T.C. İSTANBUL AREL ÜNİVERSİTESİ

Bilgisayar Mühendisliği Bölümü

Bitirme Projesi

Yüz ve Nesne Tanıma Tabanlı Self Kasa Sistemi

Doruktan KARAKURT

210309018

Danışman

Dr. Öğr. Üyesi Sibel BİRTANE AKAR

Proje Bilgisi

Dosya adı	210309018_Doruktan_Karakurt_Bitirme_Projesi_Rapor
Toplam Sayfa Sayısı	24

Sürüm	Yazarlar	Yorum	Tarih
İlk taslak	Doruktan Karakurt	İlk araştırmalar yapıldı yol haritası çizildi.	18.10.2024
Son Taslak	Doruktan Karakurt	Sistem mimarisi ve sistem ayağa kaldırıldı.	15.10.2024
Versiyon 1	Doruktan Karakurt	Tkinter ile masaüstü uygulaması olarak sistem tamamlandı.	20.11.2024
Versiyon 2	Doruktan Karakurt	QWebengine ve PyQT5 kullanılarak arayüz ve yeni sistem kuruldu.	20.05.2025

Yayın Onayı

Sibel BİRTANE AKAR		
Ad Soyad	**Ad Soyad**	**Ad Soyad**
Tez danışmanı	Tez koordinatör	Tez Koordinatör Yardımcısı

HAZIRLIK VE İNCELEME ONAYI

Doruktan KARAKURT

Öğrenci Ad Soyad

1.Giriş

Perakende sektöründe kasiyersiz self-kasa konsepti, müşterinin barkod okutmadan ürünü raftan alıp doğrudan çıkış yapabilmesini hedefler. Amazon Go mağazaları bu yaklaşımı ticarileştirse de yüksek donanım maliyeti ve kapalı kaynak algoritmalar, üniversite-sanayi ölçeğinde uygulanabilir çözümler geliştirmeyi gerekli kılmıştır. Bu bitirme projesi, düşük maliyetli tek kamera düzenekleri üzerinde **yüz tanıma ile kimlik doğrulama** ve **SSD-MobileNetV2 ile nesne (ürün) tanıma** yöntemlerini birleştirerek gerçek-zamanlı, kasiyersiz **Self-Kasa Yönetim Sistemi** tasarlamayı amaçlar. Sistem iki ardışık sürümde (Versiyon 1: Tkinter masaüstü; Versiyon 2: PyQt5 + QWebEngine kiosk) evrimleştirilmiş; ikinci sürümde hem işlem süresi hem de kullanıcı deneyimi açısından önemli iyileştirmeler sağlanmıştır.

1.1. Projenin Amacı

Bu projenin temel amacı, Amazon Go gibi gelişmiş market sistemlerinden esinlenerek geliştirilen bir yapay-görme tabanlı self-servis ödeme sistemini hayata geçirmektir. Kamera aracılığıyla müşterinin yüzünü tanıyan ve veritabanında kayıtlı olan müşteri bilgilerini doğrulayan sistem, müşterinin tanındığını onayladıktan sonra ürün tanıma modülüne geçer. Burada ürün, sistem tarafından görüntü üzerinden algılanır ve fiyatlandırma otomatik olarak gerçekleştirilir. Böylece kasiyer veya eleman ihtiyacı ortadan kaldırılarak işletmeler için personel maliyetleri düşürülürken, müşterilere hızlı ve temassız bir alışveriş deneyimi sunulur.

Açık kaynaklı Python kütüphaneleri ve uygun maliyetli donanımlar kullanılarak geliştirilen bu sistem, kullanıcı dostu bir arayüzle tamamlanarak hem işletmeler hem de müşteriler için pratik ve erişilebilir bir çözüm sağlamayı amaçlamaktadır.

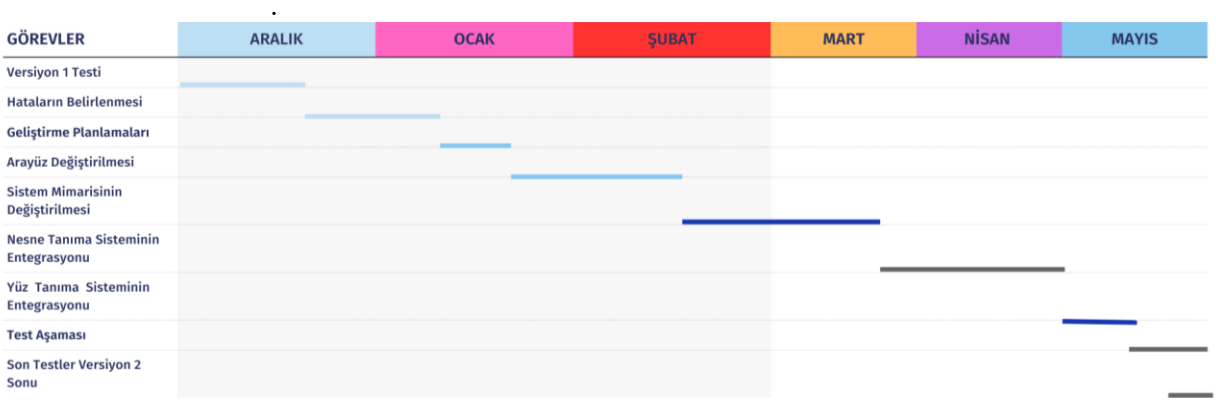
1.2. Projenin Kapsamı, Çıktıları ve Tablolar

Bu projenin kapsamı, bir market ortamında müşterilerin self-servis şekilde alışveriş yapabilmelerine olanak sağlayan bir sistem geliştirmeyi içermektedir. Sistem, yüz tanıma ve nesne tanıma modülleriyle donatılmış, Python tabanlı yapay görme algoritmaları kullanılarak tasarlanmıştır. Müşteri verilerinin kaydı, yüz tanıma ile müşteri eşleşmesi ve ürün algılandıktan sonra bakiyeden düşüş işlemi gibi tüm aşamalar yazılım tarafında kontrol altına alınmıştır.

Projenin çıktıları arasında, kayıtlı müşterilerin yüz tanıma işlemiyle tanımlanması, nesne tanıma yoluyla alınan ürünlerin otomatik fiyatlandırılması ve bakiyelerinin güncellenmesi gibi temel işlevler bulunmaktadır. Ayrıca, PyQt tabanlı kullanıcı arayüzüyle desteklenen sistem; müşterilerin hızlı, temassız ve güvenilir bir alışveriş deneyimi yaşamalarını sağlarken, işletmeler için kasiyer ve eleman maliyetlerini düşürmeyi hedeflemektedir. Geliştirilen çözüm; özelleştirilebilir fiyat tabloları, müşteri bakiyesi güncellemeleri ve anlık işlem bildirimleri gibi detaylı özelliklerle de tamamlanarak, son kullanıcı ve işletme açısından pratik bir araç haline getirilmiştir.

Projenin geliştirilmesi için yapılan çalışmalar, **Gantt şemalarında (Şekil 1.1 ve Şekil 1.2)** gösterildiği gibi iki ana versiyon halinde yürütülmüştür. İlk versiyon aşamasında planlama ve araştırma süreçleriyle başlandıktan sonra, temel yüz tanıma sistemi geliştirilmiş ve test edilmiştir. Bunu takiben nesne tanıma sistemi kurulmuş ve sistem genelindeki ilk testler başarıyla tamamlanmıştır. Ayrıca hızlandırma denemeleri ve performans testleriyle versiyon 1 sonlandırılmıştır. Versiyon 2 aşamasında, versiyon 1'de tespit edilen hataların düzeltilmesi ve eksik kalan kısımların tamamlanması sağlanmıştır.

Şekil 1.1 - Proje Bölüm I Zaman Çizelgesi.



Şekil 1.2 - Proje Bölüm I Zaman Çizelgesi.

1.3 Projenin Hedefleri ve Başarı Kriterleri

i. Yüz tanıma teknolojisi kullanılarak müşterilerin hızlı ve doğru bir şekilde tanımlanması sağlanacaktır.

ii. Nesne tanıma sistemi ile müşterilerin tuttukları ürünlerin doğru şekilde tanımlanması hedeflenmektedir.

iii. Satın alınan ürünlerin fiyatlarının otomatik olarak müşterinin bakiyesinden düşmesi sağlanacaktır.

iv. Kasiyersiz self-servis kasa deneyimi kolaylaştırılacak ve hızlandırılacaktır.

v. Mağaza veya spor salonu gibi ortamlarda kasiyer ihtiyacını azaltarak işletme maliyetleri düşürülecektir.

vi. Sistem performansının yüksek tutulması ve bekleme sürelerinin minimuma indirilmesi amaçlanmaktadır.

vii. Yüz tanıma ve nesne tanıma işlemlerinin %90'ın üzerinde doğruluk oranına ulaşması hedeflenmektedir.

viii. Sistemin 5 saniyeden kısa sürede kullanıcıyı tanıyıp nesneyi tespit edebilmesi sağlanacaktır.

ix. Farklı ışık ve ortam koşullarında güvenilir şekilde çalışması hedeflenmektedir.

x. Kullanıcıların karşılaştığı hataların minimum seviyede olması hedeflenmektedir.

