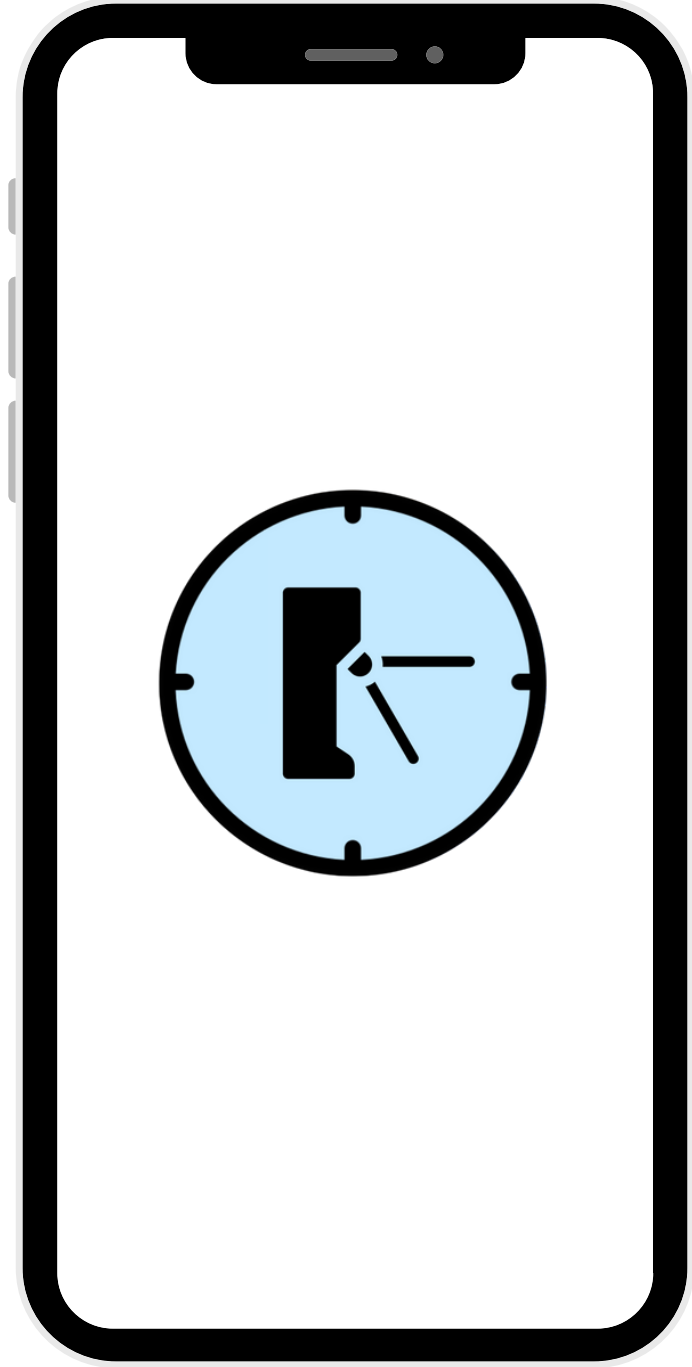


PDKS



TANITIM

Genel Tanım

PDKS, çalışanların fiziksel kart kullanmadan mobil cihazları üzerinden giriş ve çıkış işlemlerini gerçekleştirebildiği bir uygulamadır.

Özellikler

01 Kullanıcı Girişi

Sistem, önceden tanımlanmış kullanıcı ID ve şifre üzerinden giriş yapılmasını sağlar.

02 Takvim

Ana ekranda takvim bulunur; kullanıcı buradan ilgili tarihe tıklayarak son bir aya kadar olan kayıtları görüntüleyebilir.

03 İnsan Doğrulama

Kayıt atarken ve çıkış yaparken güvenlik amacıyla kamera açılır ve insan olduğu doğrulanır.

04 Giriş Çıkış Kaydı

Kullanıcı, harita üzerinde konumu bir blok içindeyse giriş/çıkış kaydı yapabilir; kayıt, insan doğrulaması başarılı olursa tamamlanır.

05 Harita Modları

Harita ekranında standart, uydu ve hibrit modlar bulunur.

06 Kayıt Görüntüleme

Kayıtlar, tarih, kayıt sırasında çekilen fotoğraf, saat ve kayıt yapılan blok bilgisi ile gösterilir.

07 Yetkili Terminaller

Kullanıcı, terminalleri koordinat, görsel ve kayıt atmak için gerekli minimum uzaklık bilgisi ile görüntüleyip bilgi alabilir.

08 VPN-Otomatik Saat Kontrolü

Kayıt sırasında VPN kapalı olmalı ve otomatik saat açık olmalıdır; aksi durumda kayıt atılamaz ve uyarı gösterilir.

09 Menü ve Ekran Geçişleri

Uygulama ekranlarında bulunan menü sayesinde kullanıcı, istediği ekrana geçiş yapabilir. Menüde kullanıcının sisteme kayıtlı ad ve soyadı ile fotoğrafı görüntülenir.

