Report of Final Project

Zhirong Lin Yi Shen Huidi Wang Lance Zeng

Content

Web Application Summary

For our web application / database, we are building a data-aggregator for patients and their medical needs. A patient's past records and medical history are important. Oftentimes, doctors need to examine past medical history and records to make a more informed decision on a patients' current needs. Nonetheless, there can be a lot of frustration when organizing your past medical records (and keeping them up-to-date), especially if you have visited multiple hospitals or even different branches of the same hospital.

To further complicate the problem, when patients are faced with changing circumstances (moved out of town, contradicted uncommon disease), patients need to spend an inordinate amount of time locating the best hospitals and clinics that accept their current (non) insurance.

The design of our web application hopes to alleviate these three issues:

1. We want users to be able to secure and track their own medical records.
2. We want users to be able to locate the nearest hospital / clinic that accepts the user's insurance.

3. We want users to find and connect with others who experienced similar affliction

A natural extension of our application is to allow users to share feedback about their doctors/healthcare workers with the greater community. Unfortunately, we did not have time to implement this part of the application. Instead, we focused our efforts in adding **a third feature** that would allow patients to find and connect (locally or online) with others that share similar ailments.

It seems like hospitals and insurance agencies exercise a lot of control over our personal medical records. Our hope is that this data management system will create a more transparent system, and simplify what is an important aspect of a patient's life.

Data Collection Issues

Unfortunately, medical records are highly regulated pieces of data. All patient data, doctor data, and nurse data is dummy data generated from: https://www.mockaroo.com/

Hospital names and addresses were randomly selected from a hospital rating dataset on Kaggle: https://www.kaggle.com/cms/hospital-general-information

Drugs and conditions were randomly pulled from a drug review dataset on Kaggle: https://www.kaggle.com/jessicali9530/kuc-hackathon-winter-2018

Insurance names were randomly pulled from a New York insurance premiums dataset on data.world: https://data.world/data-ny-gov/xek8-zfrt

Because a lot of our data was pulled from various sources, we had a difficult time building all the inner dependencies that would already exist in real data (ie. patients visiting local hospitals and seeing doctors that are actually residents in that particular hospital). With more time to develop a more realistic dataset (or with an actual real dataset of patient visit data), we could have both improved on the flow for general use as well as explore and understand more edge-cases. For our demo, we focused on creating one particular user that could display all our intended use-cases.

Entities & Organizational Structure:

This application is for patients in the United States healthcare system. We will clarify additional assumptions as they arise.

1. Person: the patient (US based) is the core user of our application. The patient entity includes all basic information of a patient including name, email, phone number, gender, address and a reference to his/her **insurance**. In our application, we do not assume that everyone has insurance (special key 0 for uninsured).

2. Visit: the patient visit is the central hub of information for our application. A patient visit is defined by a patient-doctor pair (ie. if a patient sees multiple doctors at the same hospital on the same date, it constitutes multiple visits). To make our application simpler to manage, we also assume that each visit will include a supporting nurse. Furthermore, each doctor and nurse are confined to one hospital. Each visit can have multiple treatments (purposes) including general check-up to a pre-existing ailment. Finally, the visit record will include the name of a disease if there is a doctor's diagnosis. To keep our application simple, we assumed that each visit would only include one disease diagnosis max; subsequent diagnoses are put into duplicate visit records.

3. Treatment/Purpose: this is a small entity that describes the purpose of the visit (activity), including general checkup to physical therapy. If the visit included a prescription, then it would be listed here. A particular treatment can include multiple drugs. As we were building out the front-end design, we realized that this convolution of treatment, purpose, and prescribed drugs led to an obscure visit presentation. *As a point of improvement, we would want to rethink this part of the data structure that consists of numerous one-to-many relationships, and clearly discriminate between purpose and treatment.*

4. Disease: the disease entity, is a lookup table that consists of common (and uncommon) diseases and related symptoms.

5. Drug: the drug entity shows drug id and drug name. Primary key is drug id. The entity relates to Treatment/Purpose by drug_id and it talks about each treatment subscribes what kind of drug and its specific name. We assume each treatment could have multiple drugs subscribed and each drug can be applied to multiple treatments. *As a point of*

improvement, we could relate Drug to Disease too in case patients having serious disease would like to know which drugs are used for the disease.

