Hastane Otomasyon Projesi

VERİ TABANI YÖNETİM SİSTEMLERİ DÖNEM SONU PROJESİ

Kubilay Birer 211307086 kubilaybirer@hotmail.com

Ekrem Özer 211307051 ekremozerr@hotmail.com

Halit Çelenk 201307056 hcelenkk41@icloud.com

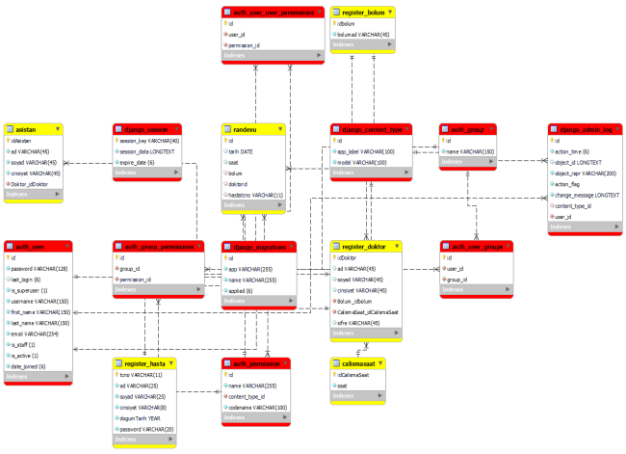
***Abstract*—**Bu proje, web üzerinde bir hastane otomasyon sistemi tasarlamayı amaçlamaktadır. Sistem, hastaların çevrimiçi olarak randevu almasını, doktorların randevu taleplerini görüntülemesini ve yönetmesini, ayrıca personelin hastane bilgilerini güncellemesini sağlar. Bu proje, kullanıcılara daha iyi bir randevu deneyimi sunmayı ve hastane yöneticilerinin randevu sürecini daha verimli hale getirmeyi hedeflemektedir. Projenin geliştirilmesi için Django web framework'ü kullanılmıştır.

Keywords—Giriş,Projenin Veri Tabanı,Frontend ve Teknik Detaylar,Sonuç,Kaynakça

I. GİRİŞ

Günümüzde sağlık hizmetlerinin daha verimli bir şekilde sunulabilmesi için hastaneler de diğer işletmeler gibi teknolojiden yararlanmaktadır. Bu amaçla birçok hastane yönetim sistemi geliştirilmiştir. Bu proje de web ortamında Django kütüphanesi kullanarak bir hastane yönetim sistemi geliştirme çalışmasıdır. Bu sistemin amacı, hastanedeki doktorların ve hastaların işlemlerini daha hızlı, güvenilir ve kolay bir şekilde yapmalarını sağlamaktır. Sistem, randevu alma, hasta kayıt işlemleri, doktor takibi, randevu alma, randevu görüntüleme gibi birçok işlemi kapsamaktadır. Bu rapor, hastane yönetim sistemi projesinin tasarımı, geliştirilmesi ve test edilmesi hakkında detaylı bilgi sunmaktadır.

II. PROJENIN VERI TABANI



Resim 1 : Database'in ER diyagramı.

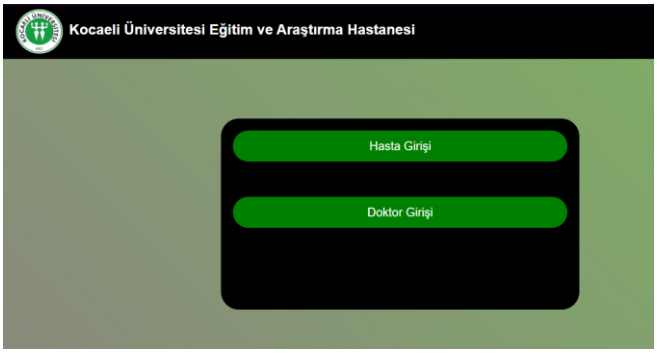
Projenin veritabanı kısmı, hastane yönetimi ile ilgili verilerin depolandığı bir alandır. Bu alanda, hasta bilgileri, doktor bilgileri, randevular bilgileri ve diğer ilgili veriler tutulur. Veritabanı, projenin temel yapısını oluşturur ve hastanemizin daha iyi yönetimine yardımcı olmak için tasarlanmıştır.

