İÇİNDEKİLER

ŞEKİL LİSTESİ

Sayfa No

SİMGELER

DC	Doğru akım / Direct Current
V	Volt

ÖZET

RASPBERRY Pİ İLE YÜZ TANIMALI GÜVENLİKLİ KASA

Beyza GÜMÜŞ
Ocak 2021, 27 sayfa

Raspberry Pi 3 modülü kullanılarak yüz tanımalı güvenlikli kasa yapılmıştır. Arayüz için Python Flask, yüz tanıma için ise OpenCV kütüphanesi kullanıldı. Yalnızca kullanıcının tanımladığı kişiler kasaya erişim sağlayabilir. Kasa erişiminde kilit 3 saniye açık kalmaktadır. Eğer erişim sağlanamaz ise önceden belirlenmiş e-posta adresine uyarı maili gider. Ayrıca kasaya erişim sağlayan kişiler ve bu kişilerin erişim tarih/saat bilgileri de listelenmektedir. Bu liste istenildiği takdirde PDF veya EXCEL formatında çıktı olarak alınabilir.

Anahtar Kelimeler: Raspberry Pi, Python, Flask, OpenCV, Selenoid kilit

ABSTRACT

FACE RECOGNIZED SAFETY CASE WITH RASPBERRY PI

Beyza GÜMÜŞ
January 2021, 27 pages

Face recognition safe box was made using Raspberry Pi 3 module. Python Flask was used for the interface and OpenCV library was used for face recognition. Only people defined by the user can access the safe. In safe access, the lock remains open for 3 seconds. If no access is available, a warning mail is sent to a predetermined e-mail address. In addition, the persons who have access to the safe and their access date / time information are also listed. This list can be printed in PDF or EXCEL format upon request.

Keywords: Raspberry Pi, Python, Flask, OpenCV, Solenoid lock

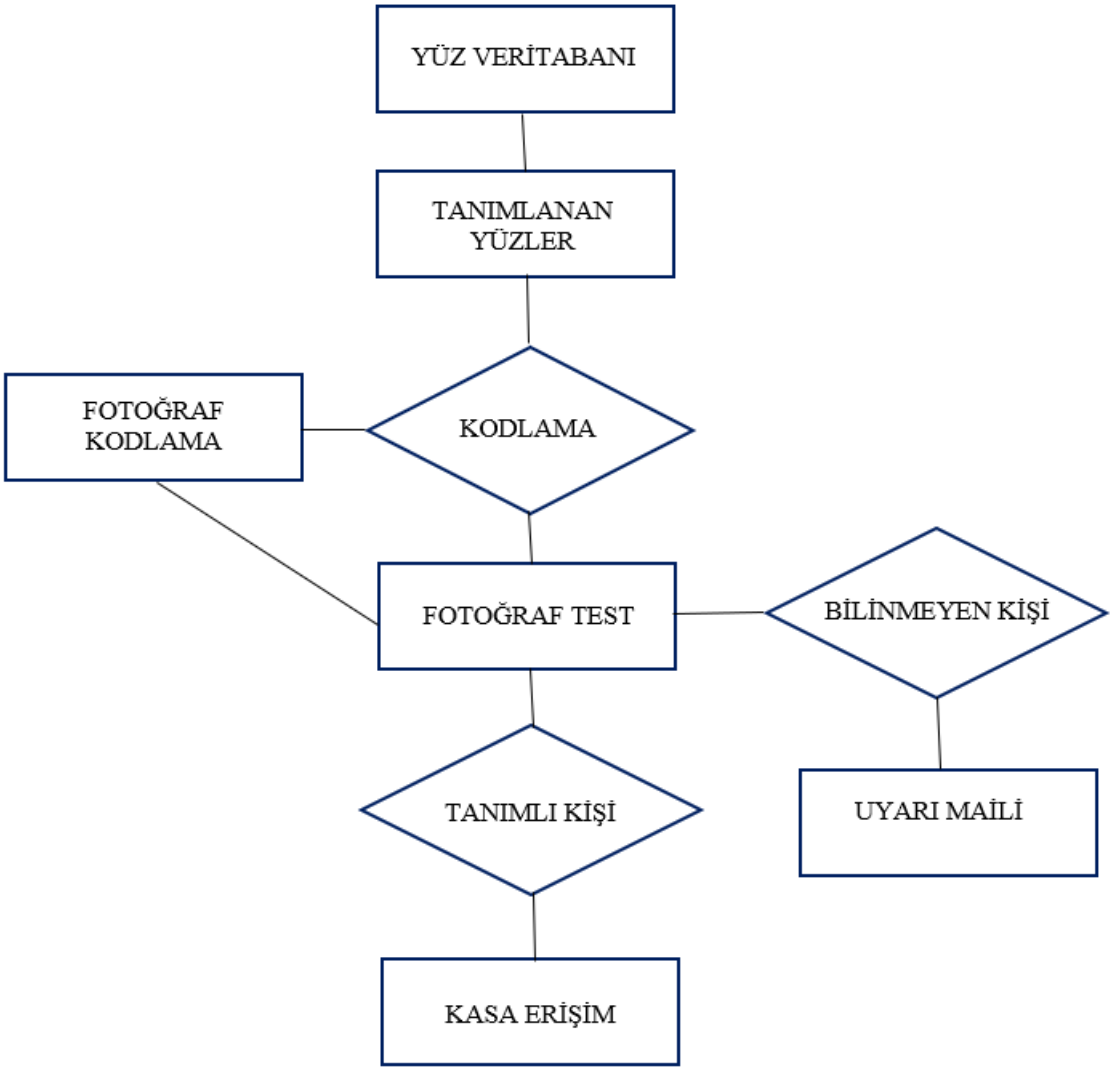
1. GİRİŞ

Günümüz teknolojisi hayatımızın her alanında yer almaktadır. 21.yy. etkileri ile kişisel yaşam ve gizlilik daha çok önem kazanmıştır. Teknolojinin her yerde bulunması ile kişiler gizliliklerine daha çok önem vermektedir. Kişi artık kişisel güvenliği ve gizliliği için teknolojiden faydalanabilir. Yapılardaki güvenlik kameraları, kapı kilit sistemi, biyometrik güvenlik sistemleri, güvenlikli kasa bunlara örnek olarak verilebilir. Bu tez çalışmasında ise güvenlikli kasa modülü oluşturulmuştur.

Güvenlikli kasa modülünde kişi, kasa içerisinde sakladığı nesneleri normal bir kilit sistemi yerine daha güvenlikli olan yüz tanıma sistemi ile saklamaktadır. Kasaya erişim yalnızca kayıtlı kişi veya kişilerin yüz taraması ile gerçekleşmektedir. Bu nedenle kasaya istenmeyen erişimler önlenir. Kişinin yüzü taranır ve kişisel bilgileri ile sisteme kayıt edilir. Erişim esnasında kayıtlı olan yüzler okunarak eşleştirme yapılır. Eşleşme sonucuna göre kasaya erişim sağlanır. Kayıtlı olmayan kişi erişiminde ise kasa sahibine fotoğraf ve uyarı maili gitmektedir.