KULLANILAN TEKNOLOJİLER

Ortam

Android Studio

Diller

Java
XML

Veritabanı

SQLite

Araç

Canva
Android Asset Studio

Harita

OSMDroid

Doğrulama

ML Kit
JavaMail API

GELİŞTİRME SÜRECİ

Intent Diyagramı

Öncelikle proje kapsamında uygulamanın sahip olacağı özellikler analiz edilmiştir. Daha sonra uygulamanın yapısı kurgulanmış, hangi ekranlarda Fragment ve hangi ekranlarda Activity kullanılacağı planlanmıştır.

Uygulama, aşağıdaki intent diyagramında da gösterildiği üzere şu ekranlardan oluşmaktadır:

- 1.LoginActivity
- 2.MainActivity
- 3.MainFragment
- 4.MapFragment
- 5.CameraActivity
- 6.SettingsFragment
- 7.DetailsFragment
- 8.TerminalFragment

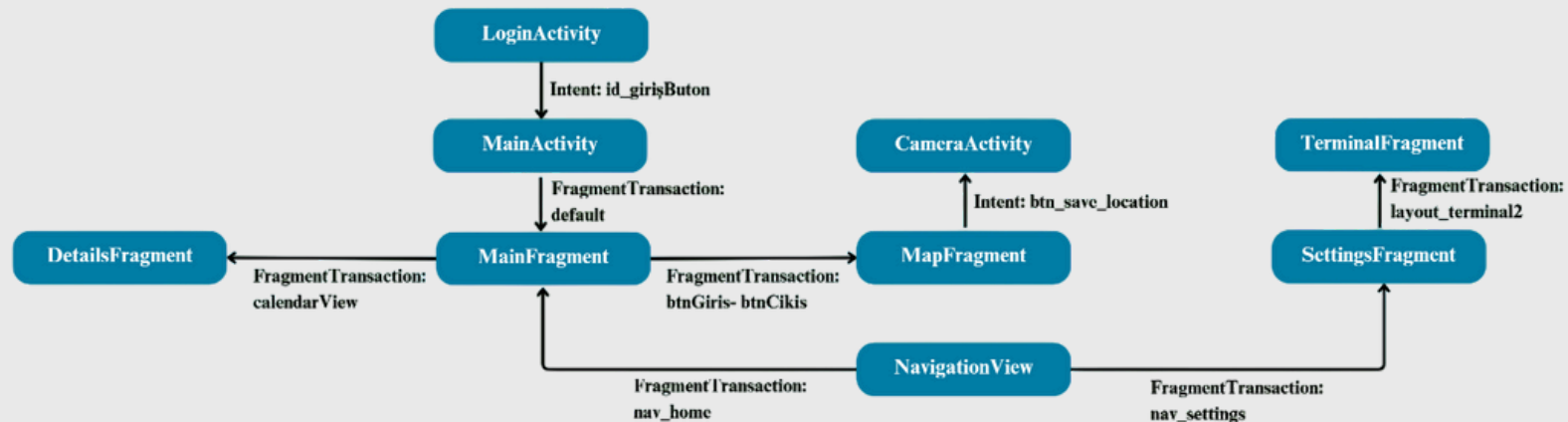
Uygulamanın daha verimli çalışması ve menünün birçok ekranda görüntülenebilmesi için iskelet MainActivity üzerine kurulmuştur. Kullanıcı tercihinə göre ilgili fragmentler bu iskelet üzerinde açılmaktadır.

Kullanıcı, LoginActivity ekranında giriş bilgilerini girdikten sonra id_girişButon id'sine sahip butona basarak MainActivity ekranına geçer. Burada varsayılan olarak MainFragment açılmaktadır.

- Kullanıcı MainFragment üzerinden takvimde ilgili güne tıkladığında DetailsFragment ekranı açılır.
- MainFragment üzerindeki btnGiris veya btnCikis butonlarına tıklanması durumunda MapFragment açılır.
- MapFragment üzerinden btn_save_location butonuna tıklanarak CameraActivity ekranına geçiş yapılır.

Kullanıcı ayrıca NavigationView üzerinden menü öğelerine tıklayarak farklı fragmentlere geçebilir:

- nav_home öğesine tıkladığında MainFragment açılır.
- nav_settings öğesine tıkladığında SettingsFragment açılır.
- SettingsFragment ekranında layout_terminal2 öğesine tıkladığında TerminalFragment açılır.



Şekil 1. Intent Diyagramı

İkonlar, Tipografi, Renk Paleti

Proje yapısı belirlendikten sonra uygulama ikonu tasarımı yapılmıştır. Ardından kullanılacak ikonlar, tipografi ve renklere karar verilmiştir.

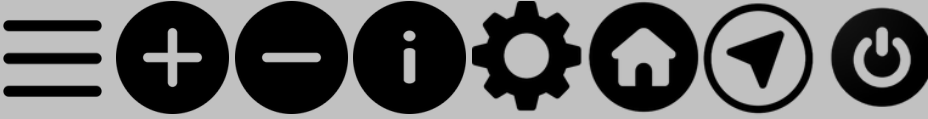
Uygulama ikonu



Şekil 2. PDKS İkonu

İkonun en dışında saat, içinde ise işe giriş çıkışlarda kart basmayı simgeleyen turnike simgesi kullanıldı. Turnikenin iki kolunun yelkovan ve akrebi temsil etmesiyle ikonun uygulamanın amacıyla uyumlu olması sağlandı.

