Bil 372

Covid-19 management center project

Haktan ÇetinTOBB Economics and Technology University, Computer Science Department, Ankara 06560, Turkey , Ahmet Burak ÖzgüngördüTOBB Economics and Technology University, Computer Science Department, Ankara 06560, Turkey, İbrahim Bahadır AltunTOBB Economics and Technology University, Computer Science Department, Ankara 06560, Turkey, Tuna BaydemirTOBB Economics and Technology University, Computer Science Department, Ankara 06560, Turkey

Abstract

This article intends to discuss a Python Flask and SQLAlchemy based web application that connects to a PostgreSQL based database to be used to monitor Covid-19 cases in a country.

© 2012 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of Global Science and Technology Forum Pte Ltd

Database; management; application; sql; Python Flask; PostgreSQL

In this paper, the layout and application of a Covid-19 database management system will be examined thoroughly, from the front-end to the back-end of development. This system’s aim is to allow healthcare workers to update and have access to a synchronized database where necessary information regarding each patient is stored.

* 1. Front end design

To implement this project’s front end design, the team used a series of HTML forms supported by CSS to construct a series of forms for the user, depending on their rank, to:

* insert new possible cases,
* update the status of said possible cases, turning them into possible cases,
* turn the possible cases into actual patients,
* record those who have been in contact with the patients,
* change the patients’ status to reflect their status(currently undergoing treatment, successfully discharged, deceased), and add new locations where necessary.
  1. Back end design

The project’s back end design was created through use of Python Flask, SQLAlchemy and PostgreSQL. Python Flask was used to create links to the pages containing the specific HTML forms discussed above and pass these forms’ contents to select SQLAlchemy statements that updated the database. These SQLAlchemy statements took the place of actual SQL queries that were connected to a database created through PostgreSQL.

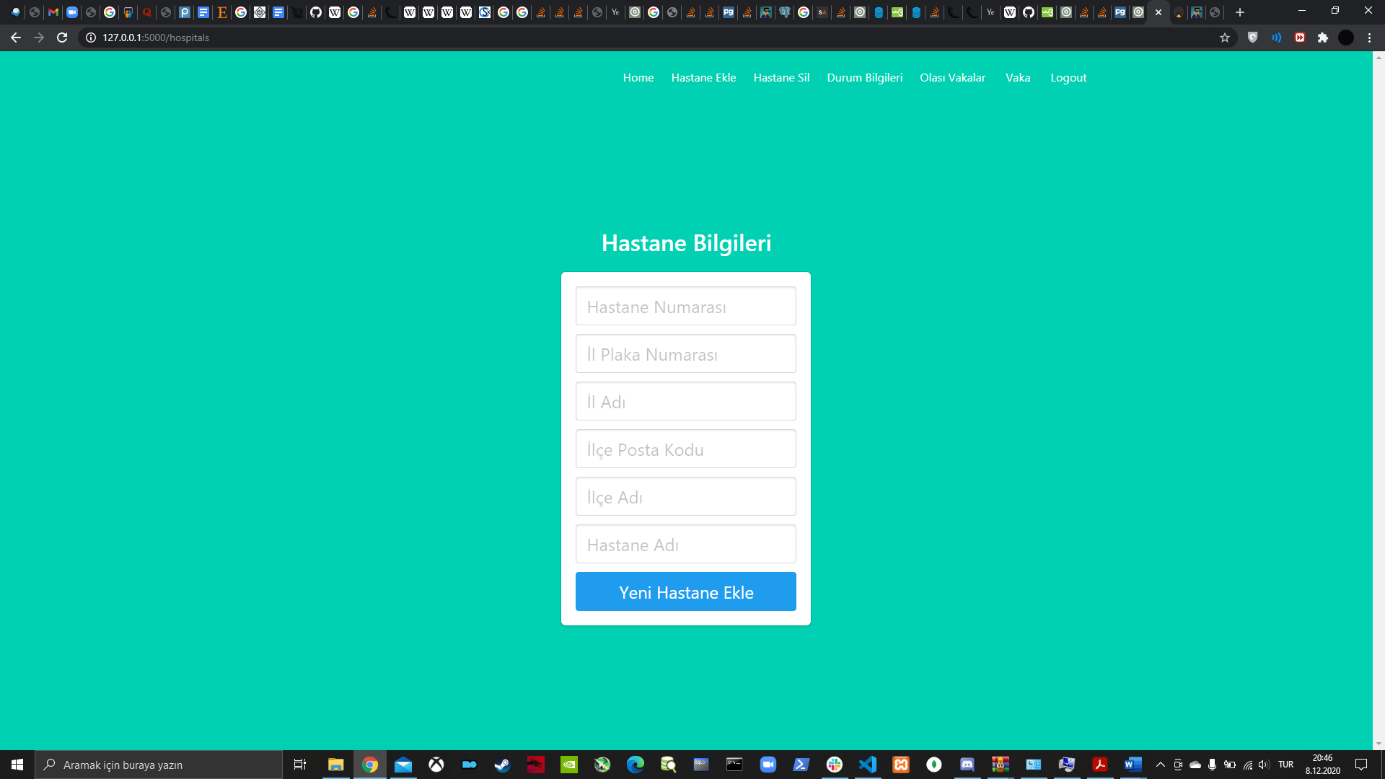
* 1. Task division

Haktan Çetin: Insertion and deletion of medicine lists per patient and hospitals, initial database design.

İbrahim Bahadır Altun:

Tuna Baydemir: Insertion and deletion of location related information along with visual representation for said information.

Ahmet Burak Özgüngördü: Insertion, deletion and updates of all patient-related forms, final database design.

1. Author Artwork
   1. GUI’s and SQL statements

This form handles the insertion of a new hospital to the database. In it, the user is asked to enter a hospital ID number ‘hastaneno’, the city license plate number that the hospital belongs to ‘ilplakano’, the name of said city ‘ilismi’, the zipcode of the county the hospital belongs to ‘ilcepostakodu’, the name of the county ‘ilceismi’ and the name of the hospital itself ‘isim’. When the blue ‘Yeni Hastane Ekle’ button is clicked, the data in the forms gets used in the following SQL statements:

il= select \* from Il where plakano=%s (ilplakano)

//Used to determine the city with given license plate number.

ilce= select \* from Ilce where plakano= %s and postakodu=%s (ilplakano ,ilcepostakodu)

//Used to determine the county with given license plate number and zipcode.

hastane= select \* from Hastane where hastaneno = %s and plakano= %s and postakodu= %s (hastaneno, ilplakano, ilcepostakodu)

//Used to determine the hospital with given hospital ID, license plate number and county zipcode.

insert into Il values(ilplakano, ilismi, 0,0,0)

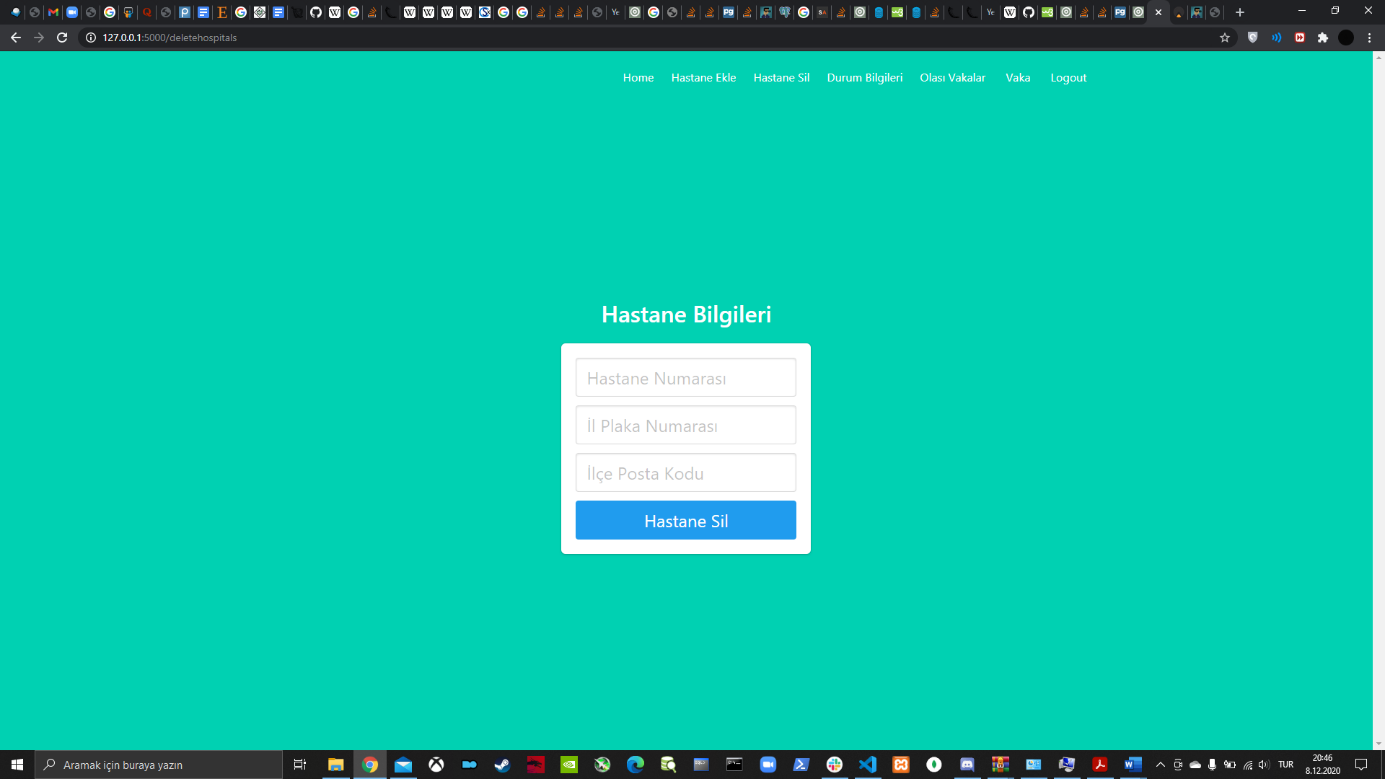
//If we cannot find a city with given license plate number, we create a new city with given license plate number and name.