2. METARYAL VE YÖNTEM

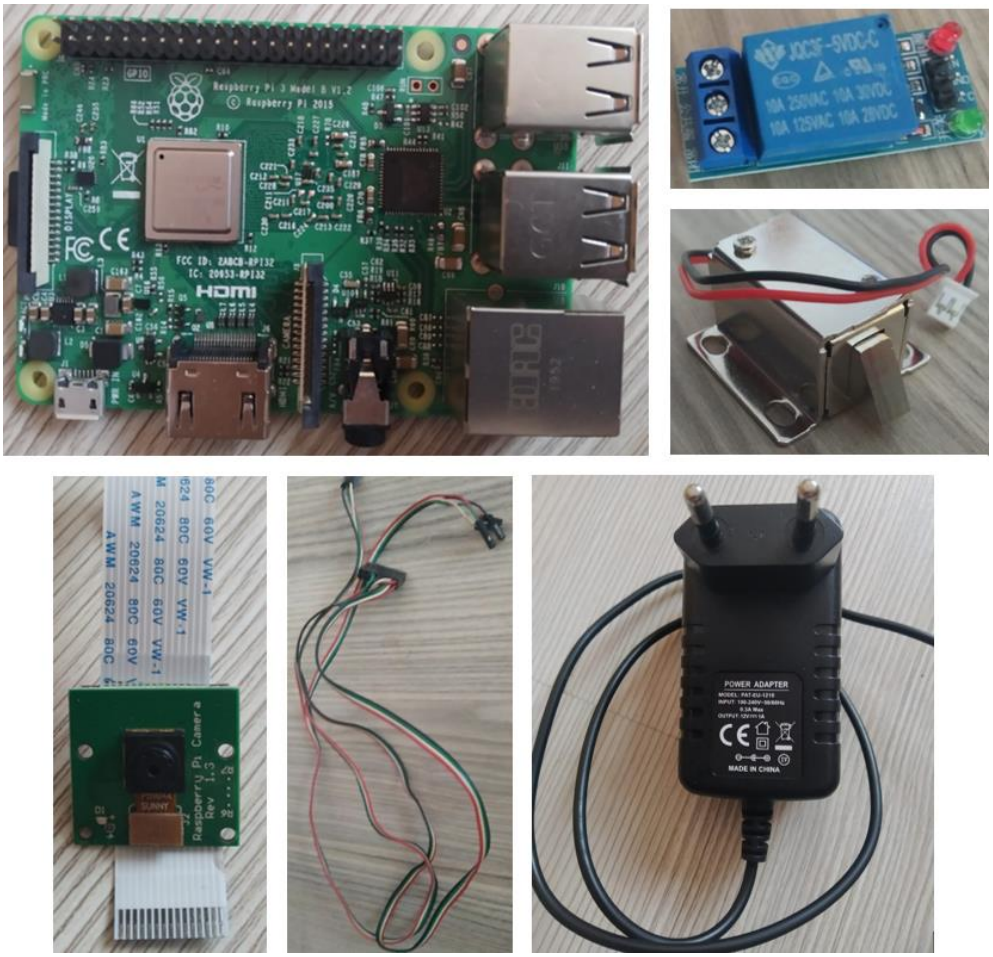
Sistem veritabanında kişi bilgileri ve yüz fotoğrafları tutulur. Fotoğraflar matris formatında kodlanır. Kasaya erişim durumunda bilgiler ile kodlanmış fotoğraf karşılaştırması yapılır. Yeteri kadar benzerlik olması halinde kasaya erişim sağlanır. Benzerlik yok ise sistem sahibine uyarı maili gider.



Şekil 1. Yüz tanıma akış diyagramı

Çalışma, proje ölçeğinde VNC Viewer arayüzü üzerinden gerçekleştirilmiştir. Bu arayüzü kullanmak için Raspberry Pi ve işlem yapılacak bilgisayar aynı ağa bağlı olmalı ve ikisinde de VNC Viewer yüklü olmalıdır. Projede kullanıcı için ise Python Flask ile arayüz oluşturulmuştur. Bunun için Python ve Flask da yüklü olmalıdır. Donanımsal ihtiyaçlar ise:

- 1 x Raspberry Pi 3
- 1 x 12V DC güç kaynağı
- 1 x Mini Selenoid kilit
- 1 x Raspberry kamera modülü
- 1 x Tek kanal röle kartı
- Atlama Kabloları



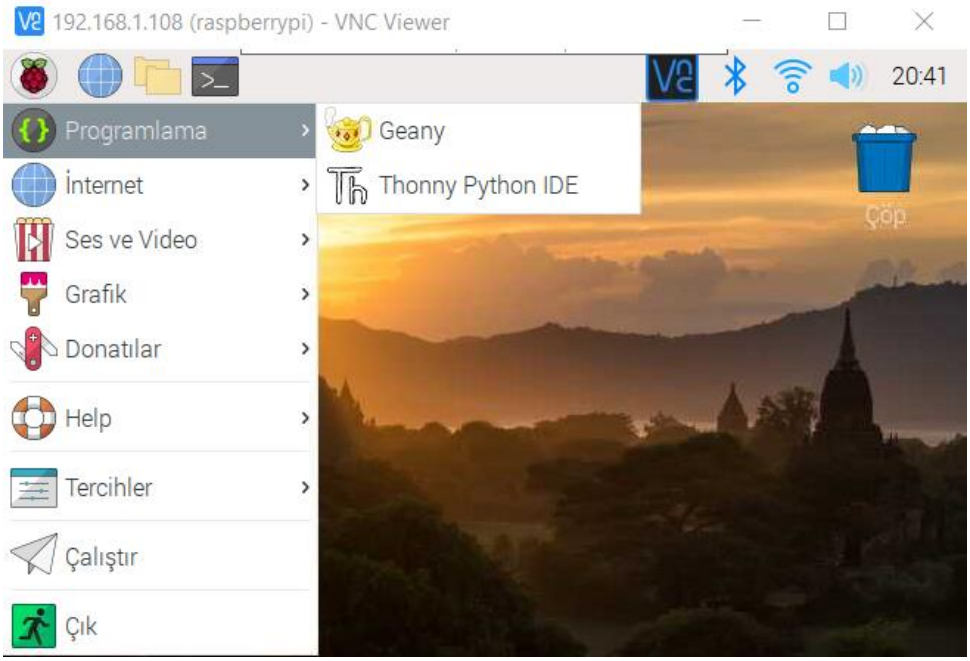
Şekil 2. Donanımsal Gereksinimler

Sistemin çalışabilmesi için kilit ve röle kart, atlama kabloları ile Rapberry'e bağlanmalıdır. Aynı zamanda kilidin ve röle kartın bir ucu 12V DC güç kaynağına bağlanmalıdır. Bu sayede kilit istenilen durumda elektrik akımı verilerek çalıştırılır.

3. KURULUM

3.1. Raspberry Pi

Çalışmada Raspberry 3 modülü kullanılmıştır. Bu modül SD kart ile çalıştırılmaktadır. Raspberry Pi küçük bir bilgisayardır. Üzerinde birçok IOT projesi geliştirip programlama öğrenilebilir. Hatta Web Server, Cloud File Server, DLNA Server, TOR Router / Proxy , VPN Server, IP Telefon Santrali olarak da kullanılabilir. Raspberry kullanımı için *'Raspian'* arayüzü Raspberry sitesinden indirilmelidir. Bu proje için '*Raspberry Pi OS with desktop*' kullanılmıştır. SD kart formatlanarak indirilen arayüz dosyası içerisinde bulunan imaj yüklenmelidir. Bu yükleme işlemi '*win32diskImager'* aracılığı ile yapılmıştır. Yükleme işleminin ardından SD kart Raspberry'e takılarak arayüzün kurulması beklenir. Arayüz kurulumu ardından Raspberry için kullanıcı adı ve şifre belirlenir. Kişisel olarak değişiklikler yapılabilir.



