Bilkent University
Department of Computer Engineering

CS 353

Database Management Systems
Term Project

Design Report
Patient Medical Treatment Tracking System

Group 7:
Kasymbek Tashbaev 21603136
Babanazar Gutlygeldiyev 21503402
Alina Zhumasheva 21503347
Islomiddin Sodiqov 21503402

Teaching assistant: **Arif Usta**

1. BRIEF DESCRIPTION

PMTTS is a web-based system application for tracking medical treatments of patients at certain hospitals. The system will be designed to be used by patients, doctors of different hospitals and pharmacists. System includes information about the hospitals and doctors working there. Doctors will be able to set or change hospitals where they work, set their schedule of working hours and available slots for appointments. Patients can book or cancel an appointment from a certain doctor. After the appointment, patients can view their diagnosis and a list of prescribed drugs. Patients then can buy drugs from a pharmacist, and if the drug is not available, patient will be able to buy similar drugs having same ingredients. The security of payment and maintenance of the data are very crucial factors in this database system so that the users will not get into any unwanted situations.
Shortly, **PMTTS** will be a web-application system that will help to ease maintaining the interactions between patients, doctors and pharmacists.

2. REVISED E/R DIAGRAM

The following changes were made based on feedback from TA and on our decision to improve our diagram.

2.1. Added

Following elements, which are bolded, were added to E/R diagram

1. **User** entity with **username, password, name, image, phone, birthday** and **gender** attributes; it is general entity for Patient, Pharmacist and Doctor.
2. **Address** entity with **add_id, country, city, street, apartment, apartment_num, xLoc** and **yLoc** attributes; it has **Lives** relation with User entity and **Located** relation with Hospital entity.
3. **Has** relation between Pharmacy and Drug entities with attribute **stock.**
4. **Transaction** entity with **trans_id, total_price, date, time, status** attributes; it has **Contains** relation with Drug and Pharmacist entities and **Pays-for** relation with Patient entity; **Contains** relation has attribute **amount**.

5. **Test** entity with attributes **test_id, name**; it has **Does** relation with Doctor and Appointment entities.
6. **Symptom** entity with attributes **sympt_name, type and description**, it has **Asks-for** relation with Appointment entity.
7. **Diagnosis** weak entity with attribute **diagnosis_id;** it has **Shows** relation with Disease, **Results** relation with Appointment and **Prescribes** relation with Drug; **Prescribes** relation has attribute **description**.

2.2. Deleted

Following elements, which are bolded, were removed from E/R diagram.

1. **Bill** entity with its relations was removed**,** since it was unimportant for our application.
2. **Payment** entity with its relations was removed, since it was unimportant for our application.
3. **Supplies** relationship between Drug, Patient and Pharmacist was removed, since it was not properly expressing the function of our application.
4. **Diagnoses & Treats** relation between Patient, Doctor and Disease was removed, since it was not properly expressing the function of our application.

2.3. Modified

1. Attributes of Patient, Pharmacist, Doctor, Hospital, Drug, Disease and Appointment were changed.
2. Wrong Mapping Cardinality Constraints were fixed.

2.4. Revised E/R Diagram

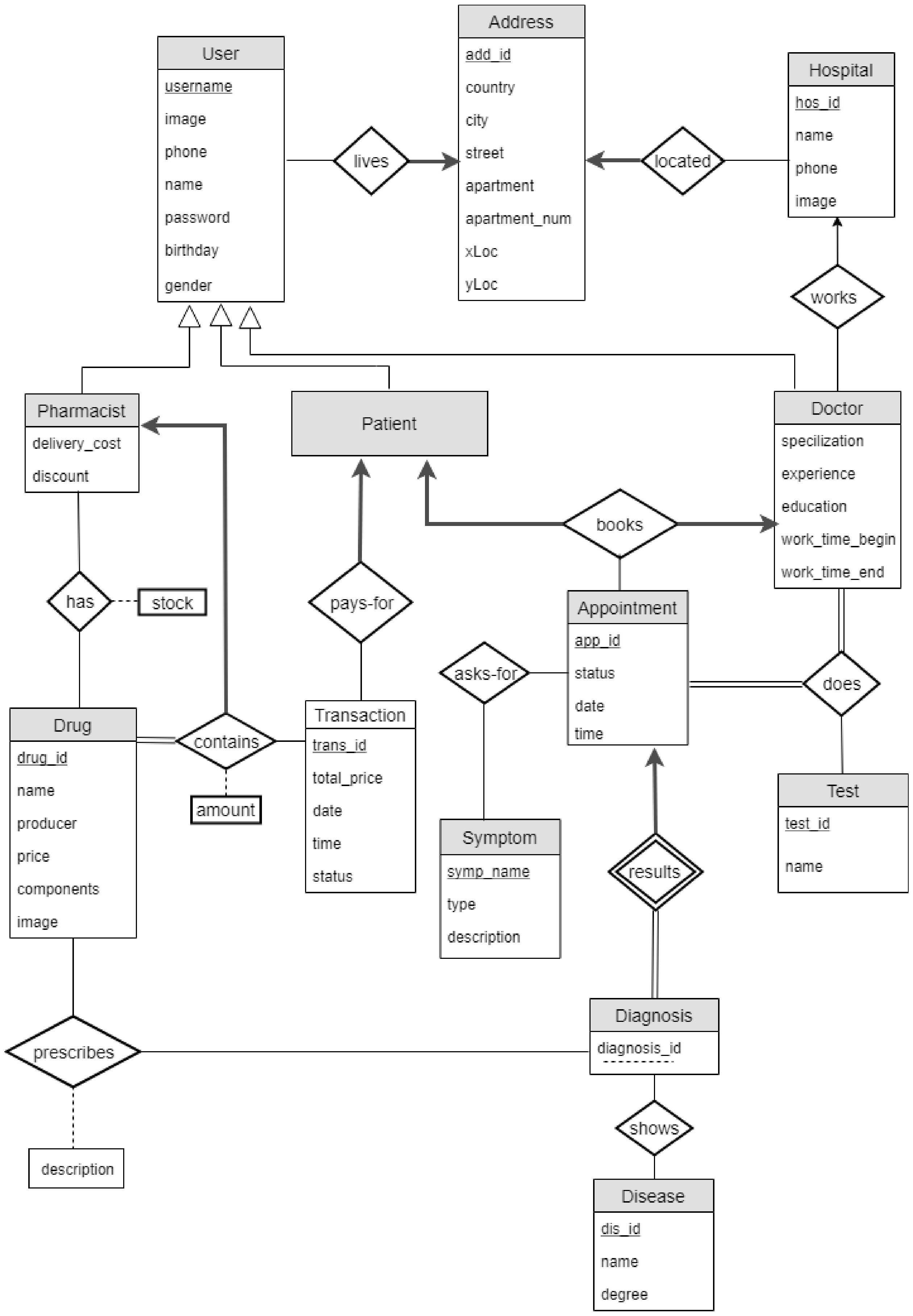
Legends:

→ : One Optional (0..1)

➡ : One Mandatory (1)

— : Many Optional (0..*)

═ : Many Mandatory (1..*)



User
username
image
phone
name
password
birthday
gender
lives
Address
add_id
country
city
street
apartment
apartment_num
xLoc
yLoc
located
Hospital
hos_id
name
phone
image
works
Pharmacist
delivery_cost
discount
Patient
Doctor
specilization
experience
education
work_time_begin
work_time_end
books
has
stock
pays-for
Appointment
app_id
status
date
time
asks-for
does
Drug
drug_id
name
producer
price
components
image
contains
amount
Transaction
trans_id
total_price
date
time
status
Symptom
symp_name
type
description
results
Test
test_id
name
prescribes
description
Diagnosis
diagnosis_id
shows
Disease
dis_id
name
degree

3. RELATIONAL SCHEMA

3.1. User

Relational Model

User(<u>username</u>, image, phone, name, password, birthday, gender, add_id)

Functional Dependencies

username $\rightarrow$ image, phone, name, password, birthday, gender, add_id

Candidate Keys

{(username)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE User (
        username      varchar(25) NOT NULL,
        image         varbinary(max),
        phone         varchar(20),
        name          varchar(20) NOT NULL,
        password      varchar(25) NOT NULL,
        birthday      date NOT NULL,
        gender        varchar(20),
        add_id        int NOT NULL,
        FOREIGN KEY(add_id) REFERENCES Address(add_id),
        UNIQUE (username)
);
```

3.2. Pharmacist

Relational Model

Pharmacist(username, delivery_cost, discount)

Functional Dependencies

username $\rightarrow$ delivery_cost, discount

Candidate Keys

{(username)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Pharmacist (
        username        varchar(25) NOT NULL,
        delivery_cost   float(2) NOT NULL DEFAULT 0,
        discount        int DEFAULT 0,
        FOREIGN KEY (username) REFERENCES User(username)
);
```

3.3. Patient

Relational Model

Patient(<u>username</u>)

Functional Dependencies

No dependencies

Candidate Keys

{(username)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Patient (
        username        varchar(25) NOT NULL,
        FOREIGN KEY (username) REFERENCES User(username)
);
```

3.4. Doctor

Relational Model

Doctor(<u>username</u>, specialization, experience, education, work_time_begin, work_time_end, hos_id)

Functional Dependencies

username → specialization, experience, education, work_time_begin, work_time_end, hos_id

Candidate Keys

{(username)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Doctor (
        username            varchar(25) NOT NULL,
        specialization      varchar(20) NOT NULL,
        experience          int DEFAULT 0,
        education           varchar(25) NOT NULL,
        work_time_begin     time NOT NULL,
        work_time_end       time NOT NULL,
        hos_id              int,
        FOREIGN KEY (username) REFERENCES User(username)
        FOREIGN KEY (hos_id) REFERENCES Hospital(hos_id)
);
```

3.5. Hospital

Relational Model

Hospital(<u>hos_id</u>, name, phone, image, add_id)

Functional Dependencies

hos_id $\rightarrow$ name, phone, image, add_id

Candidate Keys

{(hos_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Hospital (
        hos_id          int NOT NULL PRIMARY KEY AUTO_INCREMENT,
        name            varchar(20) NOT NULL,
        phone           varchar(20),
        image           varbinary(max),
        add_id          int NOT NULL,
        FOREIGN KEY(add_id) REFERENCES Address(add_id),
);
```

3.6. Appointment

Relational Model

Appointment(app_id, status, date, time, patient_username, doctor_username)

Functional Dependencies

app_id → status, date, time, patient_username, doctor_username

Candidate Keys

{(app_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Appointment (
        app_id              int NOT NULL PRIMARY KEY AUTO_INCREMENT
        status              varchar(25) NOT NULL,
        date                date NOT NULL,
        time                time NOT NULL,
        patient_username    varchar(25) NOT NULL,
        doctor_username     varchar(25) NOT NULL,
        FOREIGN KEY (patient_username) REFERENCES Patient(username)
        FOREIGN KEY (doctor_username) REFERENCES Doctor(username)
);
```

3.7. Test

Relational Model

Test(<u>test_id</u>, name)

Functional Dependencies

test_id $\rightarrow$ name

Candidate Keys

{(test_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Test (
        test_id                int NOT NULL PRIMARY KEY AUTO_INCREMENT,
        name                   varchar(25)
);
```

3.8. Symptom

Relational Model

Symptom(<u>symp_name</u>, type, description)

Functional Dependencies

symp_name $\rightarrow$ type, description

Candidate Keys

{(symp_name)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Symptom (
        symp_name     varchar(25) NOT NULL PRIMARY KEY,
        type          varchar(25) NOT NULL,
        description   varchar(25)
);
```

3.9. Disease

Relational Model

Disease(<u>dis_id</u>, name, degree)

Functional Dependencies

dis_id $\rightarrow$ name, degree

Candidate Keys

{(dis_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Disease (
        dis_id          int NOT NULL PRIMARY KEY AUTO_INCREMENT,
        name            varchar(20) NOT NULL,
        degree          varchar(20)
);
```

3.10. Diagnosis

Relational Model

Diagnosis(diagnosis_id, app_id)

Functional Dependencies

No dependencies

Candidate Keys

{(diagnosis_id, app_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Diagnosis (
        diagnosis_id    int AUTO_INCREMENT,
        app_id          int NOT NULL,
        FOREIGN KEY(app_id) REFERENCES Appointment(app_id)
);
```

3.11. Drug

Relational Model

Drug(drug_id, name, producer, price, components, image)

Functional Dependencies

drug_id $\rightarrow$ name, producer, price, components, image

Candidate Keys

{(drug_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Drug (
        drud_id         int NOT NULL PRIMARY KEY AUTO_INCREMENT,
        name            varchar(20) NOT NULL,
        producer        varchar(25) NOT NULL,
        price           float(2) NOT NULL DEFAULT 0,
        components      varchar[] DEFAULT '{}',
        image           varbinary(max)
);
```

3.12. Transaction

Relational Model

Transaction(trans_id, total_price, date, time, status, patient_username)

Functional Dependencies

trans_id $\rightarrow$ total_price, date, time, status, patient_username

Candidate Keys

{(trans_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Transaction (
        trans_id            int NOT NULL PRIMARY KEY AUTO_INCREMENT,
        total_price         float(2) NOT NULL DEFAULT 0,
        date                date NOT NULL,
        time                time NOT NULL,
        status              varchar(25),
        patient_username    varchar(25) NOT NULL,
        FOREIGN KEY(patient_username) REFERENCES Patient(username)
);
```

3.13. Address

Relational Model

Address(add_id, country, city, street, apartment, apartment_num, xLoc, yLoc)