Veritabanı, Django ORM (Object-Relational Mapping) ve MYSQL Workbench kullanılarak oluşturulmuştur. Veriler, Django model sınıfları aracılığıyla tanımlanır ve veritabanında tablolar olarak saklanır. Veri tabanında ilişkiler ilgili normalizasyon kurallarına uygun olarak projeden sorumlu kişiler tarafından özenle hazırlanmıştır. Gerekli ilişkiler ve foreign keylerin normalizasyon kurallarına uygun yapılması sonucunda Django web tarafında yanlış veri girişleri ve olası karışıklıkların önüne geçilmektedir.

III. FRONTEND VE TEKNIK DETAYLAR

Raporun bu kısmında sırasıyla web sitemizin kısımları aktarılacaktır.Genel olarak frontend kısımları aktarılacak olup gerekli yerlerde Django backend kodlarıyla da pekiştirme sağlanacaktır.

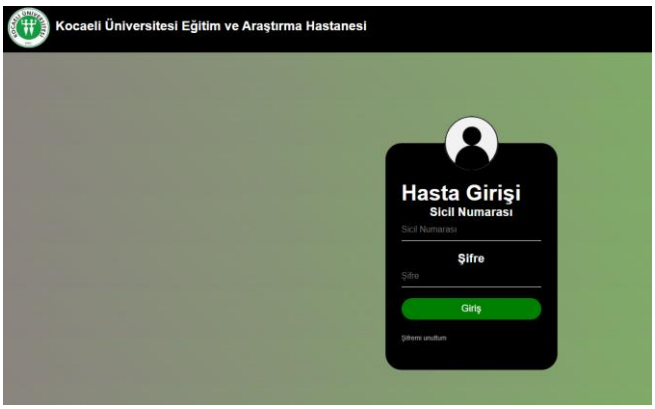
Giriş Ekranı



Resim 2 : Web sitemizin giriş sayfası.

Yukarıda görülen ekran web sitesine ilk girildiğinde açılan giriş sayfasıdır.Burada giren kişi hasta ise hasta ise hasta giriş ekranına tıkladığında , doktor ise doktor giriş ekranına tıkladığında gerekli ekrana yönlendirilmesi sağlanacaktır

Hasta Giriş Ekranı



Resim 3 : Hasta login ekranı.

Giriş ekranında hasta girişi kısmına tıklayan kişilerin yönlendirileceği ekran üstteki hasta giriş ekranı olacaktır.Bu ekranda hastalar hastanede kayıt açılma işlemi sonrasında onlara verilen şifreyle kendi girişlerini yapacaklardır.Ekran herhangi yanlış giriş olduğunda hatalı giriş mesajı vermektedir ve hastalar eğer şifrelerini unuturlarsa şifremi unuttum kısmından maillerine şifrelerini mesaj olarak alabileceklerdir.

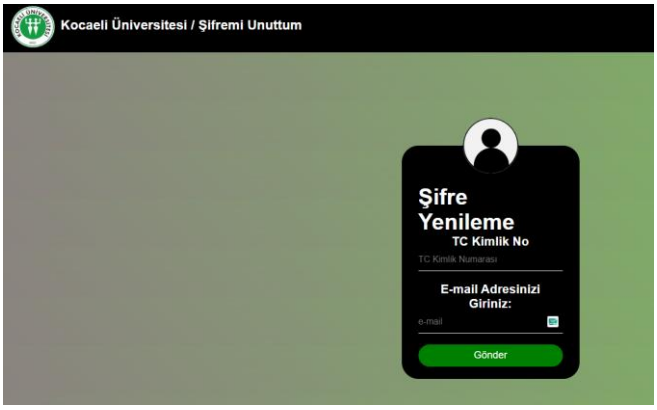
Hasta giriş ekranına ait Django üzerindeki backend kodları aşağıda belirtilmiştir.

```
def hgiris(request):
    if request.method == "POST":
        tcno =
request.POST.get('tcno')
        password =
request.POST.get('password')

        try:
            hasta =
Hasta.objects.get(tcno=tcno,
password=password)
            request.session['tcno']
= tcno
             # kullanıcı giriş
yaptıktan sonra yönlendirileceği
sayfanın adını buraya yazın
        except Hasta.DoesNotExist:
            error_message =
"Geçersiz TC Kimlik No veya Şifre"
            return render(request,
'hgiris.html', {'error_message':
error_message})

        return
render(request,'hanasayfa.html')
    else:
        return render(request,
'hgiris.html')
```

Kodda görüldüğü gibi giriş sağlandıktan sonra Mysql üzerindeki database üzerinde gerekli sorgulamalar yapılıp buna göre gerekli aksiyonlar alınmaktadır.Giriş başarısız ise hata alan kullanıcı eğer bilgilerini doğru girerse hasta anasayfasına yönlendirilmektedir.