Kullanılan İkonlar



Şekil 3. İkonlar

Yazı Tipi

- Roboto- Bold
- Roboto- Regular

Renk Paleti

#FFD180	#BC3E51	#E52022	#D8DADC	#888888	#3C3C3C	#FF000000
#D2F4B9	#90CAF9	#4D9EE2	#0090FF	#DEAE50	#CC9C3D	#FFFFFF

Şekil 4. Renk Paleti

Ekran Tasarımları



Uygulama ikonu tasarımı yapıp kullanılacak ikonlar ve tipografiler belirlendikten sonra ekran tasarımlarına geçilmiştir. Bu süreçte, şirketin mevcut olarak kullandığı PDKS uygulaması tasarımı göz önünde bulundurulmuş olup kullanıcı deneyiminin anlaşılır olmasına, arayüzün görsel olarak çekici olmasına ve kullanıcıların uygulamayı rahatlıkla kullanabilmesine özen gösterilmiştir.

Veritabanı Yapısı

Uygulamada veriler SQLite veritabanı ile saklanmaktadır. Veritabanı iki temel tabloya sahiptir:

- kullanıcılar: Kullanıcı bilgilerini tutar (entity_id, ad, soyad, sifre).
- kayıtlar: Kullanıcıların giriş-çıkış kayıtlarını saklar (entity_id, blok_adi, tarih, saat, tip, photo).

DatabaseContract sınıfı tablo ve sütun adlarını tanımlar, DatabaseHelper sınıfı ise veritabanını oluşturur, tablo yapısını yönetir ve örnek kullanıcı verilerini ekler.

kullanıcılar	kayıtlar
ID  INT	ID  INT
entity_id TEXT	entity_id TEXT
ad TEXT	blok_adi TEXT
soyad TEXT	tarih TEXT
sifre TEXT	saat TEXT
	tip TEXT
	photo BLOB

Şekil 5. ER Diyagramı

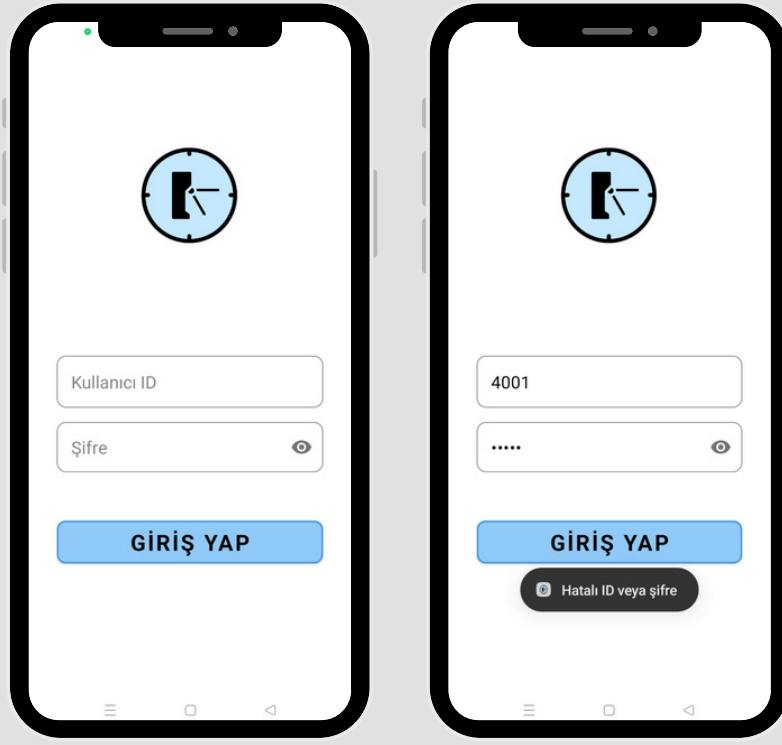
Kod Yapısı

Öncelikle kullanılacak renkler color.xml dosyasında tanımlandı. Ardından kullanılacak fontlar Google Fonts'dan çekilerek projeye aktarıldı. Kullanılacak ikonlar mipmap ve drawable klasörlerine aktarıldı. Mipmap'e aktarılan ikonlar Android Asset Studio da boyutlandırılıp ilgili klasörlere aktarılmıştır.

LoginActivity

LoginActivity'de kullanıcı uygulamaya giriş işlemini gerçekleştirmektedir. Burada önce frontend tarafında kullanıcı arayüzü tasarlandı: kullanıcı ID ve şifre için EditText alanları, uygulama ikonu için ImageView ve giriş butonu yerleştirildi.

