Database Systems Project Final Report

Patient Medical Treatment Tracking System

03.01.2019

Project URL: https://segocago.github.io/CS353_Database_Project/

Project Group No: 6

Burak Erkılıç - 21501035

Çağatay Sel - 21502938

Kaan Kıranbay - 21501103

Mert Saraç - 21401480

Course Teacher: Shervin R. Arashloo - Course TA: Arif Usta

Table of Contents

Project Description

Our project is a system for tracking medical related information for patients, doctors, and pharmacists. This tracking system allows a doctor or a patient or a pharmacist to log and monitor the medical related information and request various medical activities. This system is part of an overall information system and it interacts with the person's electronic health record, where information specific to the person is stored.

A patient can

A doctor can

A pharmacist can add or remove other pharmacist from the pharmacy that he works, can add or remove drugs and their amounts to his pharmacy, and search for a specific amount of drugs. He can add a new drug or a new vaccine to the system or can add an alternative drug for a drug to the system.

Our project's website: https://segocago.github.io/CS353_Database_Project/

1. Final E/R Model

We revised our E/R diagram according to the feedback that we got from our teaching assistant and made necessary changes:

1. We added a relation between patient, drug, and pharmacy. This will help patients to buy drugs from a pharmacy.
2. We changed examination_result to one to one relation.
3. We changed alternative_to to many to many relation.
4. We changed stores to many to many relation.
5. We deleted hospital_executive_doctor_id from hospital and made it a relation attribute of works_as_doctor..
6. We deleted date attributes from test, treatment, and prescription entities because they all came from a examination date.

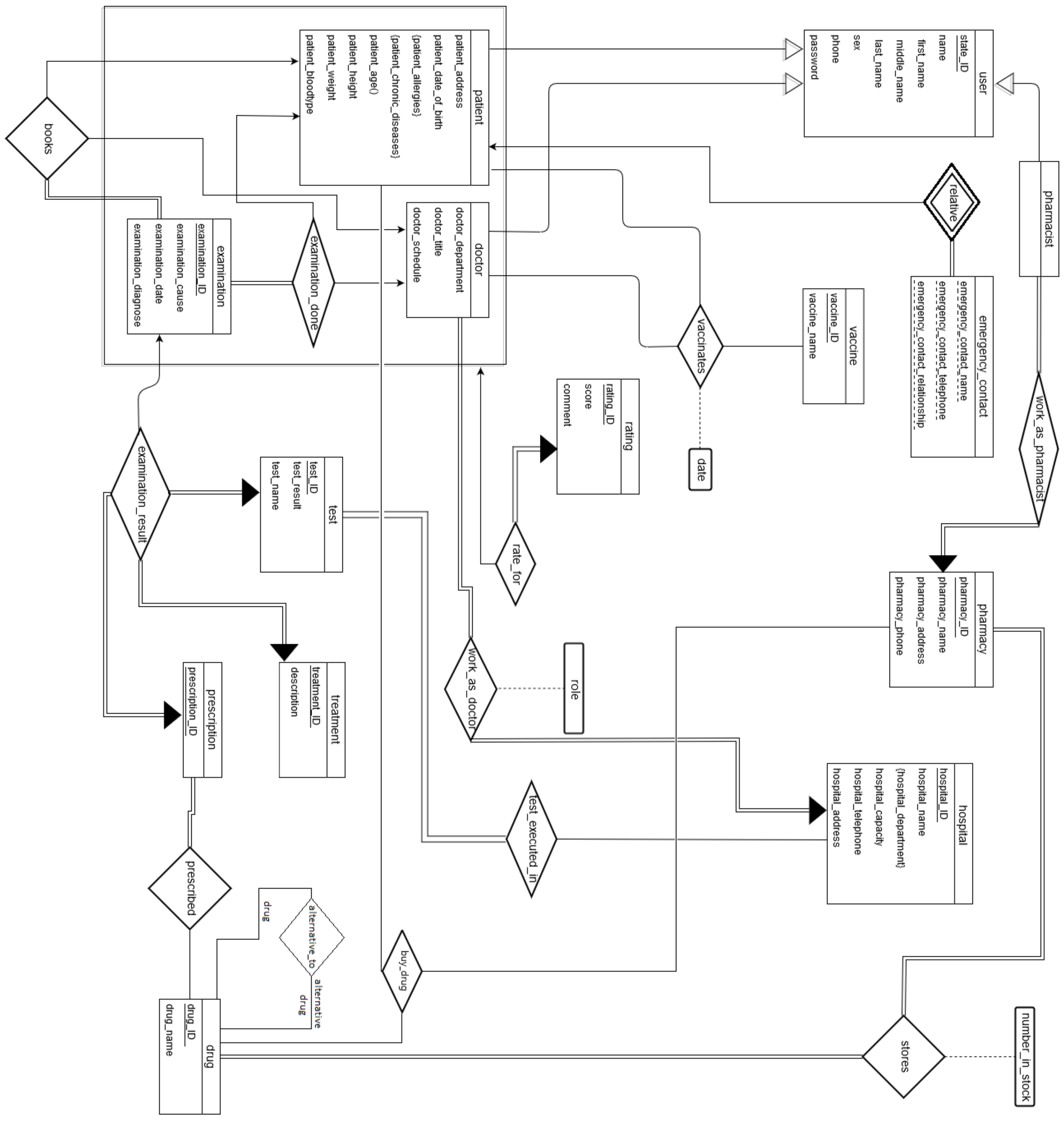


Figure 1: Final E/R Model of Project’s Database

2. Relation Schemas

2.1 User

Relational Model:

user(<u>state_ID</u>, first_name, middle_name, last_name, sex, phone, password)

Functional Dependencies:

state_ID → first_name, middle_name, last_name, sex, phone, password

Candidate Keys:

{ (state_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE user(
        state_ID        char(11) PRIMARY KEY,
        first_name      varchar(20),
        middle_name     varchar(20),
        last_name       varchar(20),
        sex             varchar(20),
        phone           varchar(100),
        password        varchar(40) NOT NULL);
```

2.2 Pharmacist

Relational Model:

pharmacist (<u>state_ID</u>)

Functional Dependencies:

none

Candidate Keys:

{ (state_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE pharmacist(
        state_ID        char(11) PRIMARY KEY,
        FOREIGN KEY (state_ID) references user);
```

2.3 Patient

Relational Model:

patient (state_ID, patient_adress, patient_date_of_birth, patient_allergies, patient_chronic_diseases, patient_height, patient_weight, patient_bloodtype)

Functional Dependencies:

state_ID → patient_adress, patient_date_of_birth, patient_allergies, patient_chronic_diseases, patient_height, patient_weight, patient_bloodtype

Candidate Keys:

{ (state_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE patient(
        state_ID                    char(11) PRIMARY KEY,
        patient_adress              varchar(100),
        patient_date_of_birth       date NOT NULL,
        patient_allergies           varchar(100),
        patient_chronic_diseases    varchar(100),
        patient_height              numeric(3,2),
        patient_weight              numeric(3,2),
        patient_bloodtype           varchar(20),
        FOREIGN KEY (state_ID) references user);
```

2.4 Doctor

Relational Model:

doctor (<u>state_ID</u> , doctor_department, doctor_title, doctor_schedule)

Functional Dependencies:

state_ID → doctor_department, doctor_title, doctor_schedule

Candidate Keys:

{ (state_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE doctor(
        state_ID                char(11) PRIMARY KEY,
        doctor_department       varchar(40) NOT NULL,
        doctor_title            varchar(40) NOT NULL,
        doctor_schedule         varchar(400) NOT NULL,
        FOREIGN KEY (state_ID) references user);
```

2.5 Examination

Relational Model:

examination (examination_ID, examination_cause, examination_date, examination_diagnose)

Functional Dependencies:

examination_ID→ examination_cause, examination_date, examination_diagnose

Candidate Keys:

{ (examination_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE examination(
        examination_ID          int PRIMARY KEY AUTO_INCREMENT,
        patient_state_ID        char(11),
        doctor_state_ID         char(11),
        examination_cause       varchar(400) NOT NULL,
        examination_date        timestamp NOT NULL,
        examination_diagnose    varchar(400) NOT NULL);
```

2.6 Rating

Relational Model:

rating (rating_ID, score, comment)

Functional Dependencies:

rating_ID → score, comment

Candidate Keys:

{ (rating_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE rating(
        rating_ID       int PRIMARY KEY AUTO_INCREMENT,
        score           int,
        comment         varchar(400),
        check (score between 0 and 5));
```

2.7 Test

Relational Model:

test(test_ID,, test_result, test_name)

Functional Dependencies:

test_ID → test_result, test_name

Candidate Keys:

{ (test_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE test(
        test_ID         int PRIMARY KEY AUTO_INCREMENT,
        test_result     varchar(400),
        test_name       varchar(100));
```

2.8 Treatment

Relational Model:

treatment (<u>treatment_ID</u>, treatment_description)

Functional Dependencies:

treatment_ID → treatment_description

Candidate Keys:

{ (treatment_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE treatment(
        treatment_ID                int PRIMARY KEY AUTO_INCREMENT,
        treatment_description       varchar(400));
```

2.9 Prescription

Relational Model:

prescription (prescription_ID)

Functional Dependencies:

none

Candidate Keys:

{ (prescription_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE prescription(
        prescription_ID        int PRIMARY KEY AUTO_INCREMENT);
```

2.10 Drug

Relational Model:

drug(drug_ID, drug_name)

Functional Dependencies:

drug_ID → drug_name

Candidate Keys:

{ (drug_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE drug(
        drug_ID         int PRIMARY KEY AUTO_INCREMENT,
        drug_name       varchar(200));
```

2.11 Pharmacy

Relational Model:

pharmacy (pharmacy_ID, pharmacy_name, pharmacy_address, pharmacy_phone)

Functional Dependencies:

pharmacy_ID → pharmacy_name, pharmacy_address, pharmacy_phone

Candidate Keys:

{ (pharmacy_ID**) }**

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE pharmacy(
        pharmacy_ID          int PRIMARY KEY AUTO_INCREMENT,
        pharmacy_name        varchar(100),
        pharmacy_address     varchar(100),
        pharmacy_phone       varchar(100));
```

2.12 Hospital

Relational Model:

hospital(hospital_ID, hospital_name, hospital_capacity, hospital_telephone, hospital_address, hospital_executive_doctor_id)

Functional Dependencies:

hospital_ID → hospital_name, hospital_capacity, hospital_telephone, hospital_address, hospital_executive_doctor_id

Candidate Keys:

{ (hospital_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE hospital(
        hospital_ID                 int PRIMARY KEY AUTO_INCREMENT,
        hospital_name               varchar(200),
        hospital_capacity           int,
        hospital_telephone          varchar(100),
        hospital_address            varchar(200));
```

2.13 Vaccine

Relational Model:

vaccine(<u>vaccine_ID</u>, vaccine_name)

Functional Dependencies:

vaccine_ID → vaccine_name

Candidate Keys:

{ (vaccine_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE vaccine(
        vaccine_ID          int PRIMARY KEY AUTO_INCREMENT,
        vaccine_name        varchar(100));
```

2.14 Emergency Contact

Relational Model:

emergency_contact (state_ID, emergency_contact_name, emergency_contact_telephone, emergency_contact_relationship)

Functional Dependencies:

state_ID, emergency_contact_name, emergency_contact_telephone, emergency_contact_ relationship → state_ID, emergency_contact_name, emergency_contact_telephone, emergency_contact_ relationship

Candidate Keys:

{ (state_ID, emergency_contact_name, emergency_contact_telephone, emergency_contact_relationship) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE emergency_contact(
        state_ID                            char(11),
        emergency_contact_name              varchar(100),
        emergency_contact_telephone         varchar(100),
        emergency_contact_relationship      varchar(100),
        PRIMARY KEY (state_ID, emergency_contact_name, emergency_contact_telephone,
emergency_contact_relationship),
        FOREIGN KEY (state_ID) references patient);
```

2.15 Hospital Departments

Relational Model:

hospitalDepartment (hospital_ID, hospital_department)

Functional Dependencies:

None

Candidate Keys:

{ (hospital_ID, hospital_department)}

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE hospitalDepartment(
        hospital_ID             int,
        hospital_department     varchar(40),
        PRIMARY KEY (hospital_ID, hospital_department),
        FOREIGN KEY (hospital_ID) references hospital);
```

2.16 Patient Allergies

Relational Model:

patientAllergies (state_ID, allergy_name)

Functional Dependencies:

state_ID, allergy_name→state_ID, allergy_name

Candidate Keys:

{ (state_ID,allergy_name) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE patientAllergies(
        state_ID        char(11),
        allergy_name    varchar(100),
        PRIMARY KEY (state_ID,allergy_name),
        FOREIGN KEY (state_ID) references patient);
```

2.17 Patient Chronic Diseases

Relational Model:

patientChronicDiseases (state_ID, chronic_disease)

Functional Dependencies:

state_ID, chronic_disease→ state_ID, chronic_disease

Candidate Keys:

{ (state_ID, chronic_disease) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE patientChronicDiseases(
        state_ID                char(11),
        chronic_disease         varchar(100),
        PRIMARY KEY (state_ID,chronic_disease),
        FOREIGN KEY (state_ID) references patient);
```

2.18 Examination Done

Relational Model:

examinationDone (patient_state_ID, doctor_state_ID, examination_ID)

Functional Dependencies:

No non-trivial functional dependency.

Candidate Keys:

{ (patient_state_ID, doctor_state_ID, examination_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE examinationDone (
        patient_state_ID        char(11),
        doctor_state_ID         char(11),
        examination_ID          char(11),
        PRIMARY KEY (examination_ID),
        FOREIGN KEY (patient_state_ID) references patient(state_ID),
        FOREIGN KEY (doctor_state_ID) references doctor(state_ID));
```

2.19 Books

Relational Model:

books (<u>state_ID</u>, examination_ID, doctor_ID)

Functional Dependencies:

none

Candidate Keys:

{ (state_ID, examination_ID,doctor_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE books (
        state_ID            char(11),
        examination_ID      char(11)
        doctor_ID           char(11),
        PRIMARY KEY (state_ID),
        FOREIGN KEY (state_ID) references patient,
        FOREIGN KEY (examination_ID) references examination);
```

2.20 Vaccinates

Relational Model:

vaccinate (<u>vaccine_ID</u>, <u>patient_state_ID</u>, <u>doctor_state_ID</u>, date)

Functional Dependencies:

vaccine_ID, patient_state_ID, doctor_state_ID → date

Candidate Keys:

{ (vaccine_ID, patient_state_ID, doctor_state_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE vaccinates(
        vaccine_ID              int,
        patient_state_ID        char(11),
        doctor_state_ID         char(11),
        date                    date,
        PRIMARY KEY (vaccine_ID, patient_state_ID, doctor_state_ID),
        FOREIGN KEY (vaccine_ID) references vaccine,
        FOREIGN KEY (patient_state_ID) references patient(state_ID),
        FOREIGN KEY (doctor_state_ID) references doctor(state_ID));
```

2.21 Works as Pharmacist

Relational Model:

worksAsPharmacist (<u>state_ID</u>, pharmacy_ID)

Functional Dependencies:

none

Candidate Keys:

{ (state_ID, pharmacy_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE worksAsPharmacist(
        state_ID            char(11),
        pharmacy_ID         int,
        PRIMARY KEY (state_ID),
        FOREIGN KEY (state_ID) references pharmacist,
        FOREIGN KEY (pharmacy_ID) references pharmacy);
```

2.22 Rate for

Relational Model:

rate_for (<u>rating_ID</u>, patient_state_ID, doctor_state_ID, examination_ID)

Functional Dependencies:

none

Candidate Keys:

{ (rating_ID, patient_state_ID, doctor_state_ID, examination_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE rateExamination(
        rating_ID               int,
        patient_state_ID        char(11),
        doctor_state_ID         char(11),
        examination_ID          int,
        PRIMARY KEY (rating_ID),
        FOREIGN KEY (patient_state_ID) references patient(state_ID),
        FOREIGN KEY (doctor_state_ID) references doctor(state_ID),
        FOREIGN KEY (examination_ID) references examination);
```