Aynı zamanda bu views.py fonksiyonunda önemli kısımlardan biri ise tcno nun request.session işlevleri ile diğer sayfalar üzerine yollanmasıdır.Bu tcno hasta tablosunun primary keyi olarak ileride randevu görüntüleme,şifre değiştirme ve bunun gibi diğer işlemlerde kullanılacaktır.

Şifremi Unuttum Ekranı



Resim 4 : Şifremi unuttum ekranı

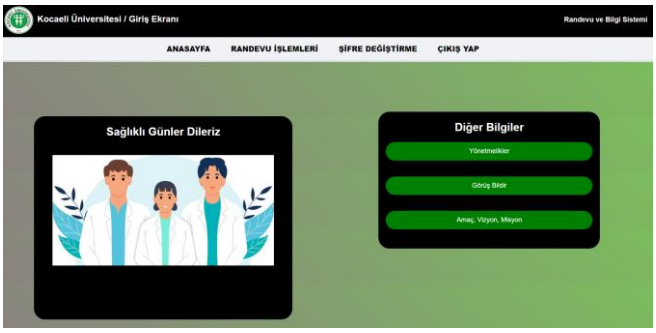
Eğer hastalar şifrelerini unuturlarsa hastaneye gelmelerine gerek kalmadan çeşitli mail servislerini kullandığımız şifremi unuttum ekranıyla şifrelerini maillerine mesaj olarak alabileceklerdir.

```
def hşifreu(request):
    if request.method == 'POST':
        tcno =
request.POST.get('tcno')
        email =
request.POST.get('email')
        try:
            hasta =
Hasta.objects.get(tcno=tcno)

                # Şifre sıfırlama
kodları buraya yazılacak
            send_mail(
                'Şifre
Hatırlatma',
                f'Hasta
Bilgileriniz:\nTC Kimlik No:
{hasta.tcno}\nŞifreniz:
{hasta.password}',
                'fakeemail123@gm
ail.com', # fake Gmail hesabınızın
kullanıcı adı
                [email],
                fail_silently=Fa
lse,
            )
            return
redirect('hgiris')
```

```
        except Hasta.DoesNotExist:
            messages.warning(request
, "Girilen TC kimlik numarası ile
kayıtlı bir kullanıcı bulunamadı.")
    return render(request,
'hsifreunut.html')
```

Şifre unuttum ekranına ait Django backendindeki views fonksiyonu şekildeki gibidir.Fonksiyonda bilgilerin doğru girilmesi durumunda Djangonun mail servisiyle maile şifre gönderimi yapılmaktadır.Eğer iki bilgiden biri hatalı girilirse de gerekli uyarıların verilmesi djangonun messages fonksiyonuyla sağlanmıştır.

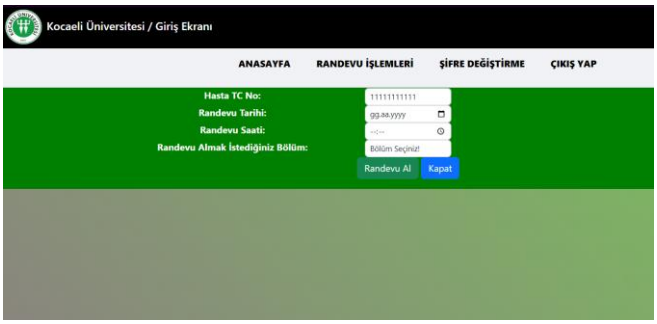
Hasta Anasayfa Ekranı



Resim 5: Hasta anasayfa ekranı

Başarılı giriş sonrası hastaların aktarıldığı anasayfa ekranı üstte görülmektedir burada hastalar gerekli ekranlara üstteki navbardan erişebilecektir.