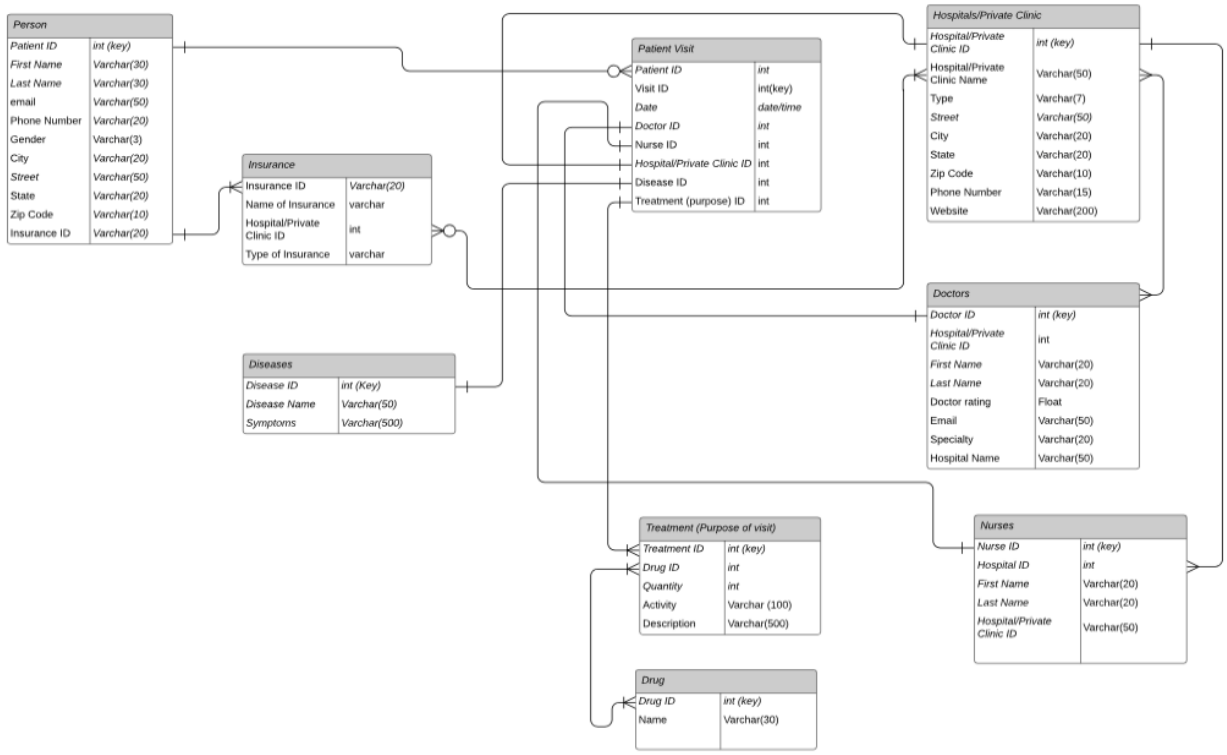
6. Insurance: the patient has their insurance records recorded in this entity including insurance company's name and corresponding insurance id. When insurance id is 0, we assume no insurance related. Insurance name will be 'Uninsured'. Insurance id is a primary, and also a foreign key to connect with insurance_hospital (this is an assistant entity between insurance and hospital. Only contain insurance_id and hospital_id) or patient entity.

7. Hospital/Private Clinics: hospital is a key resource for a patient to explore surrounding medical places and doctors. It includes records like hospital_id, hospital names, hospital type (private/public), address, phone number, and website. In our project, we assume all types of clinical or medical places belong to the 'hospital' table. We also assume each hospital could have multiple doctors and nurses. Hospital id is a primary and also a foreign key to connect Doctor/Physician, Nurses, Visit, Insurance_Hospital and further catch insurance entities. Only US hospitals are considered in this project.

8. Doctor/Physician: the entity provides information about each individual doctor. The information is about doctor id, hospital_id, doctor's first and last name, email and specialty. Each doctor is confined to one hospital. Primary key is doctor id. Foreign key is the hospital id that connects to the Hospital/Private Clinic and Visit entities.

9. Nurse: the entity provides information about nurses, including nurse id, hospital id, nurses' first and last name. One hospital can have multiple nurses, but each nurse only works for one hospital. Primary key is nurse id. Foreign key is hospital id used to connect Hospital/Private Clinics and Visit entities.