insert into Ilce values(ilplakano, ilcepostakodu, ilceismi, 0,0,0)

//If we cannot find a county with given license plate number, we create a new county with given license plate number, name and zipcode.

insert into Hastane values(hastaneno, ilplakano, ilcepostakodu, isim, 0,0,0)

//If we cannot find a hospital with given number, city license plate number, county zipcode and name, we create a new county with given number, city license plate number, county zipcode and name.

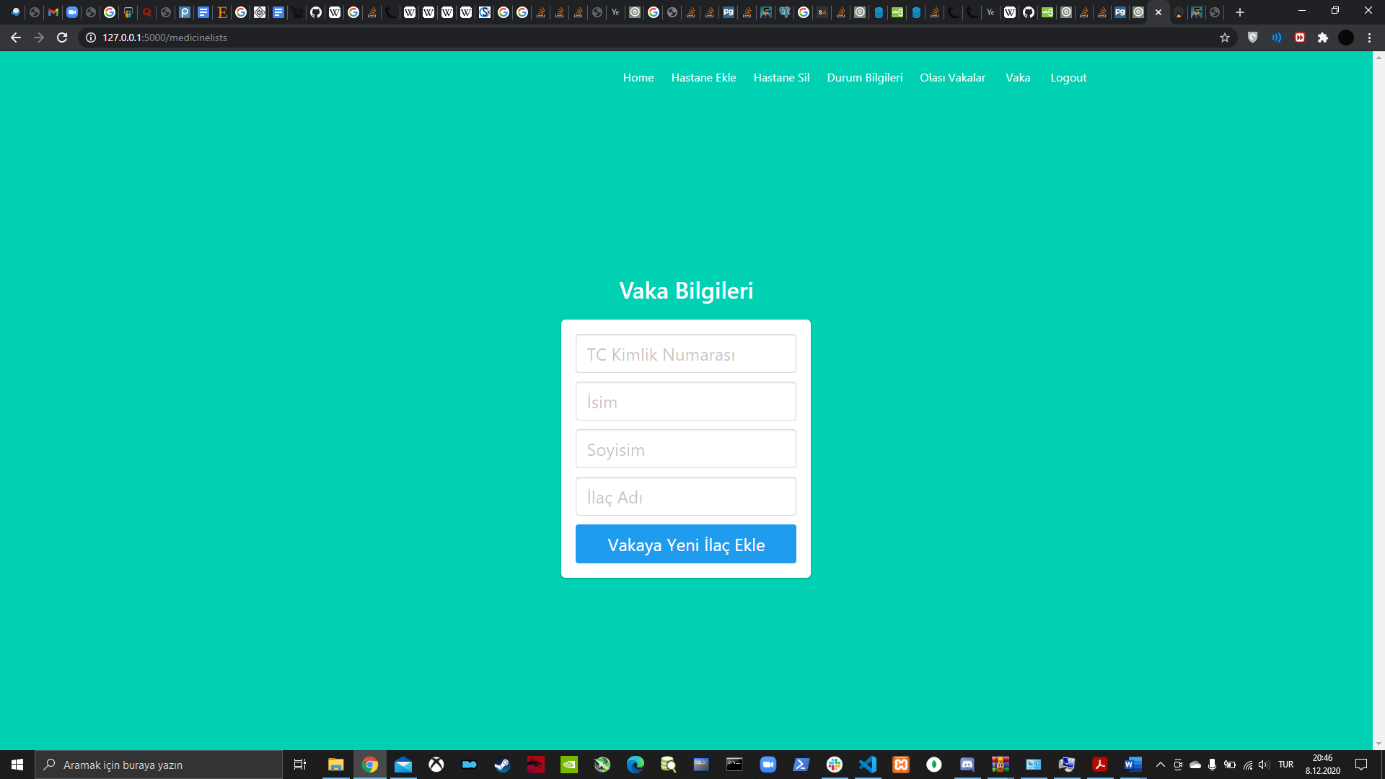


This form handles the deletion of a hospital from the database. In it, the user is asked to enter a hospital ID number ‘hastaneno’, the city license plate number that the hospital belongs to ‘ilplakano’ and the zipcode of the county the hospital belongs to ‘ilcepostakodu’.

When the blue ‘Hastane Sil’ button is clicked, the data in the forms gets used in the following SQL statements:

delete from Hastane where hastaneno= %s and plakano= %s and postakodu= %s (hastaneno, ilplakano, ilcepostakodu)

//Used to delete the hospital with given hospital ID number, city license plate number and county zipcode.



This form handles the update of a patient’s medicine list. In it, the user is asked to enter the patient’s ID number ‘tckn’, their first name ‘ad’ and their last name ‘soyad’.

When the blue ‘Vakaya Yeni İlaç Ekle’ button is clicked and if such a patient exists, the data in the forms gets used in the following SQL statements:

vaka = select \* from Vakalar where tckn=%s(tckn)

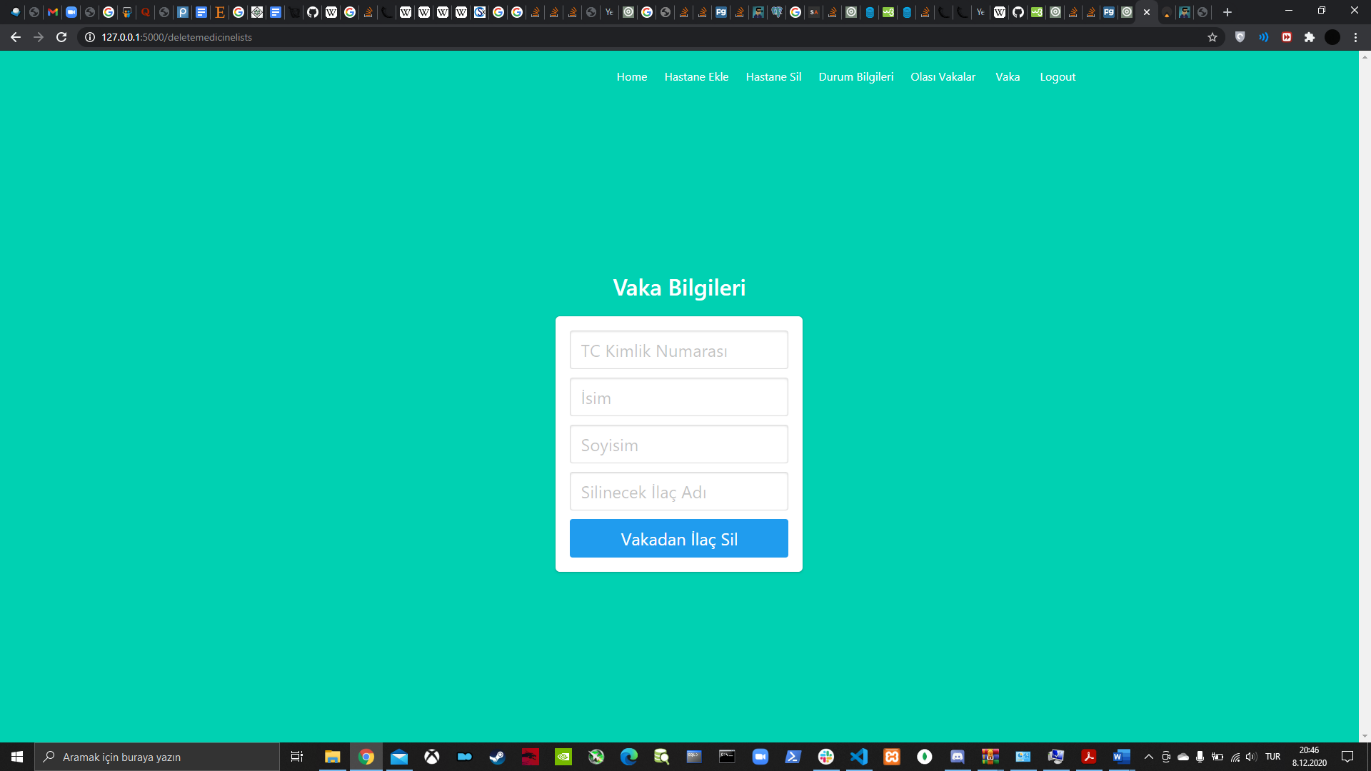
//Used to determine patient with ID number tckn.