Şekil 3. Raspberry Pi Arayüzü

3.2. VNC Viewer

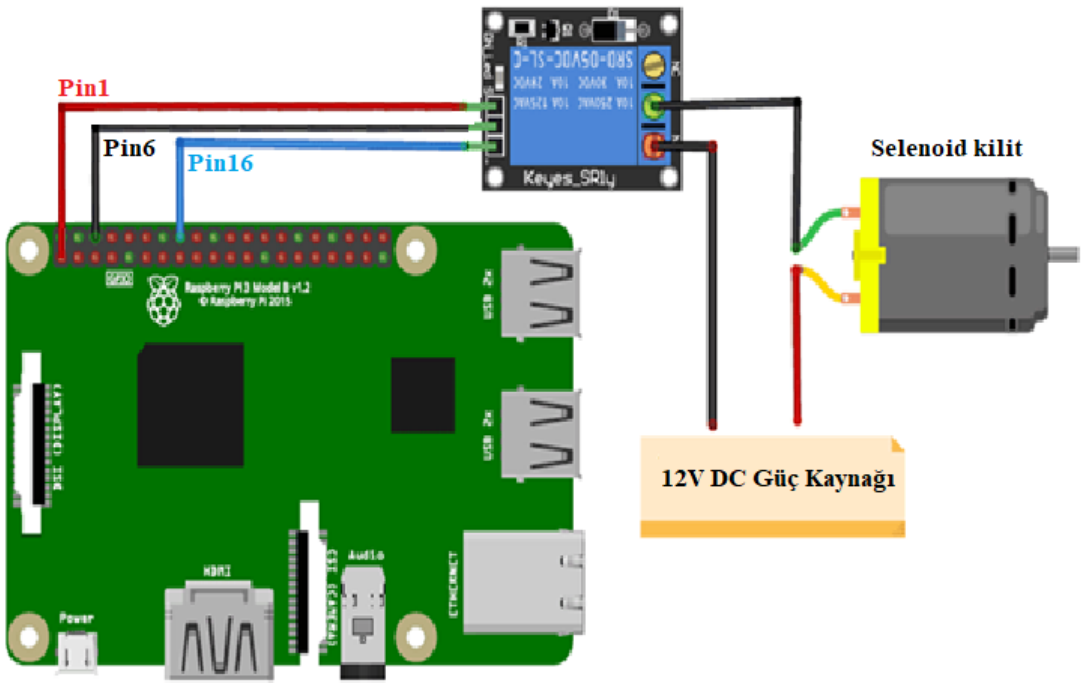
VNC (Virtual Network Computing) platform bağımsız ve açık kaynak kodlu ücretsiz uygulamadır. Uzak Çerçeve Arabelleği protokolünü kullanan bir paylaşım sistemidir. Temelde bir bilgisayarı uzaktan kontrol etmek için kullanılır.

VNC Viewer ile Raspberry bağlantısı için öncelikle aynı ağa bağlı olması gerekmektedir. Bağlantı için Raspberry ip'sine ihtiyaç vardır. Bunun için Raspberry'nin terminaline *'ifconfig'* yazılarak ip adres bulunur. Ardından *'sudo raspi-config'* komutu ile konfigürasyon aracına ulaşılır. Burada Arayüz seçenekleri altında bulunan VNC ve Kamera modülüne erişime izin verilmesi gerekir. Erişim izni olmaz ise bağlantı sağlanamaz.

Uzaktan erişim sağlanacak bilgisayara ve Raspberry'e VNC Viewer yüklenmelidir. Yükleme işleminin ardından bilgisayarda bulunan VNC arayüzüne Raspberry ip'si ile erişim sağlanabilir. Erişim esnasında Raspberry'nin kullanıcı adı ve şifresi de girilmelidir. Bu sayede Raspberry'e VNC ile uzaktan erişim sağlanır.

3.3. Bağlantılar

Sistem yüz tanıma işleminin ardından bağlantılar ile kilidin açılmasını sağlar. Röle kart, kilit ve Raspberry bağlantıları atlama kabloları ile yapılmıştır. Röle kart burada bir nevi anahtar görevindedir. Raspberry'den gelecek komuta göre kilide elektrik akımı verir. Röle kartın 3 ucu sırasıyla Raspberry'nin 1,6 ve 16. pinlerine bağlıdır. Rölenin çıkış tarafında bulunan iki uçtan birisi kilide, diğeri ise güç kaynağına bağlıdır.
Bu sayede röle aldığı güç ile kilidi kontrol edebilmektedir. Kilidin çalışabilmesi için ise bir ucu güç kaynağında olmalıdır. Ne zaman çalışacağı ise röleden gelen akıma göre belirlenir. Bu bağlantılar Şekil 4' de gösterilmiştir.



Şekil 4. Bağlantı

4. YÜKLEMELER

Çalışmada Python dili kullanılmıştır. Raspberry'de Python dili kurulu olarak gelmektedir. Eğer kurulu değilse terminalden *'sudo apt-get install python3-pip'* komutu ile yüklenebilir. Sıkıntısız işlem yapabilmek için terminalden sık sık *'sudo apt-get update'* ve *'sudo apt-get upgrade'* komutları çalıştırılmalıdır.

4.1. Flask

Flask, bir Python çerçevesidir(framework). Flask çabuk öğrenilebilir, yüksek performansta ve hızda çalışır. Güncel olan sistemimizde arayüz olarak kullanacağımız Flask için gerekli kurulumlar yapılmalıdır. Terminalde *'sudo pip3 install flask'* komutunu çalıştırarak yükleme işlemi yapılmaktadır. Eğer halihazırda yüklü ise *'flask --verison'* komutu ile hangi sürümün kullanıldığı öğrenilebilir.

4.2. OpenCV

OpenCV (Open Source Computer Vision) açık kaynak kodlu bir görüntü işleme kütüphanesidir. OpenCV platform bağımsızdır. Bu nedenle Windows, Linux, FreeBSD, Android, Mac OS ve iOS platformlarında rahatlıkla çalışabilmektedir. OpenCV kütüphanesi içerisinde görüntü işlemeye ve makine öğrenmesine yönelik 2500'den fazla algoritma bulundurur. Bu algoritmalar ile yüz tanıma, nesneleri ayırt etme, insan hareketlerini tespit edebilme, nesne sınıflandırma, plaka tanıma, üç boyutlu görüntü üzerinde işlem yapabilme, görüntü karşılaştırma gibi işlemleri yapılabilmektedir. Bu çalışmada ise OpenCV yüz tanıma işleminde kullanacağız. Öncelikli olarak OpenCV'yi Raspberry'a eklemek için gerekli bağımlılıklar yüklenmelidir. Bunun için terminale şu komutlar yazılmalıdır:

```
sudo apt-get install libhdf5-dev -y
sudo apt-get install libhdf5-serial-dev –y
sudo apt-get install libatlas-base-dev –y
sudo apt-get install libjasper-dev -y
sudo apt-get install libqtgui4 -y
sudo apt-get install libqt4-testi -y
```

Ardından OpenCV kütüphanesini yüklemeliyiz. Bunun için *'pip3 install opencv-contrib-python==4.1.0.25'* komutu çalıştırılmalıdır. Komut çalıştırılıp işlem tamamlanınca OpenCV kütüphanesi artık Raspberry'e eklenmiş olacaktır.

4.3. Paket Yüklemeleri

Yüz tanıma işlemi için gerekli paketlerin yüklenmesi gerekir. Bu paketler yüz tanımada ve yapılan işlemlerde bize kolaylık sağlar. Yüklenmesi gereken paketler aşağıda belirtilmiştir.

- **Dlib Kütüphanesi:** Makine öğrenmesi, görüntü işleme ve yapay öğrenme algoritmalarını içeren modern bir araç setidir. Bu projede görüntü işleme algoritmaları kullanılacaktır. Çoklu obje izlenmesi ve takibi de yapabilmektedir. Raspberry'e yüklenmesi için *'pip3 install dlib'* komutu kullanılmaktadır.

- **face_recognition Modülü:** Bu kütüphane, komut satırı aracılığıyla Python'daki yüzleri tanımak ve işlemek için kullanılır. Yüz tanıma kitaplığını kurmak için ise *'pip3 install face_recognition'* komutu kullanılmalıdır.

- **imutils Kütüphanesi**: Çevirme, döndürme, yeniden boyutlandırma, iskeletleştirme ve Matplotlib görüntülerini OpenCV ile görüntüleme gibi temel görüntü işleme işlevlerini kolaylaştırmak için kullanılır. Yüklemek için ise *'pip3 install imutils'* komutu yeterlidir.