Entity-relationship diagram



Data Dictionary

Person - Store User Personal Information				
Attribute	**Data Type**	**PK/FK/Neither**	**Nullable**	**Description**
Patient ID	Integer	PK	No	Numerical user unique ID - autogenerated
First Name	Varchar(30)	Neither	Yes	User first name
Last Name	Varchar(30)	Neither	Yes	User last name
Email	Varchar(50)	Neither	Yes	User Email Address
Phone Number	Varchar(15)	Neither	Yes	User contact phone number
Gender	Varchar(3)	Neither	Yes	F = Female, M = Male, UNK = Unknown
Street	Varchar(50)	Neither	Yes	User home address - street
City	Varchar(20)	Neither	Yes	User home address - city
State	Varchar(20)	Neither	Yes	User home address - state
Zip Code	varchar(10)	Neither	Yes	User home address - zip code
Insurance ID	varchar(20)	FK	No	A string represents user insurance ID

Insurance - Stores Insurance Company information				
Attribute	**Data Type**	**PK/FK/Neither**	**Nullable**	**Description**
Insurance ID	Varchar(20)	PK	No	A string represents user insurance ID
Name of Insurance	Varchar(50)	Neither	No	Name of insurance company
Type of Insurance	Varchar(8)	Neither	Yes	Type of Insurance
Hospital/Private Clinic ID	Integer	FK	No	Corresponding numerical Hospital/Private Clinic (ID) can take this insurance - (each row for different Hospital/Private Clinic)

Patient Visit - Stores patient each visit information				
Attribute	**Data Type**	**PK/FK/Neither**	**Nullable**	**Description**
Patient ID	Integer	PK	No	Numerical user unique ID - autogenerated
Visit ID	Integer	FK	No	Numerical user unique visit ID - autogenerated: New visit if patient visits for multiple doctors, nurses & diseases.
Date	Date/Time	Neither	Yes	Date of visit
Doctor ID	Integer	FK	No	Numerical Doctor (ID) that patient visited. A default ID is used if no doctor for this visit.
Nurse ID	Integer	FK	No	Numerical Nurse (ID) that patient visited. A default ID is used if no nurse for this visit
Hospital/Private Clinic ID	Integer	FK	No	Corresponding numerical Hospital/Private Clinic (ID) patient visited
Disease ID	Integer	FK	No	Numerical disease ID. A default ID is used if no disease for this visit.
Treatment ID	Integer	FK	No	Numerical treatment ID for this visit

Hospital - Stores hospital information				
Attribute	**Data Type**	**PK/FK/Neither**	**Nullable**	**Description**
Hospital/Private Clinic ID	Integer	PK	No	Numerical hospital unique ID - Autogenerated
Hospital/Private Clinic Name	Varchar(50)	Neither	No	Name of hospital/Private Clinic
Type	Varchar(7)	Neither	No	Type of Hospital/Private Clinic, Public or Private
Rating	float	Neither	Yes	Numerical average user rating for this hospital
Street	Varchar(50)	Neither	Yes	Hospital/Private Clinic address- street
City	Varchar(20)	Neither	Yes	Hospital/Private Clinic address- city
State	Varchar(20)	Neither	Yes	Hospital/Private Clinic address- state
Zip Code	varchar(10)	Neither	Yes	Hospital/Private Clinic address- zip code
Phone Number	Varchar(15)	Neither	Yes	Phone number of Hospital/Private Clinic
Website	Varchar(200)	Neither	Yes	Insurance company Hospital would take

Doctors - Store doctors information				
Attribute	**Data Type**	**PK/FK/Neither**	**Nullable**	**Description**
Doctor ID	Integer	PK	No	Numerical Doctor Unique ID - Autogenerated
Hospital/Private Clinic ID	Integer	FK	No	Numerical Hospital/Private Clinic ID that Doctor works for
First Name	Varchar(20)	Neither	No	Doctor's first name
Last Name	Varchar(20)	Neither	No	Doctor's last name
Doctor Rating	Float	Neither	Yes	Numerical average user rating for the Doctor
Email	Varchar(50)	Neither	Yes	Doctor's email
Speciality	Varchar(20)	Neither	Yes	String describes fields that the Doctor is expert in, new rows if doctor has multiple expertises
Hospital Name	Varchar(50)	Neither	No	Name of Hospital/Private Clinic that Doctor works for

Nurses- Store nurses information				
Attribute	**Data Type**	**PK/FK/Neither**	**Nullable**	**Description**
Nurse ID	Integer	PK	No	Numerical Nurse Unique ID - Autogenerated
Hospital/Private Clinic ID	Integer	FK	No	Numerical Hospital/Private Clinic ID that Nurse works for
First Name	Varchar(20)	Neither	No	Nurse's first name
Last Name	Varchar(20)	Neither	No	Nurse's last name
Hospital Name	Varchar(50)	Neither	No	Name of Hospital/Private Clinic that nurse works for

Diseases - Stores disease information				
Attribute	**Data Type**	**PK/FK/Neither**	**Nullable**	**Description**
Disease ID	Integer	PK	No	Numerical Disease Unique ID - Autogenerated
Disease Name	Varchar(50)	Neither	No	Name of disease
Symptom	Varchar(500)	Neither	Yes	Symptom of this kind of disease