2.23 Stores

Relational Model:

stores (<u>pharmacy_ID</u>, <u>drug_ID</u>, number_in_stock)

Functional Dependencies:

pharmacy_ID, drug_ID → number_in_stock

Candidate Keys:

{ (pharmacy_ID, drug_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE stores(
        pharmacy_ID         int,
        drug_ID             int,
        number_in_stock     int,
        PRIMARY KEY (pharmacy_ID, drug_ID),
        FOREIGN KEY (pharmacy_ID) references pharmacy,
        FOREIGN KEY (drug_ID) references drug);
```

2.24 Works as Doctor

Relational Model:

worksAsDoctor (<u>state_ID</u>, hospital_ID, role)

Functional Dependencies:

none

Candidate Keys:

{ (state_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE worksAsDoctor(
        state_ID        char(11),
        hospital_ID     int,
        role            varchar(20),
        PRIMARY KEY (state_ID),
        FOREIGN KEY (state_ID) references doctor,
        FOREIGN KEY (hospital_ID) references hospital(hospital_ID));
```

2.25 Test Executed in

Relational Model:

textExecutedIn (<u>test_ID</u>, <u>hospital_ID</u>)

Functional Dependencies:

none

Candidate Keys:

{ (test_ID, hospital_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE textExecutedIn (
        test_ID         int,
        hospital_ID     int,
        PRIMARY KEY (test_ID, hospital_ID),
        FOREIGN KEY (test_ID) references test,
        FOREIGN KEY (hospital_ID) references hospital);
```

2.26 Examination Result

Relational Model:

examinationResult(<u>examination_ID</u>, test_ID, treatment_ID, prescription_ID)

Functional Dependencies:

none

Candidate Keys:

{ (examination_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE examinationResult(
        examination_ID       int,
        test_ID              int,
        treatment_ID         int,
        prescription_ID      int,
        PRIMARY KEY (examination_ID),
        FOREIGN KEY (examination_ID) references examination,
        FOREIGN KEY (test_ID) references test,
        FOREIGN KEY (treatment_ID) references treatment,
        FOREIGN KEY (prescription_ID) references prescription);
```

2.27 Prescribed

Relational Model:

prescribed(prescription_ID, drug_ID)

Functional Dependencies:

none

Candidate Keys:

{ (prescription_ID, drug_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE prescribed(
        prescription_ID         int,
        drug_ID                 int,
        PRIMARY KEY (prescription_ID, drug_ID),
        FOREIGN KEY (prescription_ID) references prescription,
        FOREIGN KEY (drug_ID) references drug);
```

2.28 Alternative to

Relational Model:

alternativeTo(<u>drug_ID</u>, <u>alternative_drug_ID</u>)

Functional Dependencies:

none

Candidate Keys:

{ (drug_ID, alternative_drug_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE alternativeTo(
        drug_ID                 int,
        alternative_drug_ID     int,
        PRIMARY KEY (drug_ID, alternative_drug_ID),
        FOREIGN KEY (drug_ID) references drug,
        FOREIGN KEY (alternative_drug_ID) references drug);
```

2.29 Buy Drug

Relational Model:

buy_drug(<u>state_ID</u>, <u>drug_ID</u>, <u>pharmacy_ID</u>)

Functional Dependencies:

none

Candidate Keys:

{ (state_ID, drug_ID, pharmacy_ID) }

Normal Form:

BCNF

Table Definition:

```
CREATE TABLE buy_drug (
        state_ID       char(11),
        drug_ID        int,
        pharmacy_ID    int,
   PRIMARY KEY (state_ID,drug_ID,pharmacy_ID),
   FOREIGN KEY (state_ID) references patient(state_ID) ,
   FOREIGN KEY (drug_ID) references drug(drug_ID),
   FOREIGN KEY (pharmacy_ID) references pharmacy(pharmacy_ID))ENGINE=InnoDB;
```

3. Functional Dependencies and Normalization of Tables

Every functional dependency and every normal form are given in the relation schemas which is Section 2 of this Project Design Report. Every relation is checked in our design if the relation is in Boyce-Codd Normal Form. Since the left side of the functional dependencies in our schemas are foreign keys, they are in BCNF form and does need further decomposition.

4. Functional Components

4.1 Use Cases / Scenarios

4.1.1 Patient

- Patients can only login to the system with their state IDs and their passwords.
- Patients can only view their medical profile which are vaccine history, examinations, diagnoses treatments, prescribed drugs, allergies, test results, chronic diseases.
- Patients can view hospitals and their information with doctors who are working there.
- Patients can book an appointment from doctors.
- Patients can only view and edit their own profile which has emergency contact and profile information.

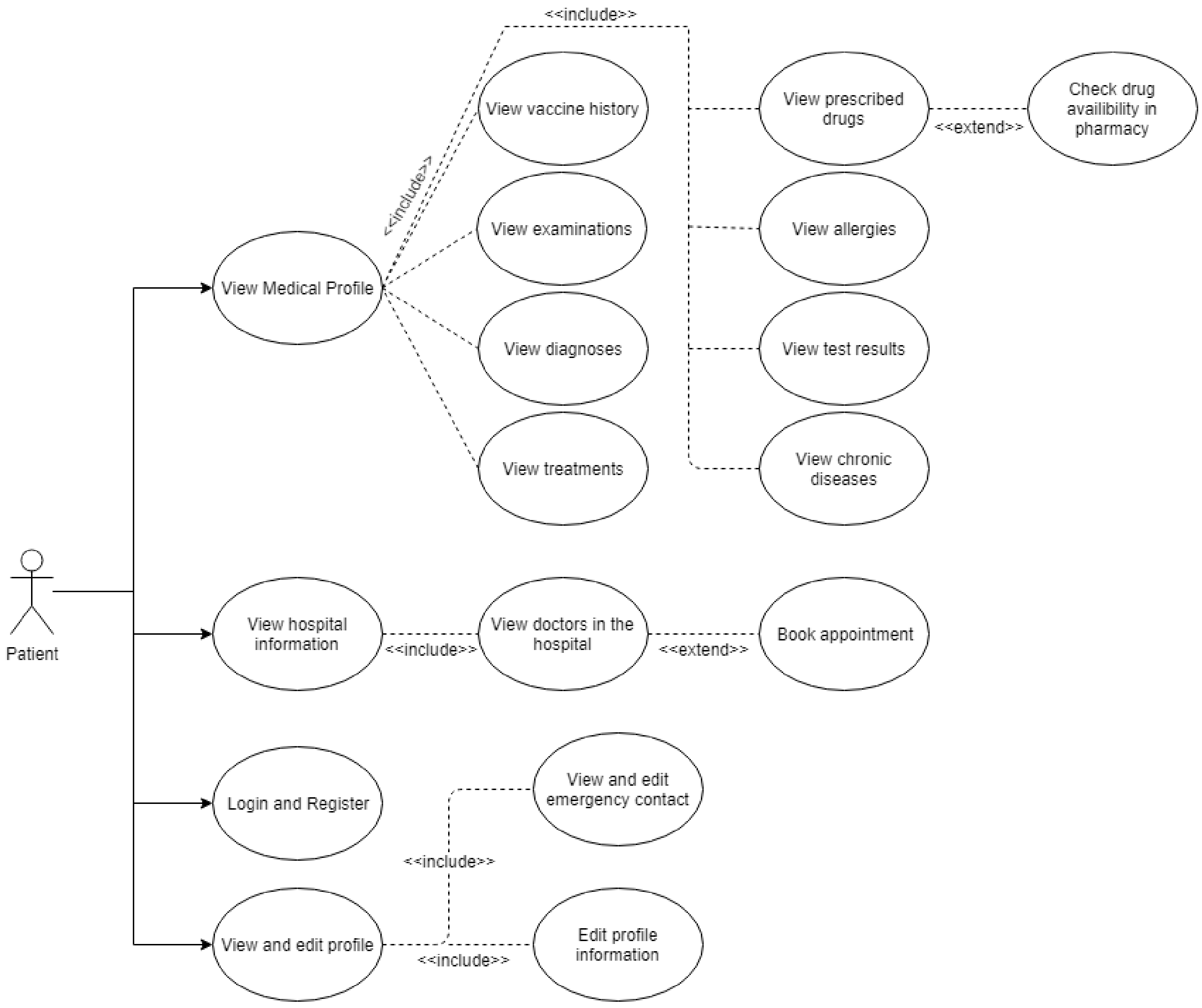


Figure 2: Patients' Use Case Diagram

4.1.2 Doctor

- Doctors can only login to the system with their state IDs and their passwords.
- Doctors will vaccinate a patient in real life then they will add this vaccination of a patient with the information of the date and the name of the vaccine with the state ID’s of the patient.
- Doctors can add the examination result of a patient with the state ID’s of the patient..
- Doctors can add the prescription of a patient after an examination with the state ID’s of the patient.
- Doctors can add the treatment of a patient after an examination.
- Doctors can add the test results after a test is done after the examination.
- Doctors can add diagnoses such as allergies or chronic disease of a patient.
- Doctors can view hospital informations.
- Doctors can view a patient’s medical information.
- Doctors can view their schedule.

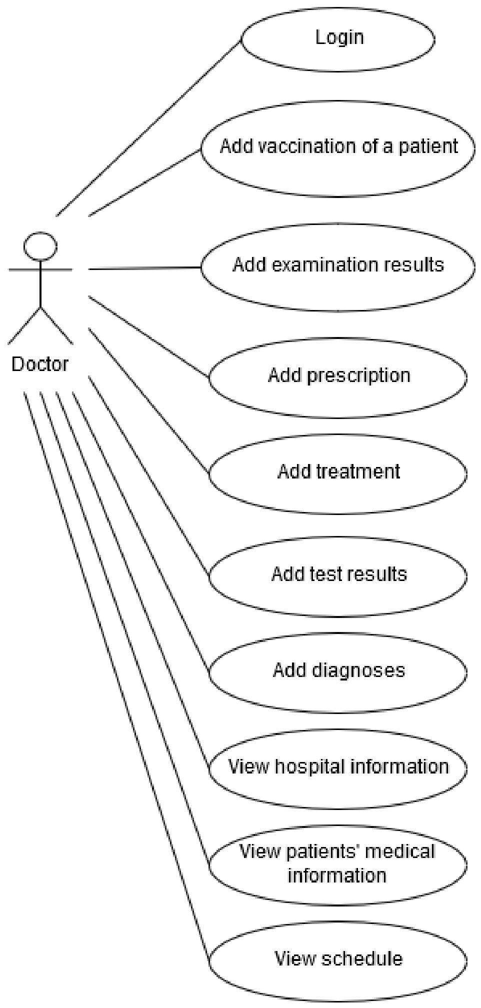


Figure 3: Doctors’ Use Case Diagram

4.1.3 Pharmacist

- Pharmacists can register and login
- Pharmacists can register their new pharmacies to the system.
- Pharmacists can manage the pharmacy stock such as adding new drugs or removing drugs from the pharmacy.
- Pharmacists can view patients' prescriptions.
- Pharmacists can edit their pharmacys' information.
- Pharmacists can add or remove other pharmacists from their pharmacies.
- Pharmacists can check whether there are no drugs left in the store or not, and can check the alternative drugs for that drug.

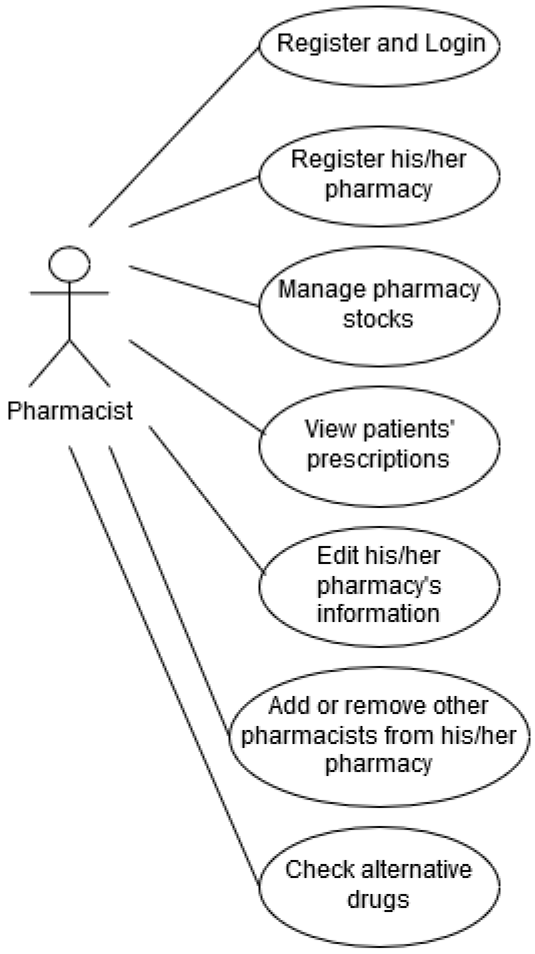

Register and Login
Register his/her pharmacy
Manage pharmacy stocks
View patients' prescriptions
Edit his/her pharmacy's information
Add or remove other pharmacists from his/her pharmacy
Check alternative drugs
Pharmacist

Figure 4: Pharmacists' Use Case Diagram

4.2 Algorithms

Since our project is mostly based on database manipulations, there are not any domain specific algorithm that will be used in the project. Application will do database queries in order to add, update or get information from the database and the information that database contains will be displayed to users. Our algorithms will be basically the queries that we write to interact with the database.

4.3 Data Structures

We have used char, varchar, date and int domains in the MySql tables. There could also be sorted array or sorted linked list structures in server side or in client side to display lists in order.

5. User Interface Design and Corresponding SQL Statements

5.1 Doctors' Page

This is page which doctors who have already registered to system will see when they login. First to sections in which hospital information and doctor list is displayed will be seen only by the executive doctor. Executive doctor will be able to click on the names of doctors to open an information card as an pop-up. In this pop-up, executive doctor will be able to change the schedule of doctors. Executive doctor will also be able to change or add departments. Other doctors will not see these sections and will not be able to edit hospital information or add new doctors to hospital.

Doctors who are not executive doctor will see their information and the top and then continue with patient medical information section so that they will not be able to change hospital related information. In the patient medical information section, they will be able to request medical information of a patient by providing the state id of the patient. View Patients Medical History button will redirect to the profile page of the patient in which medical history is displayed.

Doctors will be able to register examinations in the new examination section. They will register any diagnoses, test, treatment and prescribed drug in this section.

