Veri Tabanı Yönetim Sistemleri

Otopark Otomasyonu

Mutlucan Gökçukur
211307006
Bilişim Sistemleri
Mühendisliği/
Kocaeli Üniversitesi

Proje, Otopark Otomasyonu, bir otopark işletme sahibinin müşterilerine ve personellerine yardımcı olması için tasarlanmıştır. Proje insanların araçlarını takip edebilmelerini, araçlarının otoparkta geçirdiği süreyi ve buna bağlı olarak da otoparkta oluşacak ücretini görebilmektedir. Bu proje herhangi bir otopark için kullanılabilir, veri tabanı sistemi de bu otoparka göre kolayca ayarlanabilir. Bu çalışmada müdür, personel ve çalışan olmak üzere 3 tip kullanıcı ve bunların kendilerince yapabildikleri işlemler bulunmaktadır.

Anahtar Kelimeler – Otopark, SQLite, QR Kod, Android Studio, Kotlin

I. Başlangıç

Projemin taslağını oluşturmadan önce, nasıl bir sistem kullanabilirim ve bunları nasıl bir araya getirerek çalıştırabilirim bunları araştırmakla başladım. Teknolojinin günden güne daha da ileriye gittiğini düşünerek nasıl yenilikçi bir otomasyon sistemi getirebileceğimi araştırdım. Bu yenilikçi denemelerimi nasıl telefon üzerinde kullanabileceğimi, nasıl entegre edebileceğimi ve hangi veri tabanının bu işlemler için daha uygun olduğunu belirlemekle başladım.

Ayrıca bu uygulamanın nasıl bir tasarıma sahip olması gerektiği hakkında da çeşitli internet sitelerinde hazır tasarımlar olmasına rağmen kendi basit tasarımımı kullanmayı tercih ettim. İnternetten yenilikçi bir otomasyon için kullanmak istediğim ‘Kotlin ile QR Kod’ okuma programının nasıl yapıldığını ve bu okutma sonrasının nasıl devam etmesi gerektiğini planladım. Kafamda müşteriler kısmını oturttuktan sonra müdür ve personel kısımlarını daha basit ve düzgün bir şekilde her bilgiyi görebilecek, güncelleyebilecek şekilde tasarladım.

II. Kullanılan Teknolojiler

Android Studio

Projemi mobil bir uygulama üstüne tasarlayacağım için öncelikle hangi platform üstünde yapacağıma karar vermem gerekiyordu. Bunin içinde ‘Kotlin’ dilinin çalıştığı ve bir çok programa göre de stabil şekilde çalışan Android Studio uygulamasını tercih ettim. Bu uygulama Android uygulamaları geliştirmek için Google tarafından özel olarak tasarlanmış bir uygulamadır.

Uygulamamı test ederken bir çok farklı Android telefonu sunmasının yanı sıra uygulamamı kendi telefonumda da deneyebildiğimden dolayı bu platformu tercih ettim.

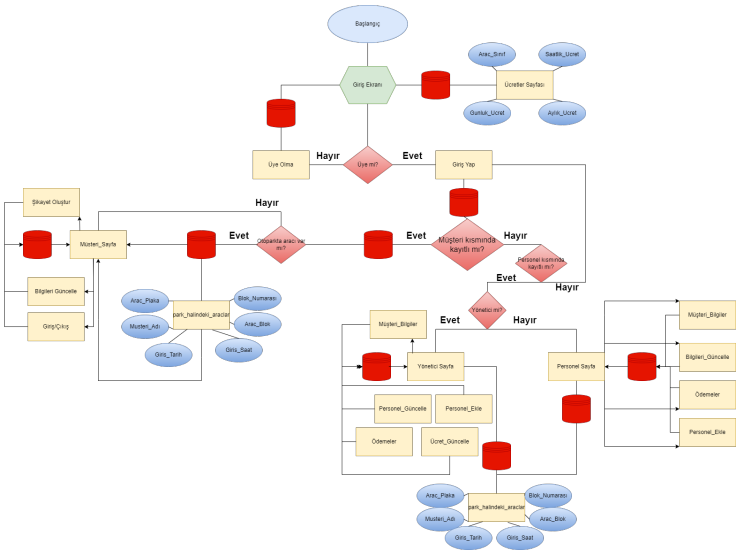
CodeScanner

CodeScanner Android Studio içerisinde ‘Kotlin’ dili için tasarlanmış basit bir QR Kod okutma sistemidir. Bu yenilikçi sistem ile otopark uygulamalarına yeni bir soluk getirmeyi hedeflemekteyim.