Treatment - Stores each treatment information				
Attribute	**Data Type**	**PK/FK/Neither**	**Nullable**	**Description**
Treatment ID	Integer	PK	No	Numerical Treatment Unique ID for patient each patient visit- Autogenerated
Drug ID	Integer	FK	No	Numerical drug ID for patient this treatment. A default ID is used if no drug used for this treatment. New row if multiples drugs used for this treatment
Quantity	Varchar(50)	Neither	No	Quantity of Drug
Activity	Varchar(100)	Neither	No	Purpose of this treatemnt
Description	Varchar(500)	Neither	Yes	Doctor's comments for this treatment

Drug - Stores all drug information				
Attribute	**Data Type**	**PK/FK/Neither**	**Nullable**	**Description**
Drug ID	Integer	PK	No	Numerical Drug Unique ID - From FDA approved drug list
Name	Varchar(30)	Neither	No	Name of drug

Database Implementation

Use case 1: Secure and track personal medical records:

```
SELECT v.date,
        h.name AS hospital_name,
        CONCAT(p2.first_name,' ', p2.last_name) AS doctor_name,
        CONCAT(n.first_name,' ', n.last_name) AS nurse_name,
        t.activity, d.name AS diagnosis
       FROM patients p
         JOIN visits v on p.id = v.patient_id
         JOIN hospitals h on v.hospital_id = h.id
         JOIN physicians p2 on v.doctor_id = p2.id
         JOIN nurses n on v.nurse_id = n.id
         JOIN visit_treatment vt on v.id = vt.visit_id
         JOIN treatment t on vt.treatment_id = t.id
         JOIN diseases d on v.disease_id = d.id
         JOIN drugs d2 on d2.id=t.drug_id
         WHERE p.id=2
         ORDER BY 1 DESC;
```

My Visit Records

Date	Hospital	Doctor	Nurse	Purpose	Doctor's Notes
2019-10-18	FAIRVIEW LAKES MEDICAL CENTER	ARDALAN ENKESHAFI	VIDYA KOLLU	therapy	
2016-07-28	PIPESTONE COUNTY MEDICAL CENTER	NICHOLAS DEFILIPPIS	RADHIKA PHADKE	check-up	
2013-08-21	FAIRVIEW LAKES MEDICAL CENTER	MAGED SOLIMAN	MICHAEL NUYLES	surgery	
2012-05-04	FAIRVIEW LAKES MEDICAL CENTER	ARDALAN ENKESHAFI	THRESA YOUNG	surgery	Opioid-Induced Constipation
2012-05-04	FAIRVIEW LAKES MEDICAL CENTER	ARDALAN ENKESHAFI	THRESA YOUNG	therapy	Opioid-Induced Constipation
2012-05-04	FAIRVIEW LAKES MEDICAL CENTER	ARDALAN ENKESHAFI	THRESA YOUNG	sickness	Opioid-Induced Constipation
2009-01-15	SAINT FRANCIS HOSPITAL MUSKOGEE	PHILIP STANDHART	JEANNE GAUTREAUX	sickness	
2006-12-06	UNITED HOSPITAL	ETAN SPIRA	AIMEE COLLINS	therapy	
2005-08-06	NORTHSIDE MEDICAL CENTER	CYNTHIA SANTIAGO	LINDSAY WILSON	sickness	Epilepsy
2005-08-06	NORTHSIDE MEDICAL CENTER	CYNTHIA SANTIAGO	LINDSAY WILSON	shot	Epilepsy
2005-08-06	NORTHSIDE MEDICAL CENTER	CYNTHIA SANTIAGO	LINDSAY WILSON	check-up	Epilepsy
2002-01-13	UNITED HOSPITAL	ETAN SPIRA	MERRY CHEN	shot	

Use case 2: Locate the nearest hospital / clinic that accepts the user's insurance:

```
SELECT h.name AS hospital_name, CONCAT(h.street, ' ', h.city, ', ', h.state, ' ', h.zip) AS address, h.phone_number, h.website

FROM patients p

LEFT JOIN insurance i ON p.insurance_id = i.id

LEFT JOIN insurance_hospital ih ON i.id = ih.insurance_id

LEFT JOIN hospitals h ON ih.hospital_id=h.id

WHERE p.id=2 AND h.state = p.state
```

Hospitals Near Me

Hospital Name	Address	Phone	Website
UNITED HOSPITAL	333 NORTH SMITH AVENUE SAINT PAUL, MN 55102	6512418802	Direct
FAIRVIEW LAKES MEDICAL CENTER	5200 FAIRVIEW BOULEVARD WYOMING, MN 55092	7633896481	Direct