Randevu Alma Ekranı



Resim 6: Hasta randevu alma ekranı

Üstteki navbarda randevu işlemlerine tıklandıktan sonra randevu alma kısmına giren hastalar üstteki Randevu alma formuyla karşılaşacaktır.Bu formda tcno giriş ekranından request sessional taşınan tcno olacaktır ve readonly modunda olduğu için değiştirilmesi olanaksızdır bu sayede ise hastanın kendinden başka bir hastaya randevu eklemesi gibi karışıklıkların önüne geçilmiştir.
Randevu bölümleri database üzerindeki bolum tablosundan çekilmektedir ve doktor seçimi ise Database üzerinde yapmış olduğumuz trigger tarafından randevu tablosuna randevu eklenirken o bölümden rastgele bir doktoru randevuya atanması şeklinde yapılmaktadır.Bu sayede doktorlarda oluşabilecek yığılmaların önüne geçilmiştir.

```
def hral(request):
    tcno =
request.session.get('tcno')
    bolumler = Bolum.objects.all()

    if request.method == 'POST':
        form =
RandevuForm(request.POST, tcno=tcno)
        form.fields['hastatcno'].ini
tial = tcno
        if form.is_valid():
            randevu =
form.save(commit=False)
            randevu.save()
            print("Form is valid and
submitted successfully.")
            messages.info(request,"R
andevunuz başarıyla eklendi.")
    else:
        form =
RandevuForm(tcno=tcno)
    return render(request,
'hrandevual.html', {'form': form,
'bolumler': bolumler})
```

Randevu alanının views fonksiyonu şekildeki gibidir burada da hatalı girişlerin önüne geçebilmek için yine gerekli kontroller yapılmaktadır.Randevu başarıyla tamamlandığında ise randevunun tamamlandığına dair mesaj hastaya iletilmektedir.

```
class RandevuForm(forms.ModelForm):
    hastatcno =
forms.CharField(widget=forms.HiddenI
nput())

    def __init__(self, *args,
**kwargs):
        tcno = kwargs.pop('tcno',
None)
        super(RandevuForm,
self).__init__(*args, **kwargs)
```

```
        self.fields['hastatcno'].ini
tial = tcno

    class Meta:
        model = Randevu
        fields = [ 'tarih', 'saat',
'bolum','hastatcno']
        widgets = {
            'tarih':
forms.DateInput(attrs={'type':
'date'}),
            'saat':
forms.TimeInput(attrs={'type':
'time'}),
            'bolum':
forms.Select(attrs={'id': 'bolum-
secimi'})
        }
```

Randevu ekranına ait forms.py fonksiyonu da şekildeki gibidir tcno direk request sessiondan alınıp diğer alanlar html üzerinde doldurulan inputlar üzerinden çekilmektedir.

Randevu Görüntüleme Ekranı

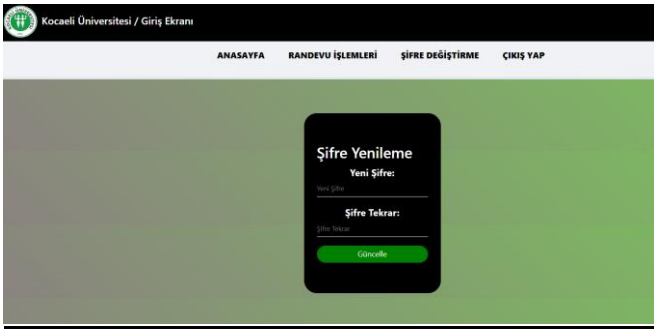
Resim 7: Hasta randevu görüntüleme ekranı

Randevu görüntüleme ekranında ise hastalar ve doktorlar kendilerine ait randevuları ekran görüntüsündeki gibi görüntüleyebilmektedir.Eğer randevu iptal edilmek istenirse sil butonundan şekildeki gibi iptal edilip aynı şekilde tekrar istenilen bölümden randevu alınabilir.

```
def hrbilgi(request):
    tcno =
request.session.get('tcno')
    if not tcno:
        return redirect('hgiris')
    randevular =
Randevu.objects.filter(hastatcno=tcn
o)
    if not randevular:
        messages.warning(request,
"Randevunuz bulunmamaktadır.")

    return render(request,
'hrandevubilgi.html', {'randevular':
randevular})
```

Randevu görüntüleme ekranının views fonksiyonu üstteki gibidir randevular sayfalar arası aktarılan request session daki tcno ya göre çekilir ve eğer herhangi bir randevu yoksa gerekli hata mesajı kullanıcıya gösterilir.