Functional Dependencies

add_id → country, city, street, apartment, apartment_num, xLoc, yLoc

Candidate Keys

{(add_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Address (
        add_id              int NOT NULL PRIMARY KEY AUTO_INCREMENT,
        country             varchar(20) NOT NULL,
        city                varchar(20) NOT NULL,
        street              varchar(25) NOT NULL,
        apartment           varchar(20),
        apartment_num       int,
        xLoc                int NOT NULL,
        yLoc                int NOT NULL
);
```

3.14. Doctor_Test_Appointment

Relational Model

Does(<u>test_id, app_id, doctor_username</u>)

Functional Dependencies

No dependencies

Candidate Keys

{(test_id, app_id, doctor_username)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Does (
        test_id             int NOT NULL,
        app_id              int NOT NULL,
        doctor_username     varchar(25) NOT NULL,
        PRIMARY KEY(test_id, app_id, doctor_username),
        FOREIGN KEY(test_id) REFERENCES Test(test_id),
        FOREIGN KEY(app_id) REFERENCES Appointment(app_id),
        FOREIGN KEY(doctor_username) REFERENCES Doctor(username)
);
```

3.15. Pharmacist_Drug

Relational Model

Has(<u>pharmacist_username, drug_id</u>, stock)

Functional Dependencies

pharmacist_username, drug_id $\rightarrow$ stock

Candidate Keys

{(pharmacist_username, drug_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Has (
        pharmacist_username         varchar(25) NOT NULL,
        drug_id                     int NOT NULL,
        stock                       int NOT NULL,
        PRIMARY KEY(pharmacist_username, drug_id),
        FOREIGN KEY(pharmacist_username) REFERENCES Pharmacist(username),
        FOREIGN KEY(drug_id) REFERENCES Drug(drug_id)
);
```

3.16. Appointment_Symptom

Relational Model

Asks-for(symp_name, app_id)

Functional Dependencies

No Dependencies

Candidate Keys

{(symp_name, app_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Asks-for (
        symp_name          varchar(25) NOT NULL,
        app_id             int NOT NULL,
        PRIMARY KEY(symp_name, app_id),
        FOREIGN KEY(symp_name) REFERENCES Symptom(symp_name),
        FOREIGN KEY(app_id) REFERENCES Appointment(app_id)
);
```

3.17. Drug_Diagnosis

Relational Model

Prescribes(app_id, diagnosis_id, drug_id, description)

Functional Dependencies

app_id, diagnosis_id, drug_id $\rightarrow$ description

Candidate Keys

{(app_id, diagnosis_id, drug_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Prescribes (
        app_id          int NOT NULL,
        diagnosis_id    int NOT NULL,
        drug_id         int NOT NULL,
        description     varchar(25),
        PRIMARY KEY(app_id, diagnosis_id, drug_id),
        FOREIGN KEY(app_id) REFERENCES Appointment(app_id),
        FOREIGN KEY(diagnosis_id) REFERENCES Diagnosis(diagnosis_id),
        FOREIGN KEY(drug_id) REFERENCES Drug(drug_id)
);
```

3.18. Pharmacist_Drug_Transaction

Relational Model

Contains(<u>trans_id</u>, <u>pharmacist_username</u>, <u>drug_id</u>, amount)

Functional Dependencies

trans_id, pharmacist_username, drug_id $\rightarrow$ amount

Candidate Keys

{(trans_id, pharmacist_username, drug_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Contains (
        trans_id                    int NOT NULL,
        pharmacist_username         varchar(25) NOT NULL,
        drug_id                     int NOT NULL,
        amount                      int NOT NULL,
        PRIMARY KEY(trans_id, pharmacist_username, drug_id),
        FOREIGN KEY(trans_id) REFERENCES Transaction(trans_id),
        FOREIGN KEY(pharmacist_username) REFERENCES Pharmacist(username),
        FOREIGN KEY(drug_id) REFERENCES Drug(drug_id)
);
```

3.19. Diagnosis_Disease

Relational Model

Shows(<u>diagnosis_id</u>, <u>disease_id</u>)