Figure 6: Doctor's Page

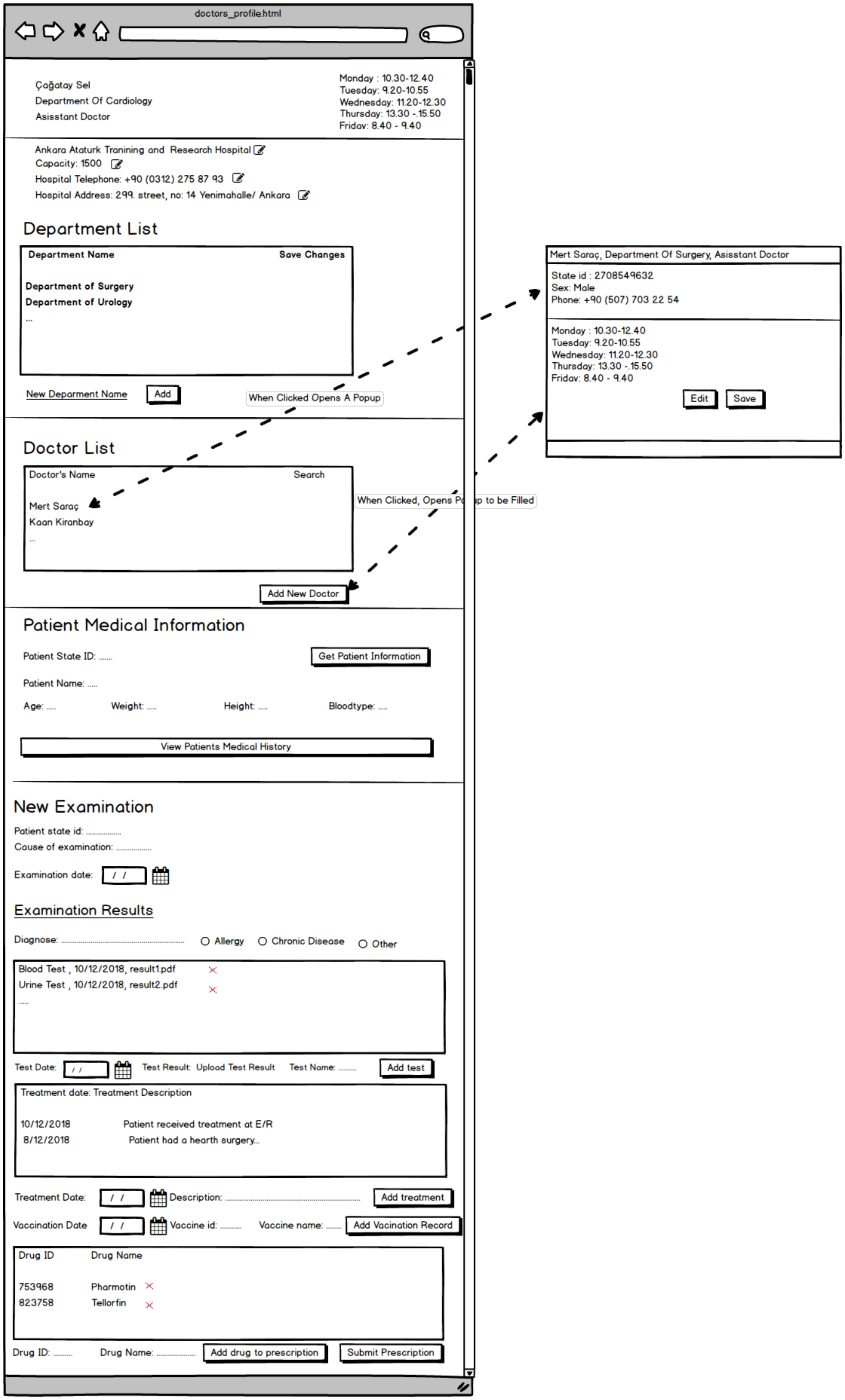
SQL Statements

Retrieving Doctor's Information

```
SELECT doctor_department, doctor_title, doctor_schedule

FROM doctor

WHERE doctor.state_ID = @state_ID;
```

Retrieving Hospital Information

```
SELECT hospital_ID hospital_name, hospital_capacity, hospital_telephone, hospital_address

FROM hospital
```



doctors_profile.html

Çağatay Sel
Department Of Cardiology
Asisstant Doctor

Monday : 10.30-12.40
Tuesday: 9.20-10.55
Wednesday: 11.20-12.30
Thursday: 13.30 -.15.50
Friday: 8.40 - 9.40

Ankara Ataturk Tranining and Research Hospital
Capacity: 1500
Hospital Telephone: +90 (0312) 275 87 93
Hospital Address: 299. street, no: 14 Yenimahalle/ Ankara

Department List

Department Name	Save Changes
Department of Surgery	
Department of Urology	
...	

New Deparment Name Add

When Clicked Opens A Popup

Doctor List

Doctor's Name	Search
Mert Saraç	
Kaan Kiranbay	
...	

Add New Doctor

When Clicked, Opens Popup to be Filled

Mert Saraç, Department Of Surgery, Asisstant Doctor

State id : 2708549632
Sex: Male
Phone: +90 (507) 703 22 54

Monday : 10.30-12.40
Tuesday: 9.20-10.55
Wednesday: 11.20-12.30
Thursday: 13.30 -.15.50
Friday: 8.40 - 9.40

Edit Save

Patient Medical Information

Patient State ID: ...... Get Patient Information

Patient Name: .....

Age: ..... Weight: ..... Height: ..... Bloodtype: .....

View Patients Medical History

New Examination

Patient state id: ....
Cause of examination: ....

Examination date: / /

Examination Results

Diagnose: .... ○ Allergy ○ Chronic Disease ○ Other

Blood Test , 10/12/2018, result1.pdf	×
Urine Test , 10/12/2018, result2.pdf	×
.....	

Test Date: / / Test Result: Upload Test Result Test Name: .... Add test

Treatment date:	Treatment Description
10/12/2018	Patient received treatment at E/R
8/12/2018	Patient had a hearth surgery...

Treatment Date: / / Description: .... Add treatment

Vaccination Date / / Vaccine id: .... Vaccine name: .... Add Vacination Record

Drug ID	Drug Name	
753968	Pharmotin	×
823758	Tellorfin	×

Drug ID: .... Drug Name: .... Add drug to prescription Submit Prescription

Retrieving Departments

```
SELECT hospital_department
FROM hospitalDepartment
WHERE hospitalDepartment.hospital_ID =@hospital_ID;
```

Adding New Department

```
INSERT INTO hospital_department
VALUES (hospital_ID, new_department);
```

Listing Doctors in Hospital

```
SELECT first_name, middle_name, last_name,sex,phone,password
FROM user
WHERE user.state_ID in  (SELECT state_ID ,
                          FROM workAsDoctor
                          WHERE workAsDoctor.hospital_ID =@ hospital_ID) ;
```

```
SELECT doctor_department, doctor_title,doctor_schedule
FROM doctor
WHERE doctor.state_ID in (SELECT state_ID ,
                          FROM workAsDoctor
                          WHERE workAsDoctor.hospital_ID= @hospital_ID) ;
```

Getting Patient Medical Information

```
SELECT first_name , middle_name, last_name
FROM user
WHERE user.state_ID = @state_ID;
```

```
SELECT patient_weight, patient_height, patient_bloodtype
FROM patient
WHERE patient.state_ID = @state_ID;
```

Adding Vaccination Record

```
INSERT INTO vaccinates
VALUES (@vaccinate_ID , @patient_state_ID, @doctor_state_ID,@date);
```

Adding New Examination

```
INSERT INTO examination
VALUES (@examination_ID, @examination_cause, @examination_date,
@examination_diagnose);

INSERT INTO test
VALUES (@test_ID,@test_date,@test_result,@test_name);


INSERT INTO treatment
VALUES (@treatment_ID,@treatment_description,@treatment_date);


INSERT INTO prescription
VALUES (@prescription_ID,@prescription_date);

INSERT INTO prescribed
VALUES (@prescription_ID, @drug_id);

INSERT INTO examination_result
VALUES (@examination_ID,@test_ID,@treatment_ID,@prescription_ID);
```

```
INSERT INTO examination_done
VALUES (@examination_ID, @patient_state_ID,@doctor_state_ID);
```

<u>If patient is diagnosed with any allergy or chronic disease</u>

```
INSERT INTO patientAllergies
VALUES (@state_ID, @allergyName);

INSERT INTO patientChronicDisease
VALUES (@state_ID, @chronicDisease);
```

5.2 Login Page

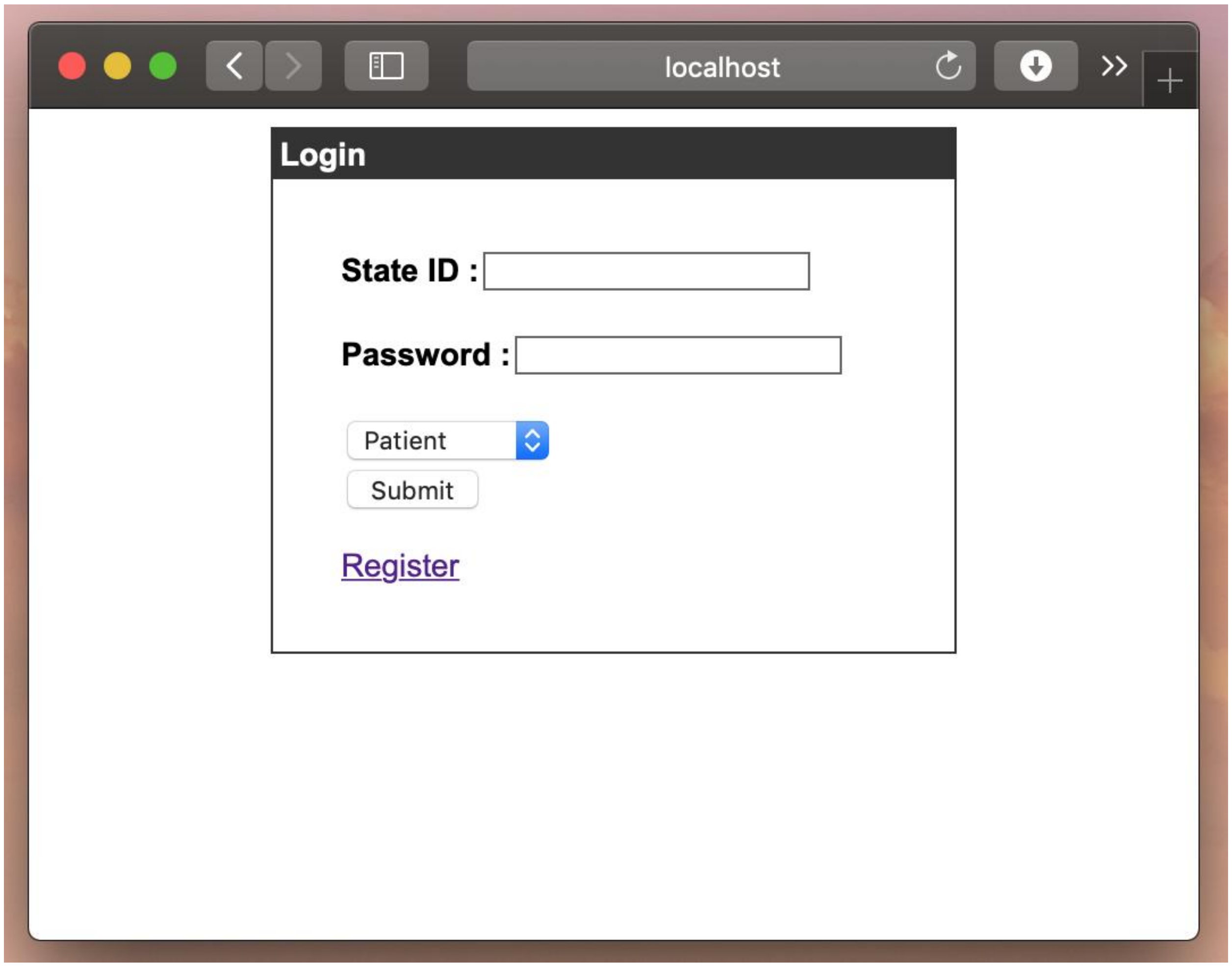


Figure 7: Login Page

In this page, user can login if he/she has already an account. Specifying type of the account (patient, account, executive doctor account or pharmacist account) is needed for login process.

Login

SELECT *

FROM user

WHERE user.state_ID = @state_ID, user.password = @password;

5.3 Register Page

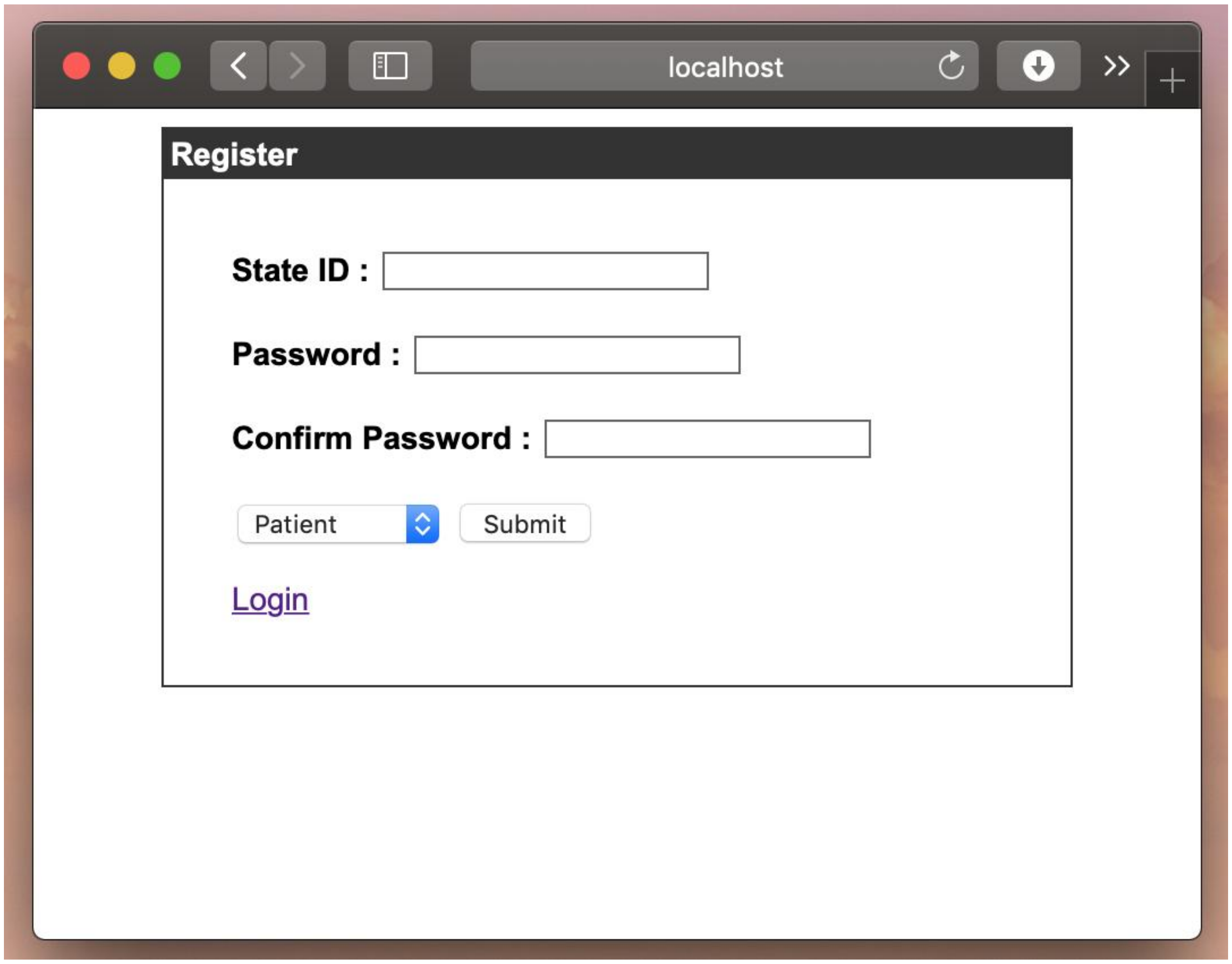


Figure 8: Register Page

If user has no account, he/she can create one easily by selecting register tab. To register, all user needs is entering state-id (TC no.) and password. Password is asked for two times in terms of reduce the likelihood of typo. Specifying type of the account is also needed here.