- **Pillow Kütüphanesi**: Açılımı Python Image Library (Python Resim Kütüphanesi). Açık kaynak kodlu grafik işleme kütüphanesidir. Fotoğraf işlemlerini kolayca yapabilmemizi sağlar. Görüntüleri farklı bir biçimde açmak, işlemek ve kaydetmek için kullanılır. *'pip3 install pillow'* komutu ile kolayca yüklenmektedir.

5. KODLAMA

Bütün kodlar *'https://github.com/Fiziyanali/YuzTanima-OpenCV'* adresinde yayınlanmıştır.

5.1.Kütüphaneler

Projemizde veritabanı, yüz tanıma, kullanıcı işlemleri ve arayüz oluşturma gibi birçok işlem gerçekleştirmekteyiz. Bütün bu işlemler için kütüphanelerden faydalanıyoruz. Bunlar oluşturduğumuz kod sayfasının(app.py) en başına eklenmelidir. Bu kütüphaneler ve açıklamaları aşağıda verilmiştir.

```
from flask import Flask, render_template, request, Response, redirect, abort, url_for, flash, ses
sion  # Arayüz gereklilikleri
from functools import wraps
import sqlite3 as sql   # Veritabanı işlemleri için
from datetime import datetime,date
from passlib.hash import sha256_crypt  #  Veri şifreleme
import gc   # Bellek temizleme için
import os
from time import sleep   # Bekleme işlemi için
# Yüz tanıma ve kaydetme işlemleri için
import cv2
import numpy as np
from PIL import Image
import RPi.GPIO as GPIO   # Raspberry pin kontrol
import shutil   # Foto taşıma
# Mail gönderme işlemleri için
import smtplib
from picamera import PiCamera
from email.mime.multipart import MIMEMultipart
from email.mime.base import MIMEBase
from email.mime.text import MIMEText
from email.utils import formatdate
from email import encoders
```

5.2. Veritabanı

Veritabanında çeşitli tablolarda bilgiler tutulmaktadır. Arayüz kullanımında ve kasaya erişimde veritabanında bulunan bilgiler ile işlem yapılır. Seçme, ekleme, güncelleme ve silme işlemleri yapılmaktır. Flask üzerinden oluşturulan veritabanı kodları aşağıda verilmiştir. Bu kodlar kütüphane eklemelerinden sonra gelmektedir. Veritabanı kodları aşağıda verilmiştir.

```
conn = sql.connect('veritabani.db')   # Veritabanı bağlantısı
# Tablo Oluşturma İşlemleri
conn.execute('Create Table If Not Exıst yetki (yetkiID Integer Prımary Key Autoıncrement,k
ullanici_adi nvarchar(20) not null, sifre nvarchar(20) not null)')  # Admin Tablosu
conn.execute('Create Table If Not Exıst kisi (kisiID Integer Prımary Key Autoıncrement, tc i
nt(11) not null, ad nvarchar(20) not null, soyad nvarchar(20) not null, aktif bit not null,daimi t
inyint not null)')   # Kişi (kasaya erişim için) Tablosu
# Kasaya Erişim Hareket Tablosu
conn.execute('Create Table If Not Exıst hareket (hID Integer Prımary Key Autoıncrement, tc
int(11) not null, ad nvarchar(20) not null, soyad nvarchar(20) not null , tarih datetime)')

AdminSifre=sha256_crypt.hash('aa')  #Veritabanina gönderilen şifreyi hashliyoruz
conn.execute("INSERT INTO yetki(kullanici_adi,sifre) values (?,?)", ('admin',AdminSifre))
conn.commit()   # İşlemleri kaydet
conn.close()   # Veritabanı bağlantısını kapat
```

5.3. Yüz Tanıma İşlemi

Yüz tanıma işlemini temelde 3 aşamada gerçekleştirmekteyiz. Bunlar sırasıyla; veri toplama, tarama(training) ve yüz tanıma işlemleridir.

5.3.1. Veri Toplama

Çalışmanın ilk aşamasında, algılanan yüzleri depolamak için bir veri kümesi oluşturulmalıdır. Yüzler farklı kimliklerle saklanacaktır. Bunun için önce çalışma verilerinin kaydedileceği bir proje dizini oluşturmalıdır. Bütün yüz verileri ‘dataset’ klasörünün altında tutulmaktadır.

Veri toplama kodları çalıştığında yüz sınıflandırıcı dosyası bir ‘*face_detector*’ değişkeni ile kullanılır. Yüz sınıflandırıcı, proje içerisinde yer alan *‘haarcascade_frontalface_default.xml’* dosyasıdır. Proje verileri toplanmadan önce sayısal yüz kimliğini girebilmesi için kişinin bilgilerini ve veritabanında kayıtlı id’si kullanır. Yüz verileri ‘Kisi.kisiID.resimSayisi.jpg’ formatında kayıt edilir. While döngüsü kullanılarak kişinin 30 adet fotoğrafı kayıt edilir. Genel veri toplama kodu ve açıklamaları aşağıda verilmiştir.

```
cam = cv2.VideoCapture(0)
cam.set(3, 640)   # Video genişliği ayarı
cam.set(4, 480)   # Video yüksekliği ayarı
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
count = 0
```

```
while(True):
    ret, img = cam.read()
    img = cv2.flip(img, -1)    # Video görüntüsünü dikey olarak cevir
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
        count += 1
        cv2.imwrite("static/dataset/aktif/Kisi." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+
h,x:x+w])   # Resimleri belirlenen adrese kayıt et
    if count >= 30:   # 30 adet foto çekildiyse
        break
cam.release()   # Kamerayı kapat
```

5.3.2. Tarama (Training)

Yüz verileri toplandıktan sonra, yüzleri doğru bir şekilde tahmin edebilmesi için yüzler matris olarak kodlanıp kayıt edilmektedir. Tanımlanan veri resimleri üzerindeki yüz ve id ile birlikte tarama işlemi yapmaktadır. ‘trainer’ klasörünün altında ‘yml’ formatında kaydedilir. Yüz tanıma işlemi yapılırken bu dosyada bulunan kişiler ve id’leri üzerinden karşılaştırma yapılır. Yüzleri kodlayabilmesi için cv2, numpy(np olarak), ve PIL kütüphaneleri eklenmelidir. Örnek görüntülerdeki yüzleri algılamak için *‘haarcascade_frontalface_default.xml’* yüz sınıflandırıcı dosyası kullanılır. Ardından, bir LBPH (Yerel İkili Model Histogramı) Yüz Tanıyıcı oluşturmak için tanıyıcı değişkeni kullanılır. Görüntü örnekleri gri tonlamaya dönüştürülmelidir. Bundan sonra, PIL görüntüsünü numpy görüntüye dönüştürülür. Genel tarama kodu ve açıklamaları aşağıda verilmiştir.

```
path = 'static/dataset/aktif'  # Veri yolu belirlenir
recognizer = cv2.face.createLBPHFaceRecognizer()
detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml');
def getImagesAndLabels(path):  # Görüntü ve etiketi al
    imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]  # Yüz örnekleri
    ids = []  # id’ler
    for imagePath in imagePaths:
        PIL_img = Image.open(imagePath).convert('L')   # gri tonlama
        img_numpy = np.array(PIL_img,'uint8')
        id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)
        for (x,y,w,h) in faces:
```

```
            faceSamples.append(img_numpy[y:y+h,x:x+w])
            ids.append(id)
    return faceSamples,ids
faces,ids = getImagesAndLabels(path)
recognizer.train(faces, np.array(ids))
recognizer.save('trainer/trainer.yml')   # Kaydedilecek adres
```