SQLite

Android Studio için en çok kullanılan ilişkisel veri tabanı yönetim sistemlerinden birisi de SQLite uygulamasıdırs. Bunun sebebi ise SQLite uygulamasının hafif ve ücretsiz bir veri tabanı olmasıdır. SQLite genel larak basit bir veri tabanı yönetim sistemidir ve SQL sorguları kullanılarak temel işlemleri (veri ekleme, veri silme, veri güncelleme, veri sorgulama) işlemlerini kolay bir şekilde yapabilirsiniz.

III. ER Diyagrami



Şekil 3.1: Otomasyon uygulamasının ER Diyagramı

IV. Yazılım Mimarisi Yöntem Ve Teknikler

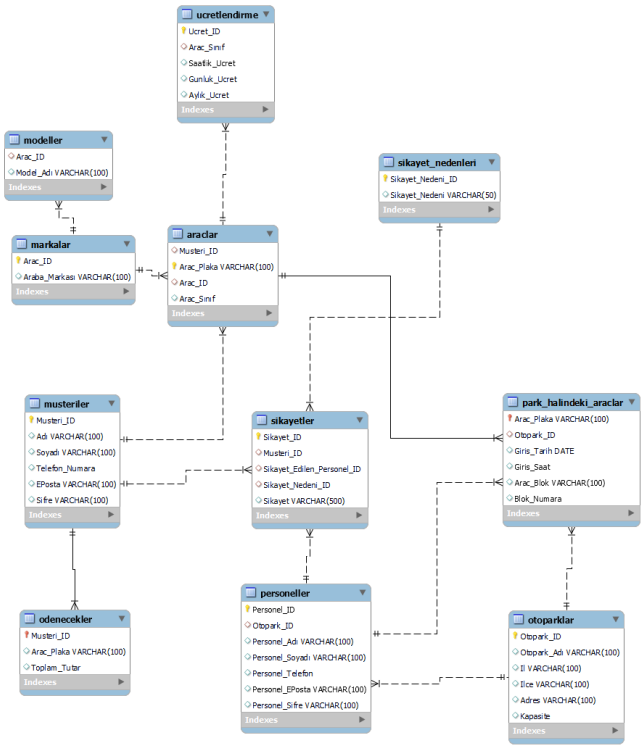
Bu projemde programlama dili olarak ‘Kotlin’ kullandım. Bunun için birkaç sebebim var.

Öncelikle Kotlin diğer dillere nazaran daha yeni bir dil olması, diğer mobil programlama dillerine göre daha kolay olması sebebiyle işimi kolaylaştırdı. Kotlin, ayrıca Android Studio ile çalıştırılması, Android uygulamaların geliştirilmesi için en popüler yöntemlerden birisi olarak görünmektedir.

V. VERİTABANI TASARIMI

Kotlin ile programlama yaparken oluşturduğum SQLite veri tabanını ayrıca ER diyagramı ile oluşturmak için veri tabanını ayrıca MySQL Workbench üzerinde de tasarladım. Bu sayede veri tabanı tasarımında oluşabilecek hataları burada görerek SQLite veri tabanında kolaylıkla düzeltebildim.

Bu tablolarda, 5N kurallarına uygun olarak veri tabanını tasarladım. Diyagramı da MySQL Workbench uygulamasında tasarladığım veri tabanından aldım.