Use case 3: Locate (nearby) patients on platform that have (had) the same ailment:

```
SELECT CONCAT(sub.first_name,' ', sub.last_name) AS name, sub.phone_number, sub.email, sub.name AS disease_name, sub.zip
```

```
FROM (
  SELECT p.first_name, p.last_name,p.phone_number, p.email, p.zip,v.disease_id,
d.name,SUBSTRING(zip,1,1) AS sub_zip
  FROM patients p
  JOIN visits v on p.id = v.patient_id
  JOIN diseases d on v.disease_id = d.id
  WHERE v.disease_id IN (
    SELECT disease_id
    FROM patients
    JOIN visits v2 on patients.id = v2.patient_id
    WHERE patients.id=2 AND disease_id<>0)
  AND p.id<>2) AS sub
JOIN
  (SELECT SUBSTRING(zip,1,1) AS sub_zip
  FROM patients p2
  WHERE p2.id=2 ) sub1
  ON sub.sub_zip=sub1.sub_zip
```

My Support Group

These people are in the same boat with you, you can try to contact them and set up a support group.

Patient Name	Phone	Email	Situation	Zipcode
Kellsie Hayer	772-442-2226	khayer33@yandex.ru	Epilepsy	54935
Elane Casson	757-832-6177	ecasson9x@globo.com	Epilepsy	59858

Sample reports

We did not have time to flesh out the recommended “doctors near me” function. The function currently lists all doctors in nearby hospitals sorted by their rank. Our idea was to have the patient identify what type of doctor (“specialty”) s/he wants to locate, and a preferred minimum rating. Our tool would then locate nearby doctors that match the criteria and potentially link to user reviews:

Doctors Recommended for Me

Doctor Name	Hospital	Specialty	Doctor Rating
KAREN BENEZRA	UNITED HOSPITAL	Osteopathic Manipulative Medicine	10.0
ETAN SPIRA	UNITED HOSPITAL	Dental	8.0
MICHAEL FISHMAN	UNITED HOSPITAL	Diagnostic Radiology	9.0
ARDALAN ENKESHAFI	FAIRVIEW LAKES MEDICAL CENTER	Internal Medicine	6.0
OBINNA UZOWULU	FAIRVIEW LAKES MEDICAL CENTER	Family Practice	7.0
MAGED SOLIMAN	FAIRVIEW LAKES MEDICAL CENTER	Psychiatry	1.0

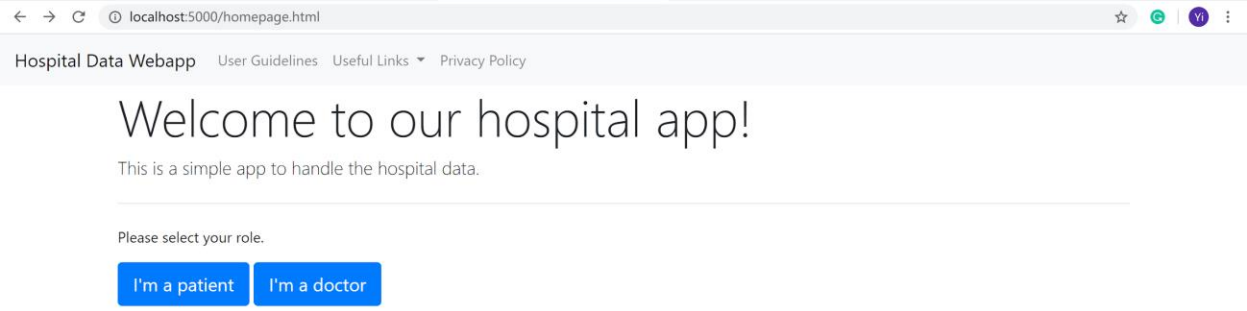
Another idea that needs to be fleshed out is the drug scheduler/cost analysis. Right now, our table only lists the prescriptions that the user has received. We could add links to more information about the drugs, and with cost information, we could cal

My Medicine

Date of Visit	Hospital Prescibed	Doctor Prescribed	Drug Name	Quantities
2012-05-04	FAIRVIEW LAKES MEDICAL CENTER	ARDALAN ENKESHAFI	Valsartan	34
2009-01-15	SAINT FRANCIS HOSPITAL MUSKOGEE	PHILIP STANDHART	Lamotrigine	38
2009-01-15	SAINT FRANCIS HOSPITAL MUSKOGEE	PHILIP STANDHART	Imitrex	28
2009-01-15	SAINT FRANCIS HOSPITAL MUSKOGEE	PHILIP STANDHART	Dulcolax	41
2009-01-15	SAINT FRANCIS HOSPITAL MUSKOGEE	PHILIP STANDHART	Buprenorphine	42
2005-08-06	NORTHSIDE MEDICAL CENTER	CYNTHIA SANTIAGO	Guanfacine	7
2005-08-06	NORTHSIDE MEDICAL CENTER	CYNTHIA SANTIAGO	Keppra	7
2005-08-06	NORTHSIDE MEDICAL CENTER	CYNTHIA SANTIAGO	Tioconazole	9