1.4 Proje Üyelerinin Görev ve Sorumlulukları

Tablo 1.4 Proje üyeleri, sorumlulukları ve görevleri

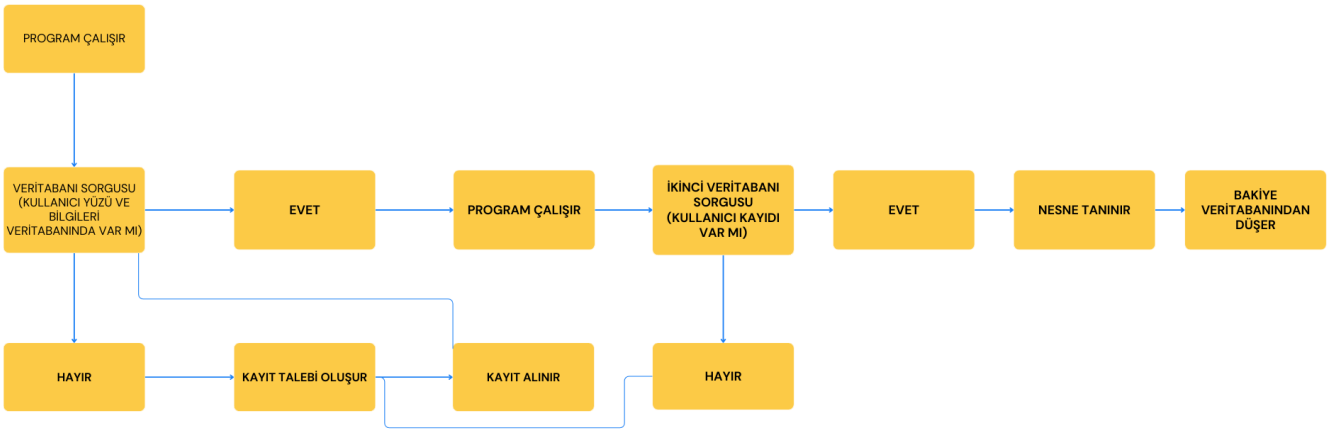
Proje Üyesi	Sorumluluklar ve Görevler
Doruktan Karakurt	Araştırma, sistem mimarisi, arayüz tasarımı, geliştirmeler, testler.

2. Yapılan Çalışma

Projenin geliştirilmesi sürecinde öncelikle yüz tanıma ve nesne tanıma sistemleri için temel algoritmalar belirlenmiş ve bu algoritmalar Python programlama dili kullanılarak hayata geçirilmiştir. Başlangıç aşamasında Tkinter kütüphanesi kullanılarak masaüstü tabanlı bir arayüz tasarlanmış ve yüz tanıma ile nesne tanıma işlevleri entegre edilmiştir. Bu ilk versiyonda yüz tanıma sürecinde threading ve multithreading yöntemleri denenmiş, performans sorunları gözlemlenmiştir.

Daha sonra, projenin kullanım senaryolarını ve kullanıcı deneyimini iyileştirmek için PyQt5 kütüphanesi ve QWebEngineView tabanlı yeni bir arayüz tasarlanmıştır. Bu ikinci versiyon geliştirme sürecinde, ilk versiyonda tespit edilen performans ve stabilite sorunları giderilmiş, yüz tanıma ve nesne tanıma algoritmalarının doğruluk oranları artırılmıştır. Nesne tanıma modülü, COCO veri seti kullanılarak yeniden optimize edilmiştir.

Ayrıca, projenin kullanıcı dostu bir hale getirilmesi amacıyla veri tabanı yönetimi güncellenmiş ve müşteri bilgileri düzenlenerek kolay erişilebilir bir yapıya kavuşturulmuştur. Projenin farklı aşamalarında yapılan iyileştirmeler; performans optimizasyonu, veritabanı entegrasyonu, arayüz geliştirme, kamera erişim süreçlerinin iyileştirilmesi ve kullanıcı etkileşimlerinin kolaylaştırılması gibi konuları kapsamaktadır. Sonuç olarak, proje iki farklı versiyon halinde tamamlanarak kullanıma hazır hale getirilmiştir. Akış diagramında **(Şekil 2.1)** süreç görselleştirilmiştir.



Şekil 2.1- Flow Diagram

2.1 Yüz Tanıma Algoritması

Projenin ilk aşamasında, kameradan alınan görüntüler kullanılarak yüz tanıma işlemi yapılmaktadır. Bu işlem, OpenCV ve Face Recognition kütüphaneleri aracılığıyla gerçekleştirilir. Veritabanı sorgusu için öncelikle `yuz_tanimla()` fonksiyonu çağrılır. Bu fonksiyon, kayıtlı yüz fotoğrafları ile kameradan alınan yüzü karşılaştırır ve eğer eşleşirse müşteri ID'sini döndürür.

Fonksiyonlar:

- `yuz_tanimla()`:
    - Kamerayı açar ve yüz fotoğrafı yakalar.
    - Veritabanındaki yüz fotoğraflarını karşılaştırır.Yüz tanınırsa müşteri ID'sini döndürür

2.2 Nesne Tanımlama Algoritması

Yüz tanıma tamamlandıktan sonra, kullanıcı elindeki nesneyi gösterir.Bu aşamada **COCO** sınıflandırmasıyla eğitimli olan **SSD Mobilenet** modelimiz devreye girer ve nesne tanıma işlemini gerçekleştirir.`nesne_tanima()` fonksiyonu çalıştırılır ve nesne tespit edilirse, ürün fiyatı alınır ve veritabanından müşteri bakiyesi düşülür.

Fonksiyonlar:

- `nesne_tanima(musteri_id):`
    - Kameradan canlı görüntü alır.
    - Algılanan nesne varsa, fiyatı alır ve veritabanındaki müşterinin bakiyesinden düşer.
- `musteriden_bakiye_dus(musteri_id, urun, fiyat):`
    - Tanınan nesneye göre müşterinin bakiyesini günceller.

2.3 Veritabanı Sorguları

Yüz tanıma ve nesne tanıma aşamalarında **SQLite** veritabanı kullanılmıştır. Veritabanında müşterilerin **ID, isim, soyisim, bakiye, yüz fotoğrafı** gibi bilgileri tutulmaktadır. Veritabanı sorguları için **SQLite** bağlantısı kurulur ve `cursor` aracılığıyla gerekli SQL komutları çalıştırılır.

Fonksiyonlar:

- `getCustomerList():`
    - Tüm müşteri listesini çeker ve kullanıcı arayüzünde listeler.
- `deleteCustomer(musteri_id):`
    - Seçilen müşteriyi veritabanından siler.
- `yeni_musteri_ekle(isim, soyisim, telefon):`
    - Kullanıcıdan aldığı bilgilerle yeni müşteri kaydı oluşturur.

2.4 Kayıt Aşaması

Kullanıcı yüzü veritabanında bulunamazsa, sistem otomatik olarak kullanıcıyı tanımadığını anlar ve **yeni müşteri kaydı** için yönlendirme sürecini başlatır. Bu durum, proje kapsamında **kullanıcı dostu self-servis sürecinin bir parçası** olarak tasarlanmıştır.

İşleyiş:

1. Kullanıcının yüzü taranır ve **`yuz_tanimla()`** fonksiyonu veritabanında arama yapar.
2. Eğer kayıt bulunmazsa, sistem otomatik olarak bir **alert (uyarı penceresi)** verir.
3. Bu uyarı, "Kullanıcı bulunamadı. Yeni kayıt oluşturmak ister misiniz?" şeklinde olur.
4. Kullanıcı onaylarsa, **`yeni_musteri_ekle()`** fonksiyonu devreye girerek kullanıcıdan **isim, soyisim, telefon** gibi bilgileri alır ve kayıt oluşturur.

Çalışan Fonksiyonlar ve Sorgular:

- **`yuz_tanimla()`:**
    1. Kayıtlı yüzlerle karşılaştırma yapar.
    2. Kayıt yoksa, False döndürerek yeni müşteri kaydı sürecine geçilmesini sağlar.
- **`yeni_musteri_ekle(isim, soyisim, telefon)`:**
    1. Yeni müşteri kaydı için bir form açar ve gerekli verileri alır.
    2. SQLite veritabanına kayıt işlemi yapar.
- **Veritabanı Sorguları:**
    - Müşteri tablosunda `SELECT` sorgusu yapılır ve eşleşme olmazsa kayıt açılır.

Sistemin Kullanıcıya Verdiği Alert:

1. "Kullanıcı bulunamadı. Yeni müşteri kaydı oluşturmak ister misiniz?"
2. Kullanıcı **onay verirse** yeni kayıt oluşturma formu açılır, **reddederse** işlem iptal edilir.

3. Benzer Çalışmalar

2.1. Amazon GO

Amazon Go **(Şekil 3.1)**, Amazon'un geliştirdiği kasiyersiz market konseptidir. Bu mağazalarda müşteriler, Amazon Go uygulamasını kullanarak giriş yapar ve ürünleri aldıktan sonra kasada ödeme yapmaksızın çıkabilirler. Sistem, gelişmiş bilgisayarla görü (CV) algoritmaları, raf sensörleri ve derin öğrenme teknikleriyle donatılmıştır. Müşteriler mağazaya girerken, alışveriş sırasında ve çıkışta sürekli olarak izlenir; alınan ürünler otomatik olarak sanal alışveriş sepetine eklenir ve müşterinin Amazon hesabından ücret tahsil edilir.

Şekil 3.1- Amazon GO **[12]**

3.2 BingoBox

BingoBox, Çin merkezli bir kasiyersiz market projesidir. Küçük konteyner benzeri mağazalarda hizmet veren BingoBox, yüz tanıma sistemleri ve nesne algılama kameralarıyla donatılmıştır. Kullanıcılar alışverişlerini yaptıktan sonra kasada sıra beklemeden mağazadan ayrılır; ürünlerin ödemesi ise mobil uygulama üzerinden ya da otomatik sistemler aracılığıyla gerçekleşir. BingoBox, Amazon Go'dan farklı olarak daha kompakt ve konteyner bazlı mağaza konseptine odaklanır.