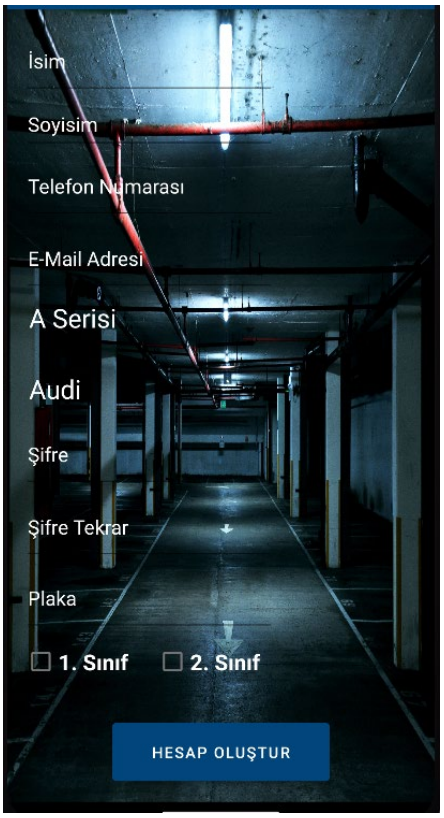


Şekil 5.1: Veri Tabanı ER Diyagramı

VI. MÜŞTERİ GENEL YAPISI

Öncelikle programıma nelerin gerekli olduğuna karar vermekle başladım. Her otoparkta bulunması kesin olacak şeyleri belirledim. Müşteri ve müdür kısımları ile başlamak ilk aşama için mantıklı bir tercih olarak gördüm. Öncelikle gelen müşterinin sisteme üye olması gerekmektedir. Bu sayede müşteriler sürekli olarak verilerini girerek zaman kaybetmesin ve bu verilerde benim elimde bulunması oluşabilecek herhangi bir sorunda müşteriye kolayca ulaşabileceğimi düşündüm.

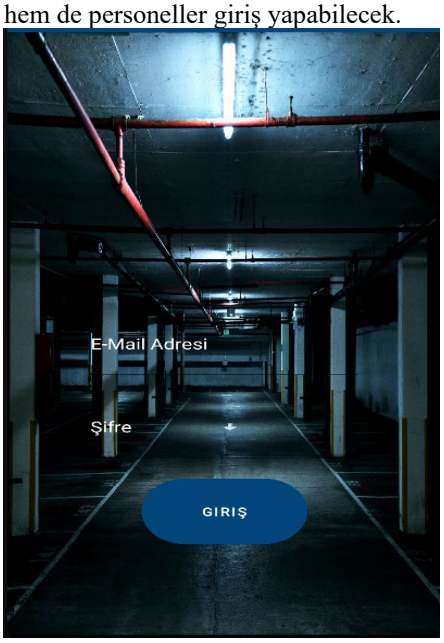
Müşteriler kayıt olurken ayrıca veri tabanına istediğiniz kadar araç marka ve model ekleyebilirsiniz. Ben örnek olması açısından birkaç örnek koydum. Bu sayede sistemin çalışma mantığını görmüş oluyoruz.



Şekil6.1: Müşteri Hesap Oluşturma Sayfası

Buradan müşteri arabasının sınıfını da yine kendisi belirtecek. Bu sayede veri tabanındaki fiyatlandırma sistemi yine burada kaydedilen araç sınıfına göre hesaplanacak.

Daha sonra ise müşterilerin girdikleri E-Posta adresleri ve şifreleri ile sisteme giriş yapmaları gerekmektedir. Bunun içinde bir giriş sayfası tasarladım. Burada hem müşteriler hem de personeller giriş yapabilecek.