User Interface

- Homepage:

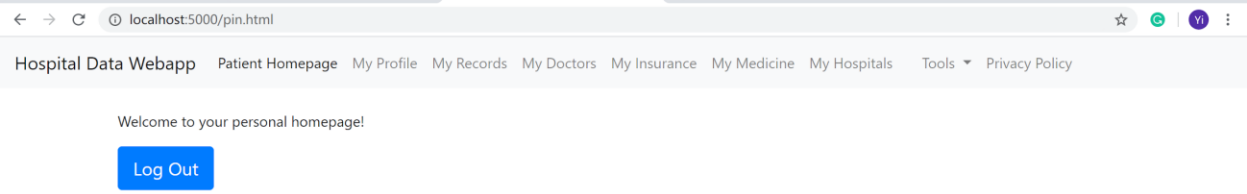


This is the page that users will be rendered to while using the url localhost:5000. We use the Bootstrap nav-bar to present options for users. All pages inherited this kind of nav-bar layout, just the items on the nav-bar might vary due to different purposes. We also use Bootstrap Jumbotron to display welcome messages, and Bootstrap buttons to let our users select the roles. Currently only the functions on the patient side is available

- Patient Login Page:

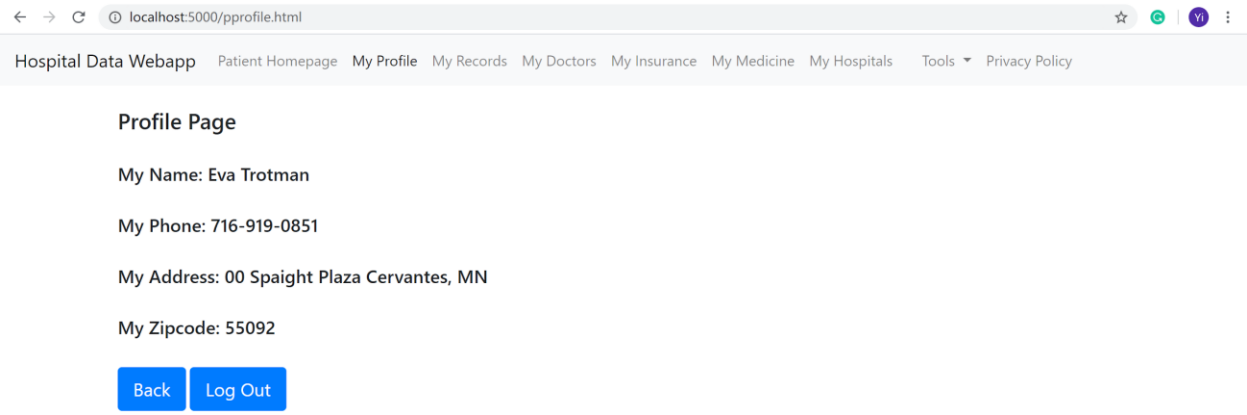
After tabbing the "I'm a Patient" button, users will be rendered to this page. The nav-bar is exactly the same as the homepage on this page. We Leveraged a Bootstrap Form here to interact with our users. We ask them to enter their name and phone number to query the patient record. By doing this we also fulfill the authentication purpose. If we found a record in the database, users will be rendered to the patient homepage. For the rest of this section, we'll use Eva Trotman, an existing virtual patient in the database to illustrate the rest of the functions.