Backend kısmında ise veritabanı üzerinden cursor ile kullanıcılar tablosu sorgulanarak girilen ID ve şifre kontrol edildi. Doğrulama başarılıysa kullanıcı bilgileri Intent ile MainActivity'ye taşındı, başarısızsa Toast mesajı ile uyarı verildi.



Şekil 6. Giriş Ekranı

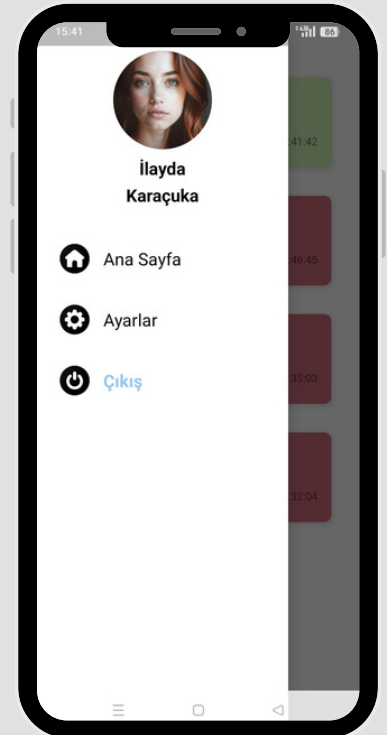
MainActivity

MainActivity uygulamanın yönetildiği ana aktivitedir. İlk başta frontend tarafında uygulama menüsü oluşturuldu. DrawerLayout ve NavigationView kullanılarak yan menü tasarlandı, ekranın ortasında fragmentlerin yerleştirileceği FrameLayout ve menüyü açan ImageView eklendi. Menü öğeleri options_menu.xml ile tanımlandı. Menü üzerinde kullanıcı fotoğrafı, ad ve soyadın görüntülenebilmesi için navigation_header adında bir dosya oluşturuldu ve header kısmına ImageView ile TextView'ler eklenerek tasarımı yapıldı.

Menü öğelerine tıklandığında fragment geçişlerini sağlamak için replaceFragment metodu oluşturuldu. Bu metod ile seçilen menü öğesine göre MainFragment veya SettingsFragment ekrana yüklendi ve ENTITY_ID bilgisi fragmentlere aktarıldı.

Menüye çıkış seçeneği eklendi. Bu seçenek seçildiğinde özel tasarlanmış logout_box.xml üzerinden bir uyarı açılmakta, kullanıcıya iptal veya çıkış seçenekleri sunulmaktadır. Çıkış işlemi öncesinde kullanıcı doğrulaması için CameraActivity çalıştırılmakta, doğrulama başarılı olduğunda SharedPreferences temizlenerek oturum sonlandırılmakta ve LoginActivity ekranına yönlendirilmektedir.

Ayrıca onResume metodu ile NavigationView header kısmındaki kullanıcı fotoğrafı oluşturulan ImageStorage sınıfı üzerinden alınarak dinamik olarak güncellenmektedir.