olasivaka = select \* from Olasivakalar where tckn=%s(‘tckn’)

//Used to determine possible patient with ID number tckn.

update Vakalar set ilaclistesi=newilaclistesi where tckn=%s(‘tckn’)

//Used to update the medicine list of the patient with ID number tckn. Here, newilaclistesi is the old medicine list with the new medicine appended.



This form handles the deletion of a certain medicine from a patient’s medicine list. In it, the user is asked to enter the patient’s ID number ‘tckn’, their first name ‘ad’, their last name ‘soyad’ and the name of the medicine to be deleted from the aforementioned medicine list ‘ilacismi’.

When the blue ‘Vakadan İlaç Sil’ button is clicked and if such a patient exists that has been prescribed the medicine with the given name, the data in the forms gets used in the following SQL statements:

vaka = select \* from Vakalar where tckn=%s(tckn)

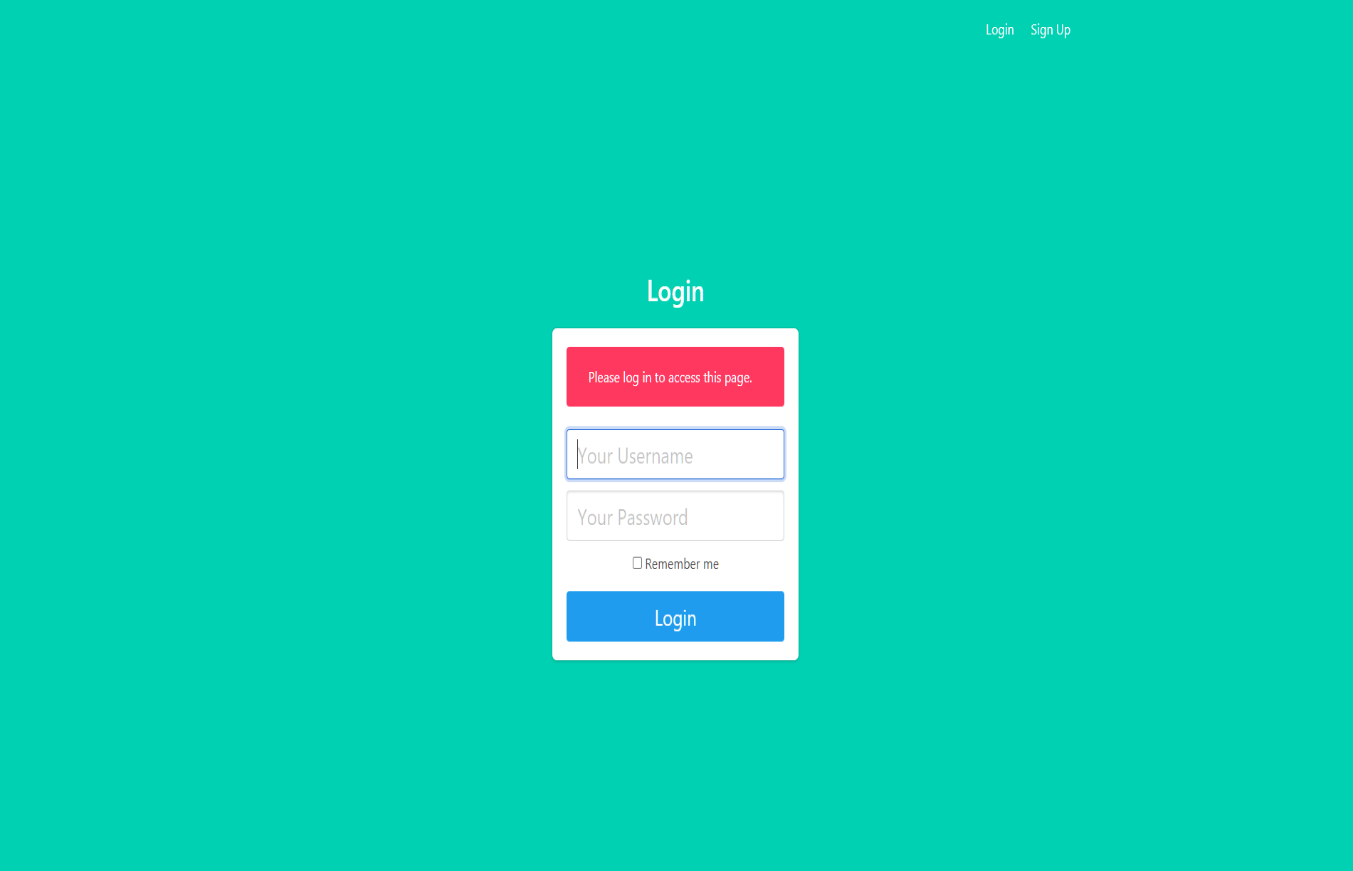
//Used to determine patient with ID number tckn.

olasivaka = select \* from Olasivakalar where tckn=%s(‘tckn’)

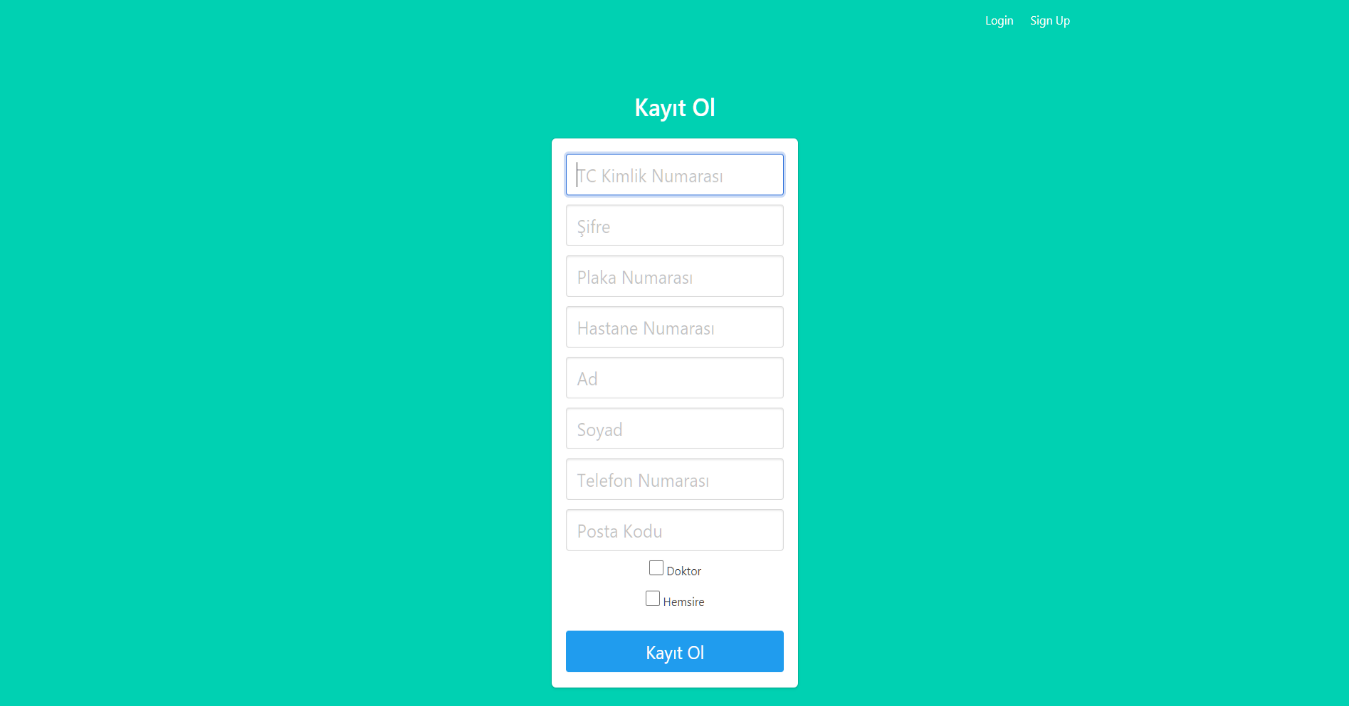
//Used to determine possible patient with ID number tckn.

update Vakalar set ilaclistesi=newilaçlistesi where tckn=%s(‘tckn’)

//Used to update the medicine list of the patient with ID number tckn. Here, newilaclistesi is the old medicine list with the given medicine removed.