Sifre Değiştirme Ekranı



Resim 8: Şifre yenileme ekranı

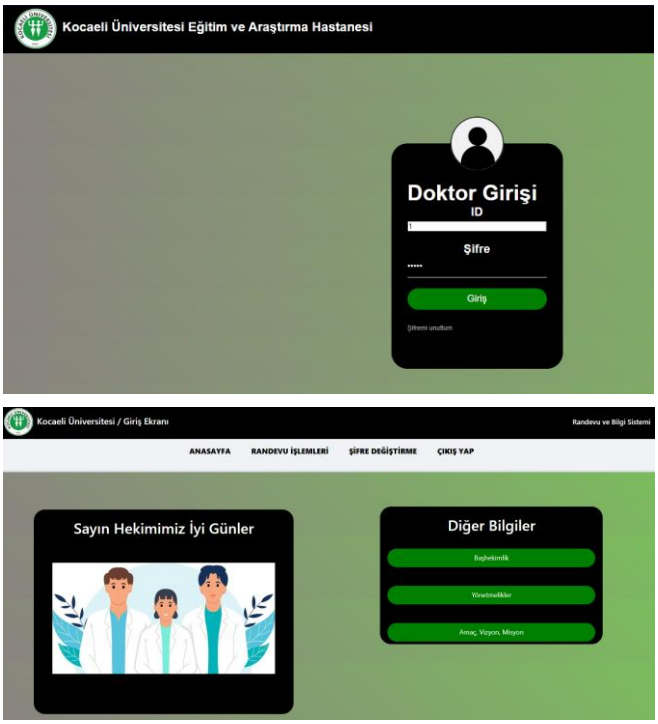
Bu ekran hasta ve doktor kısımlarında frontend olarak ortaktır sadece backenddeki gerekli sorgu kodları hasta ve doktor olarak ayrılmaktadır.Bu ekranda kişi yenilemek istediği şifreyi iki kez girerek mevcut şifresini değiştirebilmektedir.

```
def hsdegis(request):
    tcno =
request.session.get('tcno')
    hasta =
Hasta.objects.get(tcno=tcno)

    if request.method == 'POST':
        yeni_sifre =
request.POST.get('yeni_sifre')
        yeni_sifre_tekrar =
request.POST.get('yeni_sifre_tekrar'
)

        if yeni_sifre ==
yeni_sifre_tekrar:
```

```
            hasta.password =
yeni_sifre
            hasta.save()
            return
redirect('hanasayfa')
        else:
            messages.error(request,
'Şifreler eşleşmiyor.')

    return render(request,
'hsifredegis.html')
```

Ekran görüntüsünde de görüldüğü gibi eğer şifreler eşleşiyorsa html sayfasına aktarılmış request sessiondaki tcno ile şifre değiştirme işlemi başarıyla gerçekleşmektedir.Eğer şifreler uyuşmuyor ise de gerekli hata mesajı kullanıcıya gösterilmektedir.

Doktor Ekranları



Resim 9-10:Doktor giriş ve anasayfa ekranları

Doktor ekranları da frontend açısından bakılacak olursa hasta ekranlarıyla aynı tasarıma sahiptir aynı şekilde backend kısmında da sütun adları harici kodlarda tcno yerine doktorid kullanılması gibi küçük farklar dışında çok büyük farklar yoktur.

IV. Sonuç

Sonuç olarak, bu hastane projemiz birçok kullanıcının ihtiyaçlarını karşılayacak birçok özellik sunan kullanıcı dostu bir web uygulamasıdır. Veritabanı tasarımı, sistemdeki verilerin etkili bir şekilde saklanmasını ve erişilebilmesini sağlamaktadır. Projenin ileriye dönük olarak geliştirilmesi için, kullanıcı geri bildirimleri göz önünde bulundurularak, yeni özellikler ve iyileştirmeler eklenebilir.

V. Kaynakça

Django. (2023). Django Project. https://docs.djangoproject.com/en/4.2/

MySQL :: MySQL Documentation. (2023). Mysql.com. https://dev.mysql.com/doc/

3.11.3 Documentation. (2023). Python.org. https://docs.python.org/3/

Otto, M. (2023). *Get started with Bootstrap*. Getbootstrap.com. https://getbootstrap.com/docs/5.3/getting-started/introduction/

DevDocs. (2023). Devdocs.io. https://devdocs.io/html/

DevDocs. (2023). Devdocs.io. https://devdocs.io/css/

Github: https://github.com/Eozerr/djangoProjects