Functional Dependencies

No Dependencies

Candidate Keys

{(diagnosis_id, disease_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE Shows (
        diagnosis_id            int NOT NULL,
        disease_id              int NOT NULL,
        PRIMARY KEY(diagnosis_id, disease_id),
        FOREIGN KEY(diagnosis_id) REFERENCES Diagnosis(diagnosis_id),
        FOREIGN KEY(disease_id) REFERENCES Disease(disease_id)
);
```

4. FUNCTIONAL COMPONENTS

4.1. Use Cases

Use Case Diagram for **PMTTS** is provided below. It provides multiple use cases for a superactor User, and Patient, Doctor and Pharmacist subactors.

User:

- User can switch between his accounts (User can log in as a patient, doctor or pharmacist).
- User can edit his/her profile information(address, phone number, etc.).
- User can change settings of his/her account.
- User can view information about the developers of the system.

Patient:

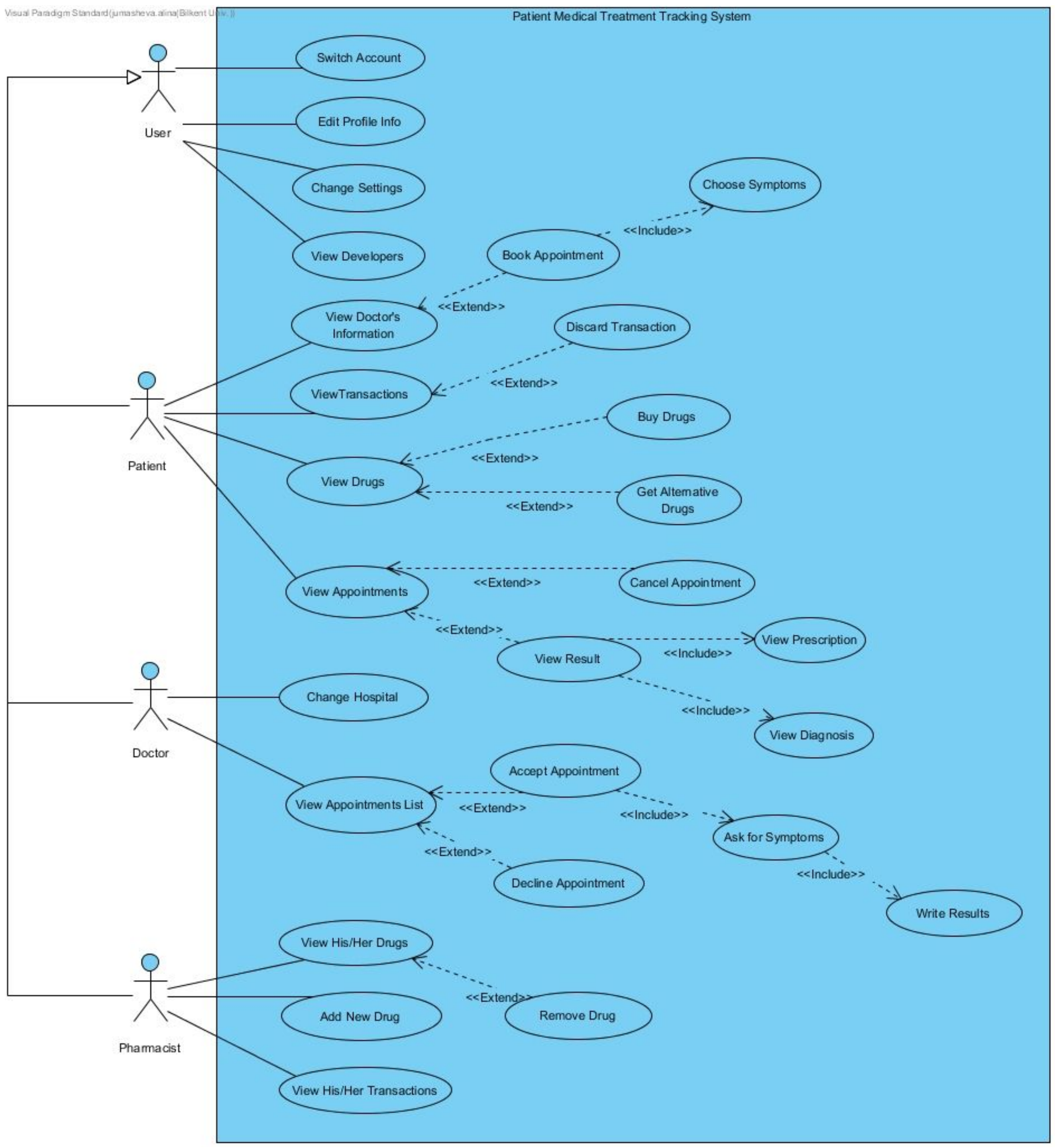
- Patient can view information about doctors and book an appointment from a particular doctor along with providing information about symptoms he/she has.
- Patient can view his/her transactions and discard transactions that he/she wants to.
- Patient can view his/her appointments list and view results of his appointments or cancel certain appointment.
- Patient can view drugs which he/she can buy and buy certain drugs or get alternative drugs.

Doctor:

- Doctor can change hospital he/she is working at.
- Doctor can view appointments and accept/decline them. If doctor accepts appointment, then he/she will ask patient to provide symptoms and then doctor can write results of the appointment.

Pharmacist:

- Pharmacist can view drugs that he/she has at the moment and remove certain drugs from the list.
- Pharmacist can add new drugs to his/her drugs list.
- Pharmacist can view information about his/her transactions.


Visual Paradigm Standard(jumasheva.alina(Bilkent Univ.))
Patient Medical Treatment Tracking System
User
Patient
Doctor
Pharmacist
Switch Account
Edit Profile Info
Change Settings
View Developers
Choose Symptoms
<<Include>>
Book Appointment
<<Extend>>
View Doctor's Information
Discard Transaction
ViewTransactions
<<Extend>>
Buy Drugs
<<Extend>>
View Drugs
Get Alternative Drugs
<<Extend>>
View Appointments
<<Extend>>
Cancel Appointment
<<Extend>>
View Result
View Prescription
<<Include>>
<<Include>>
View Diagnosis
Change Hospital
Accept Appointment
View Appointments List
<<Extend>>
<<Include>>
Ask for Symptoms
<<Extend>>
Decline Appointment
<<Include>>
Write Results
View His/Her Drugs
<<Extend>>
Remove Drug
Add New Drug
View His/Her Transactions

4.2. Algorithms

Transaction calculation algorithm

To calculate the full amount of payment for a transaction for a patient to buy drugs, the amount of drugs will be multiplied by its price and reduced according the discount pharmacist has. In addition, according to x-y location of addresses of patient and pharmacist distance will be calculated, and distance times delivery cost will be added to total price of transaction.

Alternative drugs detection algorithm

In order to detect alternative drugs for a certain drug, this drug will be compared with other drugs in the database and if at least one component of the first drug exist in a components of another drug, second drug will be shown as an alternative drug which patient can buy.

Appointment date and time algorithm

When patient views information about a particular doctor, there will be a list of appointments already booked appointments by other patients. If the patient wishes to book an appointment from that doctor, he/she will write in the date in the date field and for time he/she will be able to choose only between time slots from the drop-down list of available time slots for that date that he/she entered. The time slots will be in a period between work_start_time and work_end_time of that doctor and occupied slots for that date will be excluded from the list.

Checking for prescription before buying drug

In our system, user can buy a drug if and only if that drug is prescribed by doctor in the last six month. Therefore, there will be algorithm, which checks whether the patient has the drug, that he wants to buy, in list of prescribed drugs. If the condition is satisfied, patient can buy any amount of drugs.

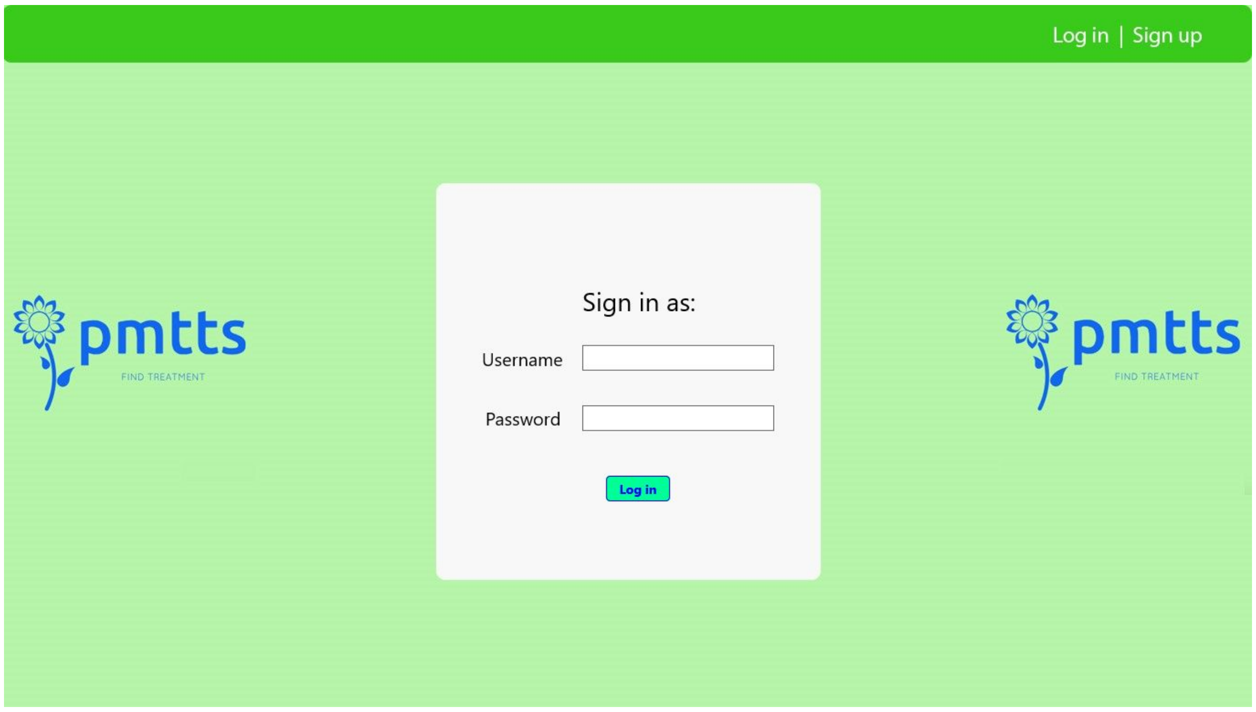
Experience increment algorithm
In our application doctors have experience integer attribute, which in units of year. Therefore, as year passes, experience should be increased as well. There will be an algorithm, that will increment experience by one each year on date when Doctor registered to the system.

4.3. Data Structures

We are going to use Char, Varchar, Date, Time, Int, Float and Varbinary data types of MySQL and array.

5. USER INTERFACE DESIGN

5.1. Sign in page



Inputs: @username, @password

Process: This page allows user to login if username entered contained in database and password entered matches username.

SQL Statements:

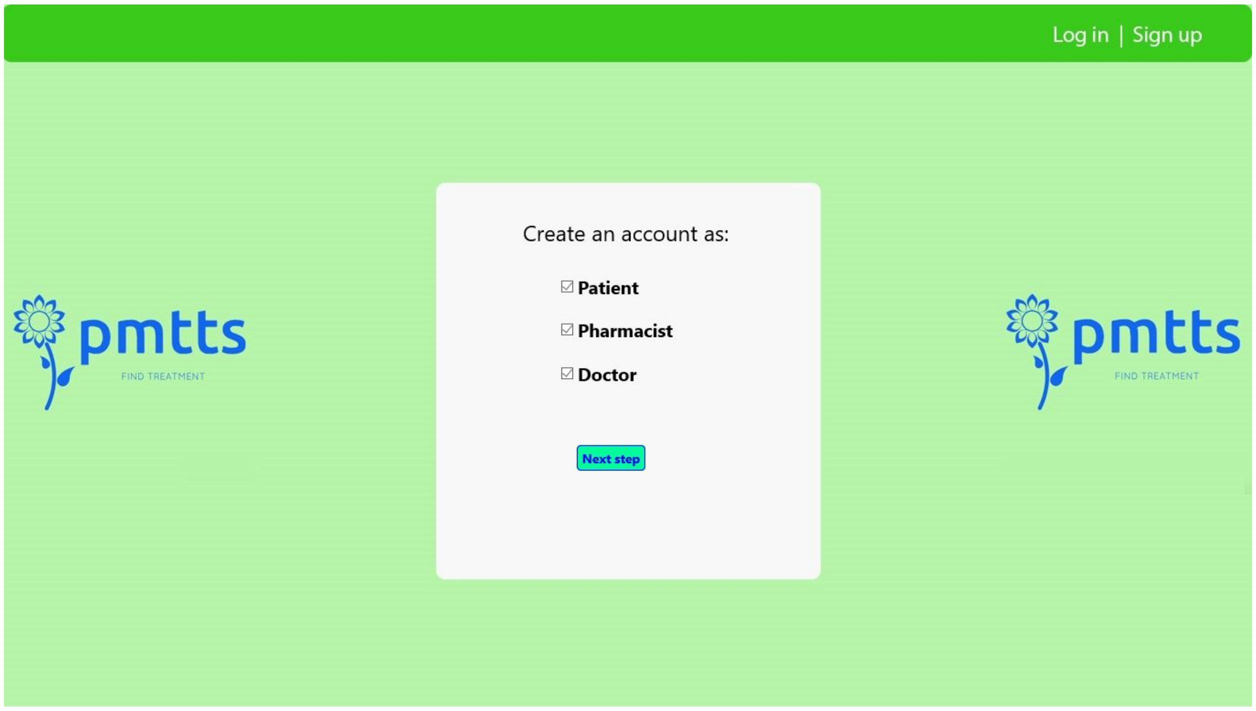
user retrieval

SELECT *

FROM user

WHERE username = @username AND password = @password

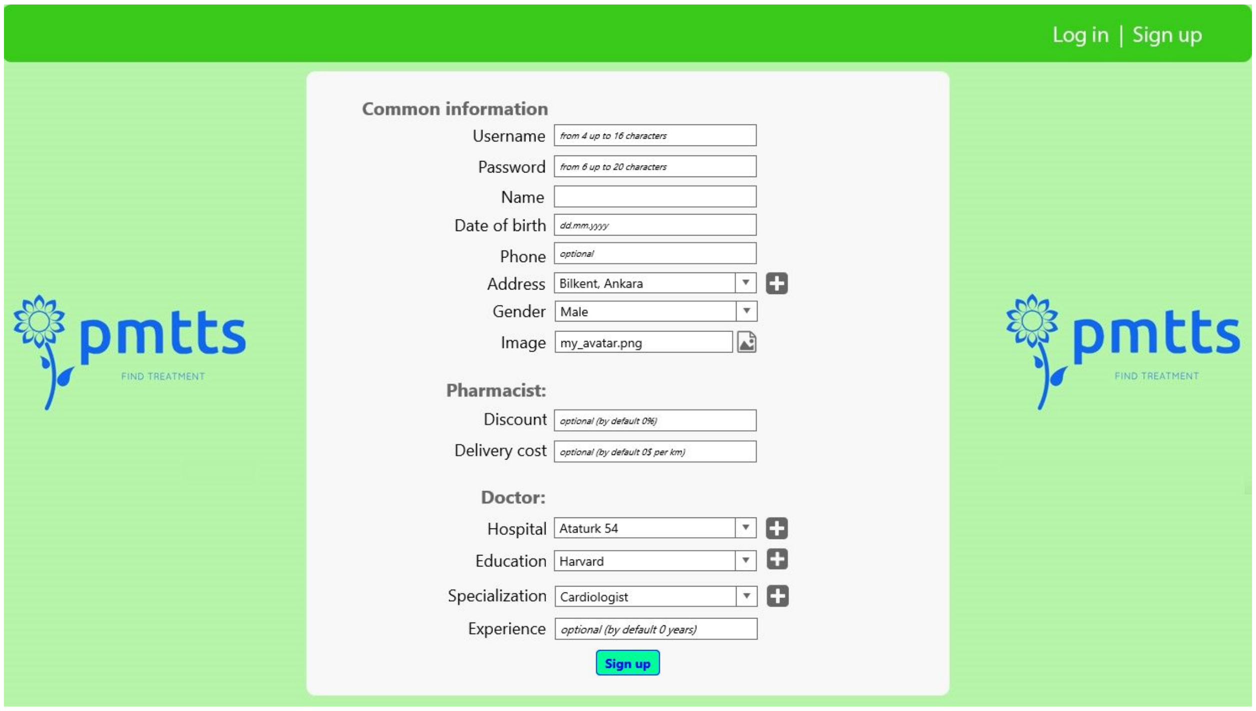
5.2. Sign up page



Create an account as:

- ☑ **Patient**
- ☑ **Pharmacist**
- ☑ **Doctor**

Process: There are three different types of accounts: Patient, Pharmacist, Doctor. In this page user should choose which account he is going to create. He/She can choose more than one type.



Common information

Username	from 4 up to 16 characters
Password	from 6 up to 20 characters
Name	
Date of birth	dd.mm.yyyy
Phone	optional
Address	Bilkent, Ankara
Gender	Male
Image	my_avatar.png

Pharmacist:

Discount	optional (by default 0%)
Delivery cost	optional (by default 0$ per km)

Doctor:

Hospital	Ataturk 54
Education	Harvard
Specialization	Cardiologist
Experience	optional (by default 0 years)

Process: According to the types of account user chosen in the previous page, this page asks him/her to enter user's information, some of them are optional. If user cannot find his/her address, hospital, education or specialization, he/she can enter a new data by clicking on the plus button.

Inputs: @username, @password, @name, @birthday, @phone, @image, @address, @gender, @discount, @delivery_cost, @hospital, @education, @specialization, @experience

SQL Statements:

insertion of inputs

INSERT INTO User

values(@username, @password, @name, @birthday, @phone, @image, @gender, @adress);

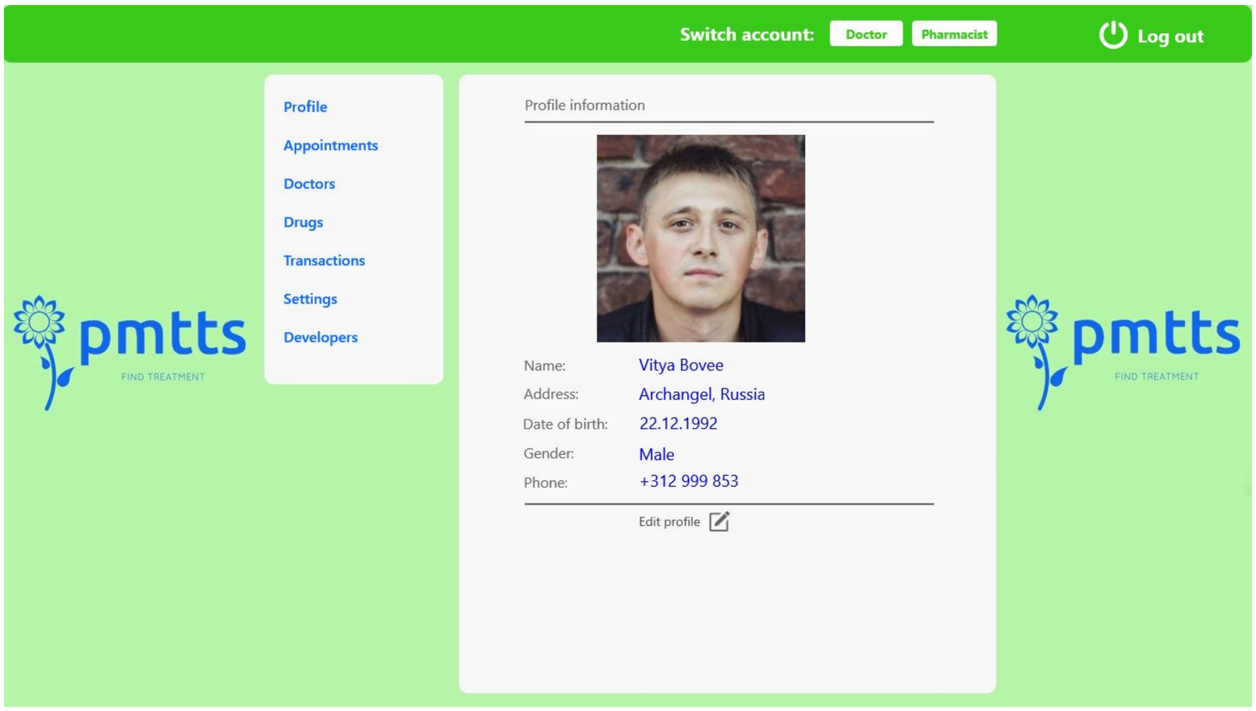
INSERT INTO Pharmacist

values(@username, @discount, @delivery_cost);

INSERT INTO Doctor

values(@username, @hospital, @education, @specialization, @experience, TIME '9:00', TIME '18:00');

5.3. Patient’s Profile menu



Profile information

Name:	Vitya Bovee
Address:	Archangel, Russia
Date of birth:	22.12.1992
Gender:	Male
Phone:	+312 999 853

Process: In this page user can view his/her profile information of patient account. He/She can edit it by clicking edit button. In addition, here he/she can switch his account to Pharmacist or Doctor account. Patient account has Profile, Appointments, Doctors, Drugs, Transactions, Settings and Developers menu.

Inputs: @username

SQL Statements:

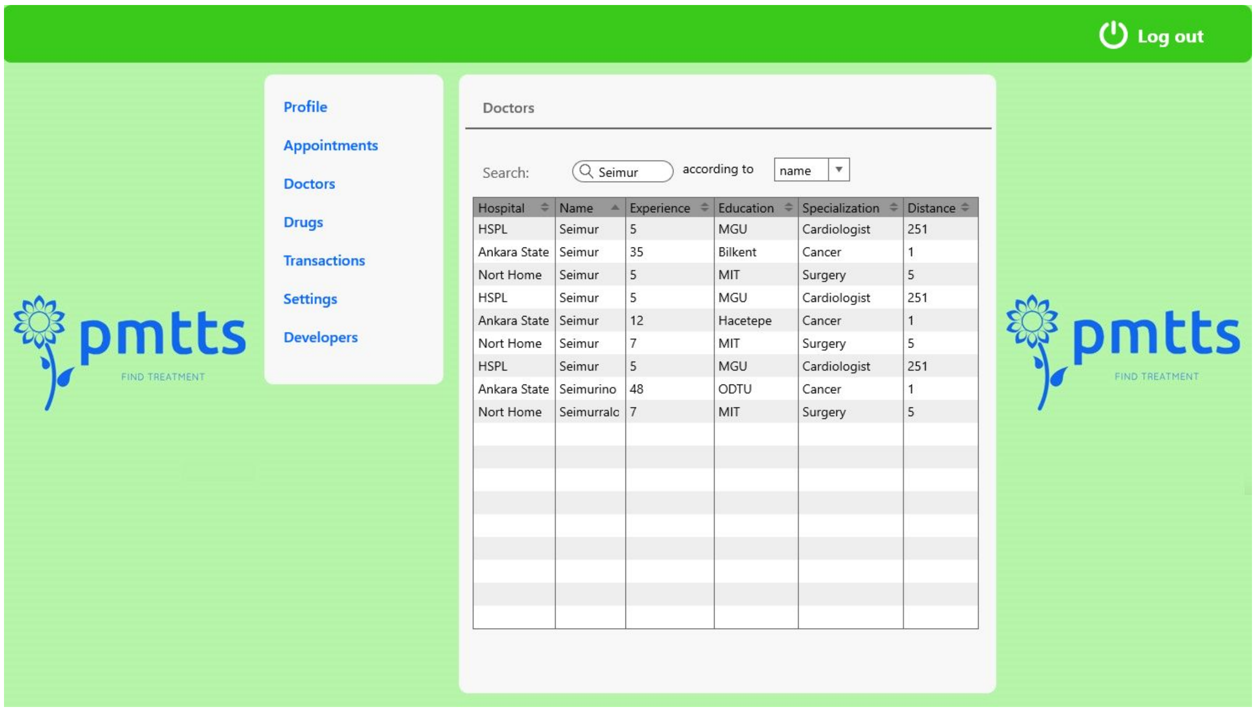
Patient’s information retrieval

SELECT name, birthday, gender, phone, country, city, street, apartment, apartment_num

FROM User join Address using (add_id)

WHERE username = @username

5.4. Patient’s Doctors menu



Doctors

Search: Seimur according to name

Hospital	Name	Experience	Education	Specialization	Distance
HSPL	Seimur	5	MGU	Cardiologist	251
Ankara State	Seimur	35	Bilkent	Cancer	1
Nort Home	Seimur	5	MIT	Surgery	5
HSPL	Seimur	5	MGU	Cardiologist	251
Ankara State	Seimur	12	Hacetepe	Cancer	1
Nort Home	Seimur	7	MIT	Surgery	5
HSPL	Seimur	5	MGU	Cardiologist	251
Ankara State	Seimurino	48	ODTU	Cancer	1
Nort Home	Seimurralo	7	MIT	Surgery	5

Process: In this page user can view information about doctors. He/She can search doctors and get sorted table according the attribute chosen or by distance, which is calculated by algorithm.

Inputs: @patient_username, @search, @according_to, @sort_by

SQL Statements:

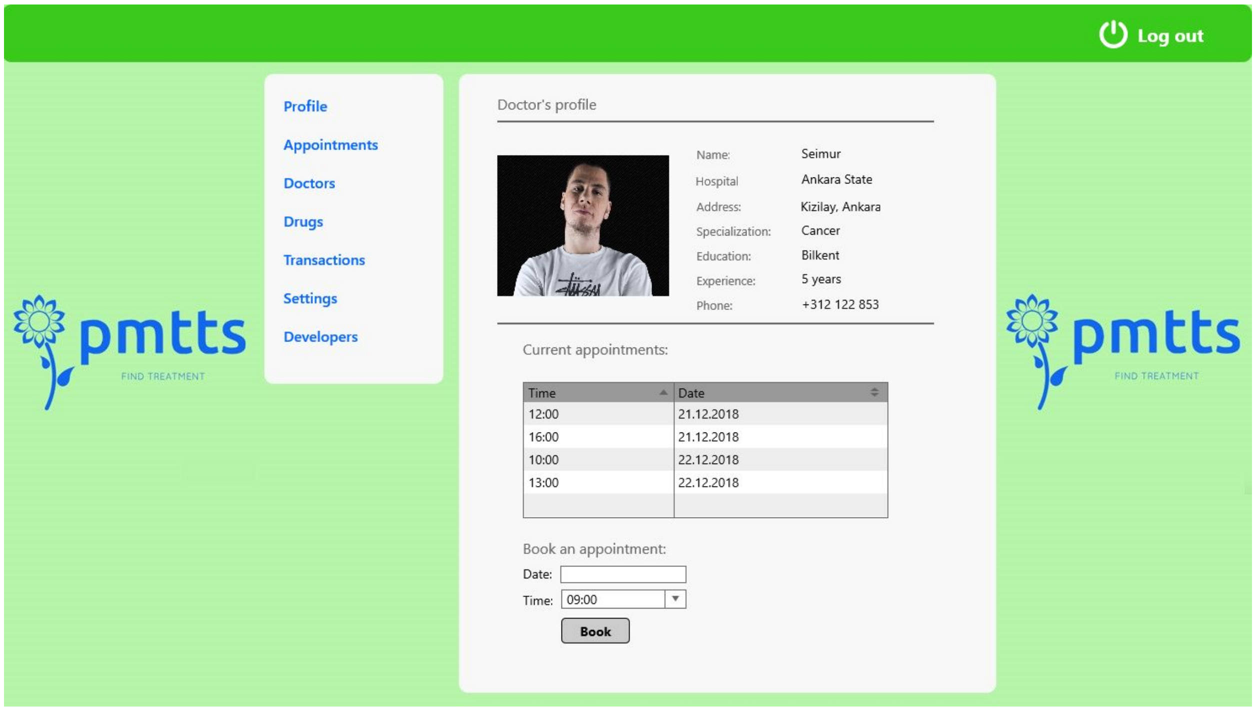
doctors’ information retrieval

SELECT hospital.name, doctor,name, experience, education, specialization, xLoc, yLoc

FROM Doctor join Hospital using (hos_id) join Address using (add_id)

WHERE @according_to like “%@search%” and username <> @patient_username

Order by @sort_by



Doctor's profile

Name:	Seimur
Hospital	Ankara State
Address:	Kizilay, Ankara
Specialization:	Cancer
Education:	Bilkent
Experience:	5 years
Phone:	+312 122 853

Current appointments:

Time	Date
12:00	21.12.2018
16:00	21.12.2018
10:00	22.12.2018
13:00	22.12.2018

Book an appointment:

Date:

Time: 09:00

Process: When user clicks one of the doctors, his/her information and his/her current booked appointments will be shown. In addition, user can book appointment by entering available date and time, then clicking "book" button. To make our implementation easier, we assume that each appointment lasts 1 hour.

Inputs: @doctor_username, @patient_usrname, @date, @time, @app_id

SQL Statements:

doctor's information retrieval

SELECT doctor.name, hospital.name, hospital.address, experience, education, specialization

FROM Doctor join Hospital using (hos_id)

WHERE username = @doctor_username

doctor's appointments retrieval

SELECT time, date

FROM Doctor join Appointments

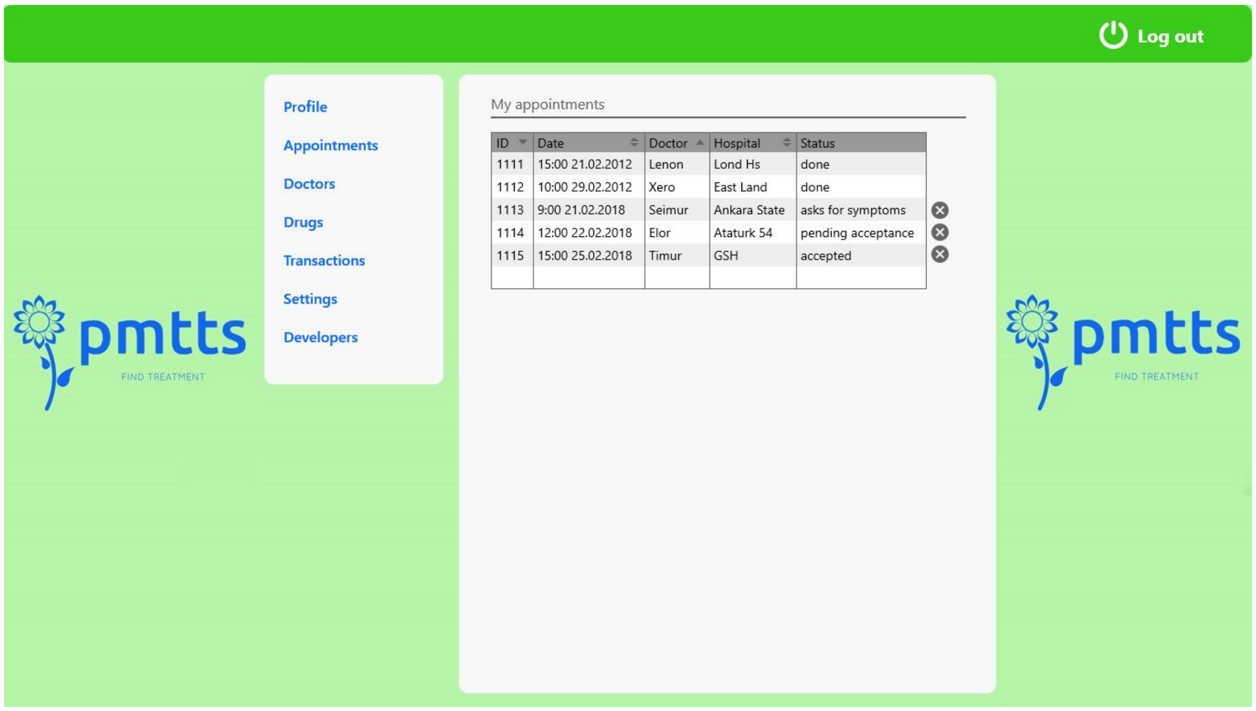
WHERE username = @doctor_username, status <> "done"

booking appointment

INSERT INTO Appointments

values(@app_id, “pending acceptance”, @date, @time,@patient_username, @doctor_username);

5.5. Patient’s Appointments menu



My appointments

ID	Date	Doctor	Hospital	Status
1111	15:00 21.02.2012	Lenon	Lond Hs	done
1112	10:00 29.02.2012	Xero	East Land	done
1113	9:00 21.02.2018	Seimur	Ankara State	asks for symptoms
1114	12:00 22.02.2018	Elor	Ataturk 54	pending acceptance
1115	15:00 25.02.2018	Timur	GSH	accepted

Process: In this page user can view information about his appointments. User can view detailed information about appointment by clicking its ID or go to symptoms page by clicking “asks for symptoms” status. In addition, he/she can cancel appointment by clicking cancel button. Appointments have 4 different statuses: firstly “pending acceptances”, when it is accepted, “asks for symptoms”, when symptoms are provided, “accepted”, when appointment occurs and doctor enters its results, “done”.

Inputs: @patient_username

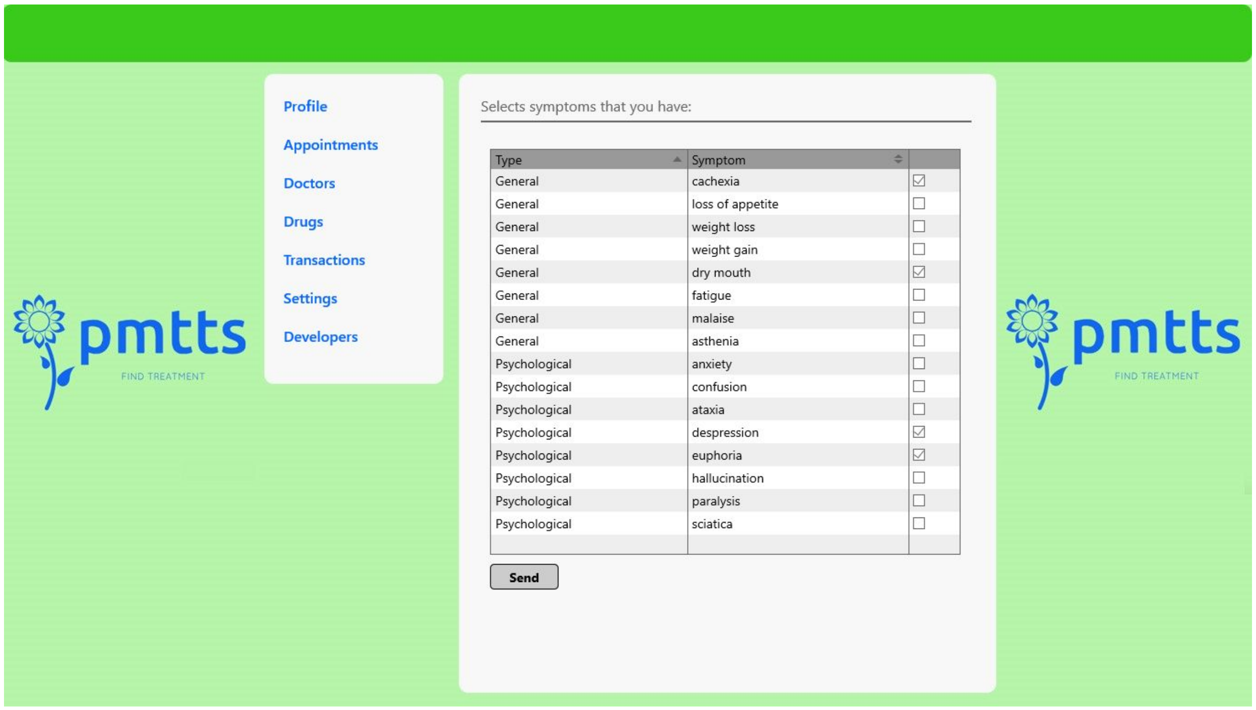
SQL Statements:

Patient’s appointments retrieval

SELECT app_id, date, time, Hospital.name, status

FROM Appointment join Doctor on (doctor_username = username) join Hospital using (hos_id)

WHERE patient_username = @patient_username



Selects symptoms that you have:

Type	Symptom	
General	cachexia	☑
General	loss of appetite	☐
General	weight loss	☐
General	weight gain	☐
General	dry mouth	☑
General	fatigue	☐
General	malaise	☐
General	asthenia	☐
Psychological	anxiety	☐
Psychological	confusion	☐
Psychological	ataxia	☐
Psychological	despression	☑
Psychological	euphoria	☑
Psychological	hallucination	☐
Psychological	paralysis	☐
Psychological	sciatica	☐

Process: In this page user should choose symptoms that he/she has and press send button.

Inputs: @app_id, @symptom_selected

SQL Statements:

symptoms retrieval

SELECT type, symp_name, description

FROM Symptom

entering symptoms

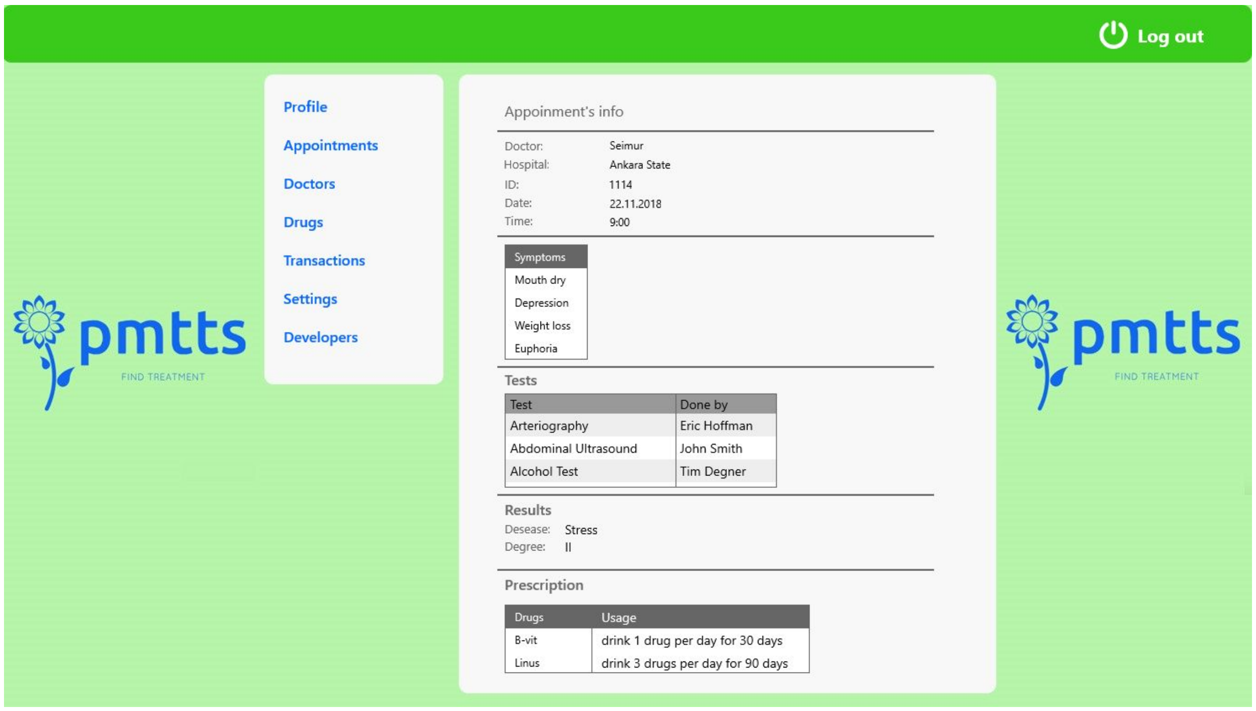
INSERT INTO Asks-for

value (@app_id, @symptom_selected)

changing status of appointment

UPDATE Appointment

SET status = “accepted”



Appoinment's info

Doctor:	Seimur
Hospital:	Ankara State
ID:	1114
Date:	22.11.2018
Time:	9:00

Symptoms
Mouth dry
Depression
Weight loss
Euphoria

Tests

Test	Done by
Arteriography	Eric Hoffman
Abdominal Ultrasound	John Smith
Alcohol Test	Tim Degner

Results

Desease:	Stress
Degree:	II

Prescription

Drugs	Usage
B-vit	drink 1 drug per day for 30 days
Linus	drink 3 drugs per day for 90 days

Process: In this page user can view detailed information about appointment and its results

Inputs: @app_id

SQL Statements:

for appointment info retrieval

SELECT *

FROM Appointment

WHERE app_id = @app_id

for symptoms retrieval

SELECT symp_name

FROM Asks-for join Appointment using (app_id)

WHERE app_id = @app_id

for tests retrieval