5.3.3. Yüz Tanıma

Yüz tanımlama işlemi kasaya erişim esnasında gerçekleşir. Canlı video akışı ile önceden tanımlanmış, bilgileri ve yüzleri kodlanmış kişiler arasında karşılaştırma yapılır. Karşılaştırma sonucuna göre kasaya erişim sağlanır. Erişim sağlayan kişi bilgileri 'hareket' tablosuna kayıt edilir. Eğer erişim sağlanamaz ise, kişinin fotoğrafı önceden belirlenen e-posta adresine uyarı maili ile birlikte gönderilir. Yüz tanıma kodları tarama kodları ile benzerdir. Aynı kütüphane dosyalarını ve sınıflandırıcı dosyasını kullanır. Canlı video akışında kamera genişlik ve yükseklik ayarlamasından sonra, while döngüsünün içinde videoyu görüntülere böler ve ardından gri tonlamaya dönüştürür. Yüzün örneklerle ne kadar eşleştiğini kontrol etmek için recognizer.predict işlevi kullanılır. Benzerlikte ise 0 mükemmel eşleşmeyi temsil eder. Kasaya erişim için benzerlik sınırı %40 üzeridir. Genel yüz tanıma kodu ve açıklamaları aşağıda verilmiştir.

```
# Pin ayarları
relay = 23
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(relay, GPIO.OUT)
GPIO.output(relay ,1)   # Kilit kapalı
recognizer = cv2.face.createLBPHFaceRecognizer()
recognizer.load('trainer/trainer.yml')
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
font = cv2.FONT_HERSHEY_SIMPLEX
id = 0
# names, id'lere bağlı: Örnek ==> Beyza: id=1
# ID'ler 1'den başladığı için 0.indisler bos
names = ['Bos']
surnames = ['Bos']
conn = sql.connect("veritabani.db")
c= conn.cursor()
```

```
c.execute("SELECT ad FROM kisi where aktif=?", [True])
adlar = c.fetchall()   # İsimleri ata
for i in adlar:
    names.extend(i)
c.execute("SELECT soyad FROM kisi where aktif=?", [True])
soyadlar = c.fetchall()   # Soyadları ata
for i in soyadlar:
    surnames.extend(i)
cam = cv2.VideoCapture(0)
cam.set(3, 640)
cam.set(4, 480)
# Yüz olarak tanınacak minimum pencere boyutu
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)
cc=0
while cc==0:
    ret, img =cam.read()
    img = cv2.flip(img, -1)
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray, scaleFactor = 1.2, minNeighbors = 5,
minSize = (int(minW), int(minH)), )
    for(x,y,w,h) in faces:
        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
        id, confidence = recognizer.predict(gray[y:y+h,x:x+w])
        # Benzerlik kontrolü. 100 ==> "0" mükemmel eşleşme
        if (confidence < 60): # Benzerlik %40 üzerinde ise
            id2 = names[id]
            id3 = surnames[id]
            ad_soyad= str(id2) +" "+ str(id3)
            c.execute("SELECT * FROM kisi where kisiID=?", [id])
            data = c.fetchone()
            now = datetime.now()
            tarih = now.strftime("%d/%m/%Y %H:%M:%S")
            c.execute("INSERT INTO hareket (tc, ad, soyad, tarih) values (?,?,?,?)", [data[1],data[
2],data[3],tarih])
            conn.commit()
            confidence = "{0}%".format(round(100 - confidence))
            GPIO.output(relay, 0)   # Kilit açık
            sleep(3)   # 3 saniye bekle
            GPIO.output(relay, 1)   # Kilit kapalı
            flash("Kilit açılıyor. Kişi >>>> " + ad_soyad)
            cc+=1
        else:   # Benzerlik %40 ve altında ise
            id = "Bilinmiyor"
```

```
        confidence = " {0}%".format(round(100 - confidence))
        GPIO.output(relay, 1)   # Kilit kapalı
        ad_soyad= str(id)
        flash("Kilit açılmıyor. Kişi >>>> " +ad_soyad)
        cc+=1
        return redirect(url_for('mail'))   #Bilinmeyen kişi ise uyarı mail gönder
cam.release()
```

5.4. Mail Gönderme

Mail göndermek için SSMTP aracını kullanacağız. Bu araç maili yerel bir bilgisayardan yapılandırılmış bir posta barındırıcısına (mailhub) ileten bir programdır. Bu bir posta sunucusudur ve posta almaz. Birincil kullanımlarından biri, otomatik e-postayı (sistem uyarıları gibi) makinenizden harici bir e-posta adresine iletmektir. Bu çalışmada ise kasaya geçersiz giriş yapıldığında belirlenmiş e-posta adresine kişinin fotoğrafı ile birlikte uyarı maili gönderilecektir.

Öncelikli olarak Raspberry'e SSMTP aracı yüklenmelidir. Bunun için terminale *'apt-get install ssmtp'* komutu yazılmalıdır. Yükleme işleminin ardından konfigürasyon dosyası olan ssmtp.conf'a gmail hesap bilgileri tanımlanmalıdır. Bunun için *'vi /etc/ssmtp/ssmtp.conf'* komutu ile dosyaya erişim sağlanır. Burada girilecek bilgiler uyarı maili hangi e-posta üzerinden gönderilecek ise o olmalıdır. Örnek düzenleme Şekil 5'de verilmiştir. Düzenlemelerin ardından mail hesabınızda 'Güvenlik → Daha Az Güvenli Uygulama Erişimi' seçeneği açık olmalıdır. Aksi taktirde mail gönderimi yapılamaz.

```
#
# Config file for sSMTP sendmail
#
# The person who gets all mail for userids < 1000
# Make this empty to disable rewriting.
root=kullanici_adiniz@gmail.com

# The place where the mail goes. The actual machine name is required no
# MX records are consulted. Commonly mailhosts are named mail.domain.com
mailhub=smtp.gmail.com:587

# Where will the mail seem to come from?
#rewriteDomain=

# The full hostname
hostname=kullanici_adiniz@gmail.com

UseSTARTTLS=YES
AuthUser=kullanici_adiniz@gmail.com
AuthPass=parolaniz

FromLineOverride=YES

#Debug=YES

# Are users allowed to set their own From: address?
# YES - Allow the user to specify their own From: address
# NO - Use the system generated From: address
#FromLineOverride=YES
```