3.3 Projem ile Benzerlikler ve Önerilen Unsurlar

Projemiz, Amazon Go ve BingoBox gibi öncü kasiyersiz market projelerinden ilham alınarak geliştirildi. Bu projelerin ortak noktası, insan müdahalesine gerek kalmadan kullanıcıları tanıma ve satın alma sürecini hızlandırma amacını taşımalarıdır. Amazon Go'da, müşteriler Amazon hesaplarıyla mağazaya giriş yaparken, BingoBox'ta ise yüz tanıma teknolojileri ile mobil ödeme sistemleri entegre edilmiştir.

Bizim projemizde de benzer şekilde, müşterinin yüzü tanındığında doğrudan veritabanı sorguları aracılığıyla kaydı bulunup bulunmadığı kontrol edilir. Böylece müşteriler, sisteme herhangi bir kart okutmadan veya kasiyere bilgi vermeden kolaylıkla giriş yapabilirler. Projemin geliştirilmesi sırasında, bu öncü projelerden sadece teknik olarak değil, kullanıcı deneyimini iyileştirme açısından da esinlendim. Örneğin, Amazon Go'daki "alışveriş yap ve çık" konsepti gibi, bizim sistemimiz de müşterinin yüzü tanındığında otomatik olarak bakiyesinden ürün ücretini düşmektedir. Eğer müşteri sistemde bulunmuyorsa, yeni bir kayıt talebi oluşturularak ilk alışveriş deneyimi sırasında kayıt süreci tamamlanır. Bu yaklaşım, kullanıcı memnuniyetini artırırken aynı zamanda mağaza çalışanı ihtiyacını minimuma indirir. Ayrıca, projemiz spor salonu, market gibi farklı ortamlara kolayca adapte edilebilecek esnek yapısıyla, Amazon Go ve BingoBox projelerine benzeyen, hızlı bir alışveriş ortamı sunmayı hedeflemektedir.

3.4 Yapılabilecek Geliştirmeler

1. Projemiz, mevcut haliyle yüz tanıma, nesne tanıma ve veritabanı üzerinden alışveriş sürecini otomatikleştirme yeteneklerini başarıyla sunmaktadır. Ancak, projenin daha profesyonel ve yüksek performanslı hale getirilmesi için çeşitli geliştirme adımları mümkündür. Örneğin, yüz tanıma algoritmalarının daha hassas ve hızlı çalışabilmesi için özel olarak eğitilmiş derin öğrenme modelleri entegre edilebilir. Böylece, farklı yüz ifadeleri veya aydınlatma koşullarındaki tanıma hataları minimuma indirilebilir.

2. Nesne tanıma modülü için de daha gelişmiş yapay zeka modelleri (örneğin YOLOv8, EfficientDet gibi) kullanılarak, ürün algılama süresi ve doğruluğu artırılabilir. Ayrıca, veritabanı yönetimi için gelişmiş bir veritabanı sunucusu (örneğin PostgreSQL gibi) ve gelişmiş bir API entegrasyonu eklenerek sistemin ölçeklenebilirliği artırılabilir. Bir diğer geliştirme alanı ise çoklu kamera desteğiyle birlikte daha büyük alanlarda eş zamanlı tanıma imkanı sağlamaktır. Böylece, mağaza gibi geniş ortamlarda bile müşterilerin akışı ve işlemleri daha iyi yönetilebilir.

3. Son olarak, projenin kullanıcı deneyimini iyileştirmek için modern ve kullanıcı dostu bir web paneli geliştirilerek, müşterilerin ve yöneticilerin kullanım kolaylığı sağlanabilir. Böylece projemiz, sadece küçük işletmeler için değil, büyük market zincirleri ve profesyonel spor salonları gibi kurumsal yapılar için de uygun ve ölçeklenebilir bir çözüm haline gelebilir.

3.5 Kalite Gereksinimleri

a) Güvenilirlik

Bu projede, insan gücü ve müdahalesi minimum seviyeye indirilerek tüm süreçler dijital olarak yürütülmektedir. Sistem, kamera ve veritabanı tabanlı yüz tanıma ile nesne tanıma işlemlerini otomatik şekilde yürütür ve müşteri bakiyesini güvenli bir şekilde düşer. Her bir işlem, manuel müdahaleye gerek kalmadan, hızlı ve hatasız şekilde sonuçlanır.

b) Kullanılabilirlik

Arayüz, basit ve kullanıcı dostu bir yapıya sahiptir. Kullanıcı, yalnızca sisteme yaklaşarak işlemleri başlatır. Tüm süreçler otomatik olarak tamamlandığından, üçüncü şahıs çalışanlara ihtiyaç duyulmaz. Böylece müşterilerin kişisel bilgileri (telefon, kart bilgileri gibi) hiçbir görevli tarafından erişilmeden korunur

c) Güvenlik

Proje, müşteri verilerini üçüncü taraf personellerin erişimine kapalı tutacak şekilde tasarlanmıştır. Tüm veriler, sadece sistemin dijital modülleri arasında işlenir ve saklanır. Kasa personelinin ya da başka birinin müşteri bilgilerini görüntülemesi veya değiştirmesi mümkün değildir.

d) Sürdürebilirlik

Sistem, dijital olarak tüm süreçleri yönettiği için insan hatası riskini minimuma indirir ve bakım-güncelleme gibi işlemler yazılım geliştiriciler tarafından yapılabilir. Yeni bir çalışan eğitimi gerekmeden, sistem kendi kendine çalışabilir.

3.6 Performans Gereksinimleri

a) Zamanlama

Müşteri yüz tanıma ve nesne tanıma işlemleri, gerçek zamanlı olarak dijital olarak tamamlanır. Her bir işlem saniyeler içinde sonuçlandığı için, müşterilerin bekleme süresi minimuma indirilir.

b) Hız

Sistem tamamen dijital olduğu için insan gücü beklemeleri veya müdahaleleri ortadan kalkar. Yüz ve nesne tanıma algoritmaları, hızlı sonuç verecek şekilde optimize edilmiştir.

c) Hacim

Yüz ve nesne tanıma işlemleri ile yapılan her işlem veritabanında saklanır. Ancak sistemin genel yapısı hafif ve optimize edilmiş olduğu için düşük depolama alanı kullanır.

d) Verim

Dijital modüller sayesinde sistem, müşteri deneyimini hızlandırır ve herhangi bir personel bekleme süresine gerek bırakmaz. Bu sayede hem zaman tasarrufu hem de müşteri memnuniyeti sağlanır.

3.7 Gerçek Olmayan Gereksinimler

1. Fiziksel

Sistem, Windows işletim sistemi yüklü bir cihaz ve bir kamera ile çalışır. Bu fiziksel ortam dışında, insan faktörüne ihtiyaç duyulmaz.

2. Yasal

Tüm süreçler dijital olarak ve müşteri rızası dahilinde yürütülür. Çalışanlar veya üçüncü şahıslar müşteri verilerine erişemez.

3. Kültürel

Müşteri bilgileri sadece işlem için kullanılır. Yüz tanıma ve nesne tanıma algoritmaları, kullanıcı kimliği dışında herhangi bir sosyal/kültürel veri toplamaz veya işlemez.

4. Çevre

Sistemin otomatik çalışması, ek personel ihtiyacını ortadan kaldırır ve bu da çevresel olarak daha az kaynak tüketimine katkıda bulunur.

5. Tasarım ve Uygulama

Sistem, Python programlama dili ve PyQt5 kütüphaneleri ile tasarlanarak dijital ortama uygun hale getirilmiştir. Çalışanlar veya üçüncü şahıslar için ek bir öğrenme eğrisi gerekmez.

6. Arayüz

Arayüz, kullanıcının kasiyersiz self-servis sistemi kolaylıkla kullanabilmesi için geliştirilmiştir. Dijital yönlendirmeler ve sade tasarım, işlemleri kendi başına tamamlamasını sağlar.

4. Senaryo: Self-servis kasa sisteminin kullanımı

1. Müşteri, kameranın önüne geçtiğinde sistem, müşterinin yüzünü tanıyarak veritabanındaki bilgilerini kontrol eder.

2. Eğer müşteri daha önce kayıtlı ise, sistem “Hoş geldiniz [Müşteri Adı]!” gibi bir mesaj gösterir ve müşteri satın almak istediği ürünleri kameraya göstererek alışverişe başlar.