```sql
SELECT Test.name, Doctor.name
FROM Appointment join Does using (app_id)  join Doctor using (username)  join Tests using (test_id)
Where app_id = @app_id
```

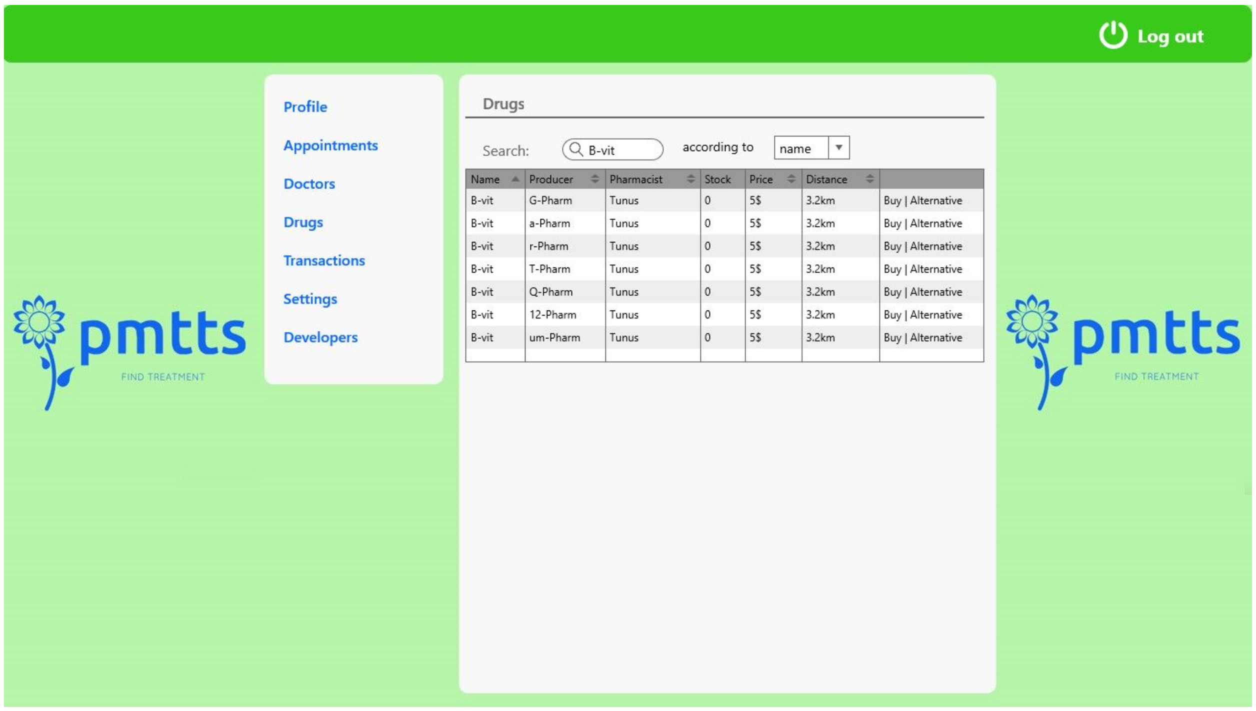
for results retrieval