Şekil 5. SSMTP Konfigürasyonu

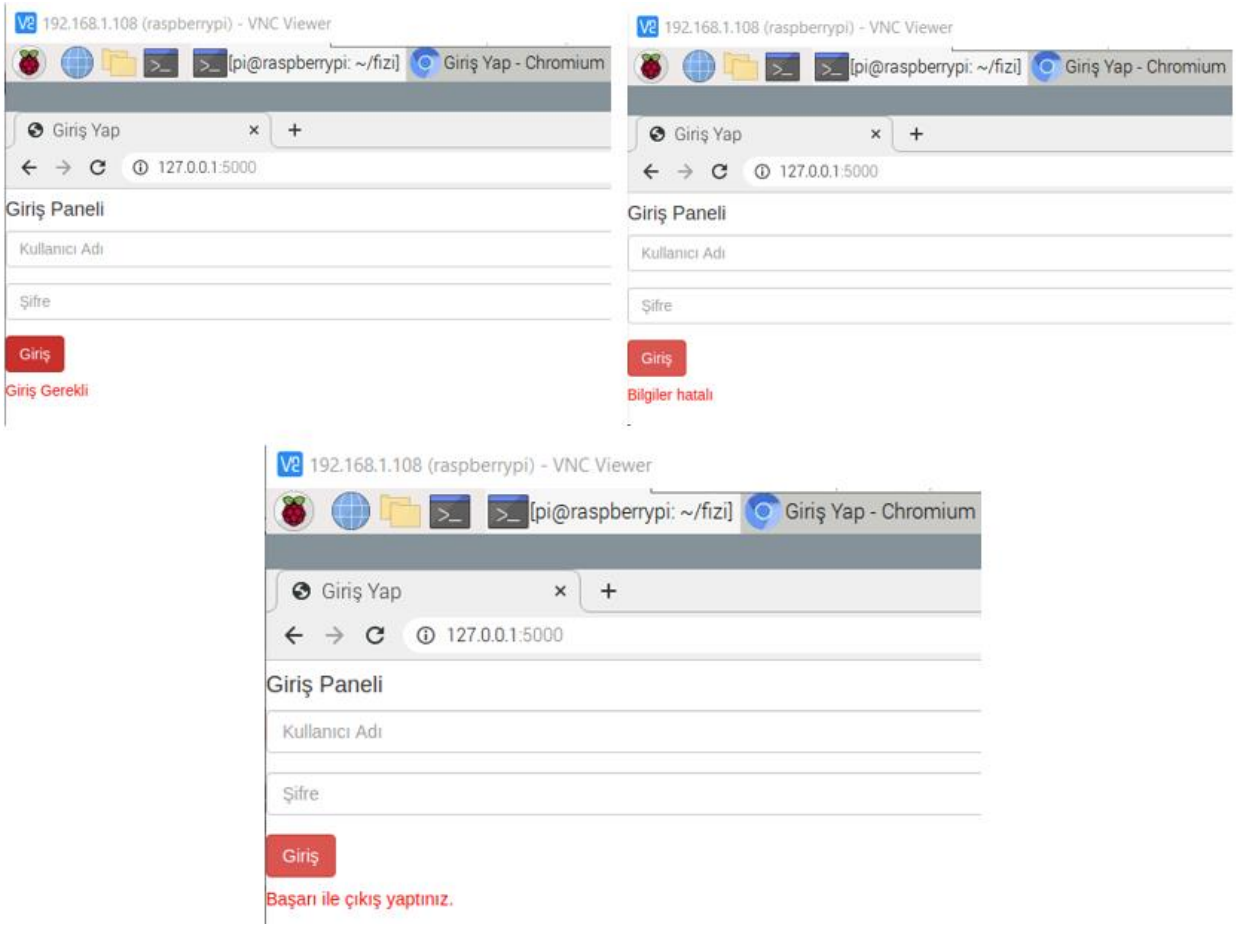
Kurulum ve konfigürasyon işleminin ardından gerekli kodlar ile sistem çalıştırılabilir. Mail gönderme kodu ve açıklamaları aşağıda verilmiştir.

```
camera = PiCamera()    # Kamera tanımı
camera.start_preview()   # Mail için resim çek
sleep(1)   # 1 saniye bekle
camera.capture('/home/pi/fizi/bilinmeyen.jpg')   # Resmi bu yola kaydet
sleep(1)   # 1 saniye bekle
camera.stop_preview()   # Resim çekimini durdur
camera.close()   # Kamera kapat
to='beyzagumus64@gmail.com'   # Gönderilecek mail
me='guvenliKasaErisim@gmail.com'   # Gönderen kişi
subject = "Kasa UYARI !!"    # Mail Konu
# Mail için ayarlamalar
msg = MIMEMultipart()
msg['Subject'] = subject
msg['From'] = me
msg['To'] = to
msg.preamble ="test"
text="Bilinmeyen kişi kasaya erişmeye çalıştı !!"   # Mail içerik
msg.attach(MIMEText(text))
part = MIMEBase('application', "octet-stream")
resim= Image.open("bilinmeyen.jpg")   # Çekilen resmi aç
resim2= resim.rotate(180)   # Resmi  döndür.
resim2.save("bilinmeyen2.jpg")   # Döndürülen resmi yeni isim ile kaydet
part.set_payload(open("bilinmeyen2.jpg", "rb").read())   # Resim yükle
encoders.encode_base64(part)
# Dosya ismi ve format ismi
part.add_header('Content-Disposition', 'attachment; filename="image.jpg"')
msg.attach(part)
s = smtplib.SMTP('smtp.gmail.com', 587)    # Protokol (server)
s.starttls()
s.login("guvenliKasaErisim@gmail.com", "MailSifresiYazilacak")   # Server için giriş
# Mail gönderme işlemi
s.sendmail("guvenliKasaErisim@gmail.com", "beyzagumus64@gmail.com", msg.as_string())
s.quit()  # Server çıkış
```

6. ARAYÜZ KULLANIMI

6.1. Kullanıcı Girişi

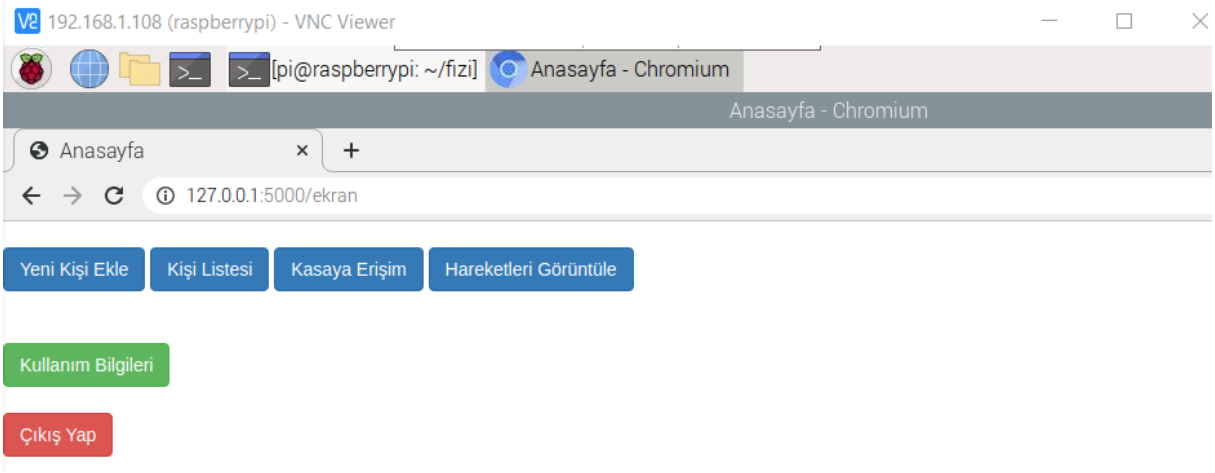
Önceden tanımlanmış kullanıcı(admin) bilgileri doğru girildiğinde kasa arayüzüne erişim sağlanabilir. Kullanıcı bilgileri veritabanına kodlanarak gönderilir ve çekilir. Kullanıcı girişi olmadan alt sayfalara erişim asla sağlanamaz. Hatalı giriş, kullanıcı çıkışı ve giriş gerekli durumlarında kullanıcı giriş sayfasına bilgi mesajı ile birlikte yönlendirilme yapılır.



Şekil 6. Kullanıcı Giriş Arayüzü

6.2. Ana Ekran

Kullanıcı girişinin ardından ana ekrana yönlendirilir. Burada sistemde kayıtlı olan kişilere, kasaya, kullanım bilgilerine ve hareket listesine erişim sağlanmaktadır. Aynı zamanda kullanıcının sistemden çıkışı da bu sayfa içerisinde bulunmaktadır.