Login

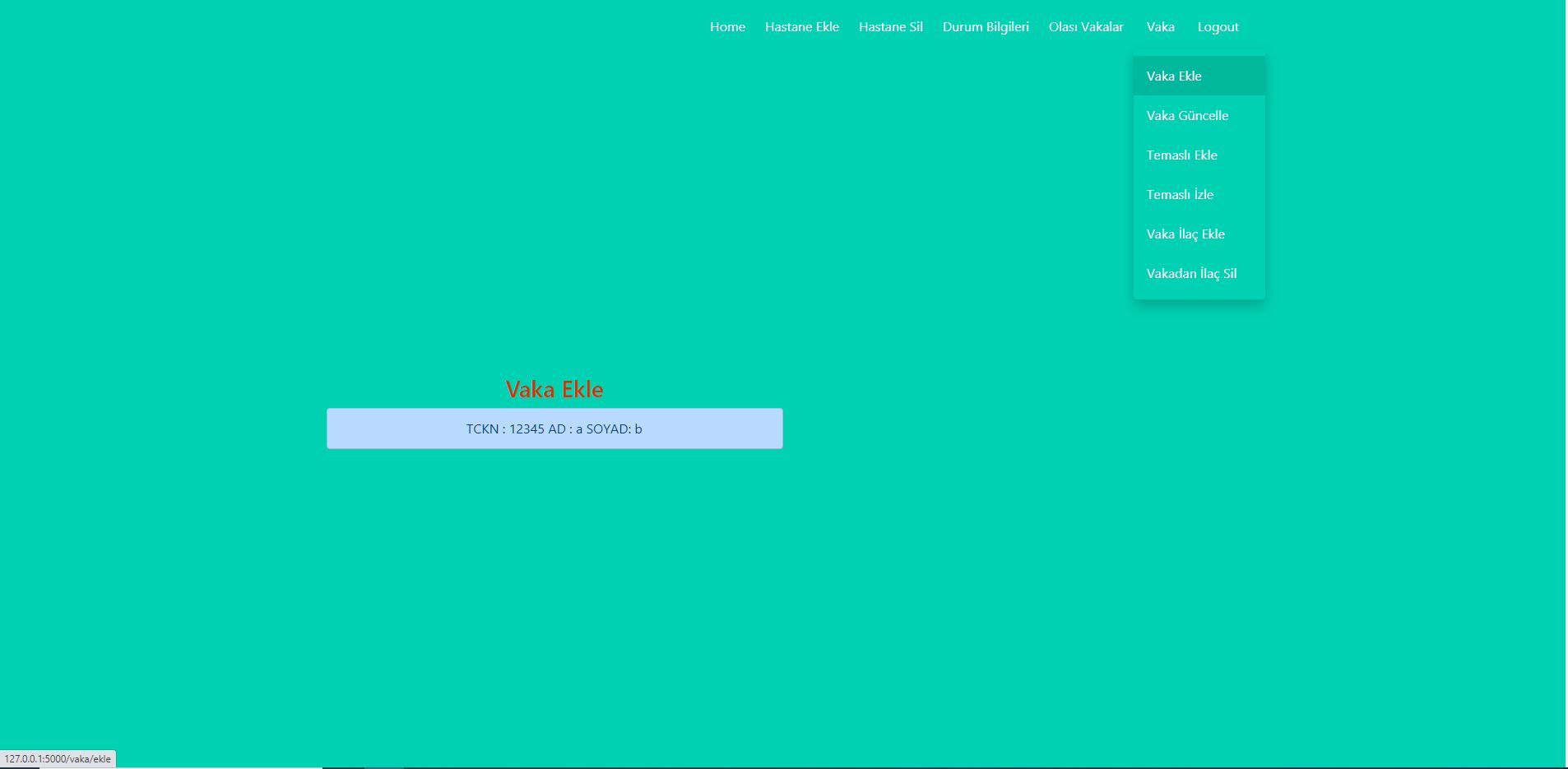
The website is start with login page if you dont login. Username is same with person’s TC number.

Register

Add doctor or nurse.

"INSERT INTO Çalışan(tckn,password,plakano,hastaneno,ad,soyad,telno,plakakodu,tur) values(%s,%s,%s,%s,%s,%s,%s,%s,%s)",

(tckn,password,plakano,hastaneno,ad,soyad,telno,pkodu,tur)



Vaka -> Vaka Ekle

People with test status 1 (positive) among the olasıvarlıklar come to the case adding screen.

When the person whose information is written is printed on, the person whose test is positive is added to the vakalar.

SQL statement for above sentence

"INSERT INTO Vakalar(tckn,durum,ilaclistesi) values(%s,%s,%s)", (olasıvaka.tckn,0,None)

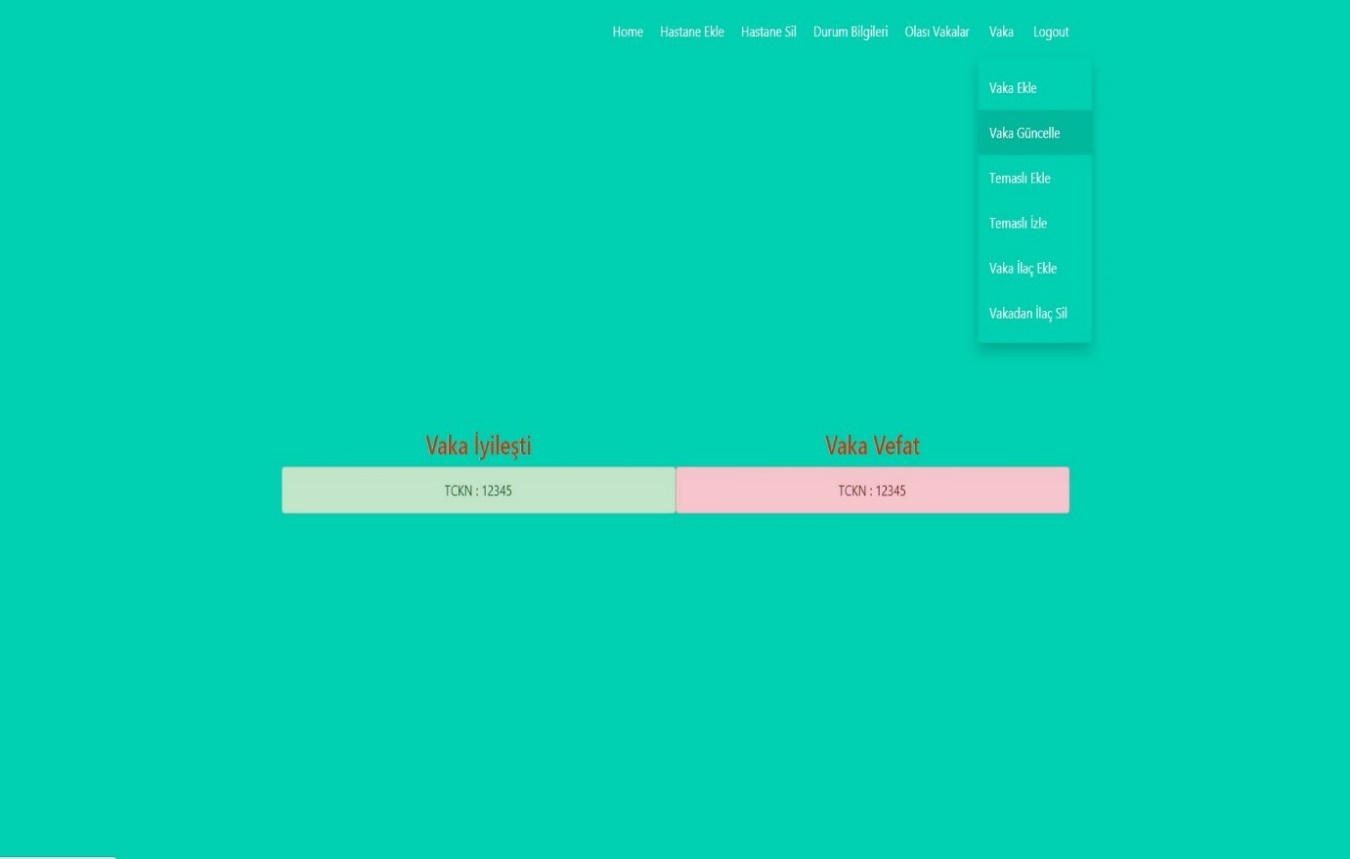
The number of patients is increased for the added case in il, ilçe and hastane tables.