Register

SELECT state_id

FROM user

WHERE user.state_id = @state_id;

Registering a User

INSERT INTO user

VALUES (@state_id, NULL , NULL, NULL, NULL ,NULL ,@ password);

5.4 Information Page

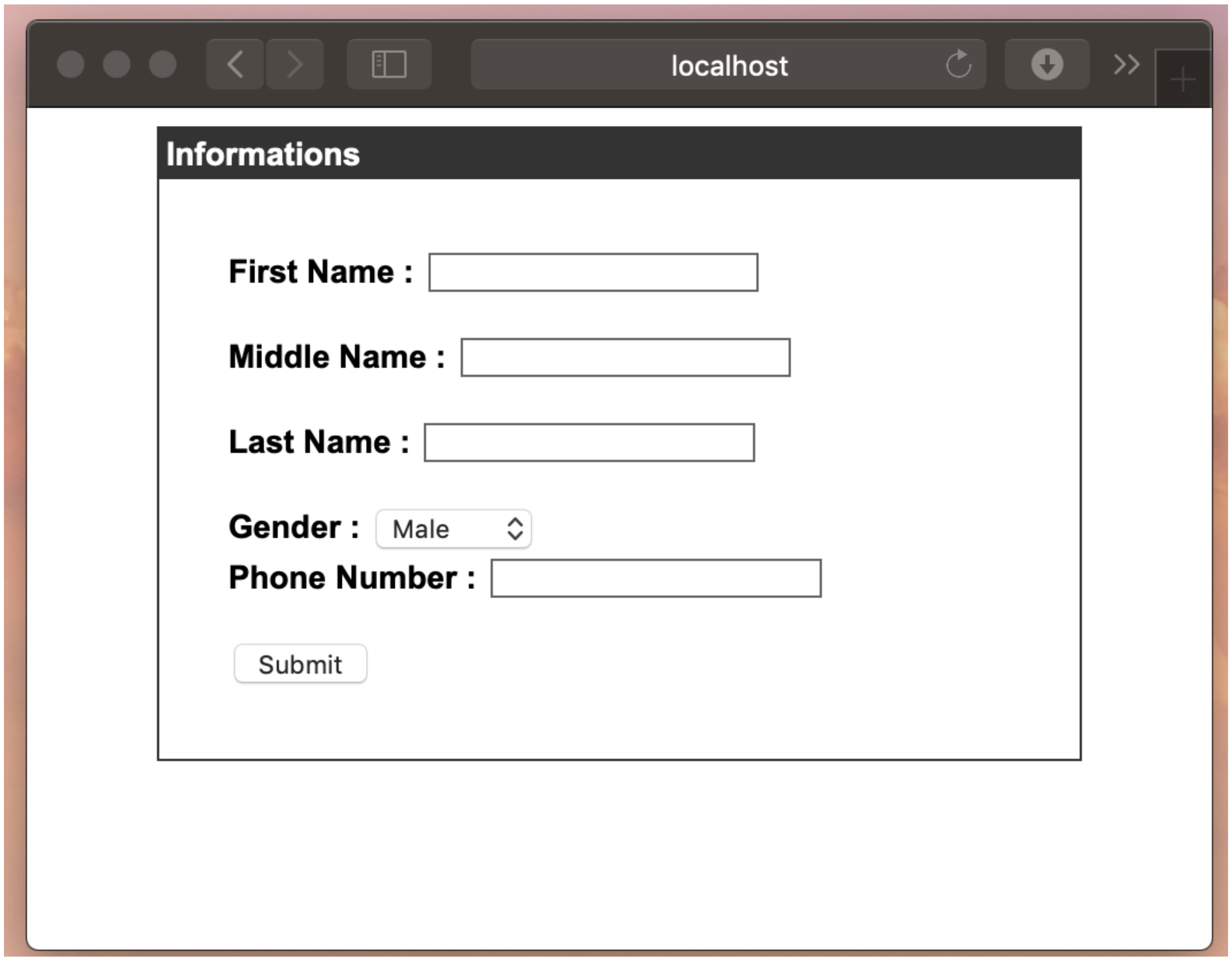


Figure 9: Information Page

All three type of the account has common features such as first name, middle name, last name, sex and phone number of the user. For doctor account, these informations belong to an executive doctor of the hospital. Similarly, if it is a pharmacist account, these informations belong to owner of the pharmacy.

Registering User Information

```
UPDATE user
SET
        first_name = @first_name,
        middle_name = @middle_name,
        last_name = @last_name,
        sex = @sex,
    phone = @phone
WHERE user.state_ID = @state_ID;
```

5.5 Hospital Information Page

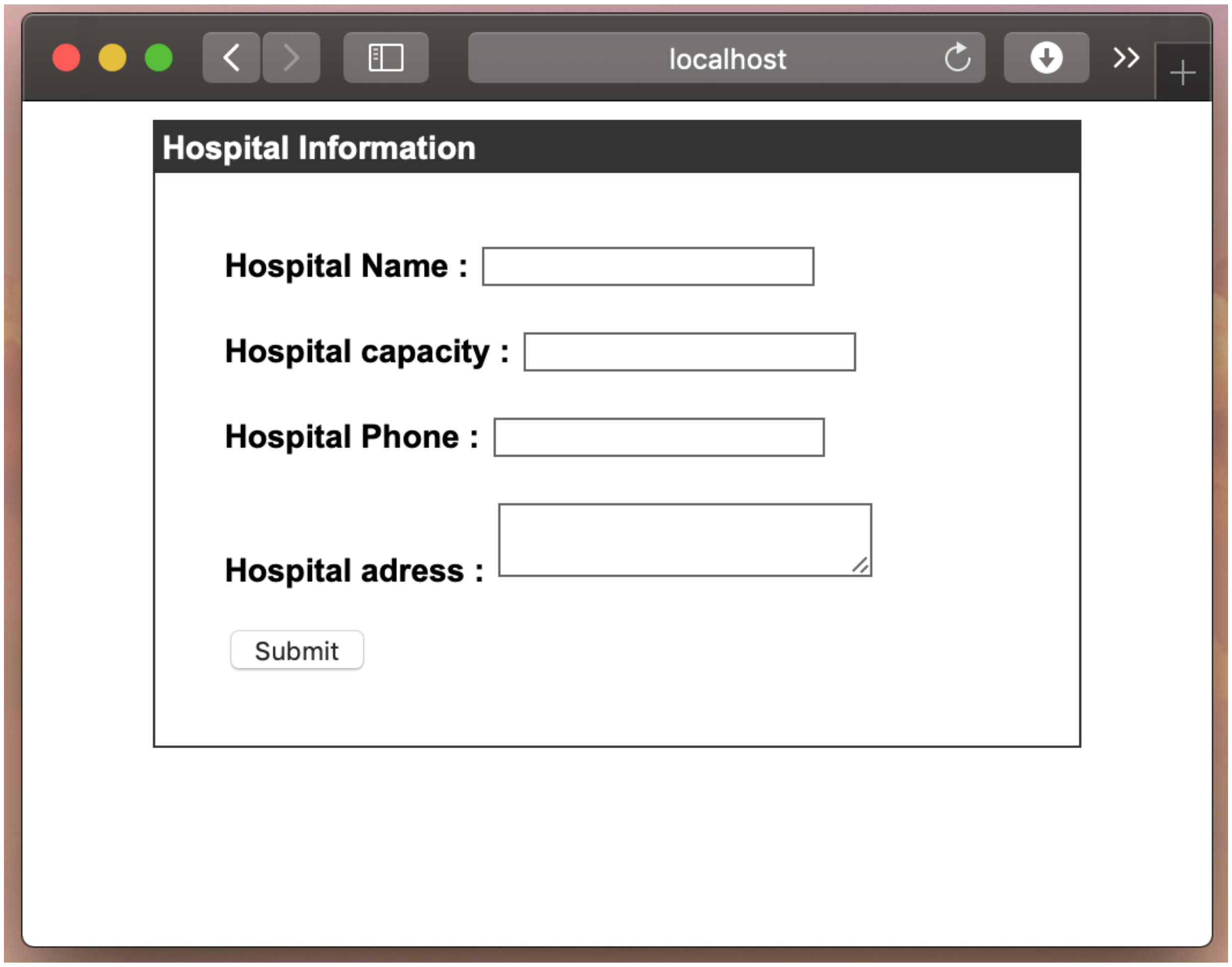


Figure 10: Hospital Information Page

In this page, user should enter informations about the hospital as it can be seen. By using "Add Hospital Department" button, he/she can create a department for the hospital and name it.

Executive Doctor Registering His/Her Hospital

INSERT INTO hospital

VALUES (NULL, @hospital_name, @hospital_capacity, @hospital_telephone, @hospital_address, @state_ID);

5.6 Patient Information Page

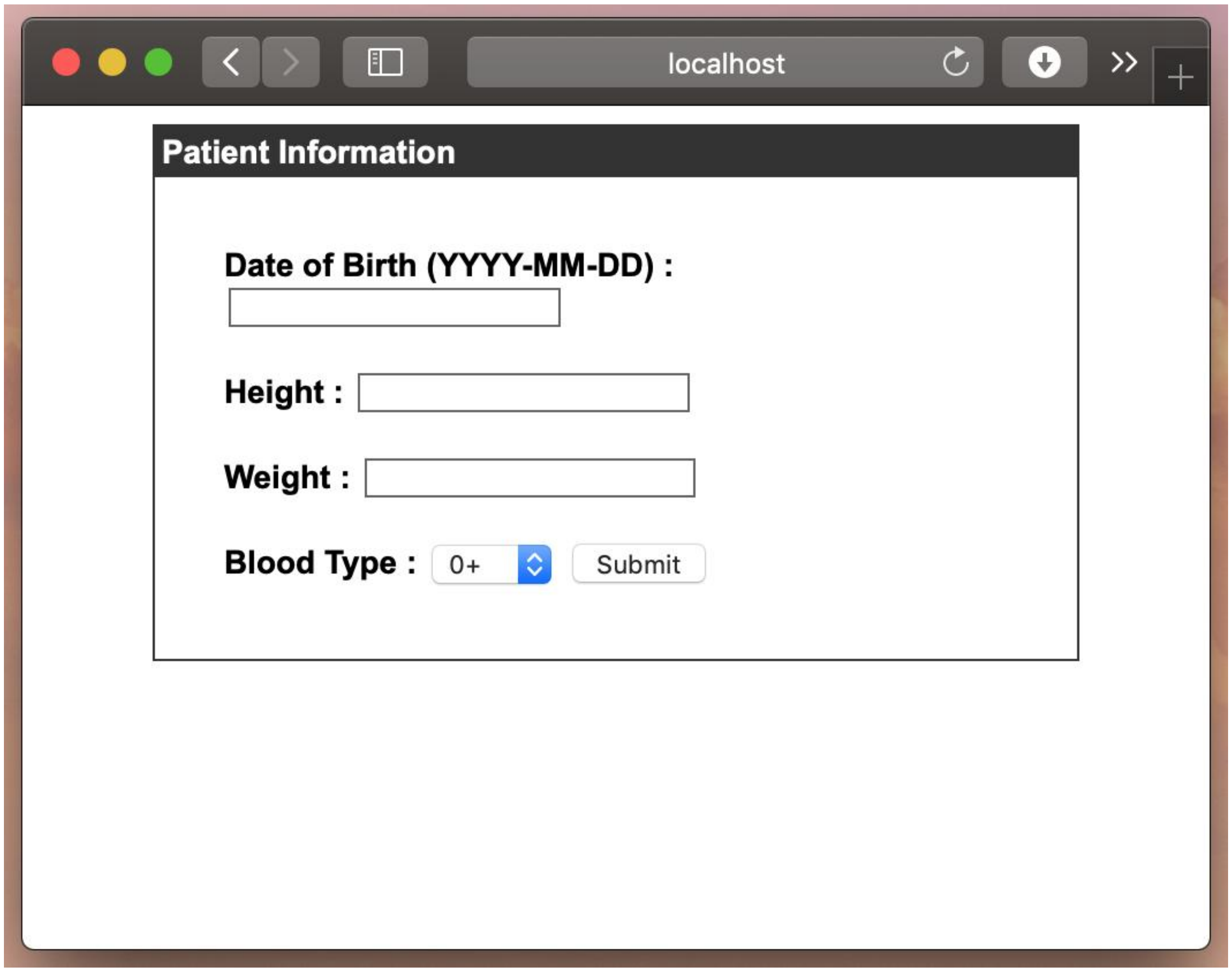


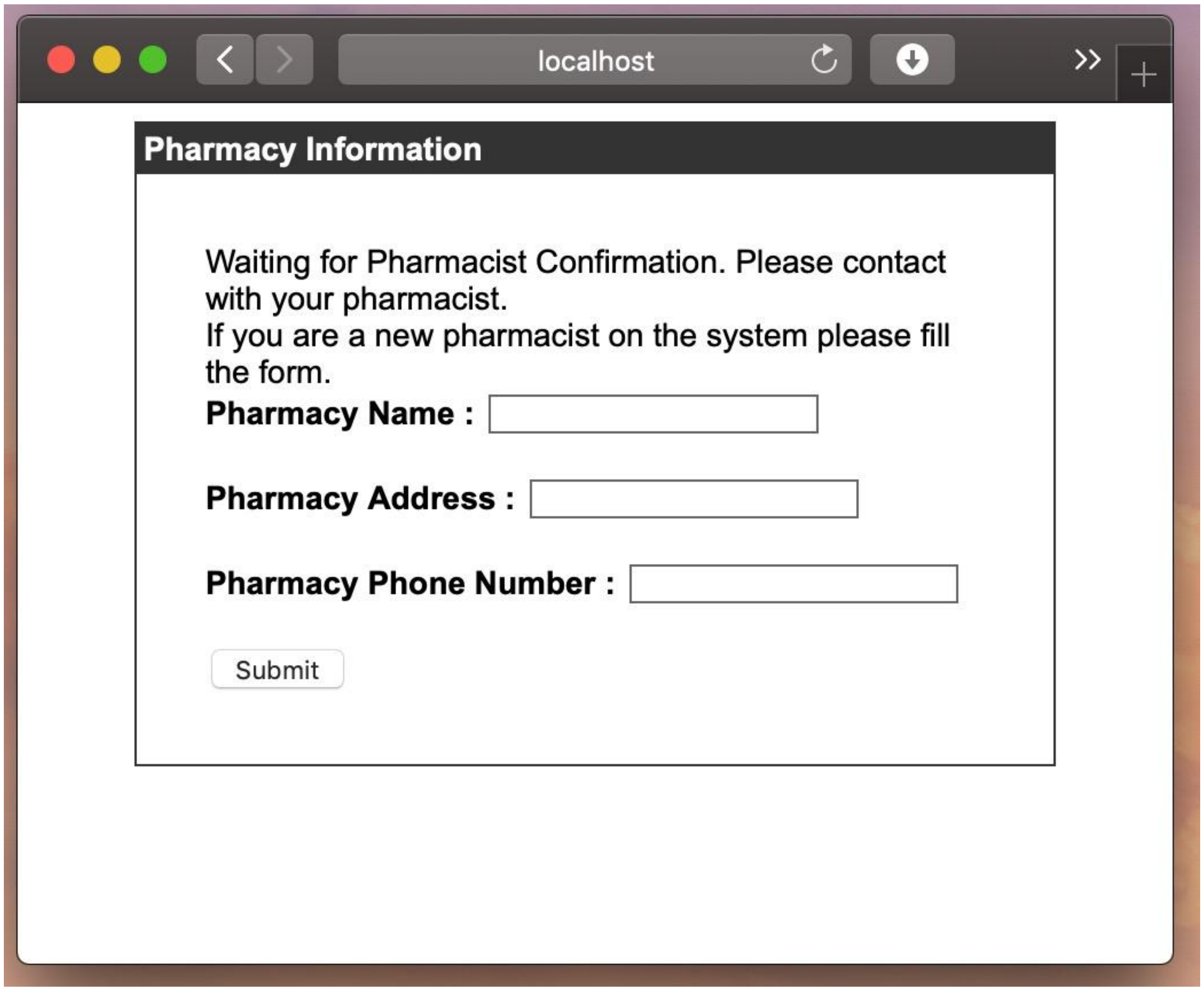
Figure 11: Patient Information Page

Patient Registering to System

INSERT INTO patient

VALUES (@state_ID, @patient_address, @patient_date_of_birth, @patient_weight, @patient_height, @patient_bloodtype);

5.7 Pharmacy Information Page



Pharmacy Information

Waiting for Pharmacist Confirmation. Please contact with your pharmacist.
If you are a new pharmacist on the system please fill the form.