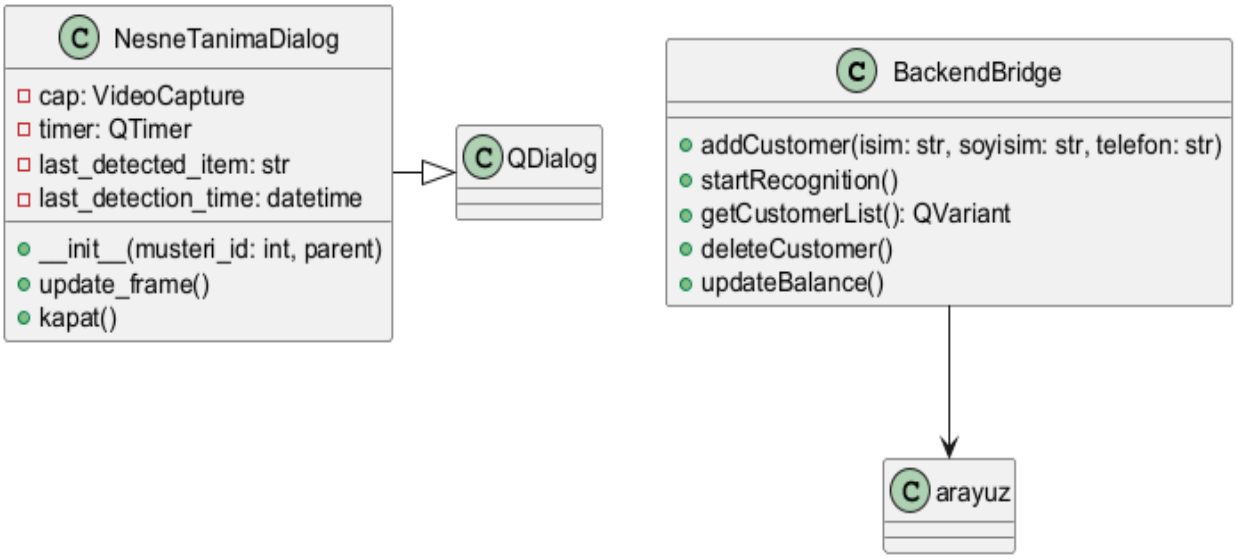
3. Eğer müşteri kayıtlı değilse, sistem müşteriden isim, soyisim, telefon gibi bilgileri girerek kayıt olmasını ister.

4. Kayıt işlemi tamamlandıktan sonra müşteri, ürünleri kameraya göstererek alışverişine devam eder.

5. Ürünler otomatik olarak algılanır, fiyatları veritabanından çekilir ve müşterinin bakiyesinden düşülür.

5.Sınıf Diyagramı

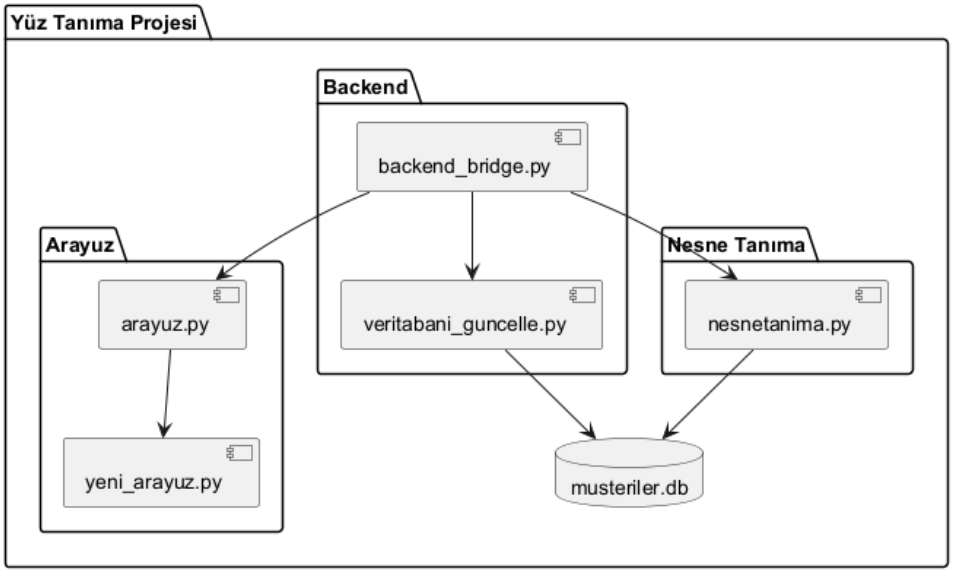
Programın sınıf diyagramı **(Şekil-5.1)** gösterilmiştir.



Şekil 5.1-Projenin Sınıf Diyagramı.

5.1 Paket Şeması

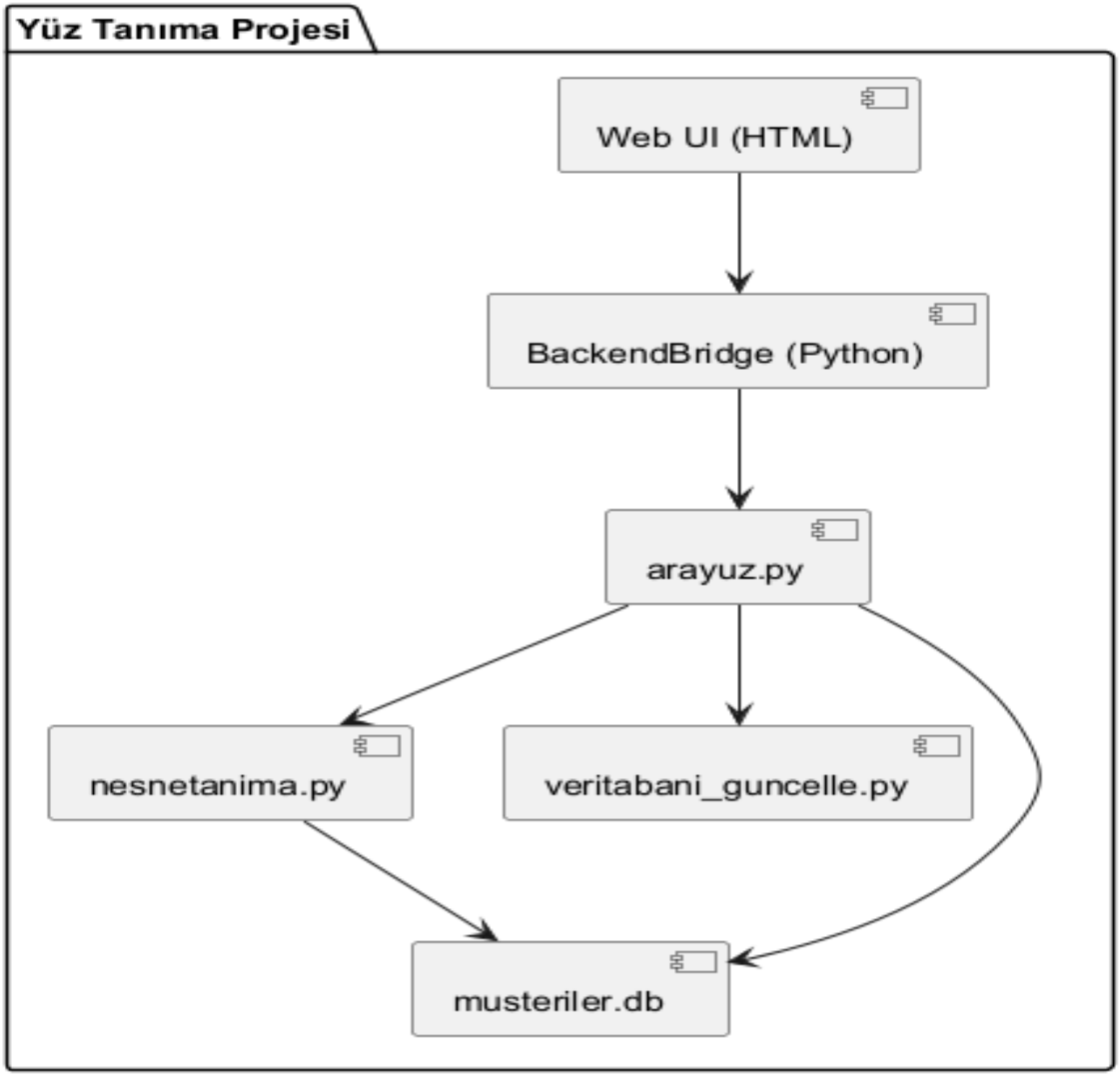
Şekil 5.2 üstünde sistemin paket şeması görselleştirilmiştir.



Şekil 5.2- Projenin Paket Şeması.

5.2 Bileşen Şeması

Bu bileşen şeması**(Şekil-5.3)**, **Yüz Tanıma Projesi**'nin farklı modüller arasındaki ilişkileri ve bileşenlerin görevlerini net bir şekilde göstermektedir. Şemanın en üstünde yer alan **Web UI (HTML)**, kullanıcı arayüzünü temsil eder ve kullanıcı ile sistem arasındaki etkileşimi sağlar. Kullanıcıdan gelen işlemler, **BackendBridge (Python)** bileşenine aktarılır; bu bileşen, PyQt altyapısıyla HTML arayüzü ile Python modülleri arasındaki iletişimi yürütür. BackendBridge, kullanıcının başlattığı tüm işlemleri **arayuz.py** dosyasına iletir. Burada hem yüz tanıma hem de nesne tanıma akışları tetiklenir. **arayuz.py** bileşeni, veritabanı güncelleme ve nesne tanıma modülleriyle doğrudan bağlantılıdır. Şemada gösterildiği gibi, **nesnetanima.py** bileşeni, tanınan nesneleri algılayarak ilgili işlemleri yürütürken; **veritabani_guncelle.py** bileşeni, tanınan müşteri bilgilerini güncellemek için **musteriler.db** veritabanına erişir. Ayrıca, nesne tanıma işlemi doğrudan veritabanı ile ilişkilidir ve güncel veriler doğrudan kaydedilir. Böylece, tüm modüller **arayuz.py** üzerinden entegre şekilde çalışarak verimli bir yüz ve nesne tanıma akışı sağlar. Diagram, projenin temel veri ve kontrol akışlarını sade ve anlaşılır bir şekilde ortaya koyar, modüllerin birbirleriyle olan bağlantılarını netleştirir ve tüm projenin mimarisini bütüncül bir bakış açısıyla sunar.