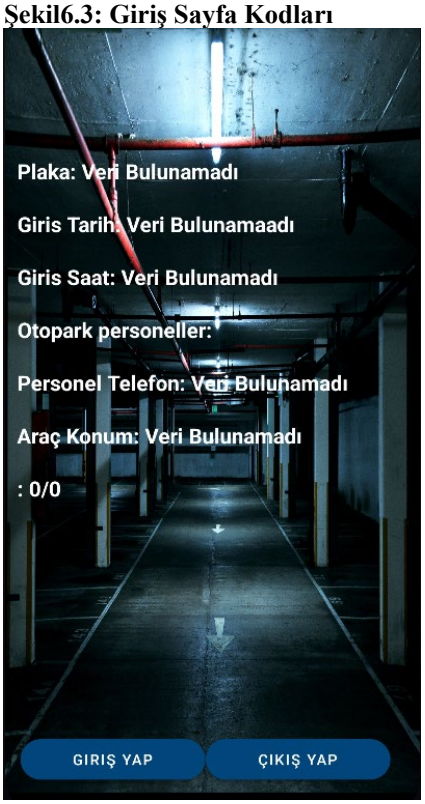


Şekil6.2: Giriş Sayfası

Giriş sayfasındaki kodlarda hem müşteriler hem de personeller tablosu kontrol edilmekte. Öncelikle girilen E-Posta ve Şifre ile uyumlu bir müşterinin veri tabanında olup olmadığı kontrol edilmektedir. Eğer varsa müşteri sayfasına yönlendirilir. Eğer yoksa personeller tablosuna bakılır. Eğer girilen bilgiler Personel_ID=1 için uyumlu ise bunun sistemdeki karşılığı müdürdür ve müdür için oluşturulan sayfayı açar. Eğer Personel_ID değeri bir değil ama başka birini işaret ediyorsa personel sayfasına yönlendirilir.

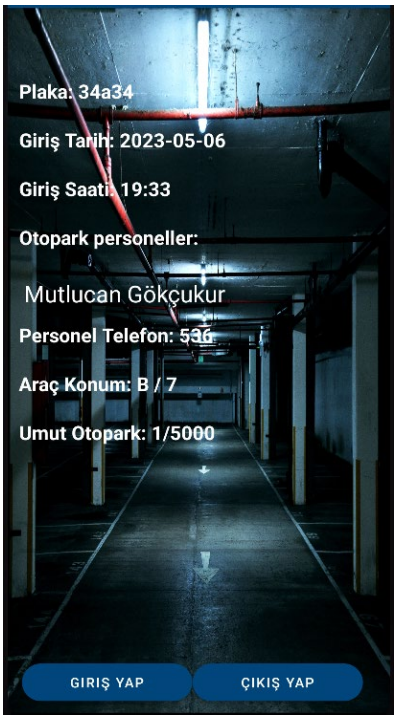
```
val Database=openOrCreateDatabase( name: "OtomasyonVeritabanı", MODE_PRIVATE, factory: null)
if(UyeGirisEPosta.text.toString()!="" &&UyeGirisSifre.text.toString()!="")
{
    val mustericursor=Database.rawQuery( sql: "SELECT * FROM musteriler WHERE EPosta=? AND Sifre=?",
        arrayOf(UyeGirisEPosta.text.toString(),UyeGirisSifre.text.toString()))
    if(mustericursor.count>0) {
        mustericursor.moveToFirst()
        val ID=mustericursor.getInt(mustericursor.getColumnIndex("Musteri_ID")).toInt()).toInt()
        val intent = Intent(applicationContext, UyeAnaSayfa::class.java)
        intent.putExtra( name: "ID",ID)
        startActivity(intent)
        finish()
    }
    else
    {
        val personelcursor=Database.rawQuery( sql: "SELECT * FROM personeller WHERE Personel_EPosta=? AND Personel_Sifre=?",
            arrayOf(UyeGirisEPosta.text.toString(),UyeGirisSifre.text.toString()))
        if(personelcursor.count>0)
        {
            personelcursor.moveToFirst()
            val ID=personelcursor.getInt(personelcursor.getColumnIndex("Personel_ID")).toInt())
            if(ID==1)
            {
                val intent = Intent(applicationContext, PersonelSayfa::class.java)
                intent.putExtra( name: "ID",ID)
                startActivity(intent)
                finish()
            }
            else
            {
                val intent = Intent(applicationContext, CalisanSayfa::class.java)
                intent.putExtra( name: "ID",ID)
                startActivity(intent)
                finish()
            }
```