SQL 3 update stataments for above sentence

"UPDATE il SET hastasayisi=%s WHERE plakano= %s" , (il.hastasayisi+1,calisanlar.plakano)

" UPDATE ilce SET hastasayisi=%s WHERE postakodu= %s and plakano= %s ", (ilce.hastasayisi+1,calisanlar.postakodu,calisanlar.plakano)

"UPDATE Hastane SET hastasayisi=%s WHERE hastaneno= %s and postakodu= %s and plakano= %s ", (hastane.hastasayisi+1,calisanlar.username,calisanlar.postakodu,calisanlar.plakano)



Vaka -> Vaka Güncelle

Case can be updated as healed or dead. Changes are made for the selected situation.

If it is healed, the case is deleted from the vakalar table.

SQL statement for above sentence

"DELETE FROM vakalar WHERE tckn = %s ", (vaka)

if it is dead, the durum is -1 for this person. The number of deaths increases by one in the il, ilçe and hastane

SQL 4 update stataments for above sentences

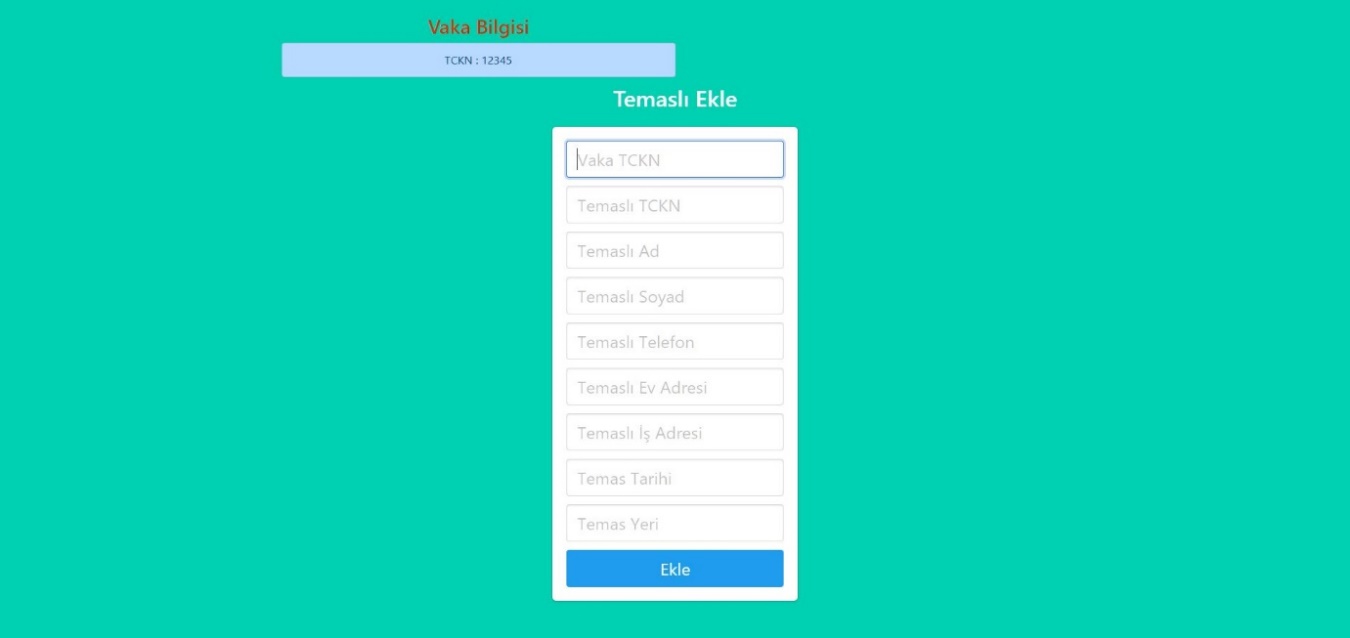
"UPDATE vakalar SET durum = -1 WHERE tckn = %s " , (vaka.tckn)

"UPDATE il SET olusayisi= %s WHERE plakano= %s", (il.olusayisi+1,calisanlar.plakano)

"UPDATE ilce SET olusayisi= %s WHERE postakodu= %s and plakano= %s ", (ilce.olusayisi+1,calisanlar.postakodu,calisanlar.plakano)

"UPDATE hastane SET hastasayisi=%s WHERE hastaneno = %s and postakodu= %s , plakano= %s", (hastaneçhastasayisi+1,calisanlar.hastaneno,calisanlar.postakodu,calisanlar.plakano)

In the page, the number of patients is decreased for the added case in il, ilçe and hastane tables. The sql stataments are same with Vaka Ekle page.



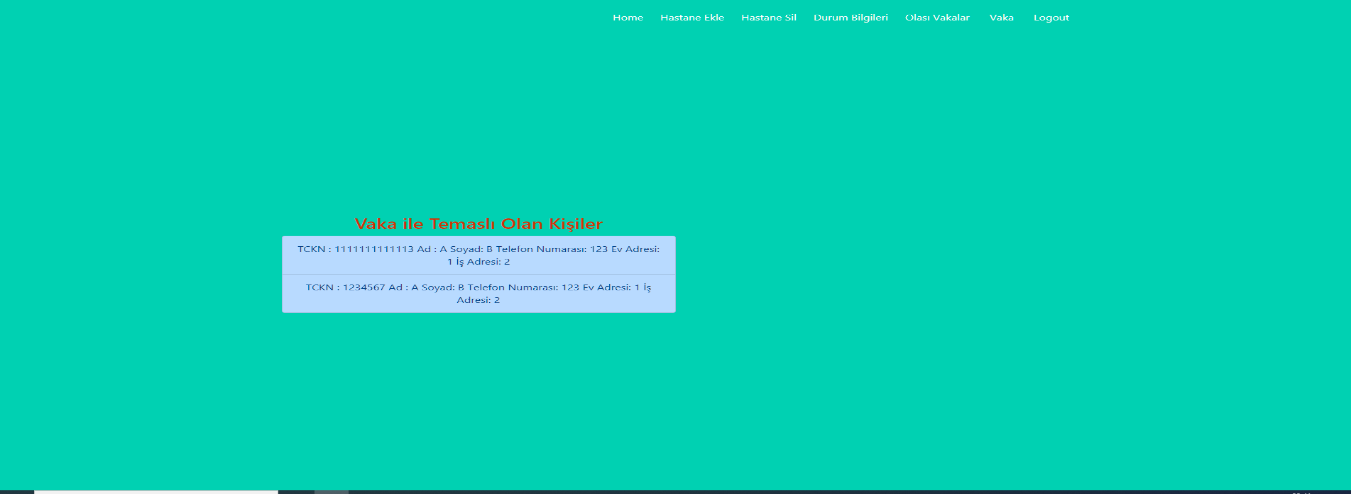
Vaka-> Temaslı Ekle

The patients information is in the table. All labels must be filled. The date format should be YYYY-MM-DD

SQL Statements for the form in above senctence

"INSERT INTO temaslılar(temaslıtckn,ad,soyad,telno,evadresi,isadresi) Values( %s,%s,%s,%s,%s) ",( ttckn,ad,soyad,telno,ev,isy)

"INSERT INTO temas(temaslitckn,tckn,temasyeri,temastarihi,temasliisim,temaslisoyisim,) Values( %s,%s,%s,%s,%s,%s) ",(ttckn,tckn,yer,ttarih,ad,soyad)