- Patient Homepage



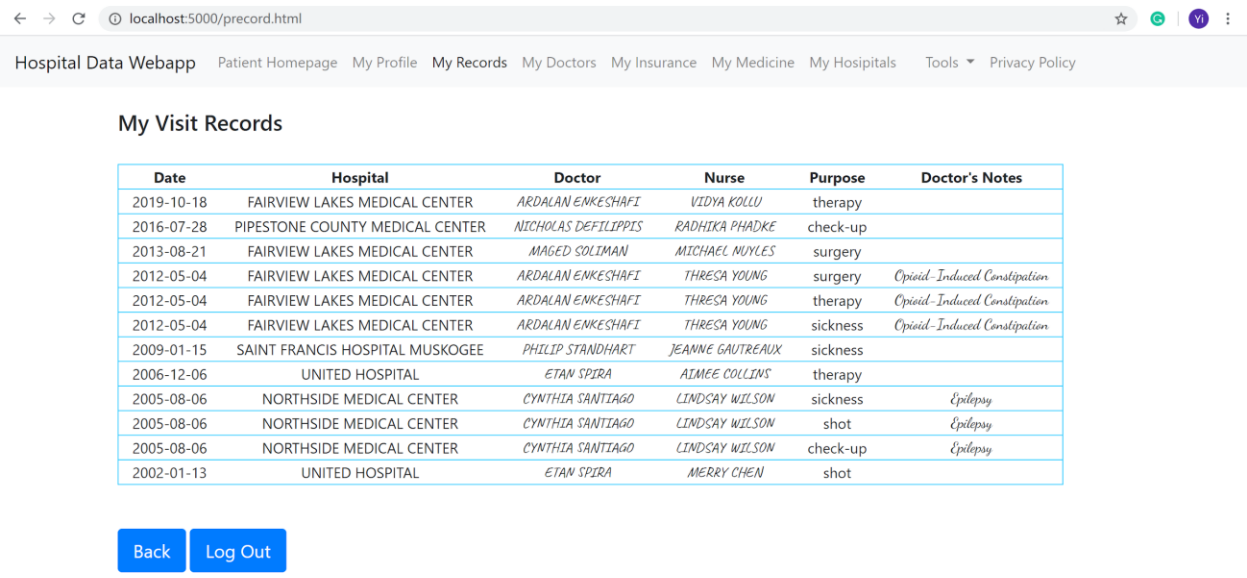
This is the UI of the patient homepage. We updated the items on nav-bar to enable different options to our users. To remind users of the current page they're in, we also use the "active" function on nav-items, so that the font indicating their current page will appear to be darker. Users can decide on which kind of medical record they want to access by tabing the nav-items displayed on nav-bar.

- Patient Profile Page



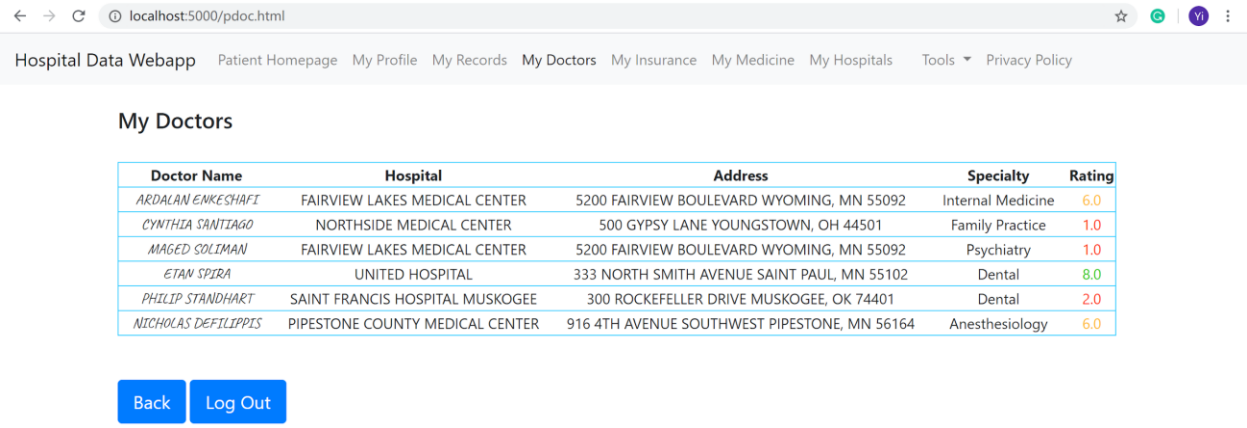
This is the page to display basic profile information including name, phone, address and Zipcode of the address.

- Patient Record Page



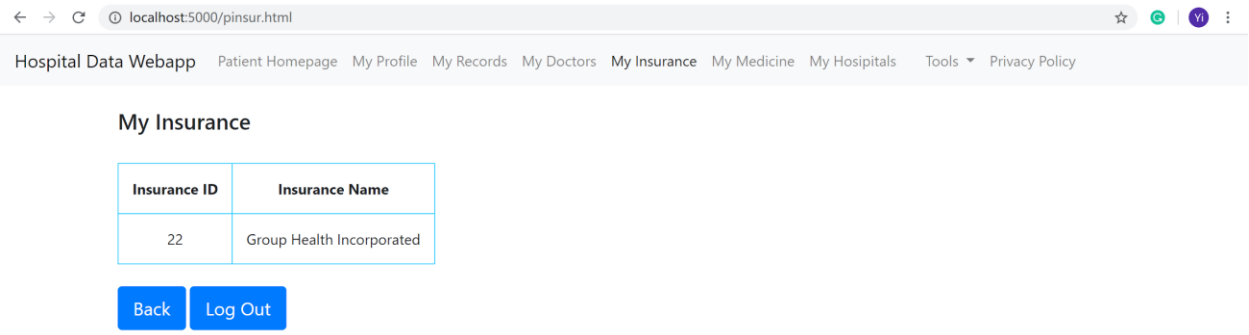
This is the page to display patient's visiting records including date of visit, hospital, doctor and nurse, purpose of visiting and doctor's note. On this page, we treat visits on the same day for different purposes, or recepted by different doctors or nurses as different visiting records. We attach a CSS file to format the table and attach different font styles to different columns for the table.