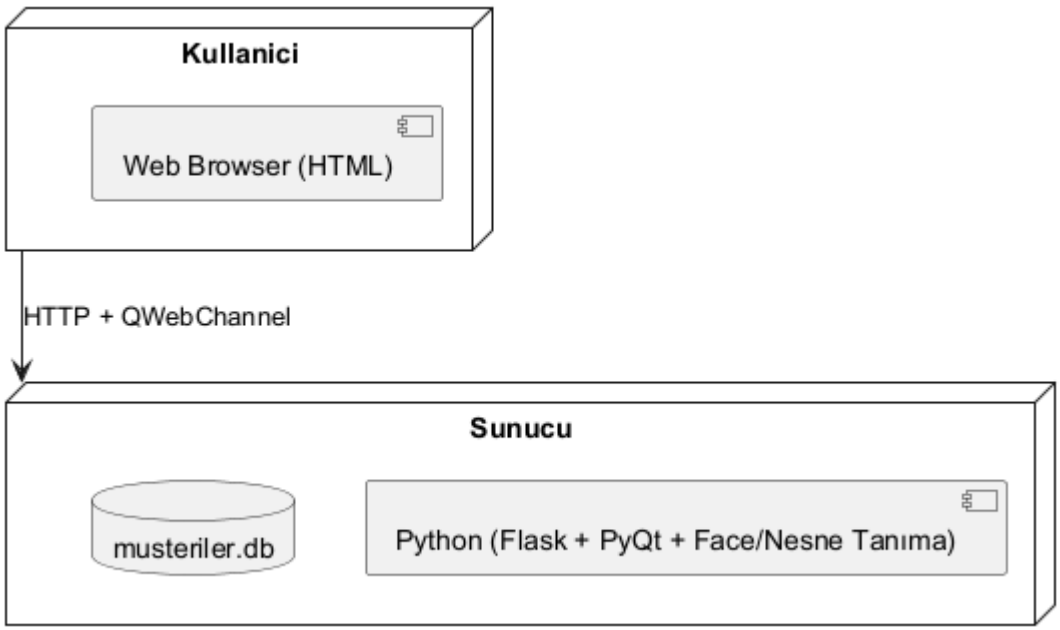


Şekil 5.3- Projenin Bileşen Şeması

5.3 Dağıtım Şeması

Diagram **(Şekil-5.4)**, iki ana bileşenden oluşuyor: **Kullanıcı (Web Browser)** ve **Sunucu**. Kullanıcı tarafında, bir web tarayıcısı (HTML) kullanılarak sunucuya istekler gönderiliyor. Bu iletişim, diagramda da belirtildiği gibi, **HTTP protokolü** ve **QWebChannel** aracılığıyla sağlanıyor. Sunucu tarafında ise **Python** dilinde geliştirilmiş bir backend yer alıyor. Bu backend, **Flask** (web server), **PyQt** (arayüz ve QWebEngineView entegrasyonu), ve **Yüz/Nesne Tanıma** bileşenlerini içeriyor. Ayrıca, **musteriler.db** veritabanı, sunucu üzerinde doğrudan erişilebilir bir veri deposu olarak diagramda doğru bir şekilde konumlandırılmış.

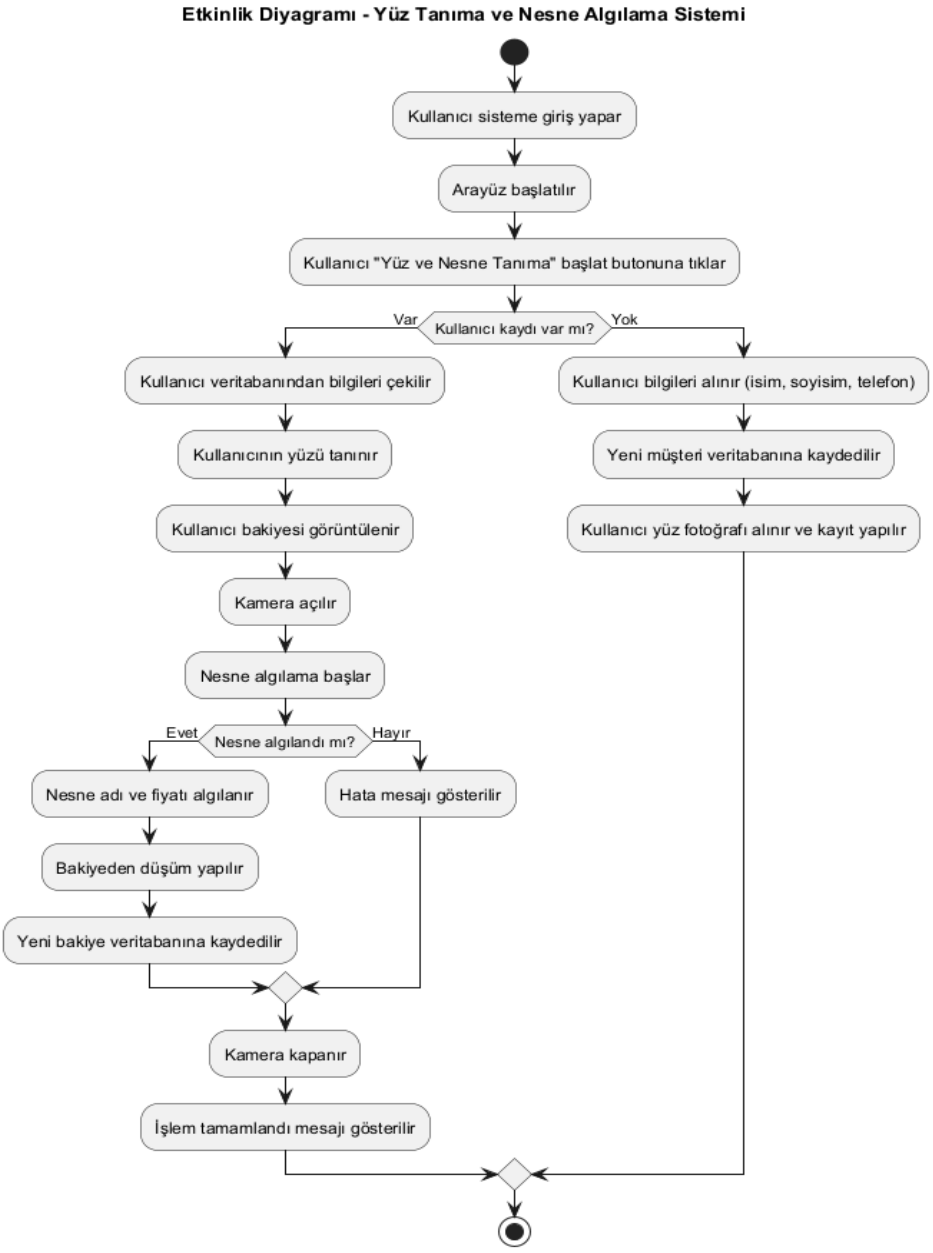
Diagramın genel akışı ve bileşenlerin yerleşimi doğru. Bu yapı, **kullanıcının tarayıcıdan gönderdiği verilerin Flask ile karşılanması**, verilerin işlenmesi (örneğin yüz tanıma ya da nesne tanıma süreçleri) ve **musteriler.db** veritabanına erişilmesi gibi temel işlevleri başarıyla gösteriyor. QWebChannel kullanımı, tarayıcı (HTML arayüz) ile Python backend arasındaki canlı veri alışverişini temsil ediyor.



Şekil 5.4 - Projenin Dağıtım Şeması

5.4 Etkinlik şeması

Etkinlik şeması **(Şekil-5.5)**, projemizde kullanıcıların adım adım gerçekleştirdiği tüm işlemleri ve bu işlemlerin nasıl bir akış izlediğini görsel olarak ortaya koyar. Şemada, kullanıcının sisteme girişinden başlayarak, yüz tanıma modülünün çalıştırılması, eğer kullanıcı bulunamazsa kayıt işleminin başlatılması ve kullanıcı verilerinin veritabanına kaydedilmesi gibi önemli adımların tümü akış halinde gösterilir. Ayrıca, kullanıcı yüzü tanındıktan sonra nesne tanıma adımı başlatılarak ürünlerin tanınması ve bakiye güncelleme işlemleri de akışa dahil edilmiştir. Bu diagram, sürecin net bir şekilde izlenebilmesini sağlayarak, hem kullanıcı deneyimini hem de sistemin genel akışını anlamayı kolaylaştırır.