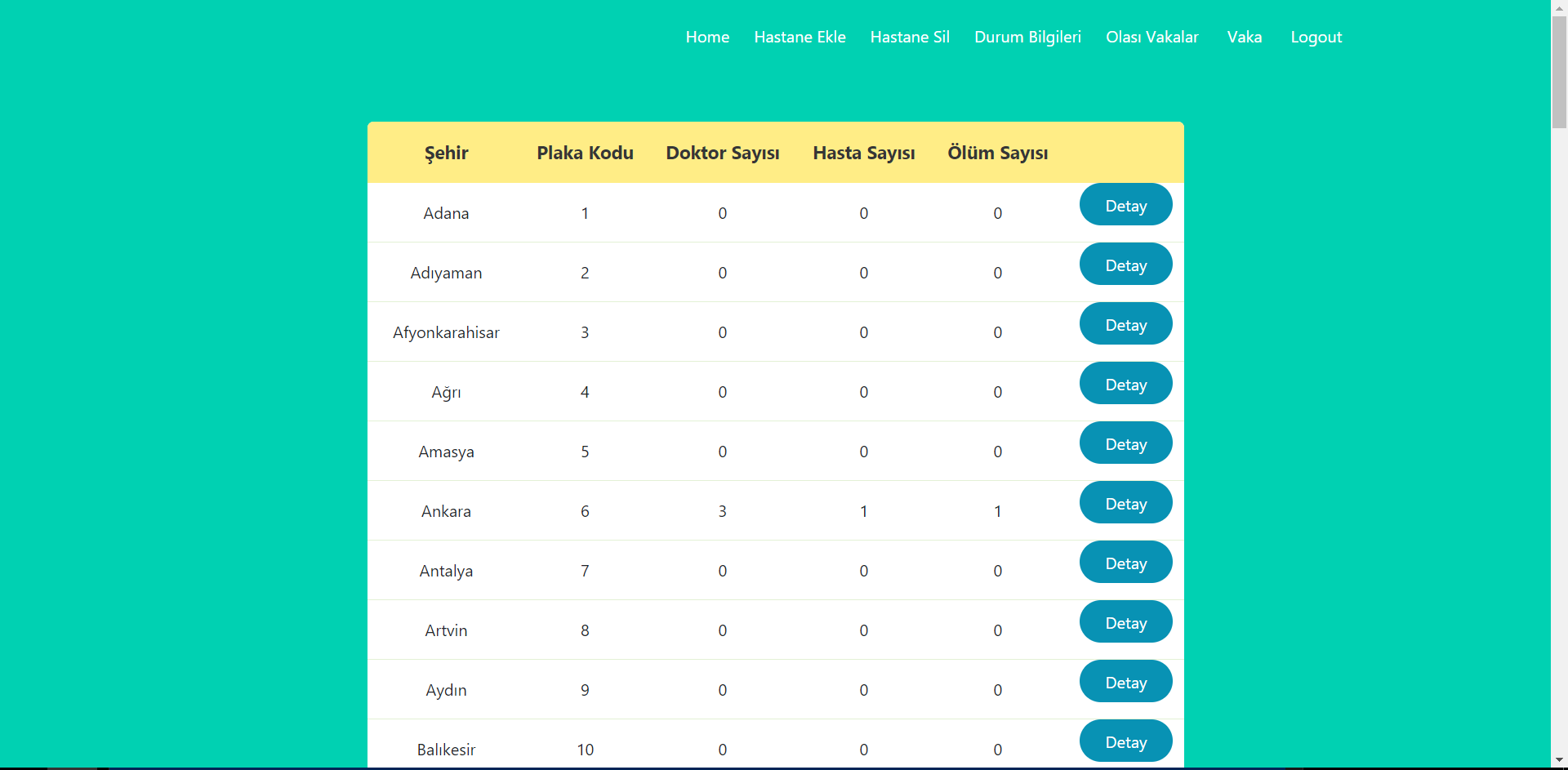


Vaka -> Temaslı İzle

This page is for contact patients information.

SQL Statement

"Select \* FROM temaslılar "

Cities

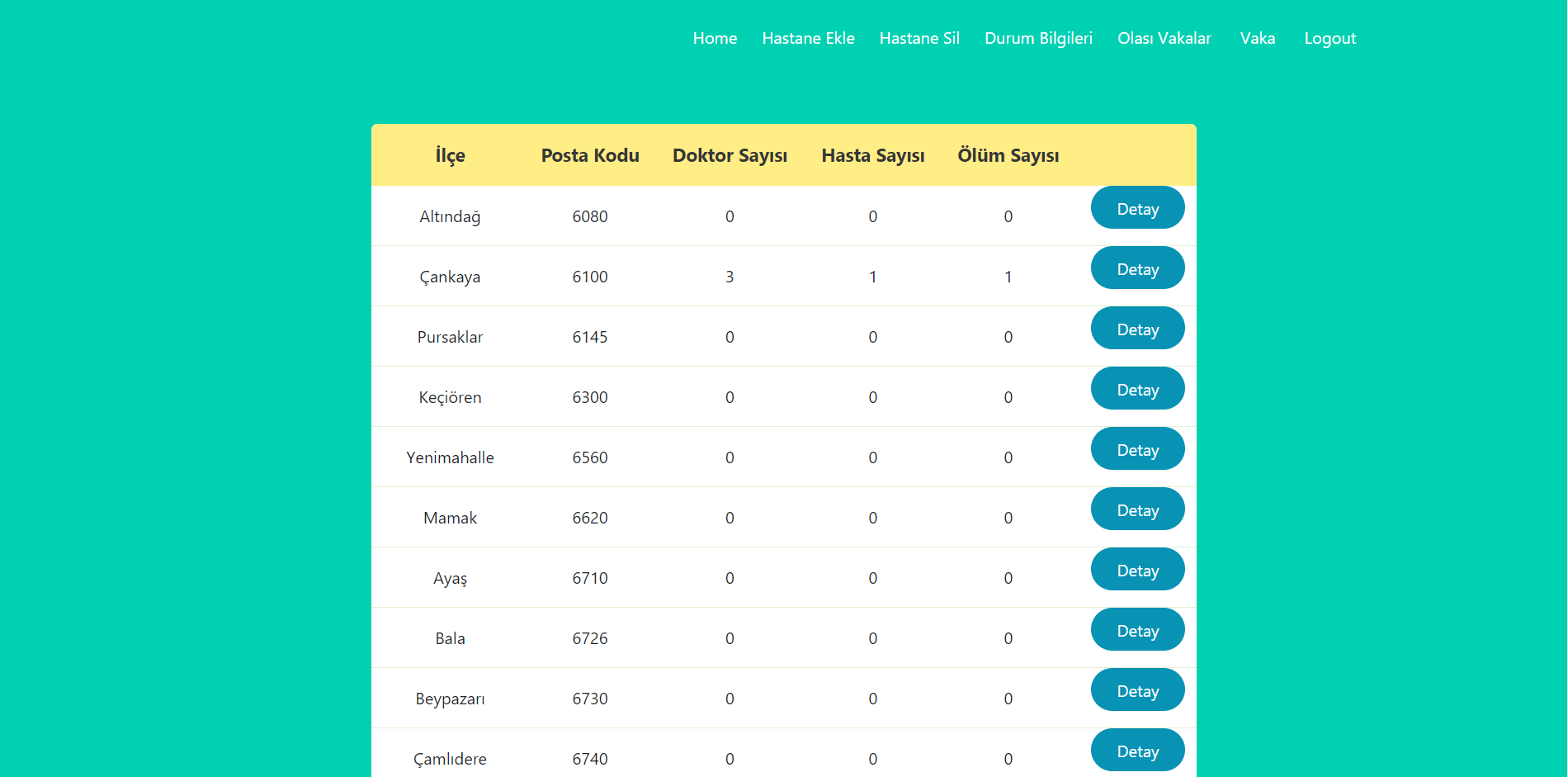
On the tables page, the information of each city in general is written in the web interface by pulling the province table in the database.

This information is in the form of city name, plate code, total number of doctors in the city, the total number of patients in the city and the total number of deaths in that city.

With the detail button next to each city, the license plate code of the selected city is sent to the server and is directed to the tables2 page that provides information about the districts of that city.