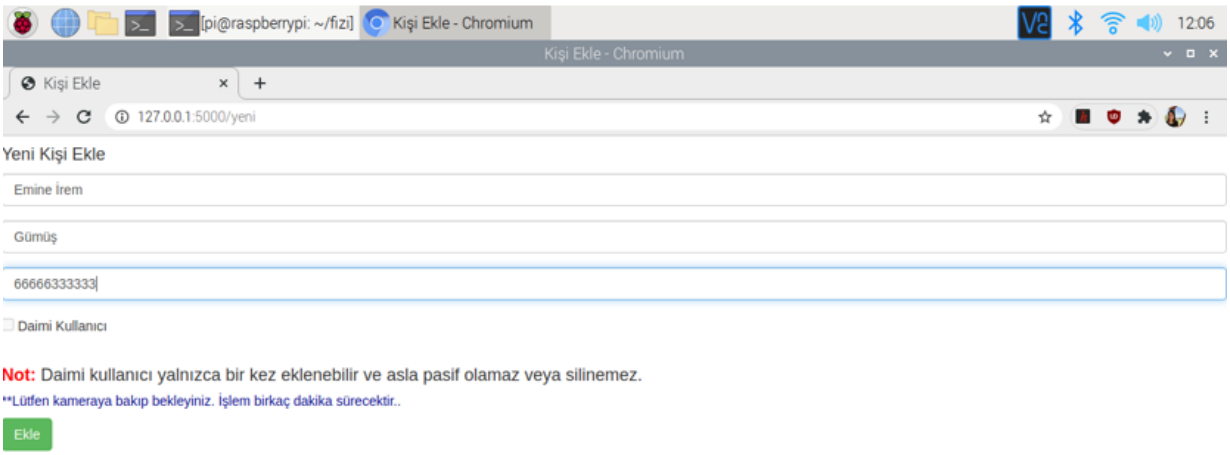


Şekil 7. Ana Ekran

6.3.Yeni Kişi Ekleme

Kasaya erişim için kişi bilgileri ve yüz tanıma işlemi gerçekleştirilmelidir. Kayıt işlemi için kişinin Adı-Soyadı-Tc numarası gereklidir. Eğer Tc numarası daha önce kayıtlı ise ekleme işlemi yapılamaz. Bilgiler girildikten sonra yüz tamamen açık ve belirgin olacak şekilde kameraya bakılmalıdır. Bu işlem birkaç dakika sürebilir. Kişi ekleme işleminde veritabanına kayıt edildikten sonra yüz taranarak kodlanır. Kasaya erişimde yüz kodlamaları ile kıyas yapılarak erişim sağlanır.

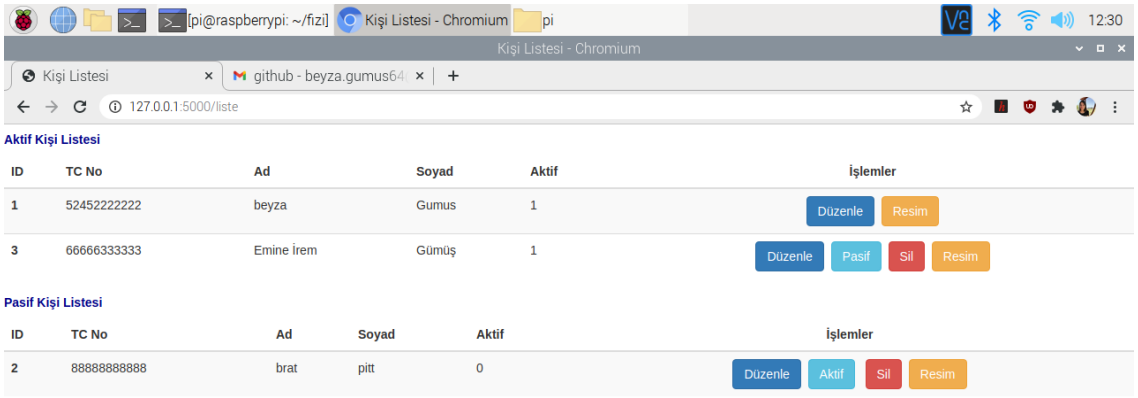
Not: Kasada tanımlı daimî kullanıcı asla pasif olamaz veya silinemez. Yalnızca 1 daimî kullanıcı tanımı yapılabilir. Bu kullanıcı tanımı genelde kasa sahibidir.



Şekil 8. Yeni Kişi Ekleme Arayüzü

6.4. Kişi Listesi

Sistemde kayıtlı olan kişilerin listesi bu ekranda yer alır. Aktif ve pasif kişiler ayrı tablolarda listelenir. Kişiler üzerinde düzenleme ve silme işlemleri yapılabilmektedir.



Şekil 9. Kişi Listesi

- **Aktif Kişiler:** Sistemde kayıtlı ve kasaya erişimi olan kişi listesidir. Bu kişiler istenildiği zaman pasif yapılabilir, bilgileri düzenlenebilir ve silinebilir.

- **Pasif Kişiler:** Sistemde kayıtlı ve kasaya erişimi olmayan kişi listesidir. Pasif olan kişilerin bilgileri ve yüz taraması silinmez. Yalnızca kasaya erişimi engellenir. Pasif halde kasaya erişmeye çalışılır ise belirlenmiş olan e-posta adresine uyarı maili gitmektedir. Bu kişiler istenildiği zaman aktif yapılabilir, bilgileri düzenlenebilir ve silinebilir.

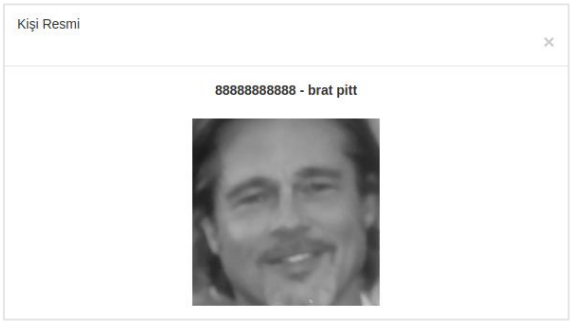
- **Silme İşlemi:** Seçilen kişinin bilgilerini ve yüz taramasını sistemden tamamen siler. Kişi kasaya erişmek isterse yeniden kayıt olması gerekir. Silme işlemi öncesi uyarı mesajı gösterilir.

Şekil 10. Silme İşlemi

- **Düzenleme İşlemi:** Seçilen kişinin bilgileri düzenlenebilir. TC no sistemde başkasına ait ise düzenleme işlemi gerçekleşmez.

Şekil 11. Düzenleme İşlemi

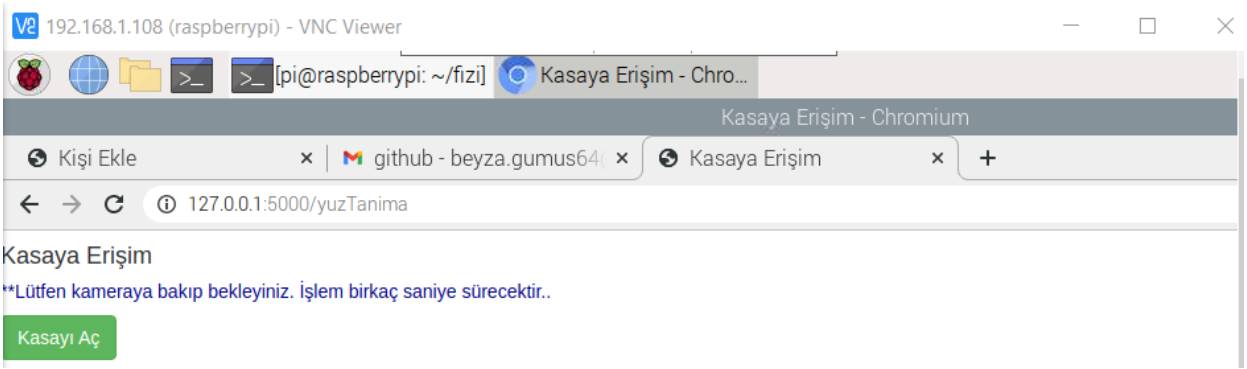
- **Resim:** Kayıtlı kişinin fotoğrafını gösterir. (Kaydedilen 5. fotoğrafı)



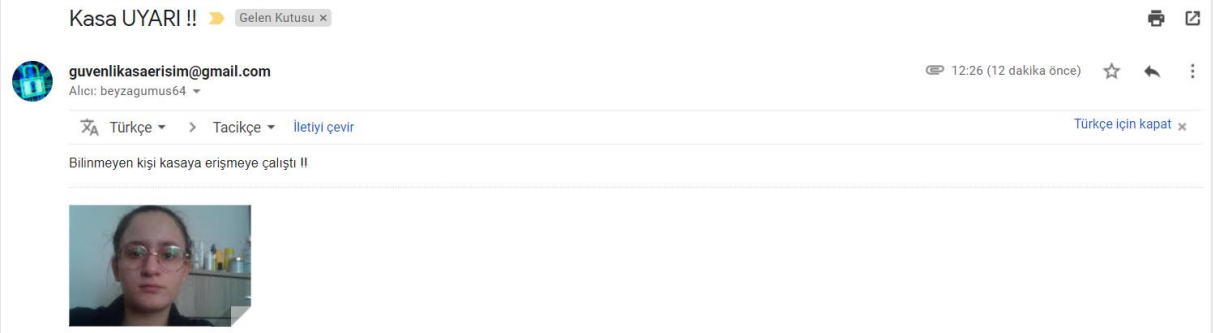
Şekil 12. Kişi Resmi

6.5. Kasaya Erişim

Kasaya erişim için kayıtlı olan kişinin yüz taraması yapılır. Tarama için yüz belirgin olacak şekilde sabit kalarak kameraya bakılmalıdır. İşlem yalnızca birkaç saniye sürer. Kişi sistemde bulunamaz veya pasif halde ise önceden belirlenen e-postaya uyarı gider. Yüzü tanımlı olan kişiler ise kasaya erişim sağlar. Erişimi sağlayan kişinin bilgileri ve erişim saati hareket listesine eklenir.