Şekil6.3: Giriş Sayfa Kodları



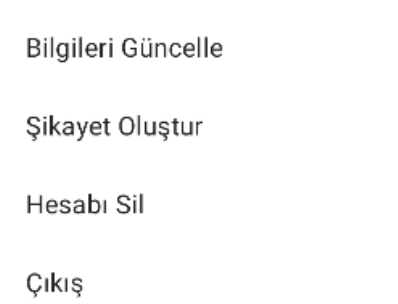
Şekil6.4: Müşteri Ana Sayfa

Müşteri sınıfına girerken sistem arka tarafta veri tabanı kontrolü yapmaktadır. Veri tabanı tablolarından park_halindeki_araclar tablosunda eğer giren müşterinin bir aracı bulunursa burada verileri yazdırılacaktır. Ama eğer bulunmazsa Şekil6.3 gibi bir ekran bizleri karşılayacaktır.



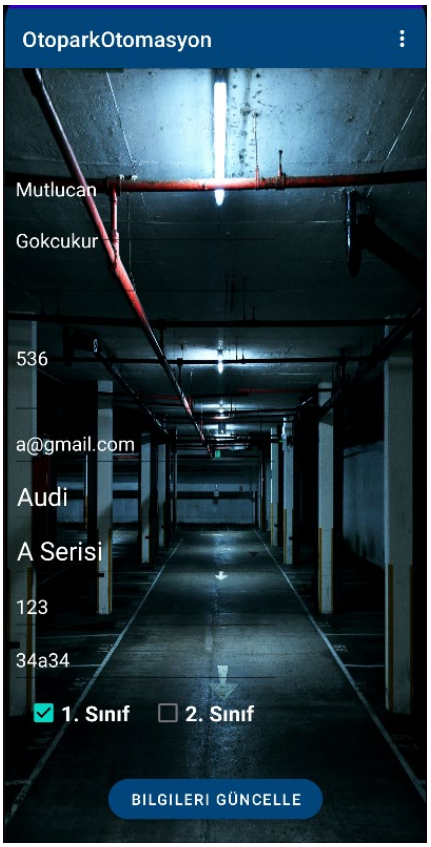
Şekil6.5: Müşteri Ana Sayfa 2

Eğer aracınızı önceden içeriye park ettiyseniz ve çıkış yapacaksanız veya arabanızın yerini öğrenmek isterseniz sizleri Şekil6.4 gibi bir ekran karşılayacaktır. Ayrıca aracınızı park ettiğiniz otoparkta çalışan personelleri de yine buradan görebilir ve herhangi bir problem durumunda onların telefonları ile kolayca iletişim sağlayabilirsiniz.



Şekil6.5: Müşteri Menü

Müşteriler kısmındaki menüde ise Şekil6.5'deki gibi işlemler yapılabilmektedir. Örneğin müşteri arabasını değiştirdiğinde ya da başka herhangi bir verisi değiştirmek istediğinde Şekil6.6 resminde göründüğü gibi bir ekranda kolayca bilgilerini veri tabanı üzerinden güncelleyebilecektir.



Şekil6.6: Müşteri Bilgi Güncelleme

Her yerde olduğu gibi bizim otoparkımızdan da memnun olmayanlar, şikayeti olanlar ve bu sitemlerini paylaşmak isteyen insanlar olabilir. Sistemimde bunu düşünerek ayrıca bir sikayetler tablosu oluşturdum ve bunu da Şekil6.5 menüsüne ekledim. Bu sayede müşteriler şikayetlerini bizlere kolay bir şekilde iletebilecekler.

Şekil6.7: Müşteri Şikayet Sayfası

Veri tabanımızda örnek olarak iki adet otopark bulunduğundan dolayı müşteriler ayrıca hangi otoparktan ve hangi personelden şikayetçi olduğunu kolay bir şekilde bizlere iletebilecekler.