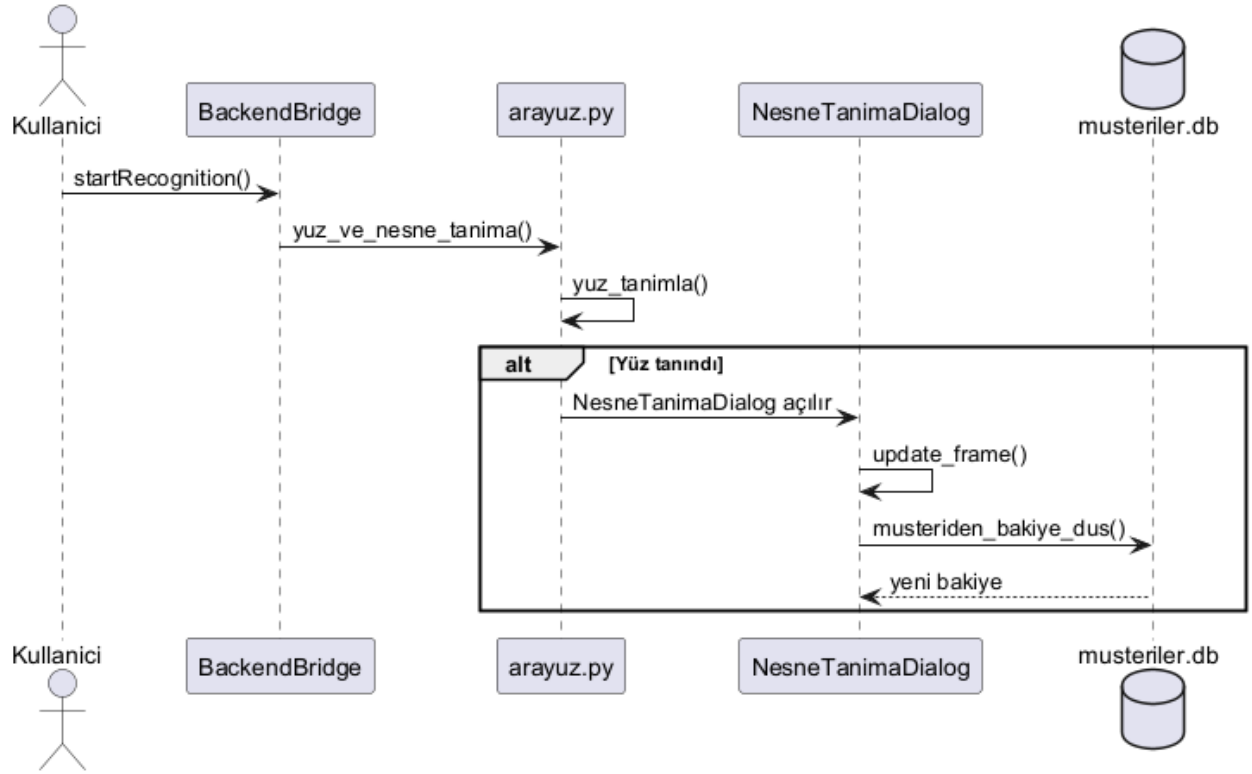


Şekil 5.5 - Projenin Faaliyet Şeması

5.5 Sıra Diyagramı

Bu sıra diyagramı, kullanıcıdan başlayarak sistemin nasıl bir işleyişe sahip olduğunu adım adım ortaya koymaktadır. Şemada ilk olarak **Kullanıcı**, "startRecognition()" fonksiyonunu çağırarak yüz tanıma ve nesne tanıma sürecini başlatır. Bu fonksiyon, **BackendBridge** modülü üzerinden "yuz_ve_nesne_tanima()" fonksiyonunu tetikler. **arayuz.py** modülü "yuz_tanimla()" fonksiyonunu çalıştırarak kullanıcının yüzünü tanıma işlemini gerçekleştirir.

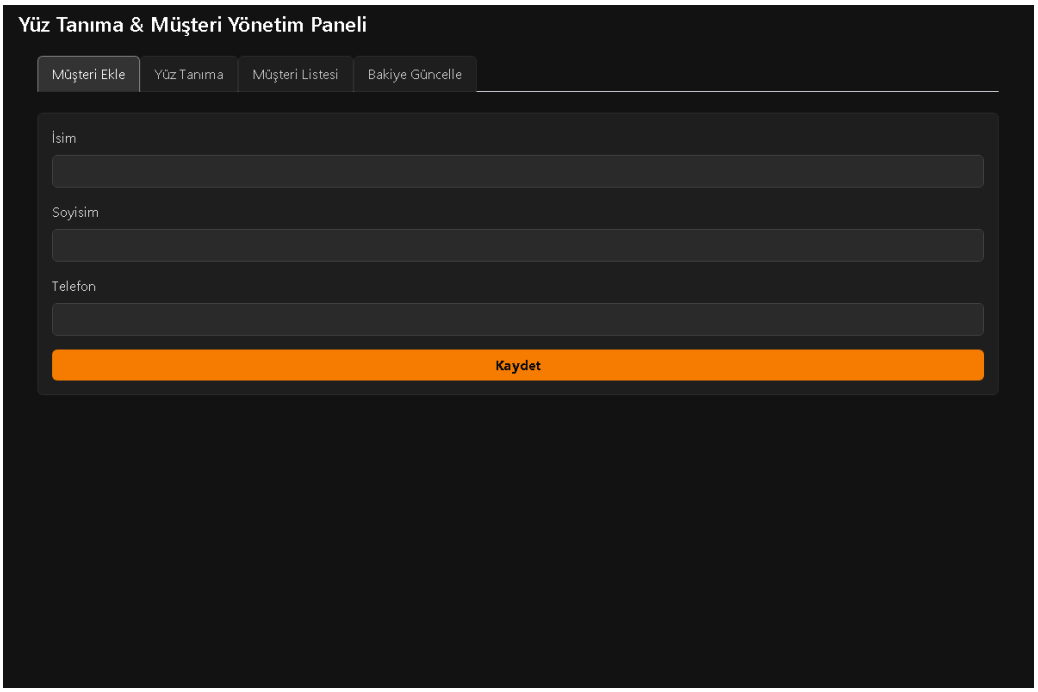
Eğer yüz tanınırsa, şemada "alt" bloğu altında görüldüğü gibi **NesneTanimaDialog** devreye girer. Bu aşamada, **NesneTanimaDialog** açılır ve "update_frame()" fonksiyonu ile kameradan sürekli olarak nesne tanıma görüntüleri alınmaya başlanır. Elde edilen nesne bilgileriyle birlikte **musteriler.db** veritabanına bir sorgu yapılır ve "musteriden_bakiye_dus()" fonksiyonu çalıştırılır. Böylece kullanıcının bakiyesi güncellenir ve yeni bakiye bilgisi kullanıcıya yansıtılır. Tüm bu akış, bileşenlerin birbirleriyle nasıl etkileşime geçtiğini ve hangi sırayla çalıştıklarını açıkça ortaya koymaktadır. Diagram, projenin çalışma mantığını ve modüller arası veri akışını görsel bir şekilde detaylandırmaktadır.



Şekil 5.6- Projenin Sıra Diagramı.

6.Kullanıcı arayüzü

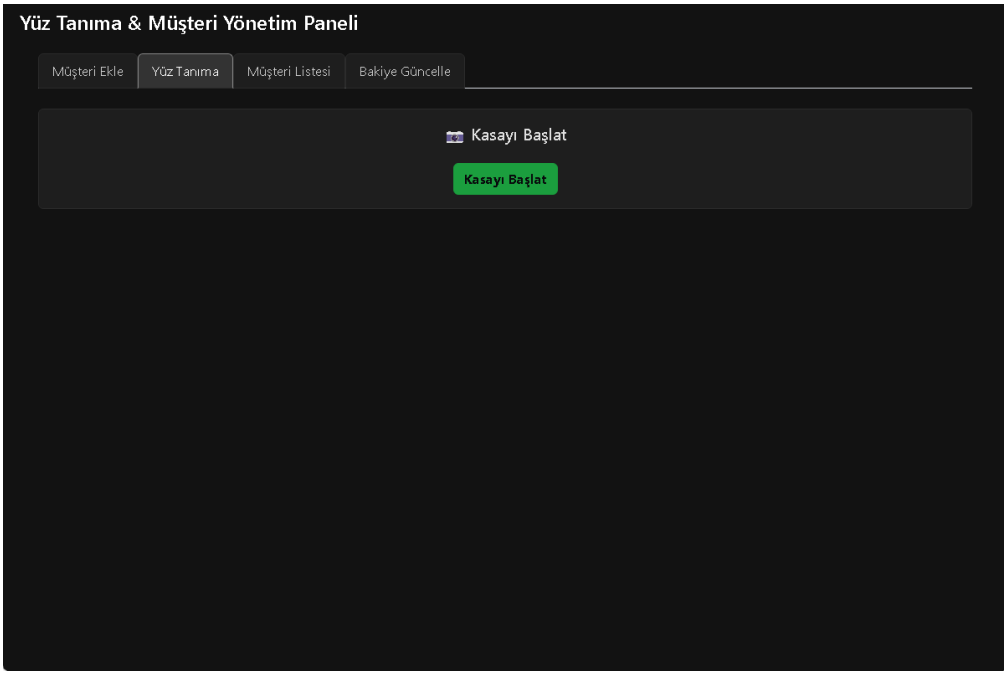
Kullanıcı arayüzü, kullanıcı dostu bir deneyim sunmak ve sistemin kullanımını kolaylaştırmak amacıyla tasarlanmıştır. Koyu tema renkleri ve minimalist tasarımıyla, kullanıcıların gözlerini yormadan uzun süreli kullanımlara olanak tanır. Arayüzde kullanılan CoreUI CSS framework'ü, modern ve profesyonel bir görünüm sağlarken, basit navigasyon ve net buton yerleşimi sayesinde kullanıcıların kolayca işlem yapmasını mümkün kılar. İlk aşamada, **Müşteri Ekle Paneli (Şekil 6.1)** kullanıcıların karşısına çıkar. Bu panelde, müşterinin ad, soyad ve telefon gibi temel bilgileri alınır. Giriş alanları `input` öğeleri ile sağlanmış ve kullanıcı tarafından eksiksiz doldurulması istenmiştir. Panelin altında bulunan "Kaydet" butonu, girilen bilgileri `addCustomer()` fonksiyonu aracılığıyla sisteme kaydeder. Bu süreçte, verilerin doğruluğu ve bütünlüğü korunurken, hızlı ve pratik bir şekilde müşteri kaydı tamamlanır. Panelin görsel ve fonksiyonel akışı, arka planda Python backend'e bağlanan `QWebChannel` teknolojisiyle desteklenerek güçlü bir yazılım altyapısı sağlar.



Şekil 6.1 -Müşteri ekle ekranı açılış görüntüsü.