```sql
SELECT name, degree
FROM Appointment join Diagnosis using (app_id)  join Shows using (diagnosis_id)  join Disease using (dis_id)
Where app_id = @app_id
```

for prescription retrieval

```sql
SELECT Drug.name, description
FROM Appointment join Diagnosis using (app_id)  join Prescribes using (diagnosis_id)  join Drug using (drug_id)
Where app_id = @app_id
```

5.6. Patient’s Drug menu



Drugs

Search: B-vit according to name

Name	Producer	Pharmacist	Stock	Price	Distance	
B-vit	G-Pharm	Tunus	0	5$	3.2km	Buy | Alternative
B-vit	a-Pharm	Tunus	0	5$	3.2km	Buy | Alternative
B-vit	r-Pharm	Tunus	0	5$	3.2km	Buy | Alternative
B-vit	T-Pharm	Tunus	0	5$	3.2km	Buy | Alternative
B-vit	Q-Pharm	Tunus	0	5$	3.2km	Buy | Alternative
B-vit	12-Pharm	Tunus	0	5$	3.2km	Buy | Alternative
B-vit	um-Pharm	Tunus	0	5$	3.2km	Buy | Alternative

Process: In this page user can view information about drugs. He/She can view detailed information about drug by clicking “buy”, get similar drugs from the same pharmacist by clicking “alternative” or view detailed information about pharmacist and drugs that he/she has by clicking pharmacist’s name. He/She can search drugs according to their name, producer or pharmacist. Table is sorted according the attribute chosen or by distance, which is calculated by algorithm. If user has both Patient and Pharmacist account, he/she cannot buy drug from himself/herself.

Inputs: @patient_username, @search, @according_to, @sort_by

SQL Statements:

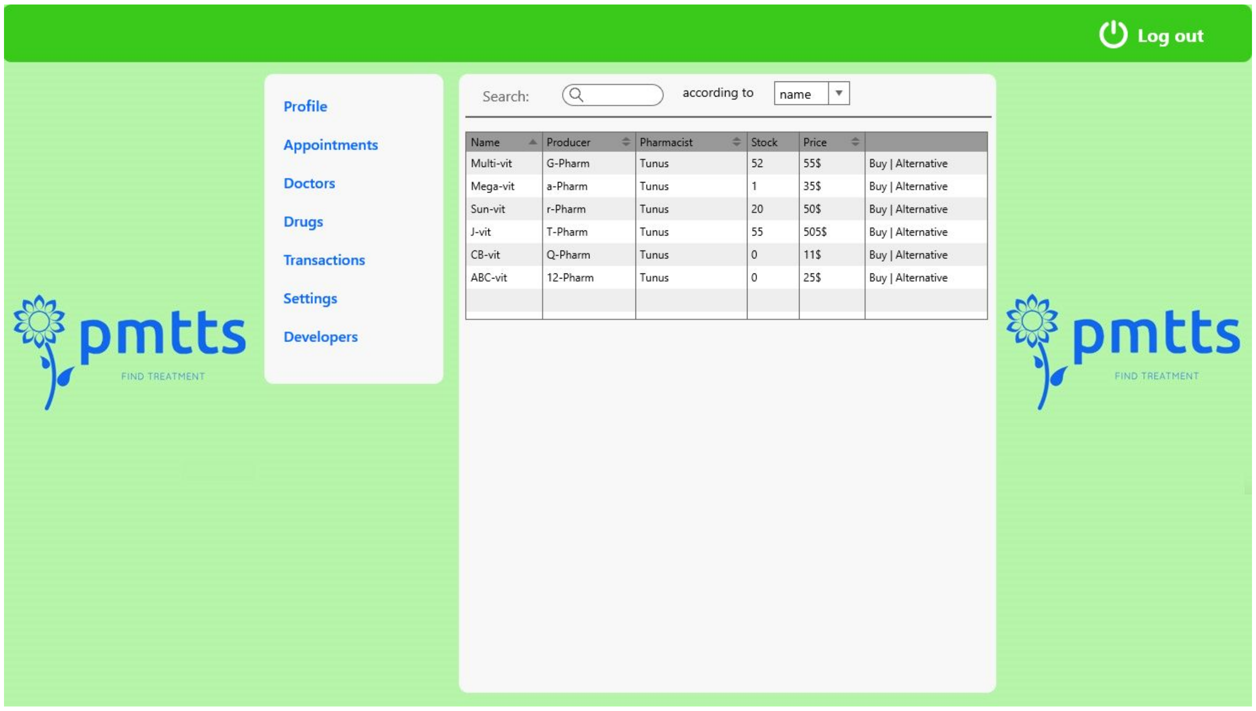
drugs’ information retrieval

SELECT Drug.name,producer, Pharmacist.name, stock, price, xLoc, yLoc

FROM Pharmacist join Has using (username) join Drug using (drug_id) join Address using(add_id)

WHERE @according_to like “%@search%” and username <> @patient_username

Order by @sort_by



Name	Producer	Pharmacist	Stock	Price	
Multi-vit	G-Pharm	Tunus	52	55$	Buy | Alternative
Mega-vit	a-Pharm	Tunus	1	35$	Buy | Alternative
Sun-vit	r-Pharm	Tunus	20	50$	Buy | Alternative
J-vit	T-Pharm	Tunus	55	505$	Buy | Alternative
CB-vit	Q-Pharm	Tunus	0	11$	Buy | Alternative
ABC-vit	12-Pharm	Tunus	0	25$	Buy | Alternative

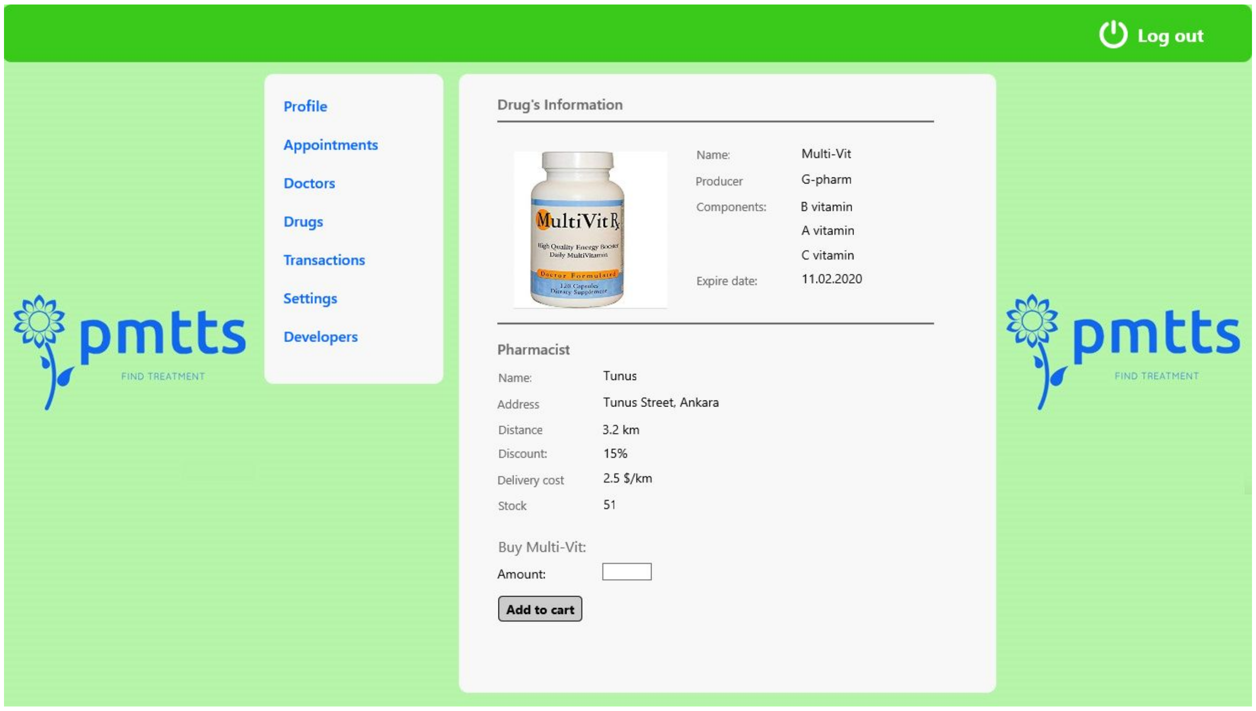
Process: This is an example of “alternative” function. This table contains all drugs similar to “B-vit” and from “Tunus” pharmacist. We assume that drugs are similar, if they have at least one same component.

Inputs: @drug_id, @pharmacist_username, @sort_by

SQL Statements:

drugs’ information retrieval