Şekil 6.5 menüsünde görünen son iki madde ise temel maddelerden oluşmaktadır. Eğer bir müşteri kendi verisini silmek isterse öncelikle ondan bir onay alınır ve bu onaya bağlı olarak da ona bağlı olan aracı ve bilgileri veri tabanı tablolarından kolaylıkla silinir.

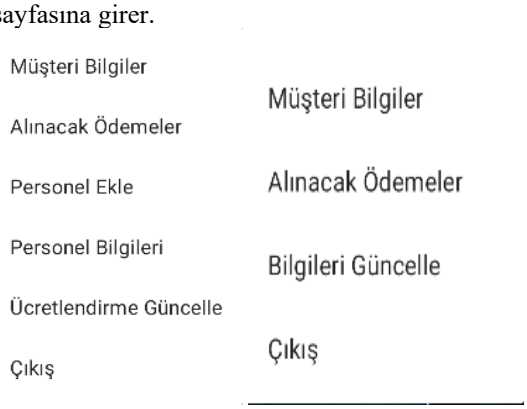
İşlemleri biten, daha fazla işlem yapmayacak müşteri ise çıkış diyerek hesabından çıkış yapabilir.

VII. MÜDÜR VE PERSONEL SAYFASI

Personel_ID	Otopark_ID	Personel_Adı	Personel_Soyadı	Personel_Telefon	Personel_EPosta	Personel_Sifre
Filtre	Filtre	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	Mutlucan	Gökçukur	536	mutlu@gmail.com	123mutlu
2	2	İsmail	Aydın	544	ismail@gmail.com	123ismail

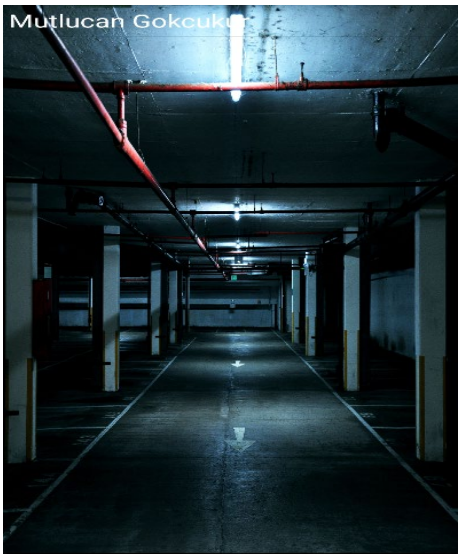
Şekil7.1: Personel tablosu

Veri tabanımız uygulamaya başlarken sisteme iki tane personel eklemektedir. Buradaki Personel_ID değeri 1 olan kişi ayrıca sistemde ‘Müdür’ yani en yüksek rütbeli çalışan olarak da görünmektedir. Sisteme giriş yapılırken personellerde aynı müşteriler gibi giriş yapmaktadırlar. Sistem öncelikle müşteriler tablosunu kontrol eder ve eğer orada istediği verileri bulamazsa personeller tablosuna gelir. Eğer veriler Personel_ID değeri 1 olan kişi ile eşleşiyorsa sistem müdür sayfasına girer, eşleşmiyor ise personel sayfasına girer.



Şekil7.2: Personeller Sayfa Farkı

Personeller sayfasındaki Personel_ID değeri 1 olan ve diğer personeller arasındaki temel farklar Şekil7.2’de görünmektedir. Personeller genel olarak sadece müşteriler ile ilgilenip, kendileri hakkında bilgileri güncelleyebilirken müdür tarafında ise personel ekleme ve ücret tarifelerini güncelleme hakkına sahiptir. Geri kalan menüler aynı sistematik şekilde çalışmaktadır.