6.1 Yüz Tanıma Ekranı

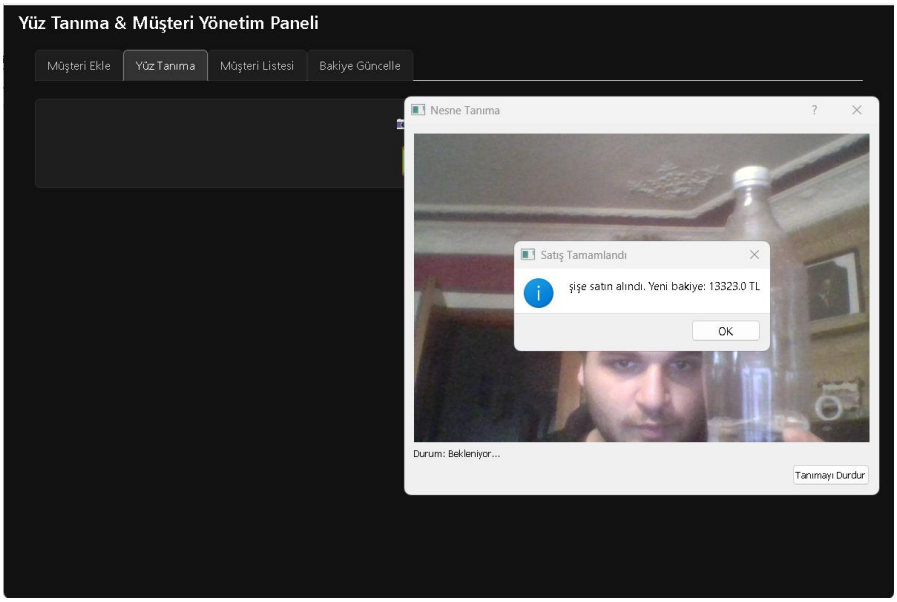
Kullanıcı arayüzünün en önemli bölümlerinden biri olan **Yüz Tanıma Arayüzü (Şekil 6.2)**, basit ve anlaşılır bir tasarımla kullanıcıya rahat bir deneyim sunmayı hedefler. CoreUI tabanlı kart tasarımıyla modern ve şık bir görünüm sağlanırken, ortada yer alan büyük boyutlu bir başlık ve ikon, sistemin görsel yönünü zenginleştirir. Arayüzde, "Kasayı Başlat" başlığı altında bulunan yeşil renkli "Kasayı Başlat" butonu yer alır ve bu buton, `backend.startRecognition()` fonksiyonunu tetikleyerek Python backend tarafında yer alan yüz tanıma modülünü başlatır. Bu sayfa, kullanıcının yüz tanıma sürecini hızlıca başlatmasına olanak tanırken, sade tasarımıyla da gereksiz bilgi kalabalığını ortadan kaldırır. Yüz tanıma ekranı, kullanıcı verilerinin doğrudan ve güvenli şekilde işlenmesi için optimize edilmiştir ve tüm işlemler arka planda hızlı ve güvenli bir şekilde gerçekleşir.



Şekil 6.2 -Yüz tanıma ekranı açılış görüntüsü.

6.2 Nesne Tanıma Ekranı Kasa İşlevi

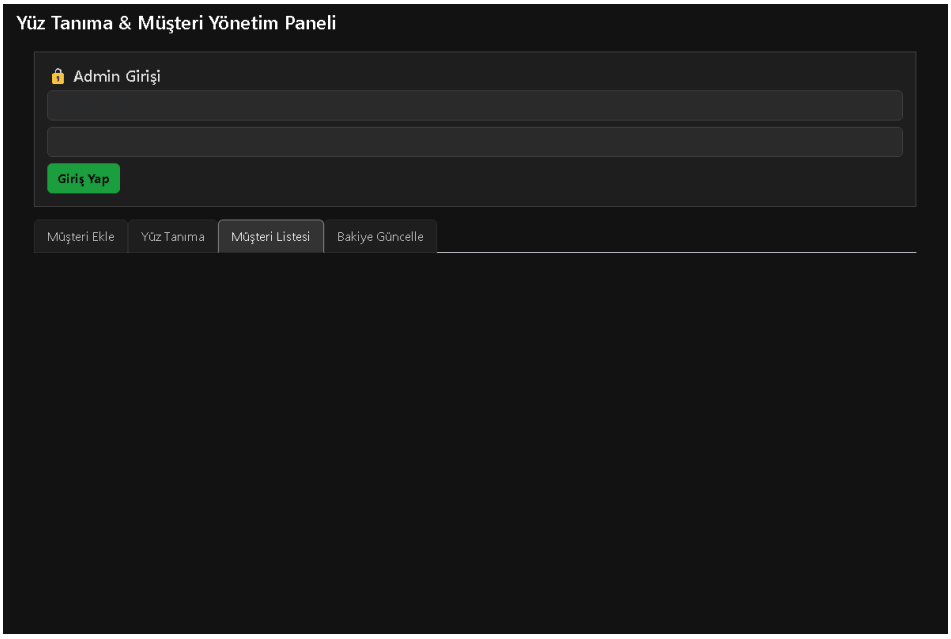
Bu panelde, kullanıcı kasayı başlat düğmesine bastığında sistem kamerayı açar ve canlı bir video akışı başlatır. Kullanıcının yüzü sistemde kayıtlı değilse, otomatik olarak yeni müşteri kayıt arayüzü açılır ve kullanıcının bilgileri alınarak kaydedilir. Eğer yüz daha önce kaydedilmişse, sistem o kişiyi tanır ve ekranın sağ üst köşesinde kullanıcının adı, soyadı, mevcut bakiyesi gibi bilgiler gösterilir. Bu aşamada, sistem gerçek zamanlı olarak yalnızca yüz tanıma değil aynı zamanda nesne tanıma işlevini de yerine getirir. Örneğin, kullanıcı kasayı başlat butonuna bastıktan sonra şişe gibi nesneleri sisteme gösterdiğinde, ekranın sağ alt köşesinde tanınan nesne ve fiyat bilgileri anlık olarak görüntülenir **(Şekil 6.3)**. Bu panel, kullanıcı dostu tasarımı sayesinde tüm işlemlerin tek tıklama ile başlatılmasını ve sistemin anında yanıt vermesini sağlar.



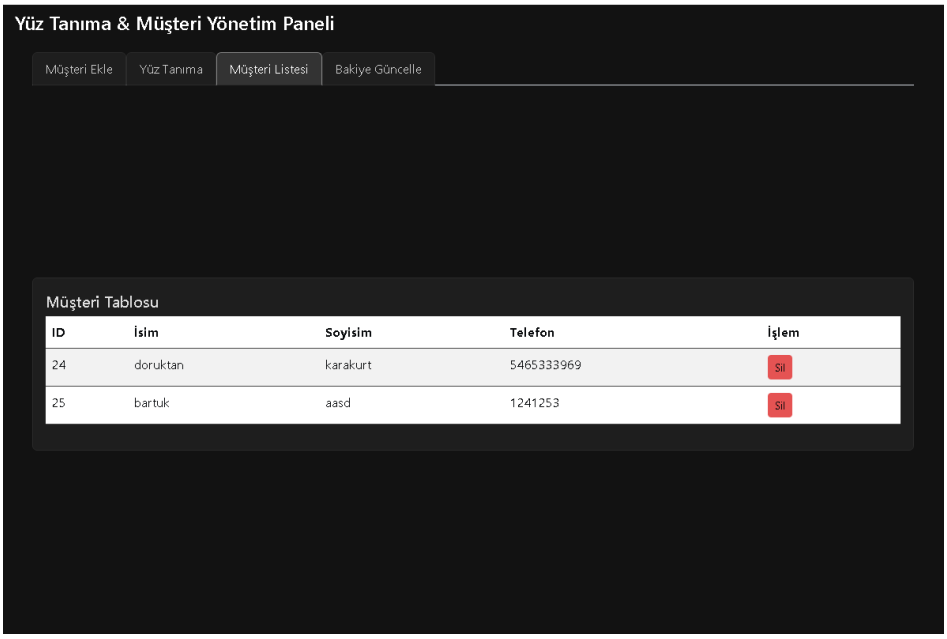
Şekil 6.3 -Nesne tanıma kasa işlevi görüntüsü.

6.3 Müşteri Listesi Paneli

Müşteri Listesi paneli **(Şekil 6.4 ve Şekil 6.5)**, sistemde kayıtlı olan tüm müşterilerin bilgilerini toplu şekilde görüntülemeyi amaçlayan bir arayüzdür. Ancak, müşteri bilgilerinin güvenliği ve gizliliği nedeniyle bu panel sadece admin girişi yapıldıktan sonra erişilebilir durumdadır. Admin kullanıcı adı ve şifre doğru bir şekilde girildiğinde, panelde müşterilerin ID, isim, soyisim ve telefon bilgileri gibi temel verileri tablo halinde görüntülenir. Bu sayede admin, müşteri verilerini kolayca yönetebilir. Panel, kullanımı kolay, açık renkli bir tablo düzeniyle tasarlanmıştır ve CoreUI bileşenleri sayesinde şık ve okunabilir bir yapıdadır. Sistem, admin girişi doğrulanmadan bu paneli gizli tutarak veri güvenliğini ön planda tutar. Yanlış kullanıcı adı ya da şifre girildiğinde ise giriş paneli açık kalır ve panel içeriği görüntülenmez.