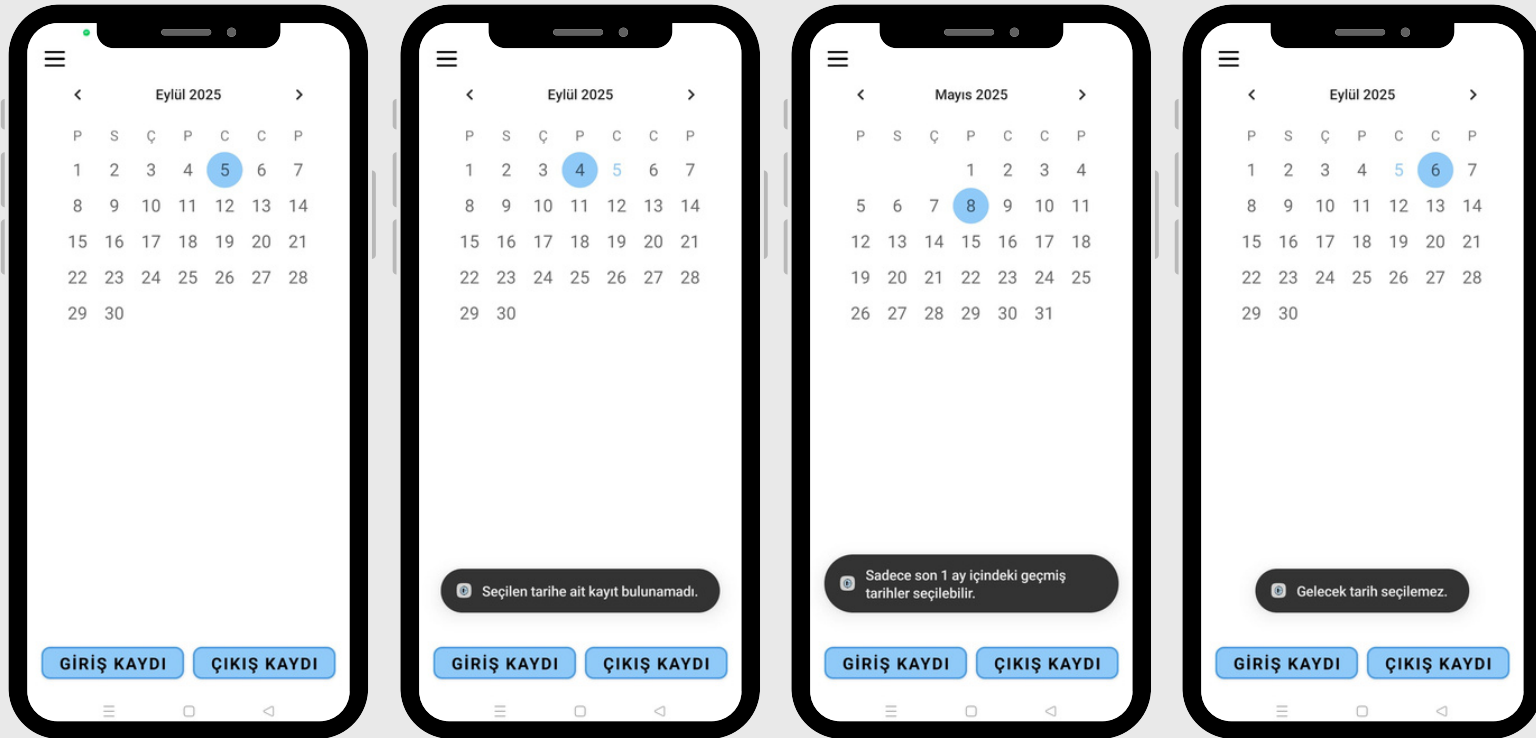
Şekil 7. Hamburger Menü

MainFragment

MainFragment, kullanıcıya tarih seçimi ve giriş-çıkış kayıtlarını yönetme imkânı sunan bir fragmenttir. Arayüzde üst kısımda takvim bulunur ve kullanıcı yalnızca son bir ay içindeki geçmiş tarihlerden seçim yapabilmektedir. Uygun tarih seçildiğinde ilgili kayıt veritabanından kontrol edilir ve varsa DetailsFragment ile ekrana yüklenir.

MainFragment'te kullanıcı arayüzü LinearLayout ile dikey olarak tasarlanmıştır. Takvim görünümü için CalendarView kullanılmıştır. Ek olarak "Giriş Kaydı" ve "Çıkış Kaydı" textli butonlar yerleştirilmiştir.

CalendarView oluşturulmuş ve kullanıcıların seçtiği tarihin geçerli olup olmadığı, yani gelecekteki tarihler veya son bir aydan eski tarihler olup olmadığı kontrol edilmiştir. Tarihe ait bir kayıt var mı diye veritabanı üzerinden cursor ile sorgu yapılmış ve kayıt varsa FragmentTransaction ile DetailsFragment'e geçiş sağlanmıştır. Ayrıca giriş ve çıkış işlemleri için Button bileşenleri eklenmiş, tıklama olayında MapFragment'e geçiş yapılmış ve hangi işlemin yapılacağını belirtmek ve MapFragment'taki butona aktarmak için "GIRIS" veya "CIKIS" bilgisi type parametresi olarak gönderilmiştir.

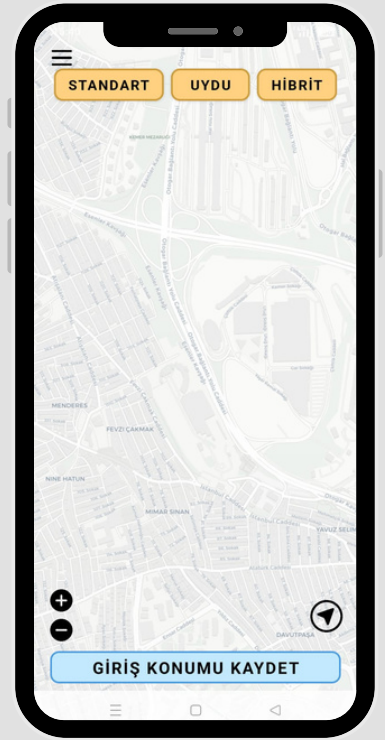
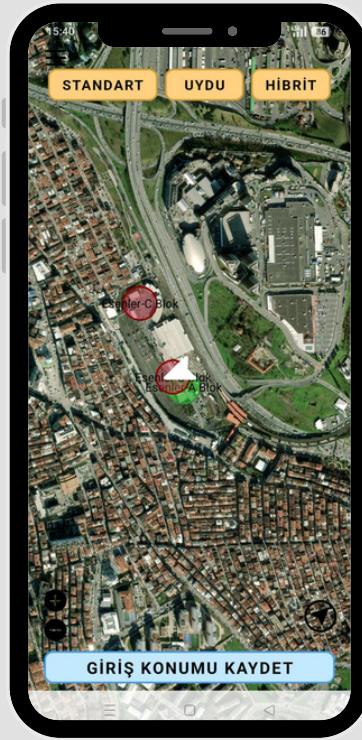