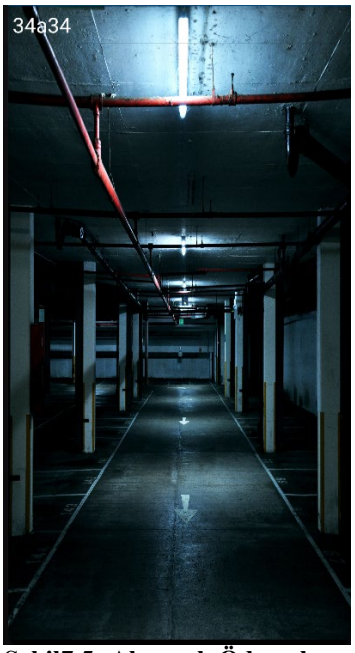


Şekil7.3: Müşteri Bilgiler Sayfası

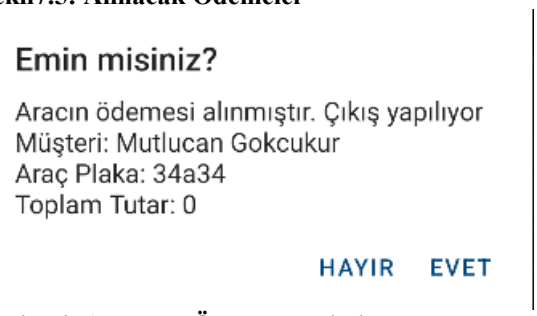
Personeller ve müdür Şekil7.2 menüsünde görünen 'Müşteri Bilgiler' kısmına basınca kendilerini böyle bir menü karşılamaktadır. Buraya müşteriler tablosunda kayıtlı olan tüm müşterilerin adı ve soyadı gelmektedir. Listeden seçilen herhangi bir personel olduğunda ise o müşterinin temel bilgilerini Şekil 7.4'deki gibi personellere gösterilmektedir.

Şekil7.4: Müşteri Bilgi

Alınacak Ödemeler kısmında ise müşteriler kısmından çıkış işlemini başarıyla tamamlamış ve bu işlem sonunda verileri Alınacak Ödemeler tablosuna eklenmiş verileri göstermektedir. Listede seçilen plakadan sonra ise personelleri Şekil 7.6 gibi bir ekran karşılamaktadır. Burada müşterinin ad soyad verisi, aracının plakası ve toplam tutarı görünmektedir. Eğer personel ücreti aldığını onaylar ve 'Evet' butonuna basarsa müşterinin park halindeki araç verisi silinir ve müşterinin işlemleri burada tamamlanır. Müşteri ileride tekrar gelmek ister ise aracı ve kendi verileri ise sistemimizde kayıt altında durmaya devam edecektir.



Şekil7.5: Alınacak Ödemeler

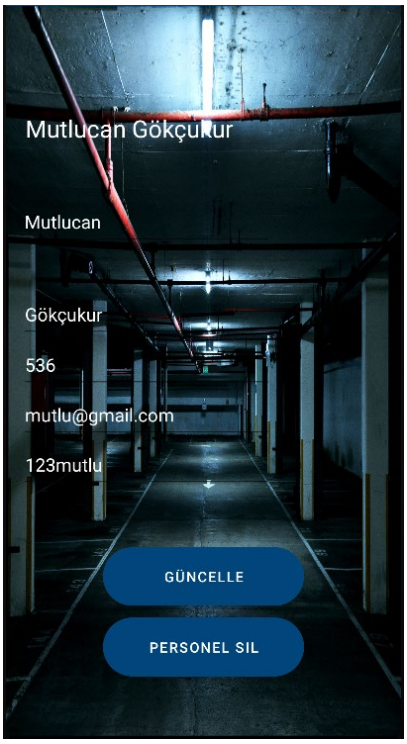


Şekil7.6: Alınacak Ödemeler Bilgi

Müdür kısmında bulunan 'Personel Ekle' sayesinde müdür kendisine yeni bir çalışan işe aldığında bunu kolaylıkla kaydedebilir ve müşteriler duruma göre bu verileri görebilir. Müdürün ayrıca yeni personeli hangi otoparka eklemek istediğini yine aynı sayfa içerisinde kendini belirleyebilmektedir. Ayrıca eklenen personel içinde ayrıca bir sistem oluşturulacağından dolayı oda sistemi kolayca kullanabilir ve müşteriler ile uygulama üzerinden iletişim kurabilirler.