```
SELECT Drug.name,producer, Pharmacist.name, stock, Price

FROM Pharmacist join Has using (username) join Drug using (drug_id) as D

WHERE exists (  SELECT *

                FROM Pharmacist join Has using (username) join Drug using (drug_id) as A

                WHERE  A.drug_id  =  @drug_id  and  A.components  =  D.components  and
                A.username = D.username)

ORDER BY @sort_by
```



Drug's Information

Name:	Multi-Vit
Producer	G-pharm
Components:	B vitamin
	A vitamin
	C vitamin
Expire date:	11.02.2020

Pharmacist

Name:	Tunus
Address	Tunus Street, Ankara
Distance	3.2 km
Discount:	15%
Delivery cost	2.5 $/km
Stock	51

Buy Multi-Vit:

Amount:

Process: In this page user can view detailed information about drug and its pharmacist. In addition, user can add this drug to his/her shopping cart.

Inputs: @drug_id, @pharmacist_username, @trans_id, @amount, @date, @time

SQL Statements:

drug's information retrieval

```
SELECT *

FROM Drug

WHERE drug_id = @drug_id
```

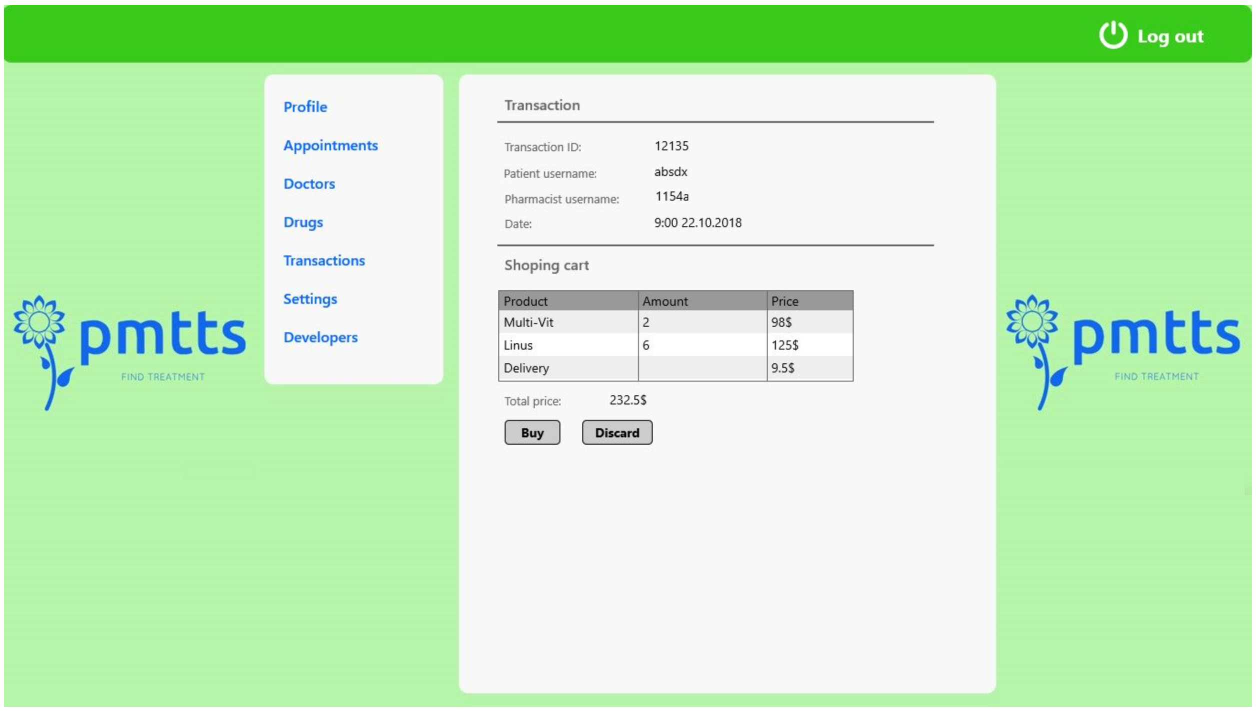
pharmacist's information retrieval

```
SELECT *

FROM Pharmacist join Address using (add_id)

WHERE username = @pharmacist_username
```

add drug to cart

```
INSERT INTO Transaction

values(@trans_id, @price * @amount, @date, @time, “not paid”)


INSERT INTO Contains

values(@trans_id, @pharmacist_username, @drug_id, @amount)
```

5.7. Patient's Transaction menu



Transaction

Transaction ID:	12135
Patient username:	absdx
Pharmacist username:	1154a
Date:	9:00 22.10.2018

Shoping cart

Product	Amount	Price
Multi-Vit	2	98$
Linus	6	125$
Delivery		9.5$

Total price: 232.5$

Process: In this page user can view information about certain transaction. He/She can pay for or cancel transaction by clicking corresponding button.

Inputs: @trans_id

SQL Statements:

transaction's information retrieval

SELECT trans_id, patient_username, pharmacist_username, date, time

FROM Transaction

WHERE trans_id = @trans_id

shopping cart's information retrieval

SELECT Drug.name, amount, price*amount

FROM Transaction join Pays-for using (trans_id) join Drug using (drug_id)

WHERE trans_id = @trans_id

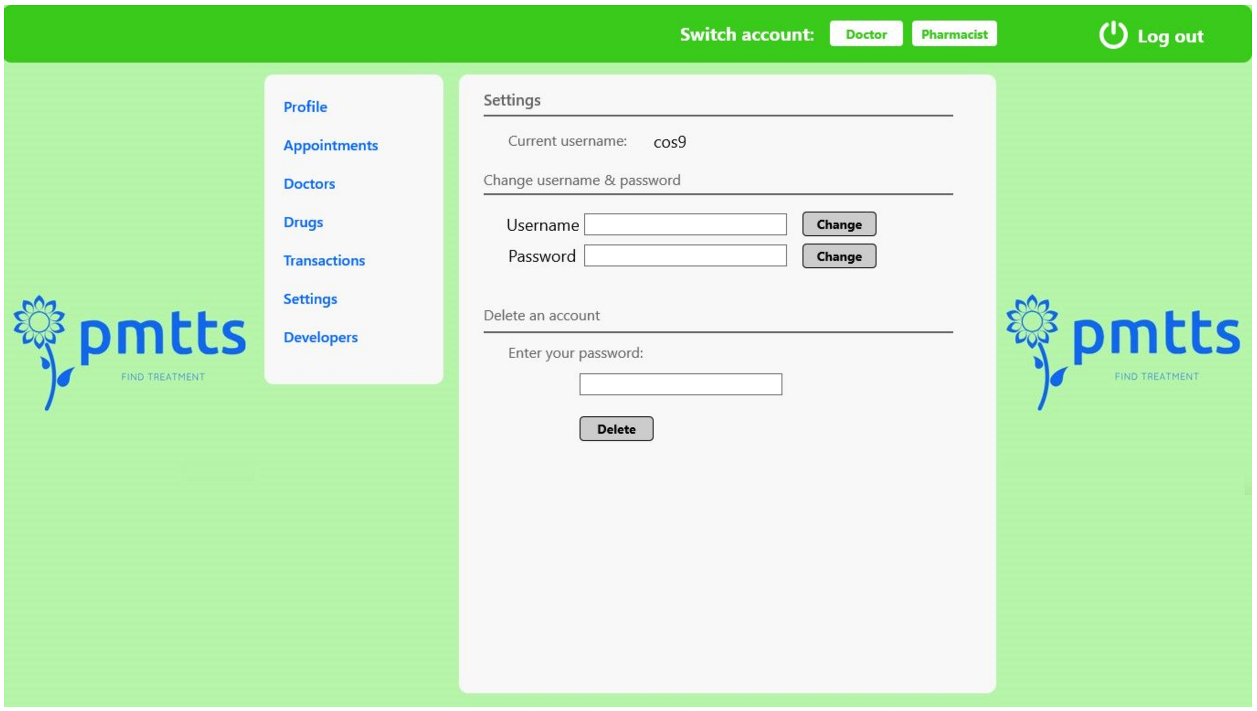
Shopping cart cancel

```
DELETE FROM Transaction

WHERE trans_id = @trans_id
```

Shopping cart pay

```
UPDATE Transaction

SET status = “paid”


UPDATE Has

SET stock = stock - C.amount

WHERE exists (SELECT *

                FROM Contains C

                WHERE C.trans_id = @trans_id AND Has.username = C.username AND

                Has.drug_id = C.drug_id )
```

5.8. Patient's Settings menu



Settings

Current username: cos9

Change username & password

Username

Password

Delete an account

Enter your password:

Process: In this page user can view his/her username, as well as change the password or username or delete his/her account.

SQL Statements:

Inputs: @username @new_username, @new_password, @entered_password

changing username

UPDATE user

SET password = @new_password

WHERE username = @username

changing username

UPDATE user

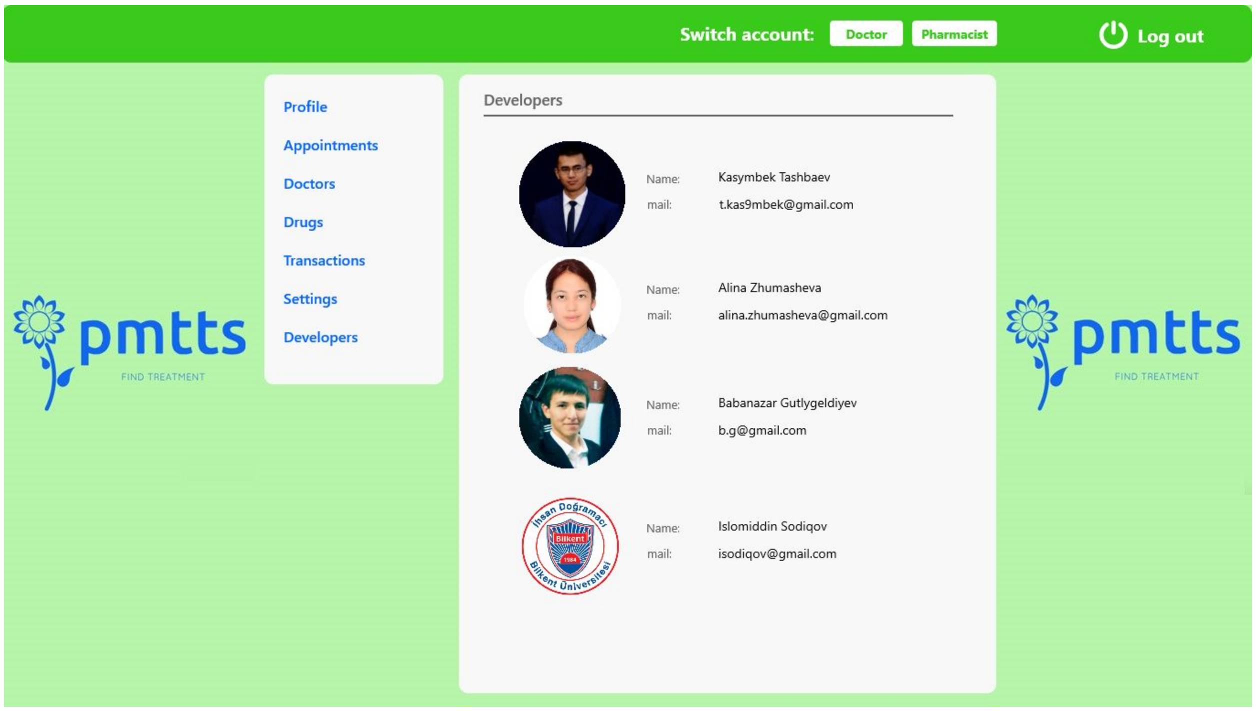
SET username = @new_username

WHERE username = @username

deleting an account