Şekil 8. MainFragment

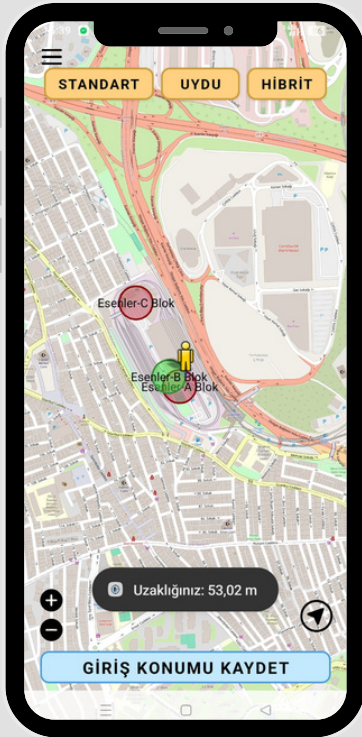
MapFragment

MapFragment'te kullanıcı konum bulma, kayıt atma, harita görüntüleme işlemlerini gerçekleştirebilir. Arayüzde MapView ile harita gösterilmiş, zoom ve harita tipi butonları eklenmiş, konum bulma için ikon yerleştirilmiştir.

Harita için osmdroid kütüphanesi kullanılmış ve varsayılan harita olarak TileSourceFactory.MAPNIK ile standart görünüm sağlanmıştır. Farklı harita türleri için TileSourceFactory ve XYTileSource kullanılarak uydu ve hibrit görünüm eklenmiştir. Kullanıcının mevcut konumu göstermek ve takip etmek için MyLocationNewOverlay ve FusedLocationProviderClient kullanılmıştır. Harita üzerindeki belirli blokların koordinatları StreetView'dan alınmış, bu koordinatlar merkez kabul edilerek Polygon ile daireler çizilmiş ve Overlay kullanılarak etiketler eklenmiştir. Kullanıcının seçtiği dairenin yeşil olması sağlanmış olup o dairenin merkezine uzaklığı Toast mesajı olarak gösterilmektedir. Eğer kullanıcının konumu o dairelerden birindeyse kayıt butonu aktifleşmektedir. Konum doğrulaması ve kayıt işlemleri, kullanıcının blok içindeki mesafesi kontrol edilerek yapılmıştır. Kayıt için veritabanına ContentValues ile giriş-çıkış bilgisi ve fotoğraf kaydedilmiştir. VPN kontrolü için ConnectivityManager, otomatik saat kontrolü için ise Settings.Global.AUTO_TIME kullanılmıştır.



Şekil 9. MapFragment Görünümleri



Şekil 10. MapFragment Uyarılar

CameraActivity

Bu ekranda, kullanıcının kayıt ve çıkış işlemlerinde yüz doğrulaması yapmak için ML Kit kütüphaneleri kullanılmıştır. Kamera erişimi CameraX API ile sağlanmış ve gerekli izinler Manifest dosyasında tanımlanmıştır. Kamera önizlemesi üzerinde grafik çizimleri yapmak için GraphicOverlay kullanılmıştır. Bu yapı, View sınıfından türetilmiş olup ölçeklendirme ve koordinat dönüşümü işlevlerini desteklemektedir.

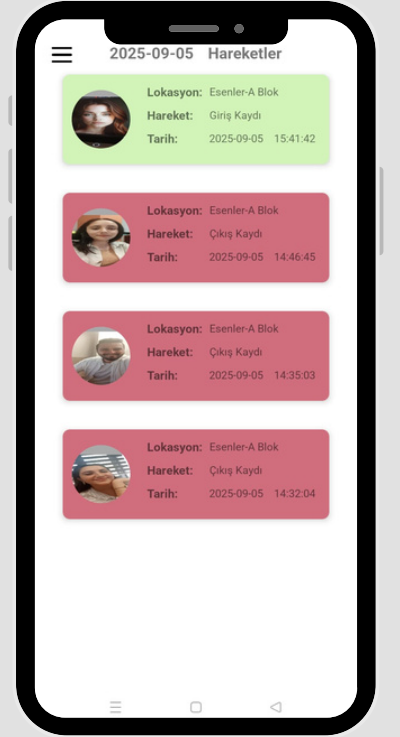
Ayrıca, utils paketinde yer alan CameraUtils, kameradan alınan koordinatların ekrana dönüştürülmesini sağlamıştır. camera paketinde oluşturulan BaseCameraAnalyzer, kamera verilerini ML Kit ile işleyip GraphicOverlay üzerinde görüntülemiş, CameraAnalyzer ise yüz algılama işlemini gerçekleştirmiştir. CameraManager ise kameranın açılıp kapatılmasını yönetmektedir.

DetailsFragment

Bu ekranda seçilen tarihe ait giriş-çıkış hareketleri listelenmiş, kullanıcıya ait lokasyon, hareket tipi, tarih, saat ve varsa kullanıcı fotoğrafı kart tasarımı içerisinde gösterilmiştir.

Listeleme işlemi için RecyclerView kullanılmış, her kayıt için özel bir kart tasarımı oluşturulmuş, bu amaçla details_card dosyası oluşturulmuş ve içerisinde TextView ile lokasyon, hareket, tarih ve saat bilgileri, ImageView ile kullanıcı fotoğrafı gösterilmiştir.