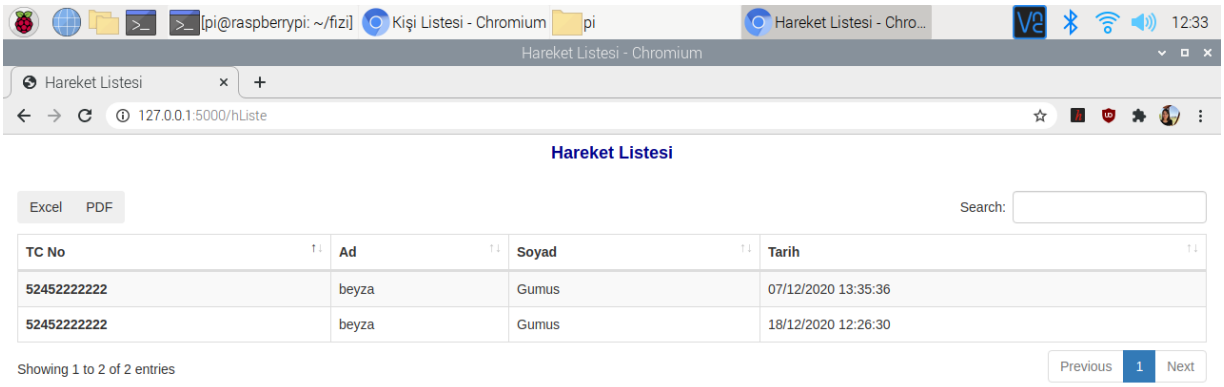
Şekil 13. Kasaya Erişim Arayüzü



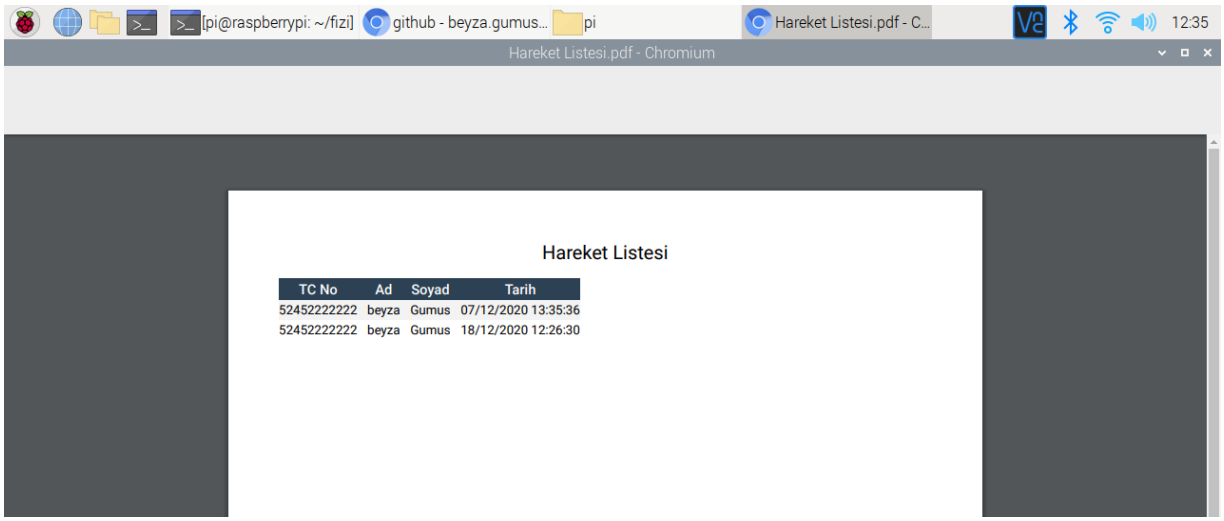
Şekil 14. Uyarı Maili

6.6. Hareket Listesi

Kasaya önceden erişim sağlamış kişiler burada listelenir. Kişinin bilgileri, erişim tarihi ve saati yer alır. Ayrıca liste PDF ve EXCEL olarak alınabilir.



Şekil 15. Hareket Listesi Arayüzü



Şekil 16. Hareket Listesi (PDF)

7. SONUÇLAR VE ÖNERİLER

Bu çalışmada Raspberry Pi kullanılarak arayüze sahip yüz tanımalı güvenlikli kasa modülü oluşturulmuştur. Çalışmada donanımsal malzemeler ve programlamadan faydalanılarak yüz tanıma yapılmıştır. Bu sayede kullanıcı güvenliğini teknoloji ile daha üst seviyelere çıkarılmıştır. Arayüze kullanıcı adı ve şifre ile giriş yapıldıktan sonra kasaya erişim sağlanabilmektedir. Yalnızca sistemde kayıtlı ve aktif kişiler kasaya erişim sağlayabilir. Erişimi olmayan kişilerin fotoğrafları kullanıcıya uyarı maili ile birlikte gönderilmektedir. Bu sayede yüz tanıması kullanılarak kullanıcının izin verdiği kişilerin erişimi sağlanır ve istenmeyen erişim denemelerinde ise kasa sahibi uyarılır.

Bu çalışmada kasanın yerinin değiştirilmesi veya zor kullanılarak erişilmesini önlemek için belirli sensörler kullanılabilir. Örneğin; basınç veya titreşim sensörü verileri sisteme kayıt edilir. Anormal bir değişim esnasında kullanıcıya uyarı mesajı gider ve bu sayede kasaya zor kullanım ile erişimin önüne geçilebilir.

KAYNAKLAR

[1] Hakan GÜNDÜZ, Web programlama, *Ders Notları,* Düzce, 2019.

[2] Hakan GÜNDÜZ, Görüntü işlemeye giriş, *Ders Notları,* Düzce, 2019.

[3] Iot Design Pro. (2020, 26 Mayıs). Face Recognition Door Lock System using Raspberry Pi [Online]. Erişim: https://www.iotdesignpro.com/projects/face-recognition-door-lock-system-using-raspberry-pi.

[4] Mjrovai . (2017). Real-time Face Recognition: an End-to-end Project [Online]. Erişim: https://www.instructables.com/Real-time-Face-Recognition-an-End-to-end-Project/.

[5] Max Countryman (2020, 13 Ekim). flask-login [Online]. Erişim: https://github.com/maxcountryman/flask-login/tree/master.

[6] I.Yugashini, S.Vidhyasri, K.Gayathri Devi, "Design And Implementation Of Automated Door Accessing System With Face Recognition," *International Journal of Science and Modern Engineering (IJISME),* ISSN: 2319-6386, Volume-1, Issue-12, November 2013.

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Beyza GÜMÜŞ

Doğum Tarihi ve Yeri : 16/12/1998

Yabancı Dili : İngilizce

E-posta : beyzagumus64@gmail.com

ÖĞRENİM DURUMU

Derece	Alan	Okul/Üniversite	Mezuniyet Yılı
Lisans	Bilgisayar Müh.	Düzce Üniversitesi	2021
Lise	Bilişim Teknolojileri	Üsküdar Mesleki ve Teknik Anadolu Lisesi	2016