Pharmacy Name :	
Pharmacy Address :	
Pharmacy Phone Number :	

Figure 12: Pharmacy Information Page

Pharmacist Registering His/Her Pharmacy to System

INSERT INTO pharmacy

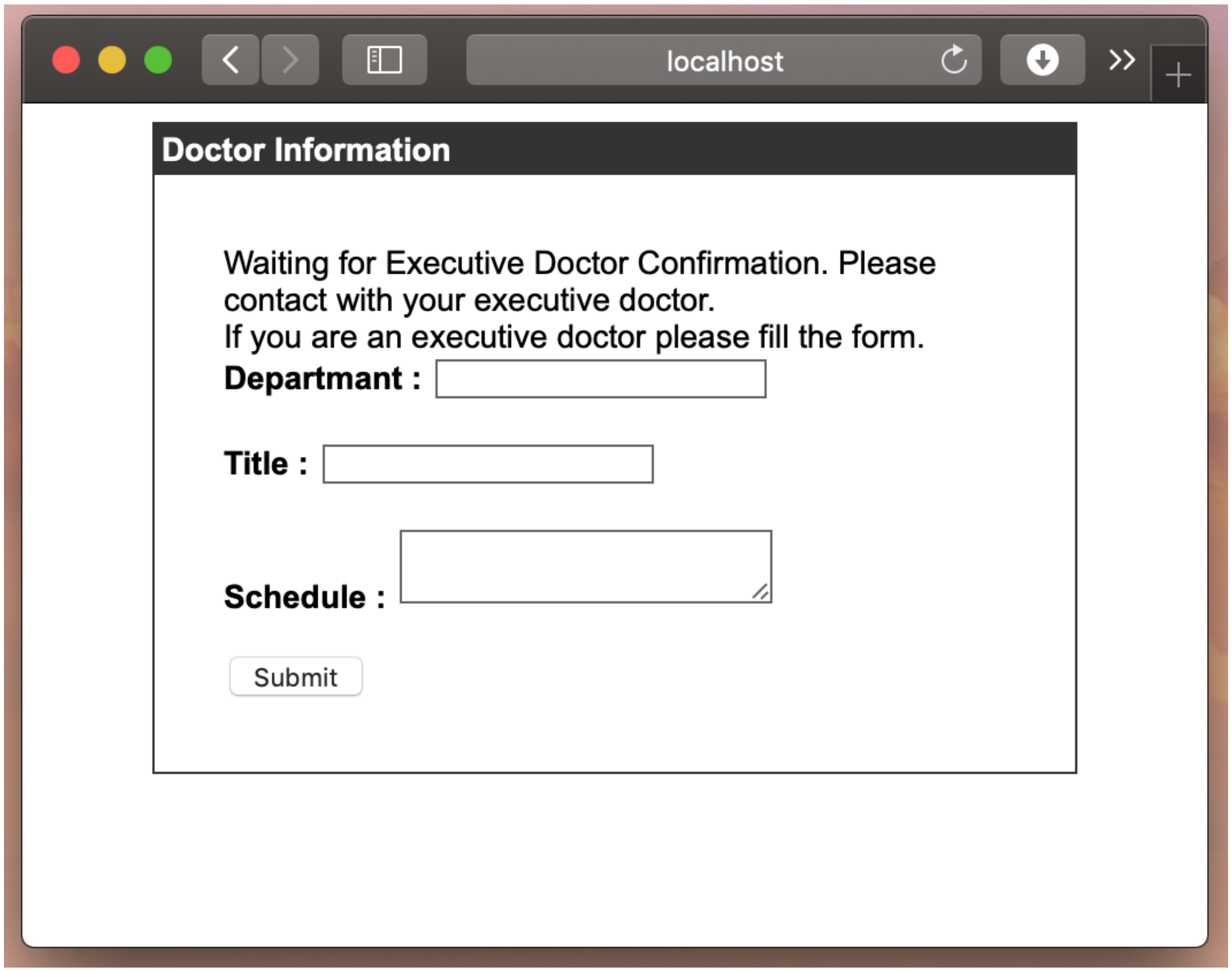
VALUES (NULL , @pharmacy_name, @pharmacy_address, @pharmacy_phone);

Adding Pharmacist as a Worker to His/Her Pharmacy

INSERT INTO worksAsPharmacist

VALUES (@state_ID , @pharmacy_ID);

5.7 Doctor Information Page



Waiting for Executive Doctor Confirmation. Please contact with your executive doctor.
If you are an executive doctor please fill the form.

Departmant :

Title :

Schedule :

Figure 13: Doctor Information Page

Pharmacist Registering His/Her Pharmacy to System

INSERT INTO pharmacy

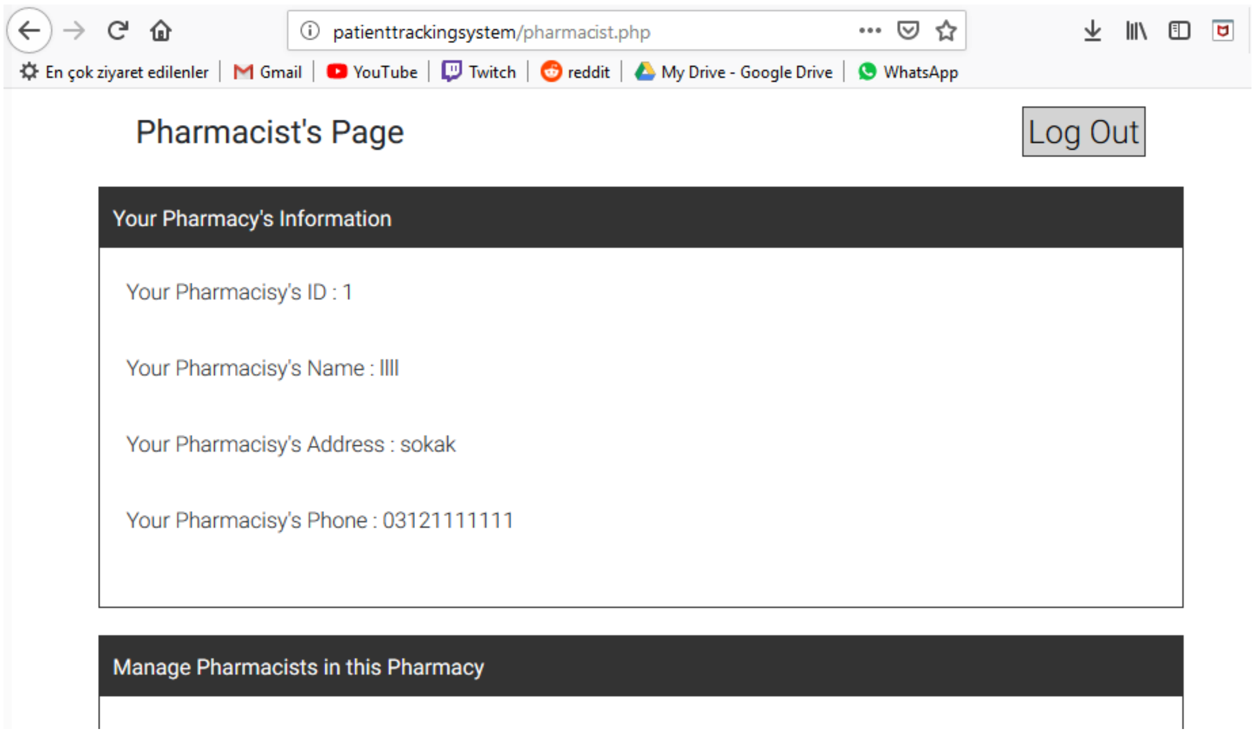
VALUES (NULL , @pharmacy_name, @pharmacy_address, @pharmacy_phone);

Adding Pharmacist as a Worker to His/Her Pharmacy

INSERT INTO worksAsPharmacist

VALUES (@state_ID , @pharmacy_ID);

5.8 Pharmacist Page



Pharmacist's Page

Your Pharmacy's Information

Your Pharmacisy's ID : 1

Your Pharmacisy's Name : llll

Your Pharmacisy's Address : sokak

Your Pharmacisy's Phone : 03121111111

Manage Pharmacists in this Pharmacy

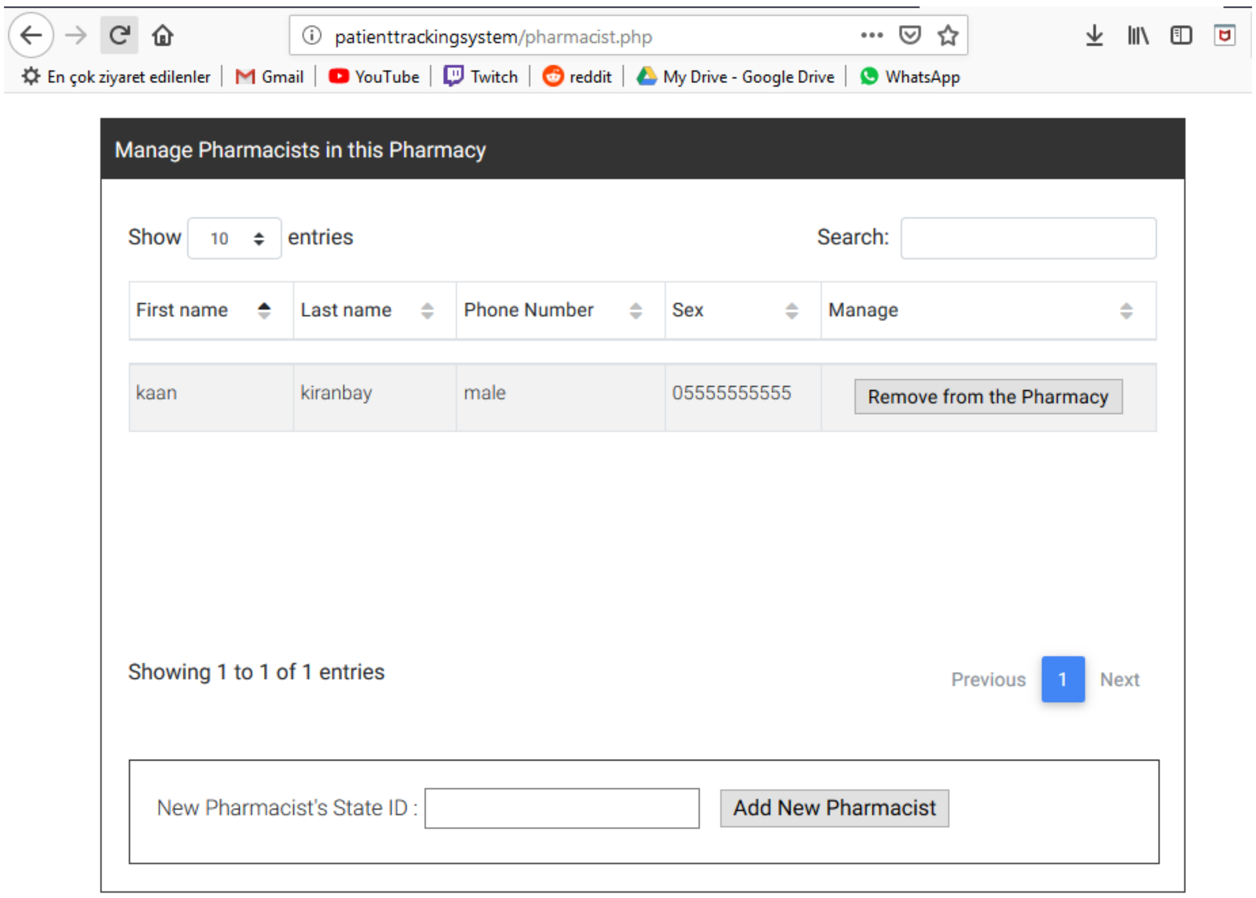
Figure 14: Pharmacist Page-1

Showing Pharmacy Information (Your Pharmacy's Information)

SELECT pharmacy_ID, pharmacy_name, pharmacy_address, pharmacy_phone

FROM pharmacy

WHERE pharmacy_ID = '$myPharmacyID'


patienttrackingsystem/pharmacist.php
En çok ziyaret edilenler | Gmail | YouTube | Twitch | reddit | My Drive - Google Drive | WhatsApp

Manage Pharmacists in this Pharmacy

Show 10 entries Search:

First name	Last name	Phone Number	Sex	Manage
kaan	kiranbay	male	05555555555	Remove from the Pharmacy

Showing 1 to 1 of 1 entries

Previous 1 Next

New Pharmacist's State ID : Add New Pharmacist

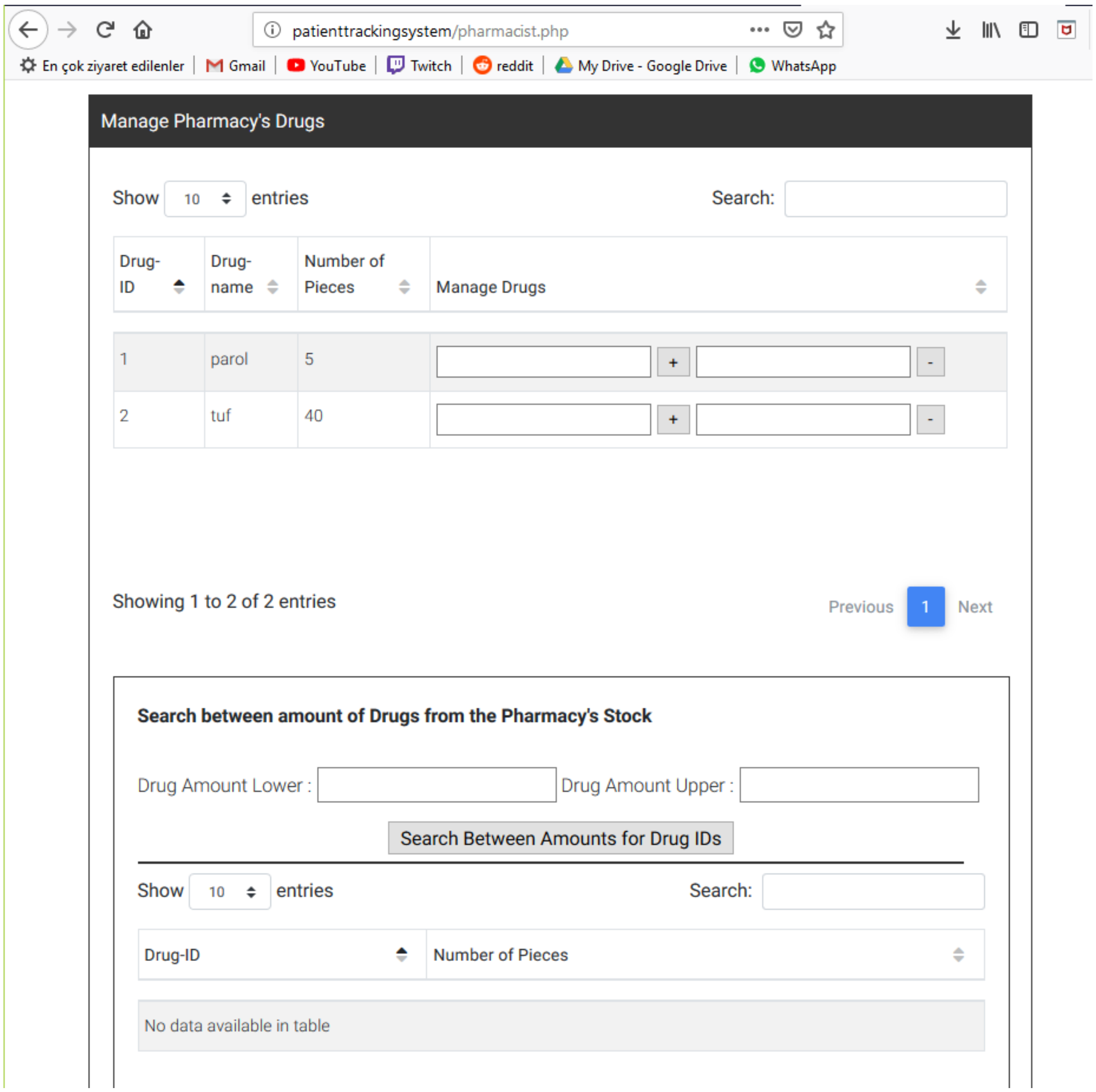
Figure 15: Pharmacist Page-2

Show Pharmacists Working in the Pharmacy

```
SELECT user.state_id, user.first_name, user.middle_name, user.last_name, user.sex, user.phone
FROM user
WHERE user.state_id in (SELECT state_id
                        FROM works_as_pharmacist
                        WHERE works_as_pharmacist.pharmacy_ID = '$myPharmacyID');
```