Veritabanından seçilen tarihe ve kullanıcıya ait kayıtlar Cursor ile çekilmiş, KayitModel sınıfı ile modele aktarılmış, fotoğraflar byte[] formatından Bitmap'e dönüştürülmek üzere ImageStorage sınıfı kullanılmıştır. DetailsAdapter aracılığıyla veriler RecyclerView'a bağlanmış, her bir kartta içerik doldurulmuş ve giriş-çıkış tipine göre arka plan rengi değiştirilmiştir.



Şekil 11. DetailsFragment

SettingsFragment

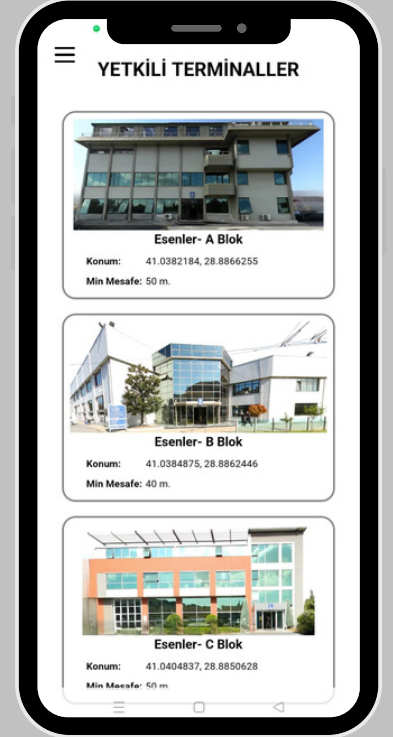
Bu ekranda FrameLayout oluşturulup içerisine LinearLayout oluşturulmuştur. Bu layout içinde kullanıcıya "Yetkili Terminaller" başlığı ile bir seçenek sunulmuştur. Kullanıcı bu alana tıkladığında bir OnClickListener ile TerminalFragment açılması sağlanmıştır.

TerminalFragment

Bu ekranda kullanıcıya yetkili terminallerin listesi sunulmuştur. Ekran, RecyclerView kullanılarak yapılandırılmıştır. Kartların tasarımı terminal_card.xml dosyasında tanımlanmıştır. Kartın içinde terminalin görseli ImageView, adı TextView, koordinat bilgisi ve minimum mesafe bilgisi de yine TextView ile gösterilmiştir.

Terminal verileri Terminal adlı sınıf ile tanımlanmıştır. Her terminalin görsel kaynağı, adı, koordinat bilgisi ve mesafesi bu sınıfta saklanmıştır.

TerminalAdapter sınıfı oluşturulmuş, bu adapter Terminal nesnelerini alarak RecyclerView'a bağlamıştır. Adapter içinde kartın her bir bileşeni ViewHolder ile referanslanmış ve bind edilmiştir. Böylece her terminalin bilgisi uygun kart üzerinde görüntülenmiştir. TerminalFragment içinde ise RecyclerView, GridLayoutManager ile dikey liste olarak ekrana yerleştirilmiş ve önceden tanımlanmış terminal verileri listeye eklenmiştir. Ayrıca fragment içinde geri tuşuna basıldığında kullanıcı SettingsFragment ekranına yönlendirilmesi sağlanmıştır.



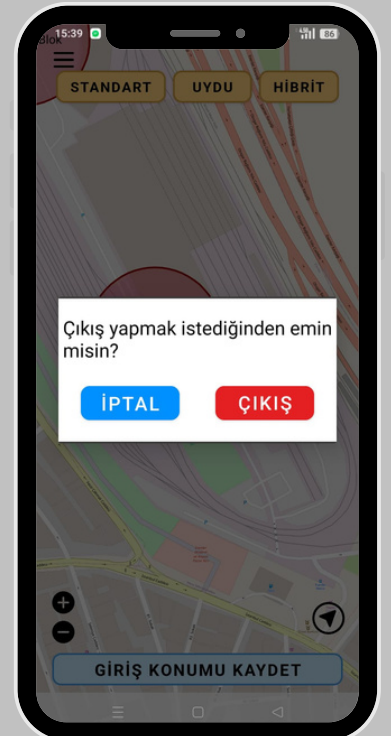
Şekil 12. TerminalFragment

Çıkış İşlemi

Menüden "Çıkış Yap" seçildiğinde çıkış için onay kutusu gösterilmekte ve çıkış öncesinde kamera ile insan doğrulaması yapılmaktadır.

Frontend tarafında, logout_box.xml ile basit bir onay penceresi tasarlanmış olup bu pencerede "İptal" ve "Çıkış Yap" butonları gösterilmektedir.

Backend tarafında, çıkış süreci MainActivity üzerinden yönetilmektedir. Kullanıcı çıkış seçtiğinde CameraActivity çağrılmakta ve ML Kit ile doğrulama yapılmaktadır. Doğrulama başarılı olduğunda logout metodu çalıştırılmakta ve kullanıcı LoginActivity ekranına yönlendirilmektedir. Başarısız doğrulama durumunda ise kullanıcıya Toast mesajı gösterilmektedir.