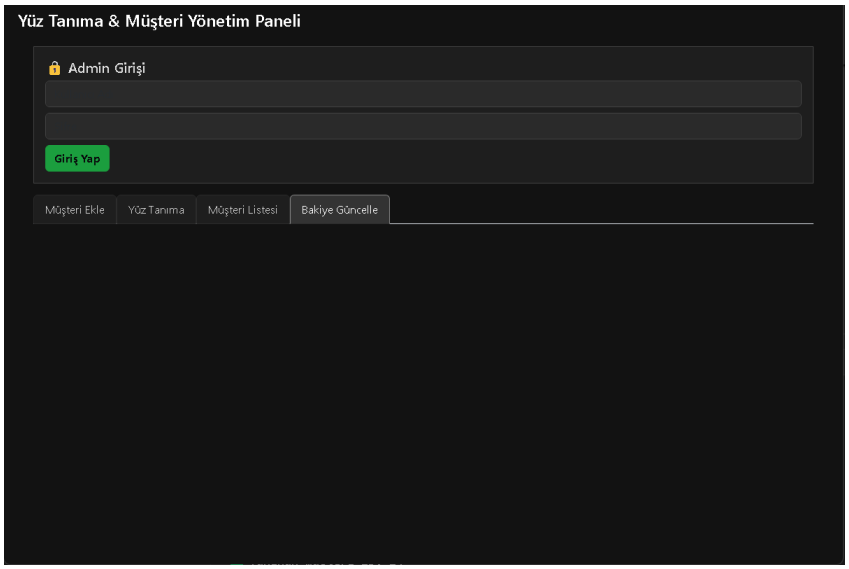
Şekil 6.4 - Müşteri Listesi açılış görüntüsü.



Şekil 6.5 - Müşteri Listesi.

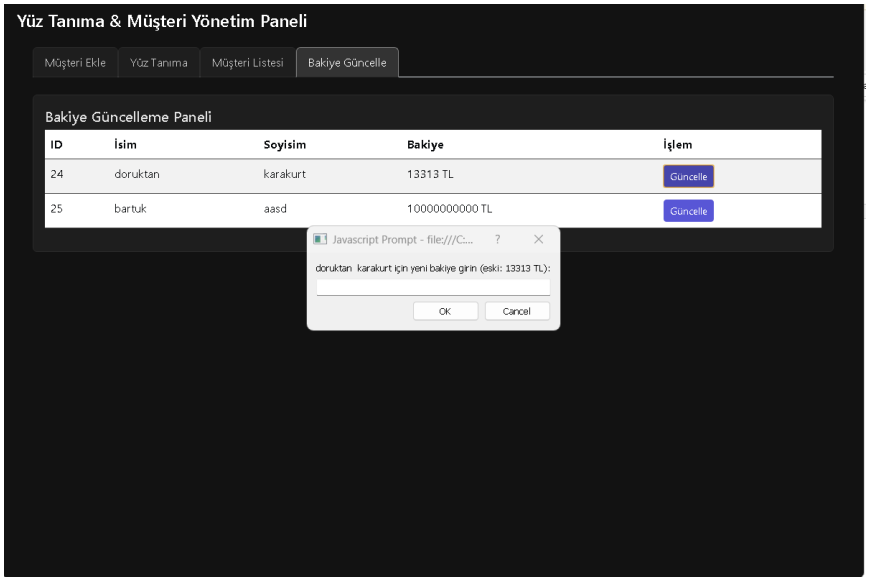
6.4 Bakiye Güncelleme Paneli

Bakiye Güncelleme paneli **(Şekil 6.6, Şekil 6.7 ve Şekil 6.8)** de benzer şekilde sadece admin girişinden sonra erişilebilir bir özelliğe sahiptir. Bu panel, sistemdeki müşterilerin mevcut bakiyelerini güncelleyebilmek için tasarlanmıştır. Panelde, her müşterinin ID, isim, soyisim ve mevcut bakiye bilgileri tablo halinde sunulur. Güncelle butonuna basıldığında, admin yeni bakiye değerini girerek müşterinin bakiyesini güncelleyebilir. Bu sayede tüm finansal işlemler yalnızca yetkili kişiler tarafından yönetilebilir. Panel, kullanıcı dostu bir tasarımla donatılmış ve CoreUI'nin özelleştirilmiş CSS yapısı ile geliştirilmiştir. Admin doğrulaması olmadan, panel tamamen gizlenir ve kullanıcı tarafından erişilemez hale gelir. Bu, sistemdeki mali verilerin bütünlüğünü ve güvenliğini sağlamak için hayati önem taşır.



Şekil 6.6 -Bakiye güncelleme paneli açılış ekranı

Şekil 6.7 -Bakiye güncelleme paneli admin girişi sağlandıktan sonra gösterilen ekran.



Şekil 6.8-Bakiye güncelleme paneli admin girişi sağlandıktan bakiye güncelleme işlemi görseli.

7. Sonuç

Bu proje kapsamında geliştirilen yüz tanıma ve nesne tanıma temelli self-servis kasa sistemi, günümüzün hız ve verimlilik odaklı müşteri deneyimi ihtiyaçlarına yanıt vermeyi amaçlamaktadır. Özellikle spor salonları ve marketler gibi yerlerde kasiyersiz ödeme süreçlerini destekleyerek insan gücü maliyetlerini düşürmek ve müşteri memnuniyetini artırmak hedeflenmiştir. Projede, kullanıcının sisteme kaydolması, yüz tanıma yoluyla kimlik doğrulaması, ardından nesne tanıma sayesinde yaptığı alışverişin otomatik olarak hesaplanması gibi fonksiyonlar bir bütün halinde çalışmaktadır. Bu akış, kullanıcı dostu ve kolay erişilebilir bir arayüzle desteklenerek, modern ticari ortamlara uygun, pratik bir çözüm sunmaktadır.

Projenin geliştirilmesi sırasında Python dili kullanılarak makine öğrenmesi ve görüntü işleme algoritmaları entegre edilmiş, veritabanı yönetimi için SQLite tercih edilmiştir. Arayüz tarafında ise HTML, CSS ve JavaScript gibi web teknolojilerinin yanı sıra CoreUI gibi modern tasarım kütüphaneleri kullanılmış, işlevselliği ve kullanıcı deneyimini güçlendiren bir yapı oluşturulmuştur. Sistemde yer alan admin paneli ise yetkilendirme bazlı erişim kontrolü sunarak, yalnızca yetkili kişilerin müşteri listesi ve bakiye güncelleme gibi kritik işlemleri yapabilmesini sağlar. Böylece sistem, hem güvenlik hem de kullanım kolaylığı açısından sağlam bir altyapıya sahip olmuştur.

Sonuç olarak, bu proje; farklı modüllerin bütünleşik bir şekilde çalıştığı, esnek ve özelleştirilebilir bir altyapı sağlamaktadır. Kullanıcıların sisteme kolayca dahil olabilmesi ve self-servis kasa mantığının hayata geçirilmesi, projenin pratikte uygulanabilirliğini göstermektedir. Gelecekte, bu sistem daha gelişmiş yüz tanıma kütüphaneleri ve daha hızlı nesne tanıma algoritmaları ile entegre edilerek performans ve doğruluk açısından iyileştirilebilir. Ayrıca, kullanıcı deneyimini daha da geliştirmek için mobil uygulama entegrasyonu gibi eklemeler de yapılabilir. Böylece, projenin vizyonu, çağın gereksinimlerine uygun, esnek ve yenilikçi bir çözüm olarak daha da ileri taşınabilir.

8. Referanslar

1. Python resmi dökümantasyonu, https://docs.python.org/3/
2. OpenCV Python kütüphanesi, https://docs.opencv.org/
3. Flask framework, https://flask.palletsprojects.com/
4. PyQt5, https://www.riverbankcomputing.com/software/pyqt/
5. SQLite veritabanı sistemi, https://www.sqlite.org/docs.html
6. COCO Dataset, https://cocodataset.org/#home
7. JavaScript Web API dökümantasyonu, https://developer.mozilla.org/tr/docs/Web/API
8. HTML ve CSS tasarım rehberleri, https://developer.mozilla.org/tr/docs/Web/HTML ve https://developer.mozilla.org/tr/docs/Web/CSS
9. CoreUI Bootstrap tabanlı tema, https://coreui.io/
10. PlantUML, https://plantuml.com/
11. Graphviz, https://graphviz.org/
12. AmazonGO, https://www.amazon.com/b?ie=UTF8&node=16008589011

9. Terimler Listesi

9.1 Tanımlar

Çerçeve (Framework) - Yazılım geliştirmek için gerekli olan temel yapı ve araçları sağlayan, geliştiricilerin projelerini daha kolay ve düzenli bir şekilde oluşturabilmelerine olanak tanıyan yazılım geliştirme altyapısıdır.

Veritabanı - Verilerin düzenli ve erişilebilir bir şekilde saklandığı sistemdir. Projemizde SQLite kullanılmıştır.

Yüz Tanıma - Kamera görüntüsünden elde edilen verilerle yüz özelliklerini tanıyan ve eşleştiren bilgisayarla görme tekniğidir.

Nesne Tanıma - Görüntülerdeki belirli nesnelerin otomatik olarak tespit edilmesi ve sınıflandırılması işlemidir.

Arayüz - Kullanıcıların yazılım veya donanımlarla etkileşimini sağlayan ekran ve kontrollerin bütünü.

9.2 Kısaltmalar

API	Uygulama programlama Arayüzü
GUI	Grafiksel kullanıcı arayüzü
HTML	Köprü Metni İşaretleme Dili
css	Basamaklı Stil Şablonları
COCO	Ortak Nesne Sınıflandırma Veri Seti
PyQt5	Python Web Geliştirme Çatısı