Adding a new Pharmacist to the Pharmacy

```
INSERT INTO works_as_pharmacist
VALUES ('$newPharmacistID', '$myPharmacyID')
```


patienttrackingsystem/pharmacist.php

En çok ziyaret edilenler | Gmail | YouTube | Twitch | reddit | My Drive - Google Drive | WhatsApp

Manage Pharmacy's Drugs

Show 10 entries Search:

Drug-ID	Drug-name	Number of Pieces	Manage Drugs
1	parol	5	+ -
2	tuf	40	+ -

Showing 1 to 2 of 2 entries

Previous 1 Next

Search between amount of Drugs from the Pharmacy's Stock

Drug Amount Lower : Drug Amount Upper :

Search Between Amounts for Drug IDs

Show 10 entries Search:

Drug-ID	Number of Pieces
No data available in table	

Figure 16: Pharmacist Page-3

Showing Current Drugs in the Pharmacy Stock

SELECT drug_ID, drug_name, number_in_stock

FROM pharmacy NATURAL JOIN stores NATURAL JOIN drug

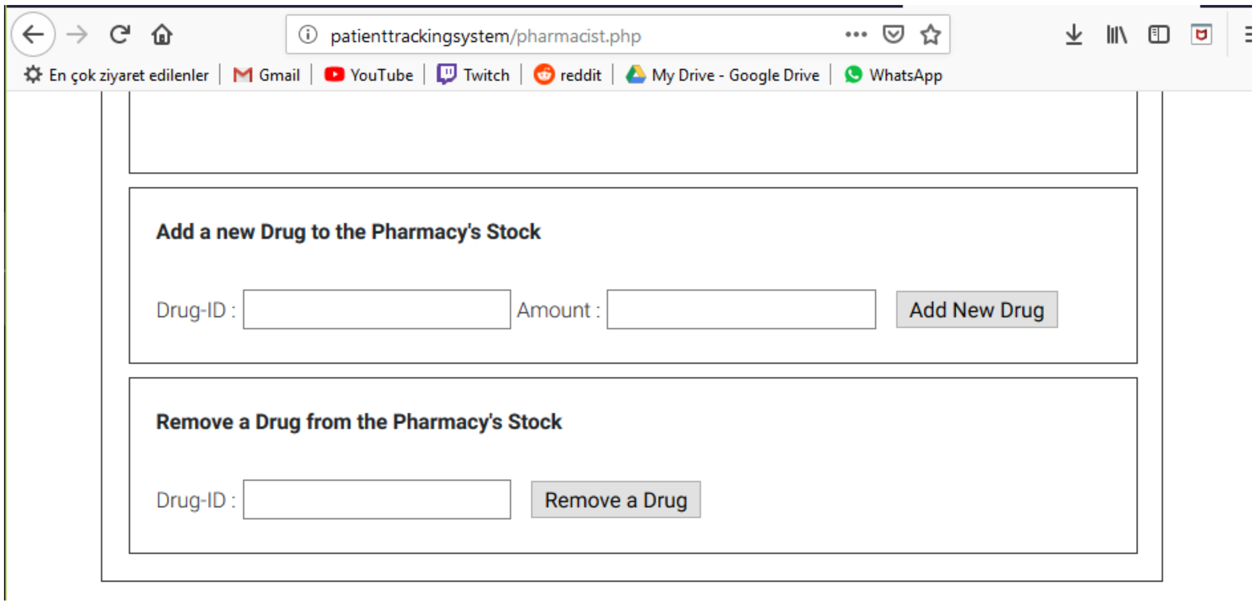
WHERE pharmacy.pharmacy_ID = '$myPharmacyID'

Search between Amount of Drugs from the Pharmacy's Stock

SELECT drug_ID, number_in_stock

FROM stores

WHERE number_in_stock BETWEEN '$curDrugAmountPharLow' AND '$curDrugAmountPharUpp'



Add a new Drug to the Pharmacy's Stock

Drug-ID : Amount : Add New Drug

Remove a Drug from the Pharmacy's Stock

Drug-ID : Remove a Drug

Figure 17: Pharmacist Page-4

Adding a New Drug to the Pharmacy’s Stock

INSERT INTO stores VALUES ('$myPharmacyID', '$newDrugIDPhar', '$newDrugAmountPhar')

Removing a Drug from the Pharmacy’s Stock

DELETE FROM stores

WHERE stores.pharmacy_ID = '$myPharmacyID' and stores.drug_ID = '$remDrugIDPhar'

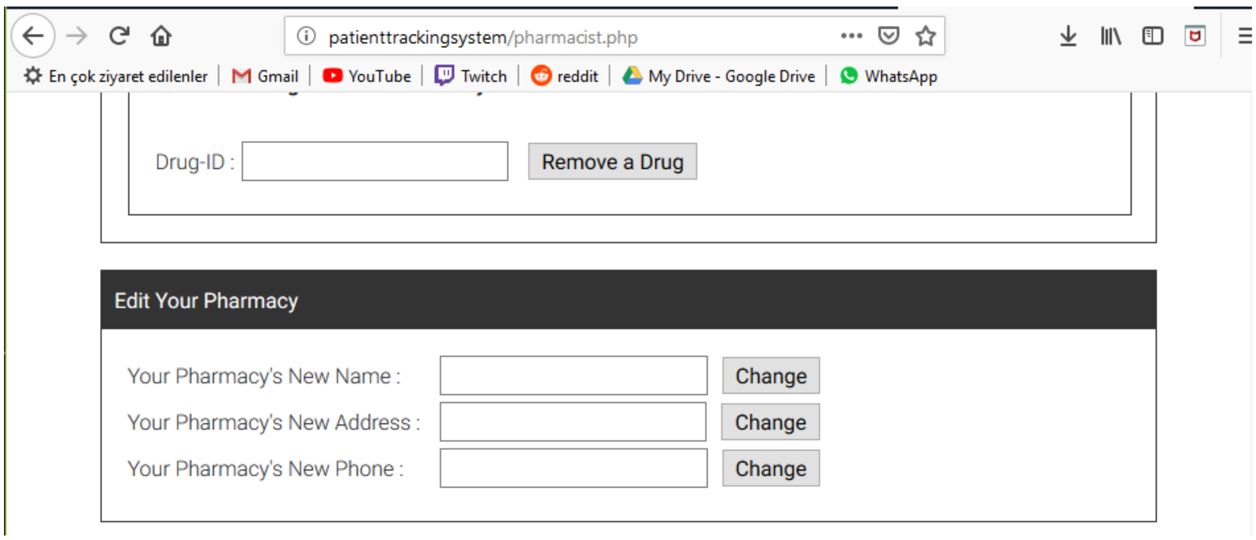


Figure 18: Pharmacist Page-5

Editing the Pharmacy’s Name

UPDATE pharmacy

SET pharmacy.pharmacy_name = '$pharNewName'

WHERE pharmacy.pharmacy_ID = $myPharmacyID

Editing the Pharmacy’s Address

UPDATE pharmacy

SET pharmacy.pharmacy_address = '$pharNewAddress'

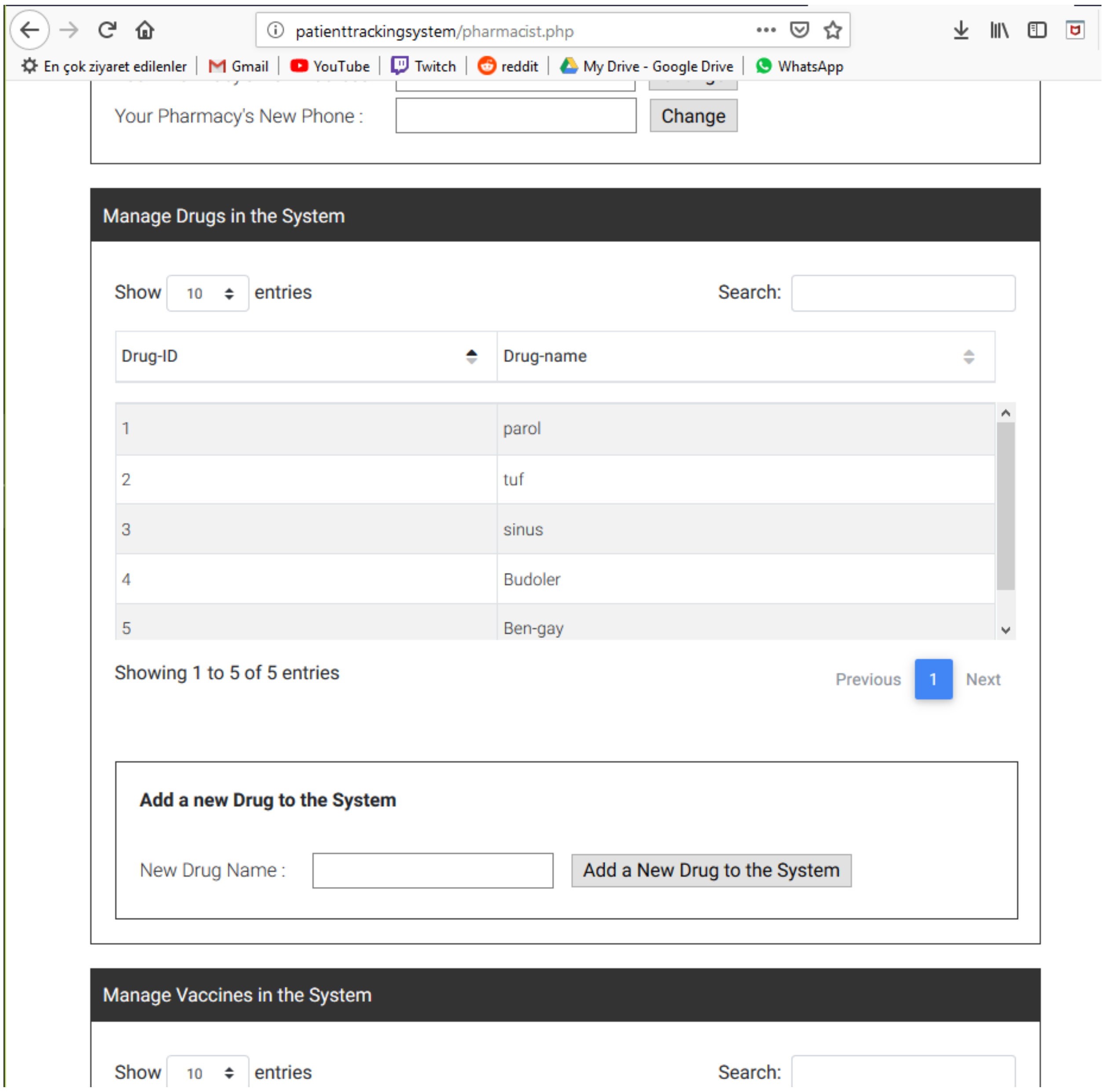
WHERE pharmacy.pharmacy_ID = '$myPharmacyID'

Editing the Pharmacy’s Phone

UPDATE pharmacy

SET pharmacy.pharmacy_phone = '$pharNewPhone'

WHERE pharmacy.pharmacy_ID = '$myPharmacyID'



patienttrackingsystem/pharmacist.php

En çok ziyaret edilenler | Gmail | YouTube | Twitch | reddit | My Drive - Google Drive | WhatsApp

Your Pharmacy's New Phone : Change

Manage Drugs in the System

Show 10 entries Search:

Drug-ID	Drug-name
1	parol
2	tuf
3	sinus
4	Budoler
5	Ben-gay

Showing 1 to 5 of 5 entries

Previous 1 Next

Add a new Drug to the System

New Drug Name : Add a New Drug to the System

Manage Vaccines in the System

Show 10 entries Search:

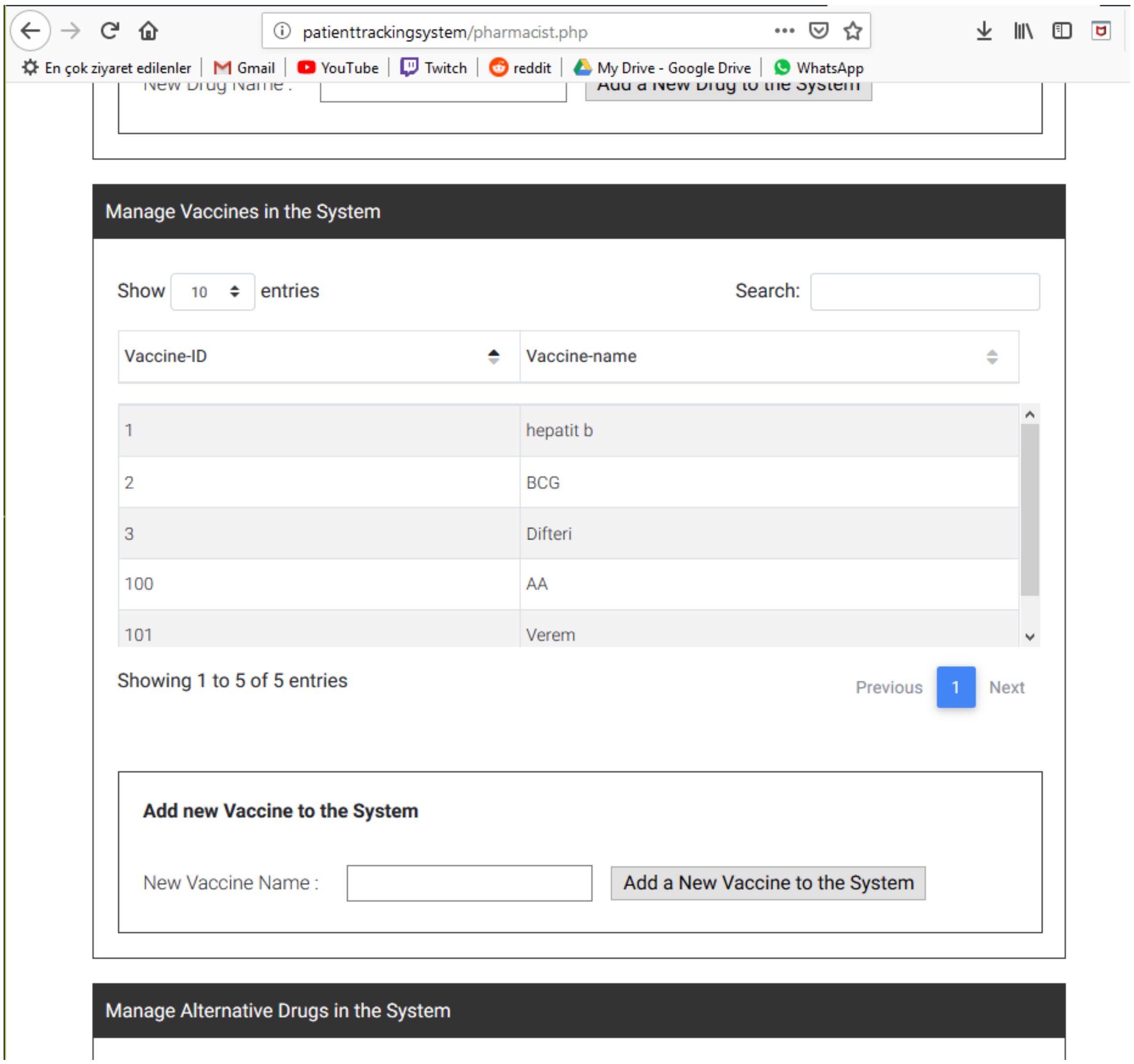
Figure 19: Pharmacist Page-6

Showing Current Drugs in the System

SELECT drug.drug_ID, drug.drug_name FROM drug

Adding a Drug to the System

INSERT INTO drug VALUES (NULL, '$newDrugNameSys')



patienttrackingsystem/pharmacist.php

En çok ziyaret edilenler | Gmail | YouTube | Twitch | reddit | My Drive - Google Drive | WhatsApp