Şekil7.7: Personel Ekleme

Personeller kendi verilerini sistemlerinde kolaylıkla güncelleyebilmektedir. Müdür kısmında ise personel güncellemeye ek olarak personeli silme hakkı vardır. Bu sayede işten müdürün işten çıkartmak istediği bir personel olduğunda onu kolaylıkla veri tabanından silebilmektedir.



Şekil7.8: Personel Bilgi Güncelleme ve Silme

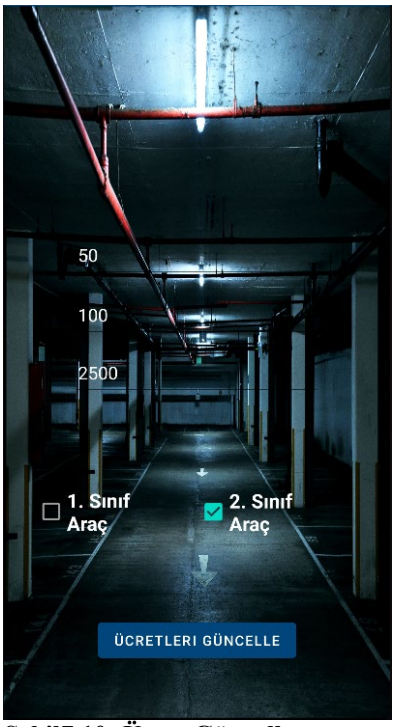
Bu otopark sisteminde her otoparkta olduğu gibi arabaları sınıflara ayırarak ücret hesaplamaktadır.

	Ucret_ID	Arac_Sınıf	Saatlik_Ucret	Gunluk_Ucret	Aylık_Ucret
	Filtre	Filtre	Filtre	Filtre	Filtre
1	1	1	20	50	1000
2	2	2	50	100	2500

Şekil7.9: Ücretler tablosu

Şekil7.8'de göründüğü gibi araçların saatlik, günlük ve aylık olmak üzere ücretleri ayrı ayrı hesaplanmaktadır. Günümüzde her türlü alanda günlük olarak her türlü ürünün fiyatına artış geldiğini düşünürsek otopark sisteminde de ücretlerin güncellenmesi gerektiğini düşünüyorum. Bu işlemi ise sadece kıdemli bir çalışan yapabileceğinden bunu müdür sayfasında yapabilmekteyiz.

Şekil7.9'da göründüğü gibi hangi araç sınıfının fiyatlandırmasını güncellemek istersek saatlik, günlük ve aylık olarak gelen verileri değiştirmemiz yeterli olacaktır.



Şekil7.10: Ücret Güncelleme

VIII. Projenin Ekibe Kattımış Olduğu Faydalar

Bu proje sayesinde artık daha kolay ve rahat bir şekilde Android Studio uygulamasını kullanabiliyorum. Bu sayede ileride yapacağım Android uygulamaları daha rahat bir şekilde yapabileceğime inanıyorum. Ayrıca SQLite veri tabanı kullanarak da verilerin saklanmasını ve veri işlemleri nasıl yapabileceğimi öğrendim. Yapacağım veri tabanında tabloların ve bu tablolar arasındaki bağlantıların önemli olduğunu fark ettim. Bu sayede bu projede varsa yapmış olduğum hatalarımı ilerideki projelerimde tekrarlamayacağıma inanıyorum.

IX. Kaynakça

[1] https://www.youtube.com/watch?v=7-d5mKjAJxc (05.05.2023)

[2] https://www.youtube.com/watch?v=N8GfosWTt44 (05.05.2023)

[3] https://www.sqlite.org/docs.html (25.04.2023)

[4] https://kotlinlang.org/docs/home.html (25.04.2023)