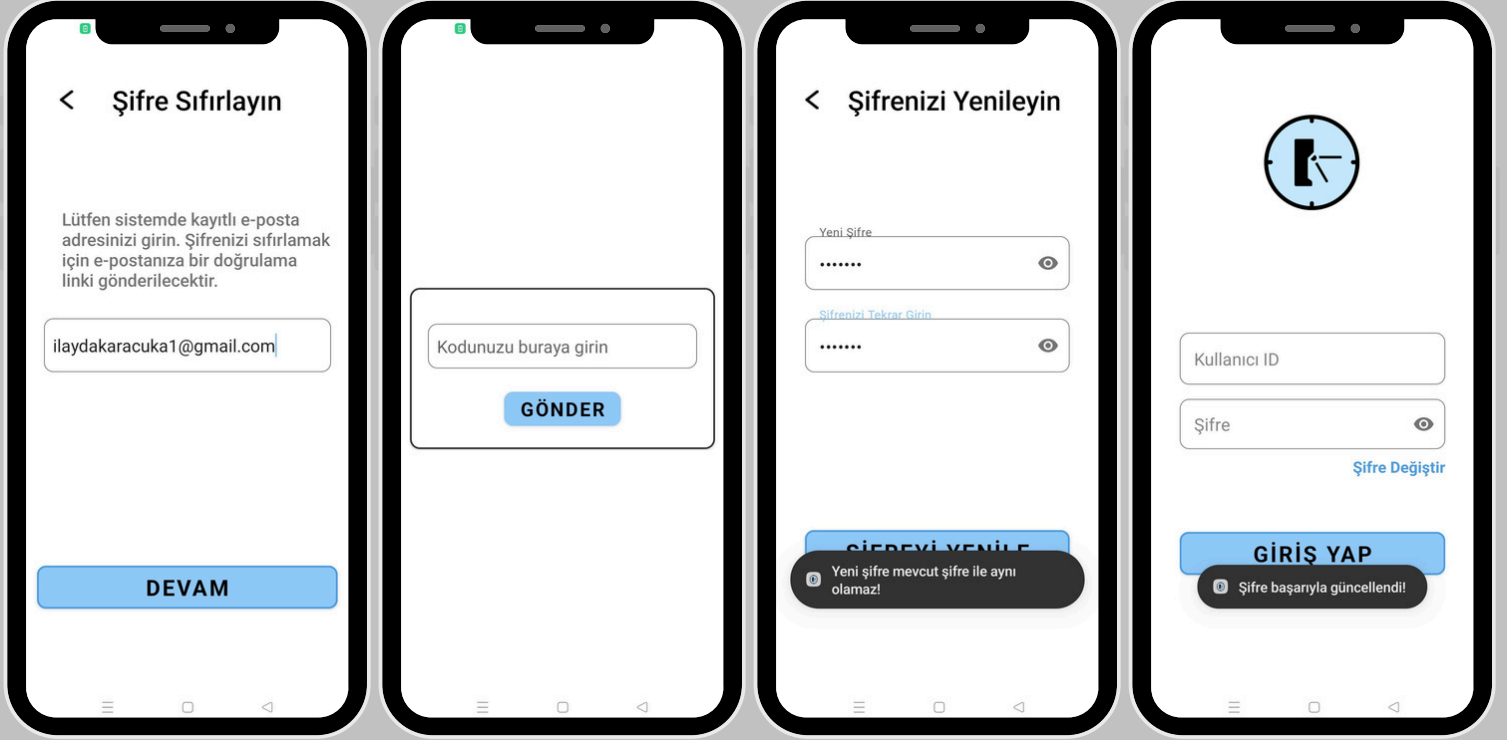


Şekil 14. Çıkış Yapma

Yeni Şifre Belirleme

Kullanıcı, LoginActivity'deki "Şifre Değiştir" metnine tıkladığında VerifyMailActivity açılır. Kullanıcı burada mailini girer; eğer mail veritabanında kayıtlıysa sisteme doğrulama kodu gönderilir. Kullanıcı doğru kodu girdiğinde ChangePasswordActivity'e yönlendirilir ve yeni şifresini belirleyebilir.

Bu süreçte kullanıcı bilgileri SQLite veritabanında tutulmuş, doğrulama kodu üretimi Java'nın Random sınıfı ile sağlanmış, e-posta gönderimi için JavaMail API kullanılarak Gmail'in SMTP servisi üzerinden mail gönderilmiştir. Arka planda e-posta gönderim işlemini gerçekleştirmek için AsyncTask sınıfından yararlanılmıştır. Ayrıca Android tarafında Activity yapısı, Intent geçişleri, Toast mesajları, TextInputEditText, Button ve ImageButton gibi arayüz bileşenleri kullanılmıştır. Şifre güncelleme işlemleri ContentValues ve SQLiteDatabase sınıfları aracılığıyla yapılmıştır.



Şekil 13. Şifre Değiştirme