```
DELETE FROM user

WHERE username = @username AND password = @password_entered
```

5.9. Patient’s Developers menu

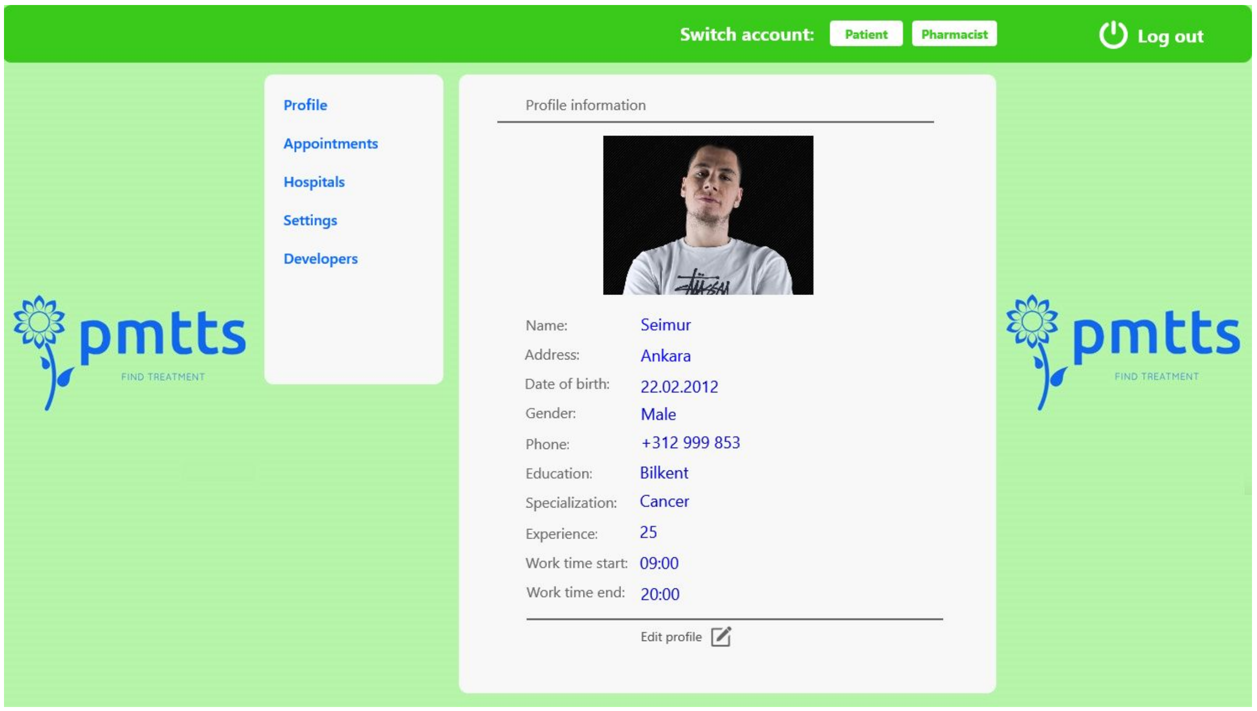


Developers

Name:	Kasymbek Tashbaev
mail:	t.kas9mbek@gmail.com
Name:	Alina Zhumasheva
mail:	alina.zhumasheva@gmail.com
Name:	Babanazar Gutlygeldiyev
mail:	b.g@gmail.com
Name:	Islomiddin Sodiqov
mail:	isodiqov@gmail.com

Process: In this page user can view information about delevelopers of this application.

5.10. Doctor’s Profile menu



Name:	Seimur
Address:	Ankara
Date of birth:	22.02.2012
Gender:	Male
Phone:	+312 999 853
Education:	Bilkent
Specialization:	Cancer
Experience:	25
Work time start:	09:00
Work time end:	20:00

Process: In this page user can view his/her profile information of doctor account. He/She can edit it by clicking edit button. In addition, here he/she can switch his/her account to Pharmacist or Doctor account. Doctor account has Profile, Appointments, Hospitals, Settings and Developers menus. All menus are similar to Patient’s menus except Hospitals.

Inputs: @username

SQL Statements:

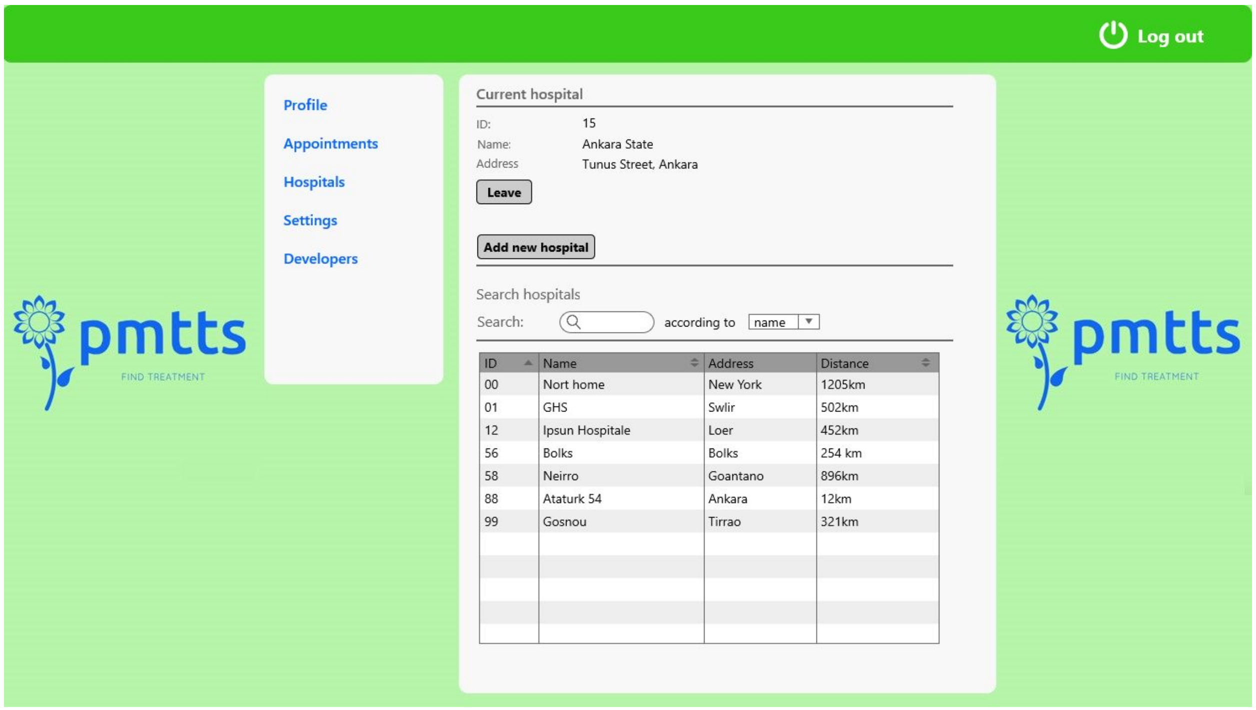
doctor’s information retrieval

SELECT name, birthday, gender, phone, country, city, street, apartment, apartment_num, education, specialization, experience, work_start_time, work_end_time

FROM Doctor join Address using (add_id)

WHERE username = @username

5.11. Doctor’s Hospital menu



Current hospital

ID:	15
Name:	Ankara State
Address	Tunus Street, Ankara

Search hospitals

Search: according to name

ID	Name	Address	Distance
00	Nort home	New York	1205km
01	GHS	Swlir	502km
12	Ipsun Hospitale	Loer	452km
56	Bolks	Bolks	254 km
58	Neirro	Goantano	896km
88	Ataturk 54	Ankara	12km
99	Gosnou	Tirrao	321km

Process: In this page doctor can view information about his/her current hospital. He/She can search other hospitals, leave his current hospital or go to page where he/she can add a new hospital to database by clicking the corresponding buttons. Detailed information about hospital is displayed, if doctor clicks hospital’s name.

Inputs: @username, @search, @according_to, @sort_by

SQL Statements:

hospitals’ information retrieval

SELECT hos_id, name, country, city, street, apartment, apartment_num

FROM Hospital join Address using (add_id)

WHERE @according_to like “%@search%”

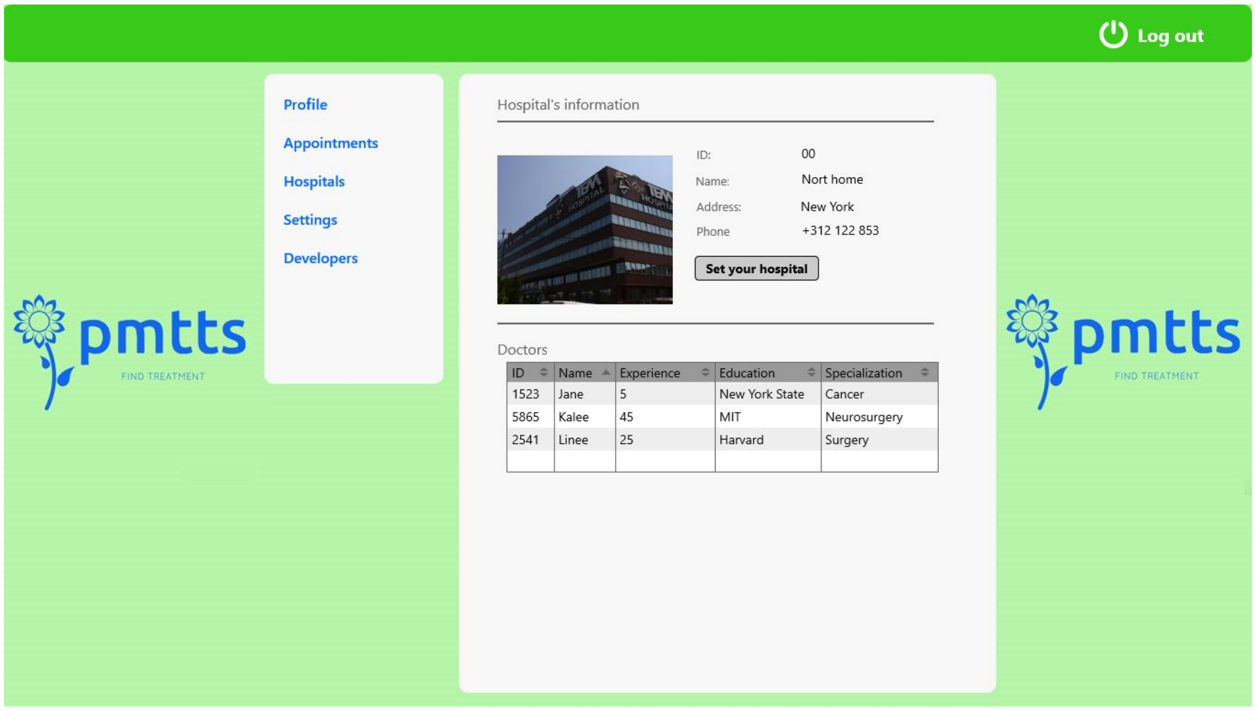
ORDER BY @sort_by

leaving current hospital

UPDATE Doctor

SET hos_id = null

WHERE username = @username



Hospital's information

ID:	00
Name:	Nort home
Address:	New York
Phone	+312 122 853

Doctors

ID	Name	Experience	Education	Specialization
1523	Jane	5	New York State	Cancer
5865	Kalee	45	MIT	Neurosurgery
2541	Linee	25	Harvard	Surgery

Process: In this page doctor can view detailed information about chosen hospital. He/She can search set this hospital as his current hospital.

Inputs: @username, @hos_id, @sort_by

SQL Statements:

hospitals' information retrieval

SELECT hos_id,name, country, city, street, apartment, apartment_num

FROM Hospital join Address using (add_id) join Doctor using (hos_id)

ORDER BY @sort_by

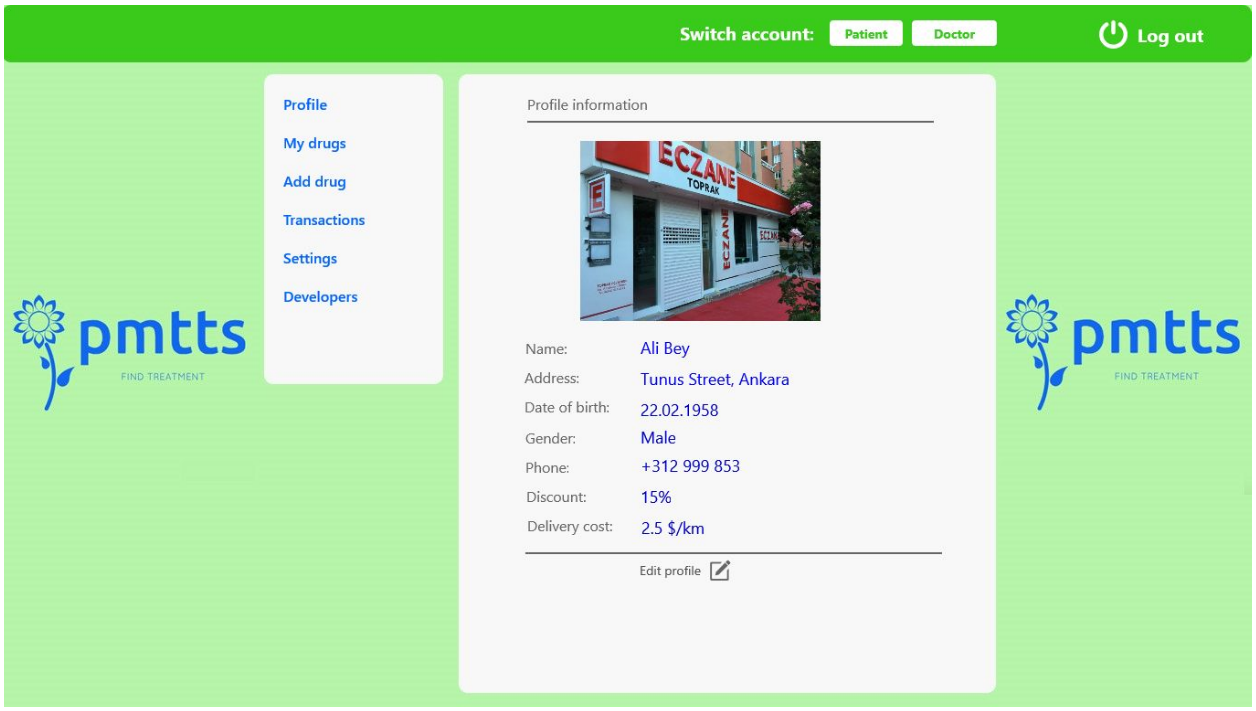
setting this hospital as current hospital

UPDATE Doctor

SET hos_id = @hos_id

WHERE username = @username

5.12. Pharmacist's Profile menu



Profile information

Name:	Ali Bey
Address:	Tunus Street, Ankara
Date of birth:	22.02.1958
Gender:	Male
Phone:	+312 999 853
Discount:	15%
Delivery cost:	2.5 $/km

Process: In this page user can view his/her profile information of pharmacist account. He/She can edit it by clicking edit button. In addition, here he/she can switch his/her account to Pharmacist or Doctor account. Pharmacist account has Profile, My Drugs, Add Drug, Transactions, Settings and Developers menus. All menus are similar to Patient's menus except My Drugs, Add Drug.

Inputs: @username

SQL Statements:

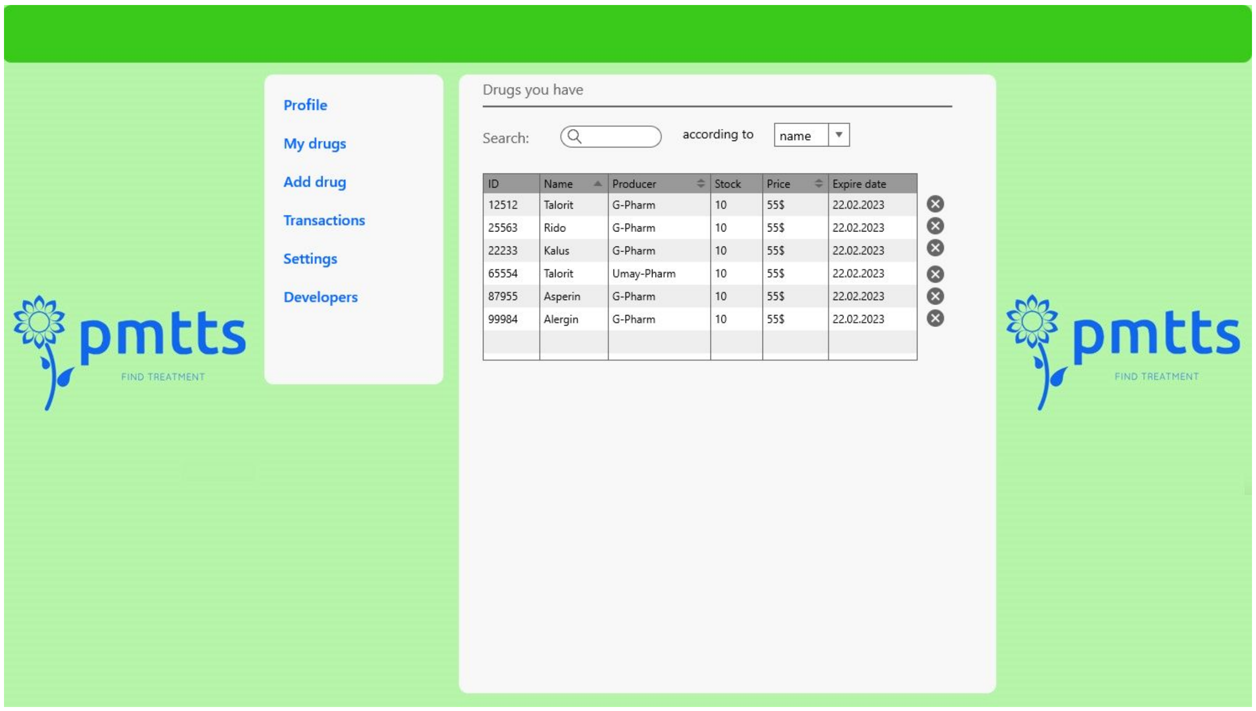
pharmacist's information retrieval

SELECT name, country, city, street, apartment, apartment_num, birthday, gender, phone, discount, delivery_cost

FROM Pharmacist join Address using (add_id)

WHERE username = @username

5.13. Pharmacist's My drugs menu



ID	Name	Producer	Stock	Price	Expire date
12512	Talorit	G-Pharm	10	55$	22.02.2023
25563	Rido	G-Pharm	10	55$	22.02.2023
22233	Kalus	G-Pharm	10	55$	22.02.2023
65554	Talorit	Umay-Pharm	10	55$	22.02.2023
87955	Asperin	G-Pharm	10	55$	22.02.2023
99984	Alergin	G-Pharm	10	55$	22.02.2023

Process: In this page pharmacists can view information about drugs that he/she has at the moment and he/she can remove certain drug.

Inputs: @username, @drug_remove, @search, @according_to, @sort_by

SQL Statements:

drugs info retrieval

SELECT drug_id, name, producer, stock, price, expire_date

FROM Pharmacist join Has using (username) join Drug using (drug_id)

WHERE @according_to like "%@search%" AND username = @username

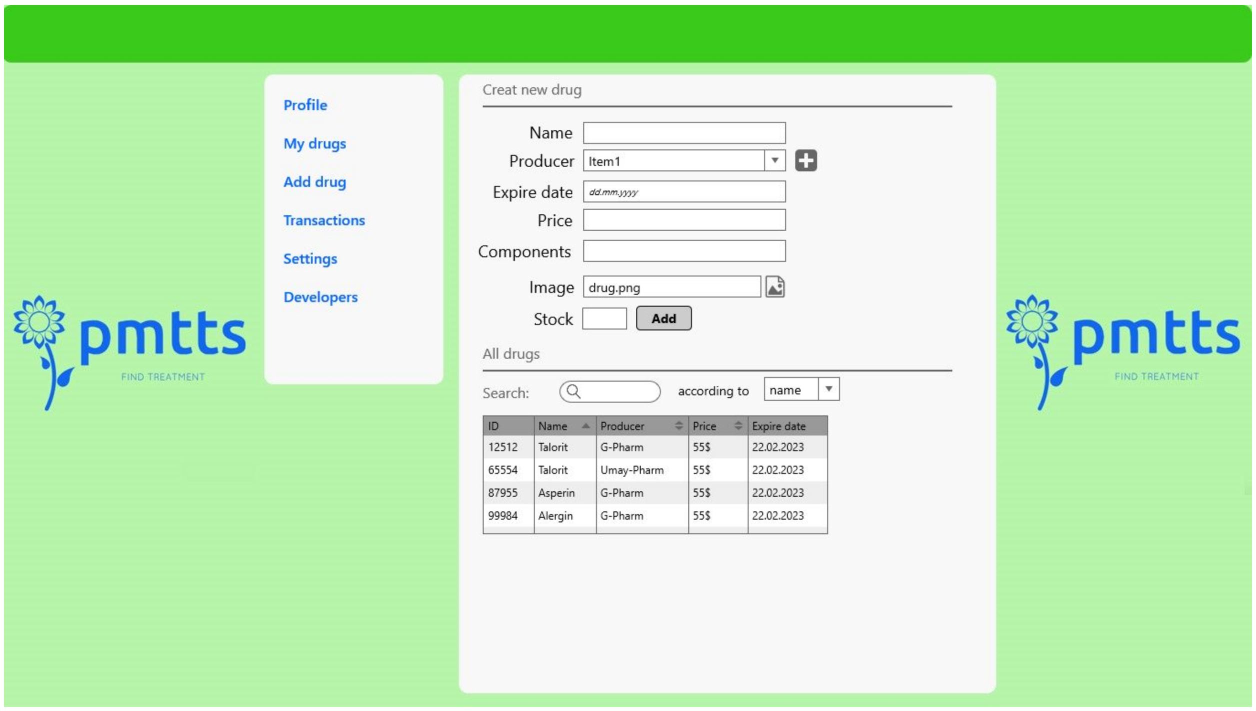
ORDER BY @sort_by

remove drug

DELETE FROM Has

WHERE drug_id = @drug_remove AND username = @username

5.14. Pharmacist’s Add drug menu



ID	Name	Producer	Price	Expire date
12512	Talorit	G-Pharm	55$	22.02.2023
65554	Talorit	Umay-Pharm	55$	22.02.2023
87955	Asperin	G-Pharm	55$	22.02.2023
99984	Alergin	G-Pharm	55$	22.02.2023

Process: In this page pharmacists can view all drugs in database and add new drug to it. He/She can go to Drug’s detailed information page where he/she can add it to the stock by clicking name of the drug.

Inputs: @username, @drug_id, @drug_name, @producer, @expire_date, @price, @components, @image, @stock, @search, @according_to, @sort_by

SQL Statements:

drugs’ information retrieval

SELECT drug_id, name, producer, stock, price, expire_date

FROM Pharmacist join Has using (username) join Drug using (drug_id)

WHERE @according_to like “%@search%”

ORDER BY @sort_by

insert new drug