select \* from il order by plakano

Provinces



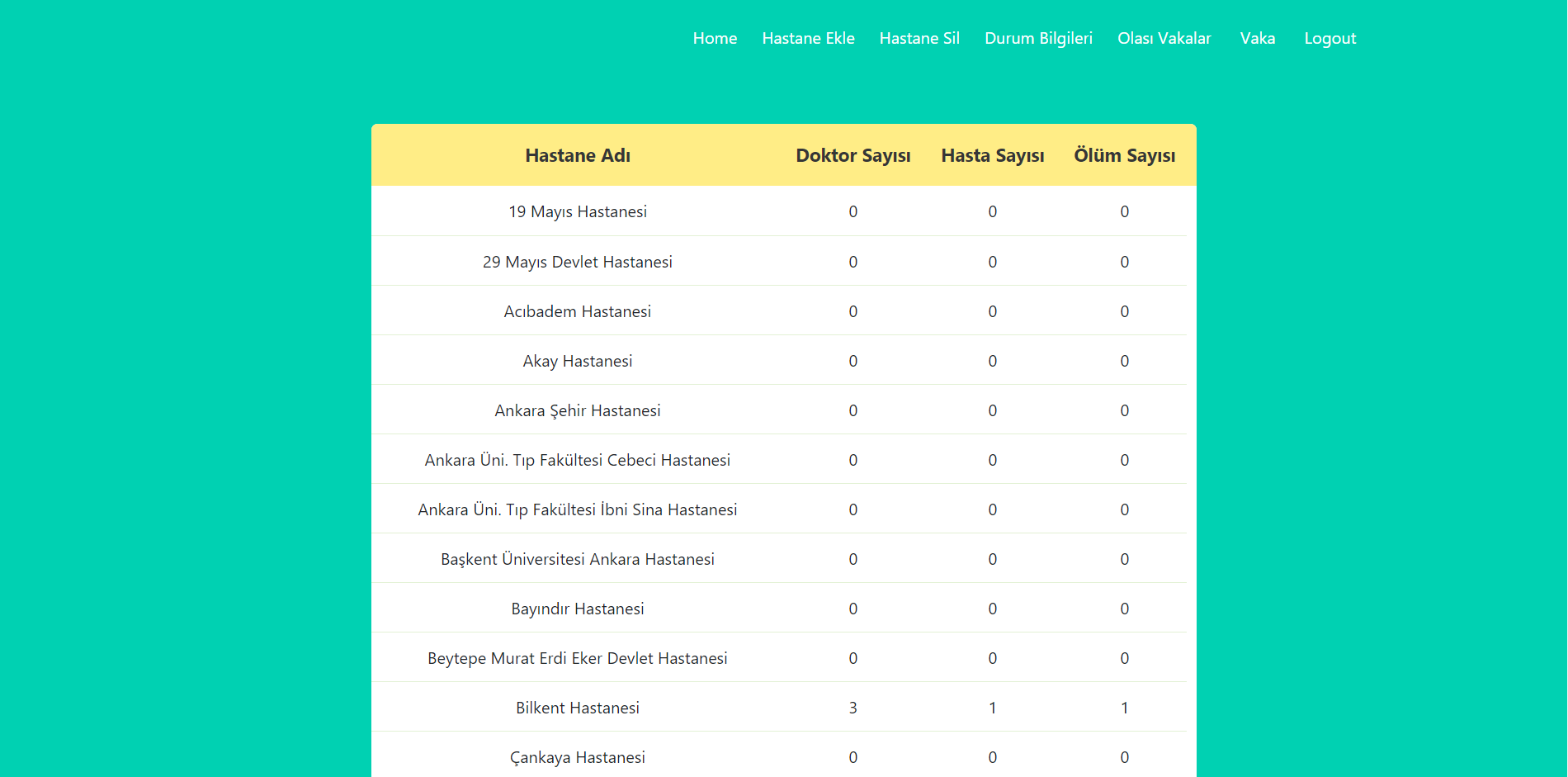
On the tables2 page,detailed information of the districts of the selected city is presented in a table by selecting the districts with the same license plate code and the plate code given as an attribute are presented in a table.

This information includes the name of the district, the postal code of the district, the number of doctors in the district, the total number of patients in the district and the total number of deaths in the district.

With the detail button next to each district presented, the zip code of the district in the selected row is sent to the server and directs it to the tables3 page, which gives detailed information about the hospitals in that district.

select \* from ilçe where plakano=%s order by postakodu (request.args.get('plaka\_no'))

Hospitals

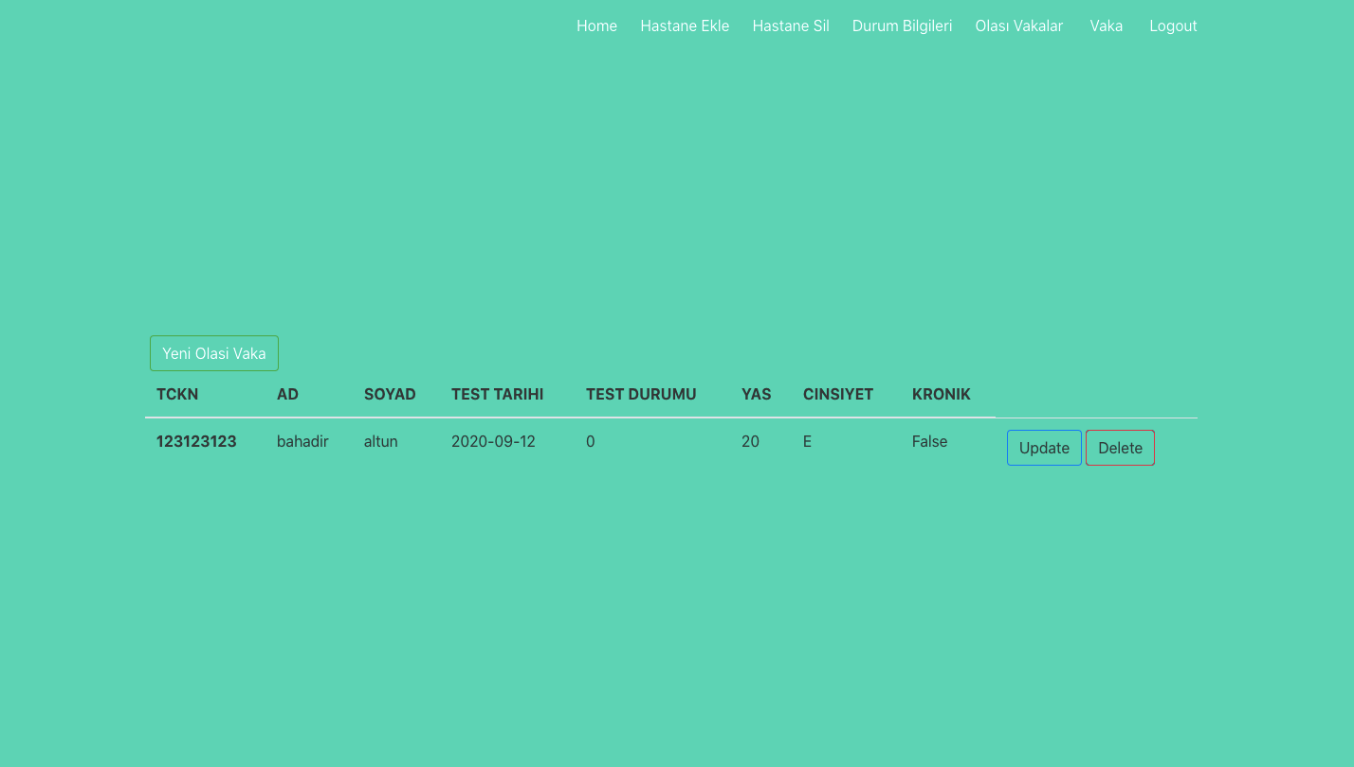


On the tables3 page, detailed information of the hospitals in the selected district is presented in a table by selecting the hospitals with the same zip code and the zip code given as an attribute are presented in a table.

This information includes the name of the hospital, the number of doctors working in the hospital, the number of patients in the hospital and the total number of deaths in the hospital.

select \* from hastane where postakodu=%s order by isim (request.args.get('posta\_kodu'))

Olasi Vakalar:



Olasi Vakalar indicates the possible cases for COVID-19. Each case has TEST DURUMU where it is set to 0 initially. Doctors can change them to 1 in case when patient has positive test results.

**Doctors** can edit rows (CRUD) and corresponding sql scripts given below:

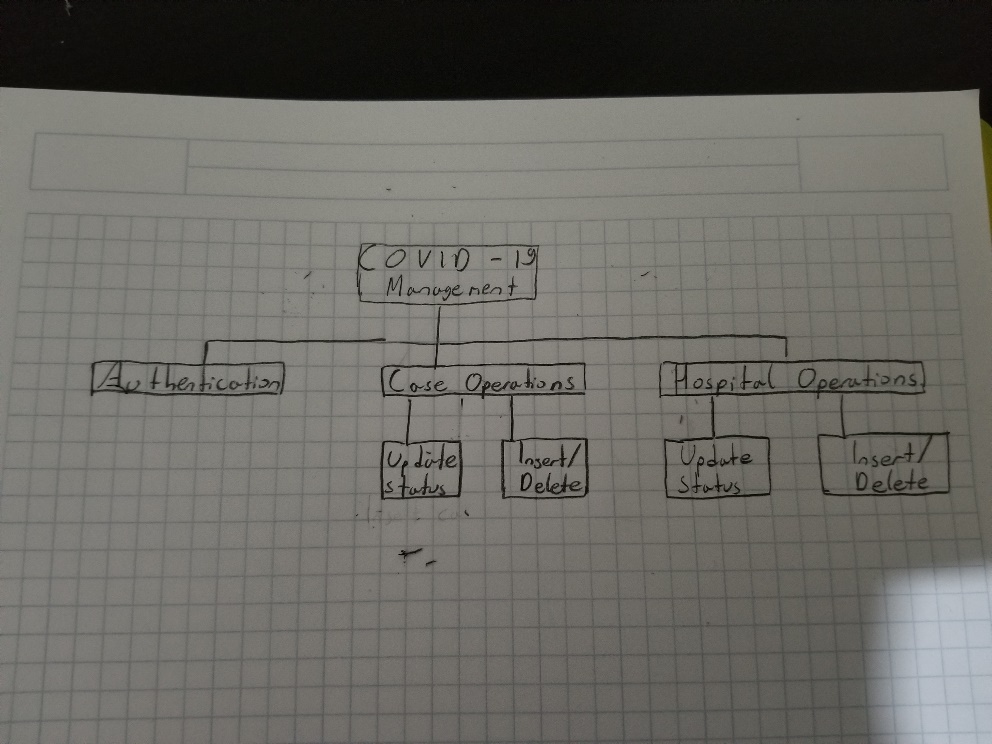
select \* from olasivakalar where testdurumu = 0;