New Drug Name : Add a New Drug to the System

Manage Vaccines in the System

Show 10 entries Search:

Vaccine-ID	Vaccine-name
1	hepatit b
2	BCG
3	Difteri
100	AA
101	Verem

Showing 1 to 5 of 5 entries

Previous 1 Next

Add new Vaccine to the System

New Vaccine Name : Add a New Vaccine to the System

Manage Alternative Drugs in the System

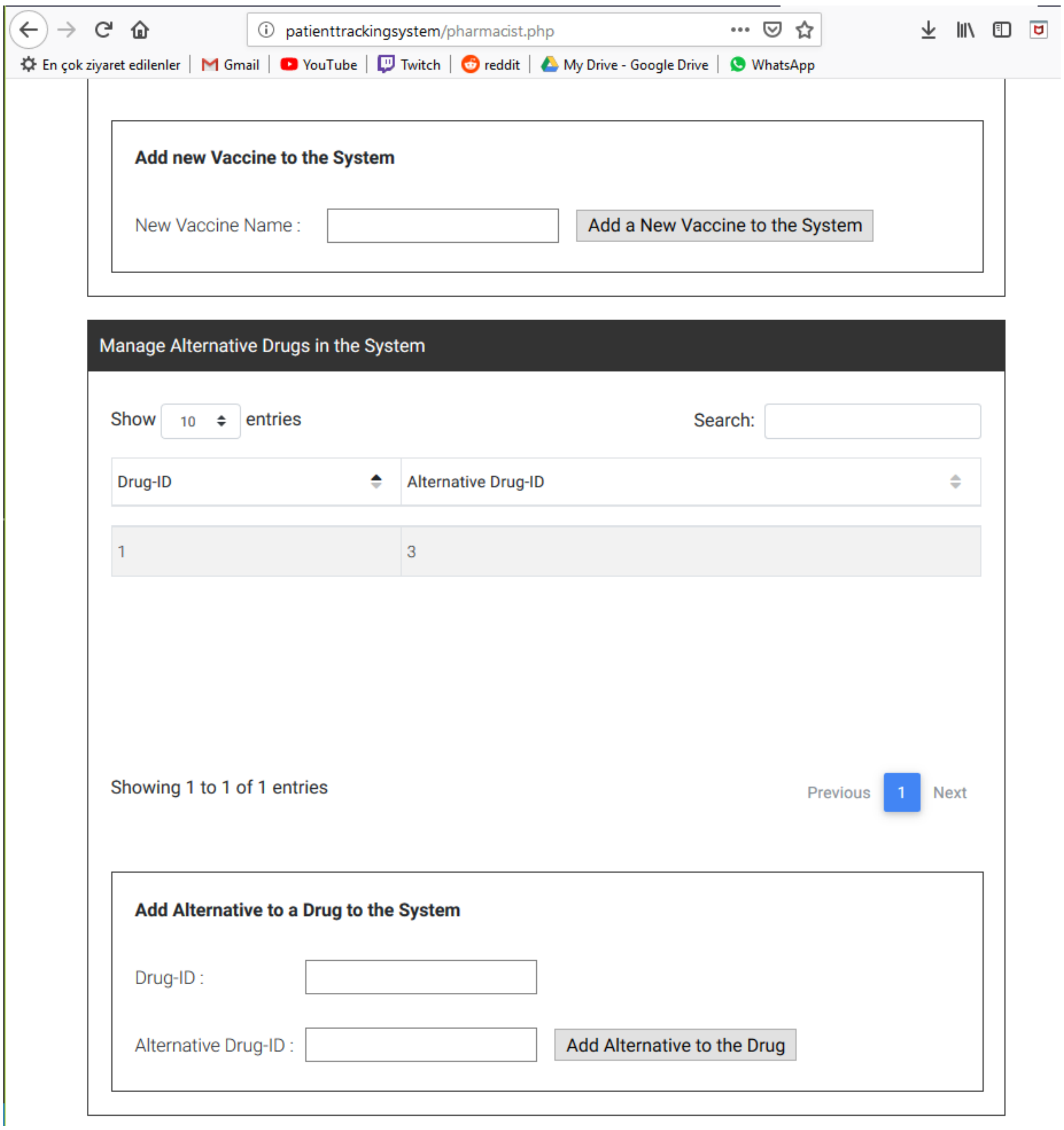
Figure 20: Pharmacist Page-7

Showing Current Vaccines in the System

SELECT vaccine.vaccine_ID, vaccine.vaccine_name FROM vaccine

Adding a new Vaccine to the System

INSERT INTO vaccine VALUES (NULL, '$newVaccNameSys')


patienttrackingsystem/pharmacist.php

En çok ziyaret edilenler | Gmail | YouTube | Twitch | reddit | My Drive - Google Drive | WhatsApp

Add new Vaccine to the System

New Vaccine Name :

Add a New Vaccine to the System

Manage Alternative Drugs in the System

Show 10 entries Search:

Drug-ID	Alternative Drug-ID
1	3

Showing 1 to 1 of 1 entries

Previous 1 Next

Add Alternative to a Drug to the System

Drug-ID :

Alternative Drug-ID :

Add Alternative to the Drug

Figure 21: Pharmacist Page-8

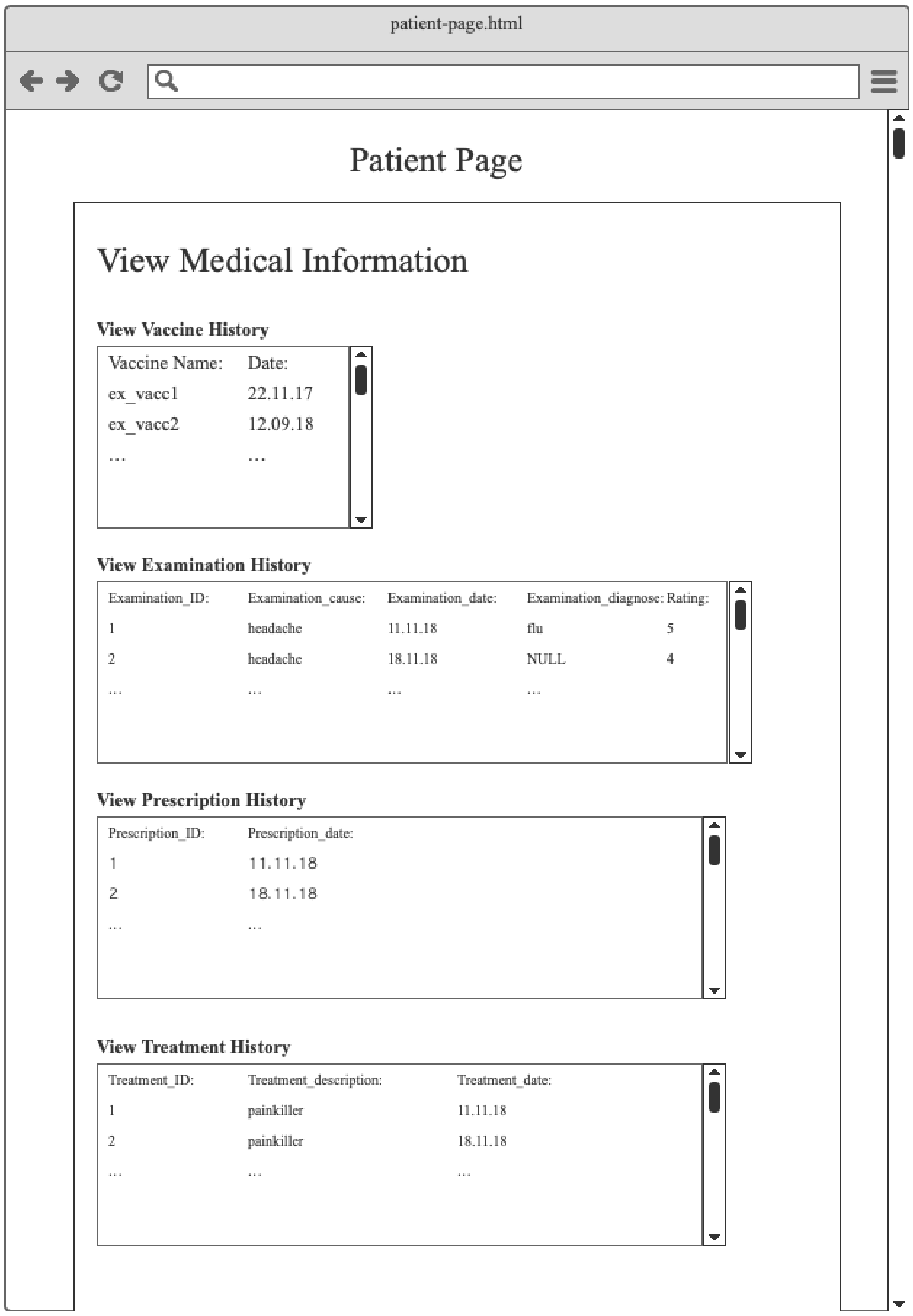
Showing Alternative Drugs in the System

SELECT alternative_to.drug_ID, alternative_to.alternative_drug_ID FROM alternative_to

Adding an Alternative to a Drug to the System

INSERT INTO alternative_to VALUES ('$newAltForDrugIDSys', '$newAltDrugIDSys')

5.9 Patient Page


patient-page.html

Patient Page

View Medical Information

View Vaccine History

Vaccine Name:	Date:
ex_vacc1	22.11.17
ex_vacc2	12.09.18
…	…

View Examination History

Examination_ID:	Examination_cause:	Examination_date:	Examination_diagnose:	Rating:
1	headache	11.11.18	flu	5
2	headache	18.11.18	NULL	4
…	…	…	…	

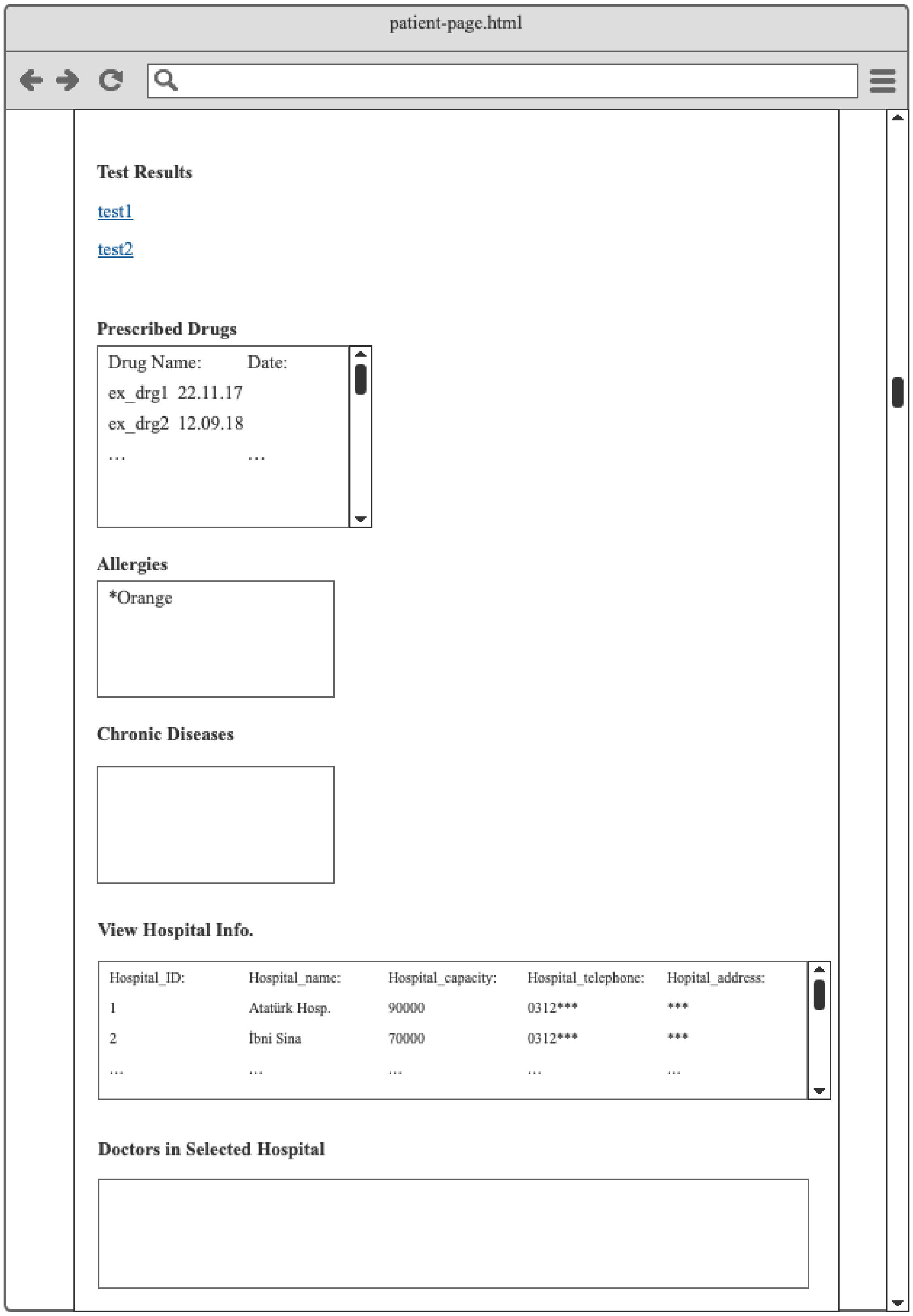
View Prescription History

Prescription_ID:	Prescription_date:
1	11.11.18
2	18.11.18
…	…

View Treatment History

Treatment_ID:	Treatment_description:	Treatment_date:
1	painkiller	11.11.18
2	painkiller	18.11.18
…	…	…