```
INSERT INTO Drug

VALUES (@drug_id, @drug_name, @producer, @components, @image)


INSERT INTO Has

VALUES (@username, @drug_id, @stock)
```

6. ADVANCED DATABASE COMPONENTS

6.1. Reports

6.1.1. User's total money spent

```
WITH Payment (username, total_paid) AS (

        SELECT patient_username, sum (total_price)

        FROM Transaction

        GROUP BY patient_username)

SELECT total_paid

FROM Payment

WHERE username = @patient_username
```

6.1.2. Pharmacist's total money earned

```
WITH Payment (username, total_paid) AS (

        SELECT pharmacist_username, sum (total_price)

        FROM Transaction

        GROUP BY pharmacist_username)

SELECT total_paid

FROM Payment
```

```
WHERE username = @pharmacist_username
```

6.1.3. Doctor’s total appointments done

```
WITH App_count (username, appointments) AS (
        SELECT doctor_username, count(*)
        FROM Appointment
        WHERE status = ‘done’
        GROUP BY doctor_username)
SELECT appointments
FROM App_count
WHERE username = @doctor_username
```

6.2. Views

6.2.1. Patient’s Appointments view

When patient opens Doctor’s detailed information page, he/she can see only the date and time of booked appointments that doctor currently has, but not the name of people who booked because of privacy reasons.

```
CREATE VIEW App_patient AS (
        SELECT date, time
        FROM appointments
        WHERE doctor_username = @doctor_username)
```

6.2.2 Doctor’s Appointments view

When doctor views appointments, he/she can see only appointments booked by patients only from him but not from other doctors.

```
CREATE VIEW App_doctor AS (
        SELECT app_id, patient_username, date, time, status
        FROM Appointment
        WHERE doctor_username = @doctor_username)
```

6.2.3 Patient's Transactions view

When patient views transactions, he/she can see only transactions made by him/her.

```
CREATE VIEW patient_transactions AS (
        SELECT trans_id, pharmacist_username, total_price, date, time, status
        FROM Transactions join Contains using (trans_id)
        WHERE patient_username = @patient_username)
```

6.3. Triggers

- When Patient buys drugs from a particular pharmacist those drugs' stock will be decreased by amount bought.
- When Patient adds drug to shopping cart, total_price in Transaction will be increased by price * amount of drug added.
- When Doctor changes or leaves hospital, all appointments booked from him will be canceled.
- When drug's stock is zero, this drug will be removed from Pharmacist's current drug list.
- When Doctor changes his work_begin_time or work_end_time, appointments that were booked for time, which not between doctor's current work_begin_time and work_end_time, will be canceled.

6.4. Constrains

- Only signed up users can use the system.
- Patient can buy drugs from a pharmacist in amount that does not exceed the number provided by stock attribute.
- Patient can book an appointment from the doctor, who currently works in hospital.
- Patient can book an appointment only for the date and time that hasn't been booked by another patient.
- Patient can book an appointment for the date and time that is between doctor's work_begin_time and work_end_time.
- Doctor's experience attribute cannot be greater than his age.

6.5. Stored Procedure

We will use stored procedures for listing the doctors for the patients. Every time the list of all doctors of all hospitals in the database will be retrieved to the patient.
We will use stored procedures when pharmacist views his/her past transactions. The list of his/her transactions will be updated at that time when patient buys some drugs from that pharmacist, so we can use every time stored procedure to show to pharmacist his/her transactions. In addition, we will use stored procedure for viewing information about developers.

7. IMPLEMENTATION PLAN

7.1. User Interface

User Interface is going to be implemented in HTML, CSS and JavaScript.

7.2. Software

Back-end of our application is going to be implemented in PHP.

7.3. Database

Database part of our application is going to be implemented in MySQL.

8. WEBSITE

Our project’s website is: https://babanazar.github.io/CS353-PatientMedicalTreatmentTrackingSystem