**For doctors only:**

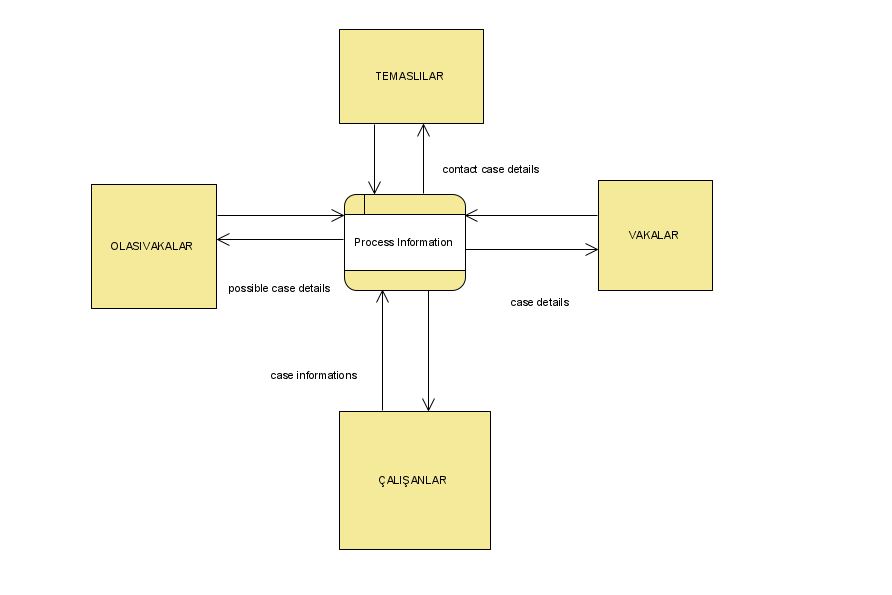
insert olasivakalar(tckn, ad, soyad, testtarihi, testdurumu, yas, cinsiyet, kronikhastalik) into values (\_tckn, \_ad, \_soyad, \_testtarihi, \_testdurumu, \_yas, \_cinsiyet, \_kronikhastalik);

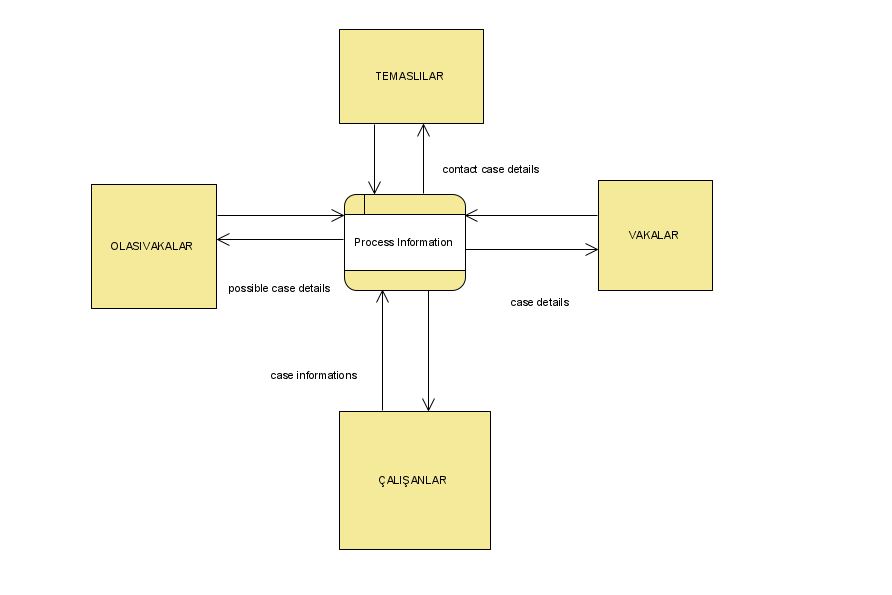
update olasivakalar set (ad, soyad, testtarihi, testdurumu, yas, cinsiyet, kronikhastalik) = (\_ad, \_soyad, \_testtarihi, \_testdurumu, \_yas, \_cinsiyet, \_kronikhastalik) where tckn = \_tckn;

*2.2 HIPO*

**

*2.3 DFD*

**

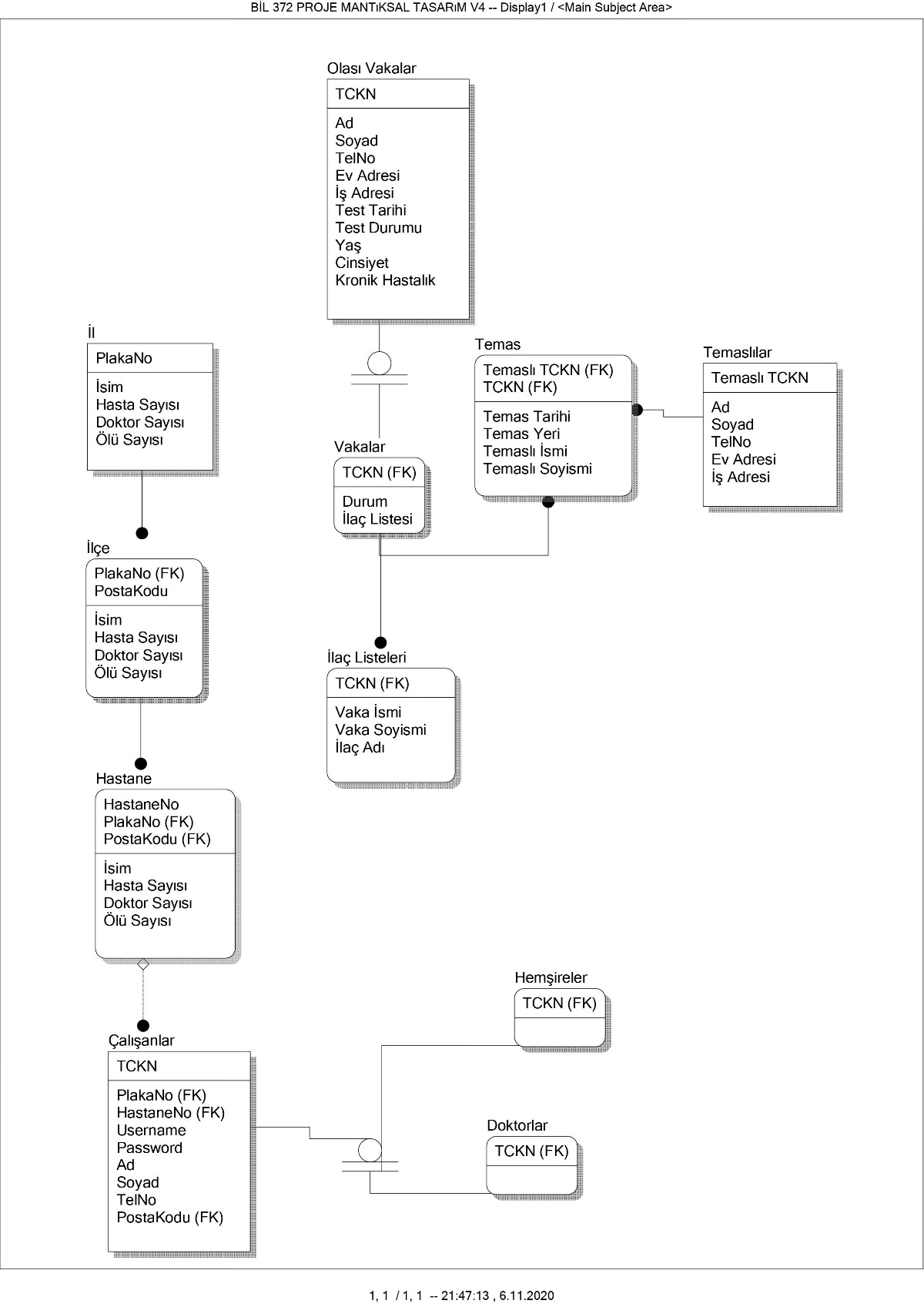


References

<https://flask.palletsprojects.com/en/1.1.x/>

<https://docs.sqlalchemy.org/en/13/>

https://www.postgresql.org/docs/

1. Database Schema
2. ER Schema

