiagram.io` platformu kullanılarak elde edilmiştir.



▪ Temel Varlıklar ve Nitelikler:

- **Kullanici**: Sistemin merkezinde yer alır. Doktor ve hasta profilleri bu varlık üzerinden türetilmektedir. `KullaniciID` birincil anahtardır. Diğer sütunlar; kimlik bilgileri, şifre hash değeri (SHA-256), profil resmi gibi detayları içerir.

- **Doktor**: Her doktor bir kullanıcıdır. `KullaniciID`, `Kullanici` tablosuna yabancı anahtardır. Doktorlara doğrudan hasta atanabilmektedir.

- **Hasta**: Hasta profili de yine bir kullanıcıya bağlıdır. Ek olarak, bir `DoktorID` alanı ile bir hastanın hangi doktorun gözetiminde olduğu belirtilir. Hem `KullaniciID` hem de `DoktorID` alanları ilişkiseldir.

- **Olcum**: Hastanın kan şekeri ölçüm kayıtlarını tutar. `HastaID` üzerinden hasta ile ilişkilidir. `TarihSaat` ve `SeviyeMgDl` sütunları, ölçümün zamanını ve değerini ifade eder.

- **DiyetEgzersiz**: Hastaların günlük diyet ve egzersiz uygulamalarını içerir. `Tip`, `Durum` ve `Tur` gibi sütunlarla detaylandırılır. Hasta ile birebir ilişkilidir.

- **Belirti**: Hastaların gün içerisinde yaşadığı semptomlar burada kaydedilir. `Tip` sütunu ENUM olarak tanımlanmış ve yaygın diyabet belirtileri (örneğin: polidipsi, yorgunluk, nöropati) seçenek olarak sunulmuştur. `TarihSaat` ile birlikte zamana bağlı analiz imkanı sağlanır.

VIII. Kaynakça

1. Python Software Foundation, “The Python Standard Library,” [Online]. Erişim: https://docs.python.org

2. Oracle Corporation, “MySQL 8.0 Reference Manual,” [Online]. Erişim: https://dev.mysql.com/doc

3. Apache Friends, “XAMPP – Apache + MariaDB + PHP + Perl,” [Online]. Erişim: https://www.apachefriends.org

4. Holistics Software, “dbdiagram.io – Database Relationship Visualizer,” [Online]. Erişim: https://dbdiagram.io

5. Şadi Evren Şeker, “Veritabanı Eğitim Serisi,” YouTube Playlist, [Online]. Erişim: https://www.youtube.com/playlist?list=PLh9ECzBB8tJOS7WQKdeUaAa5fmPLYAouD