Figure 22: Patient Page



Test Results

test1

test2

Prescribed Drugs

Drug Name:	Date:
ex_drg1	22.11.17
ex_drg2	12.09.18
…	…

Allergies

*Orange

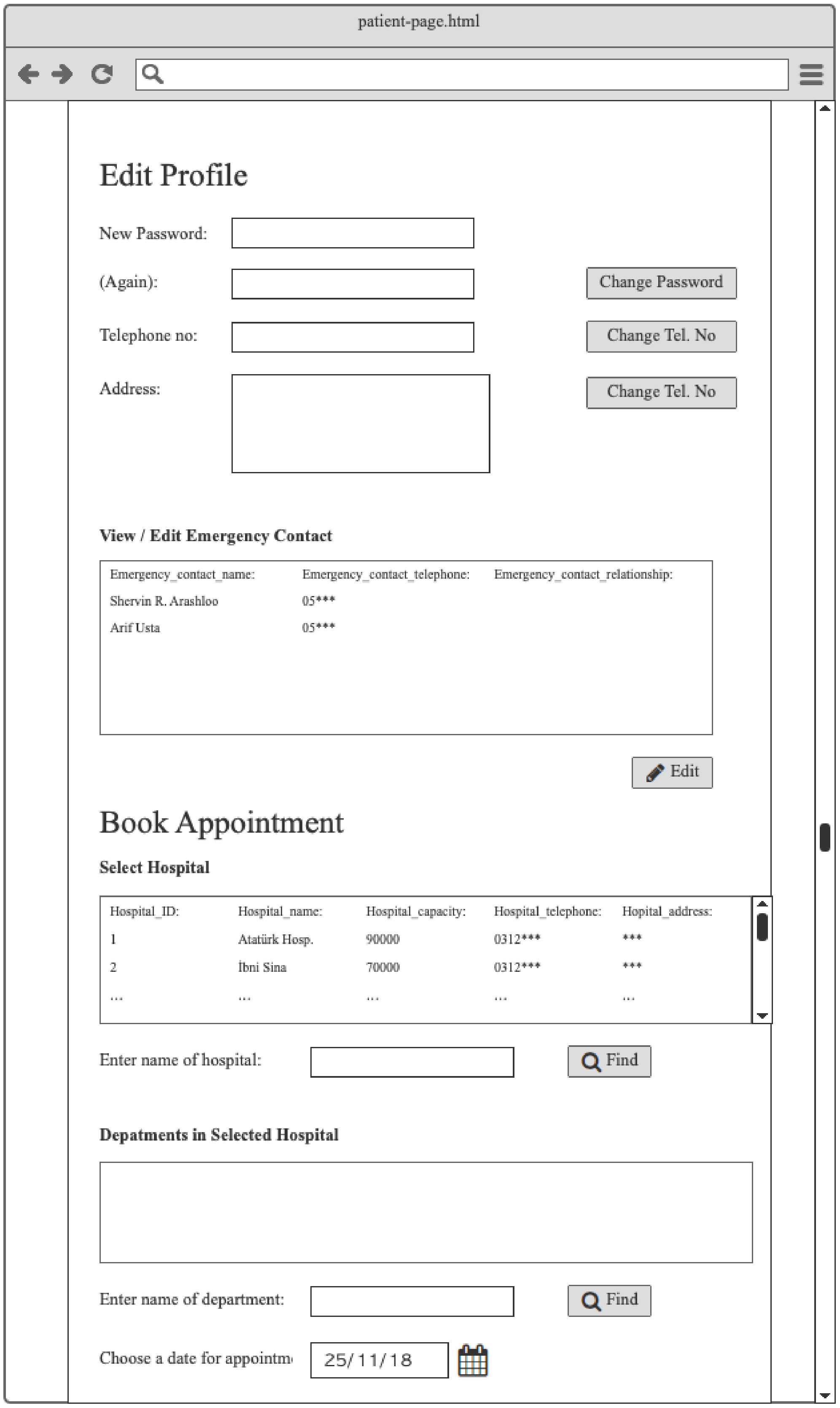
Chronic Diseases

View Hospital Info.

Hospital_ID:	Hospital_name:	Hospital_capacity:	Hospital_telephone:	Hopital_address:
1	Atatürk Hosp.	90000	0312***	***
2	İbni Sina	70000	0312***	***
…	…	…	…	…

Doctors in Selected Hospital

Figure 23: Patient Page continued



Edit Profile

New Password:

(Again):

Change Password

Telephone no:

Change Tel. No

Address:

Change Tel. No

View / Edit Emergency Contact

Emergency_contact_name:	Emergency_contact_telephone:	Emergency_contact_relationship:
Shervin R. Arashloo	05***	
Arif Usta	05***	

Edit

Book Appointment

Select Hospital

Hospital_ID:	Hospital_name:	Hospital_capacity:	Hospital_telephone:	Hopital_address:
1	Atatürk Hosp.	90000	0312***	***
2	İbni Sina	70000	0312***	***
…	…	…	…	…

Enter name of hospital:

Find

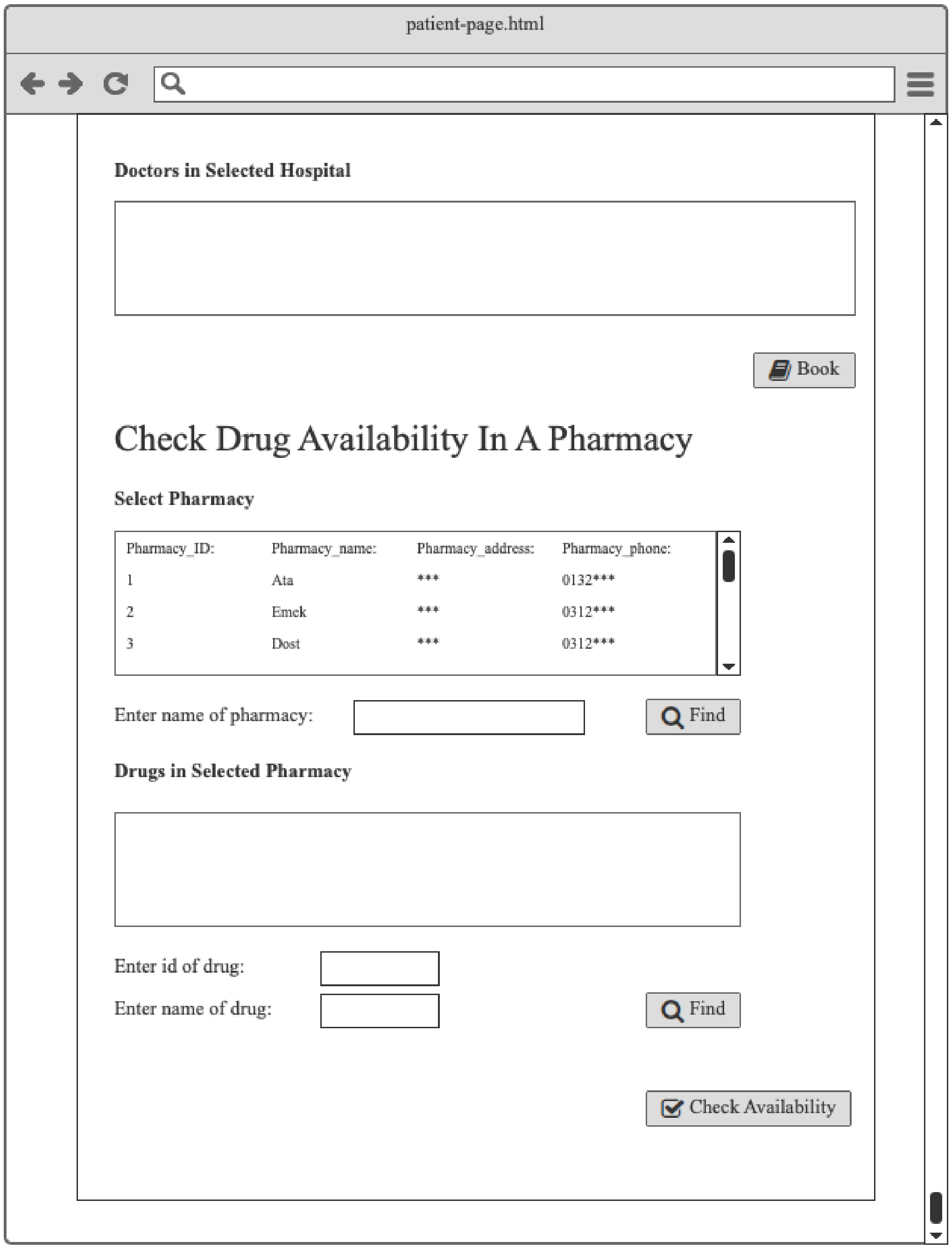
Depatments in Selected Hospital

Enter name of department:

Find

Choose a date for appointm 25/11/18

Figure 24: Patient Page continued


patient-page.html
Doctors in Selected Hospital
Book
Check Drug Availability In A Pharmacy
Select Pharmacy

Pharmacy_ID:	Pharmacy_name:	Pharmacy_address:	Pharmacy_phone:
1	Ata	***	0132***
2	Emek	***	0312***
3	Dost	***	0312***

Figure 25: Patient Page continued

Show Vaccine History

```
SELECT vaccinates.vaccine_name, vaccinates.date
FROM vaccinates NATURAL JOIN user
WHERE user.state_ID = @user_id;
```

Show Examination History

```
SELECT examination_ID, examination_cause, examination_date, examination_diagnose
FROM examination NATURAL JOIN user
WHERE  user.state_ID = @user_id
```

Show Prescription History

```
SELECT prescription_ID, prescription_date
FROM prescription NATURAL JOIN user
WHERE user.state_ID = @user_id;
```

Show Treatment History

```
SELECT treatment_ID, treatment_description, treatment_date
FROM treatment NATURAL JOIN user
WHERE user.state_ID = @user_id;
```

Show Patient's Allergies

```
SELECT patient.allergies
FROM patient NATURAL JOIN user
WHERE patient.state_ID = @state_id;
```

Show Patient's Chronic Disease

```
SELECT patient.chronic_disease
FROM patient NATURAL JOIN user
WHERE patient.state_ID = @state_id;
```

View Hospitals

```
SELECT hospital_id, hospital_name, hospital_capacity, hospital_telephone, hospital_address
FROM hospital;
```

View Doctors in Selected Hospital

```
SELECT first_name, middle_name, last_name,sex,phone,password
FROM user
WHERE user.state_ID in  (SELECT state_ID ,
                          FROM workAsDoctor
                          WHERE workAsDoctor.hospital_ID = hospital_ID) ;
```

Edit Password

```
UPDATE user
SET user.password  = @password
WHERE  user.state_id = @state_id;
```

Edit Telephone

```
UPDATE user
SET user.telephone = @telephone
WHERE  user.state_id = @state_id;
```

Edit Address

```
UPDATE user
SET user.address = @address
WHERE  user.state_id = @state_id;
```

Change Emergency Contact

```
UPDATE emergency_contact
```

```
SET emergency_contact_name = @emergency_contact_name, emergency_contact_telephone =
@emergency_contact_telephone, emergency_contact_relationship =
@emergency_contact_relationship
WHERE state_id = @state_id;
```

Book Appointment

```
INSERT INTO book
VALUES (@patient_id, @examination_ID, @doctor_id);
WHERE state_id = @state_id;
```

Check Availability of Drug

```
SELECT drug_id, drug_name
FROM drug NATURAL JOIN pharmacy
WHERE pharmacy.id in  (SELECT pharmacy_id
                       FROM store
                       WHERE number_in_stock > 0) ;
```

Give Rating To Examination

```
INSERT INTO rating
VALUES (NULL, @score, @comment)
```

6. Advanced Database Components

6.1 Secondary Indexes

Drugs can be searched with their amounts on a pharmacy for filtering.

```
CREATE INDEX number_in_stock_index USING BTREE ON stores (number_in_stock);
```

6.2 Views

6.2.1 Patient Age

This view will be used to get age of the users from their date of birth. Age was an deprived attribute in E/R diagram so it should be represented as a view.

```
CREATE VIEW patient_age as
SELECT state_ID, TIMESTAMPDIFF (YEAR, patient_date_of_birth,CURDATE()) AS age
FROM patient;
```

6.3 Stored Procedures

Some of our queries such as queries for listing doctors or patient information can be written as an stored procedure since these queries will be executed many times by many users. Also stored procedures could be used to hide the internal information.

Stored procedure will also be used to add multiple rows of drugs to prescribed relation. Since we enable doctors to add multiple drugs to prescription and submit the prescription as whole, a stored procedure can add multiple tuples in batches.

6.4 Reports

6.4.1 Total Number of Examinations Annually for Each Hospital

This report will be used to calculate the number of examinations that are done in the last 7 days of a hospital.

```
SELECT works_as_doctor.hospital_ID, count(examination_done.examination_ID) as
examination_numbers
FROM (works_as_doctor inner join examination_done on works_as_doctor.state_ID =
examination_done.doctor_state_ID) inner join examination on
examination_done.examination_ID = examination.examination_ID
WHERE examination.examination_date >= DATE(NOW()) - INTERVAL 365 DAY
GROUP BY works_as_doctor.hospital_ID
```

6.4.2 Reporting Drug_IDs' for between desired amounts in the system

This report shows drug_IDs' between desired amounts in the system.

```
SELECT `stores`.`drug_ID`, SUM(`stores`.`number_in_stock`) AS SUM_stores_number_in_s +
toc
FROM `cagatay_sel`.`stores` `stores`
WHERE ('#39'c'#39' <> '#39'c'#39' )
GROUP BY `stores`.`drug_ID
```

6.5 Triggers and Constraints

- We thought about implement them in the design report but we didn't have time to implement them.

7. Implementation Details

In our project implementation, we have used MySQL for database system. PHP was used for web application development in the server side. MDBootsrap, HTML5, CSS3x and Javascript was used for user interface development and designing.

8. User Manual

8.1 User Manual of Login and Register Pages

In login page users are waited to enter their state id and password with their intention to enter the system as patient, doctor or pharmacists. Also, User can get in to register page which they have to provide state id and a password also they can provide personal information by new page called information page. After, users fill the information page they are transferred to login page again.

If it is users first time, doctors are sent to doctor waiting page for waiting confirmation from their executive doctor. If they are the new executive doctor they may create new hospital. At first new executive doctor should fill his identification and role in the hospital then they will be sent to hospital creation page. For pharmacists, they will wait on pharmacistwait page until co-worker adds new pharmacist. Or, new pharmacist can create a new pharmacy in this waiting page. New patient should fill patient form for the first time when they enter the system and they will be sent to their profile page.

8.2 User Manual of Patients' Page

In Patient's page, at the top, one can see his/her some informations which are state id, first name, middle name, last name and age. If doctor wants to peek patient's page, these informations are belong to patient.

In this page, generally, there are some tables which contains some medical informations like which treatments are applied, what are diagnose of these treatments, one's chronic diseases and allergies etc.

In addition to these informations, one can also edit his/her profile in this page. For example, if one changes his/her phone, it can be edited in this page.

Emergency contacts are also edited in this page. One can add new emergency contacts and remove the existing ones. To add new emergency contact, this person's name, relationship with patient and telephone number.

If one prefers, drugs can be bought from this page. In order to buy drug online, firstly, one needs to choose pharmacy and then drug id.

8.3 User Manual of Doctors’ Page

In doctor’s profile page, user is displayed with his/her information. Doctor’s then see their hospital information. In hospital information section doctor’s could add departments or new doctors to their hospital by using the input fields.

In next section, doctor’s could get patient information by providing the state id of patient. With this state id, they can add new examination. After addition of examination, they can add test,treatment and prescription to this examination.

8.4 User Manual for Pharmacists' Page

After logging in as pharmacist, you will see 7 different sections if you are working on a pharmacy as follows:

- Your Pharmacy's Information

In this section, you will see your pharmacy's information.

- Manage Pharmacists in this Pharmacy

In this section, you will see pharmacists working in the same pharmacy as you including you. In the table you can click on "Remove from Pharmacy" to fire a pharmacist from your pharmacy, anyone that works in this pharmacy can do this. By entering a new pharmacist's state ID in the text field near the button named "Add New Pharmacist" and clicking on that button, you will add hire a new pharmacist to your pharmacy.

- Manage Pharmacy's Drugs

In this section, you can add or remove an amount of a drug from your pharmacy's drug. If you want to remove a drug completely, you have to go to the bottom of this section and give the drug's ID to remove. You can search between 2 different amounts for drugs that are in the pharmacy's stock. You can also add a new drug to the pharmacy's stock by giving its Drug-ID and Amount, the desired drug should be available in the system which can be searched by another section in this page.

- Edit Your Pharmacy

In this section, you can edit your pharmacy's information except its ID.

- Manage Drugs in the System

In this section, you can see the drugs that are registered to the system. If you want to add a whole new drug to your pharmacy's stock, first you have to register it to the system from this section.

- Manage Vaccines in the System

In this section, you can see the vaccines that are registered to the system. You can also register new vaccines to the system.

- Manage Alternatives Drugs in the System

In this section, you can add alternative drugs to another drug. If you want to add a whole new alternative drug for a existing drug, you need to add it to the system from the above section first.

- Logout

On the top left of this page, there is a button for logging out from the system. This returns user to the login page.