- Patient Doctors Page



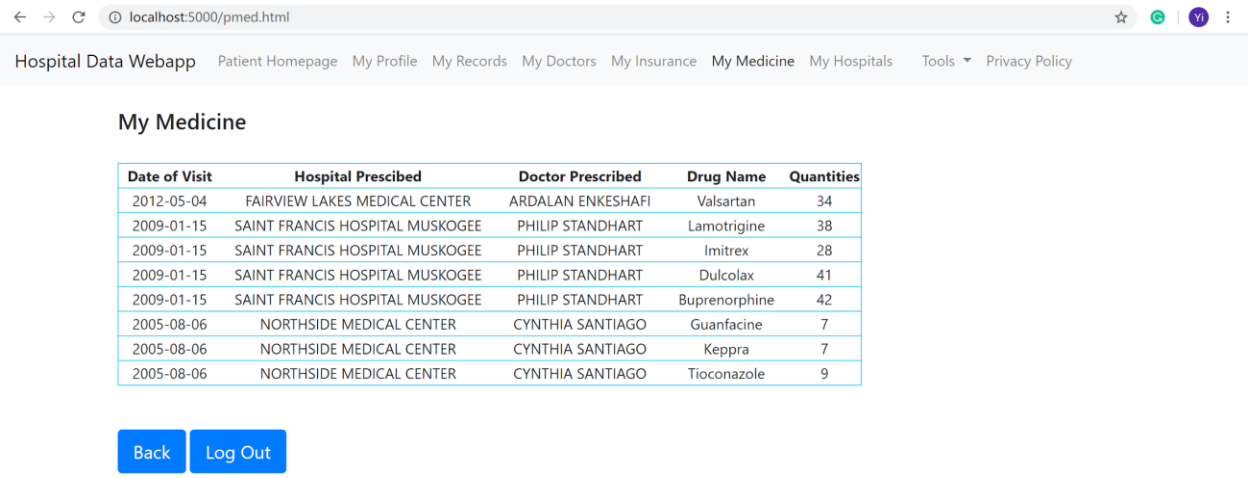
This page displays all doctors a person has visited, their hospital name and address, their specialties and their ratings. To improve user experience, we assign different colors to the ratings.

- Patient Insurance Page



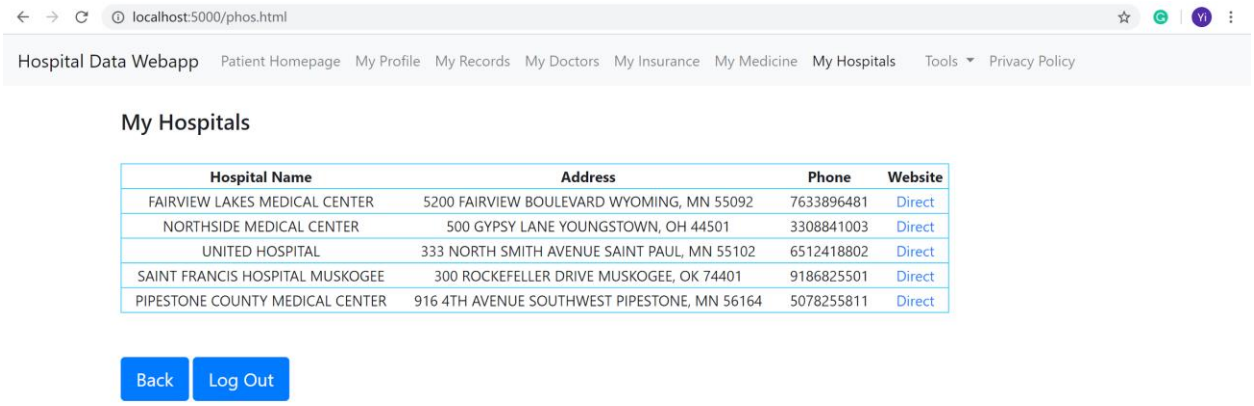
This page displays the patient's insurance ID and insurance name.

- Patient Medicine Page



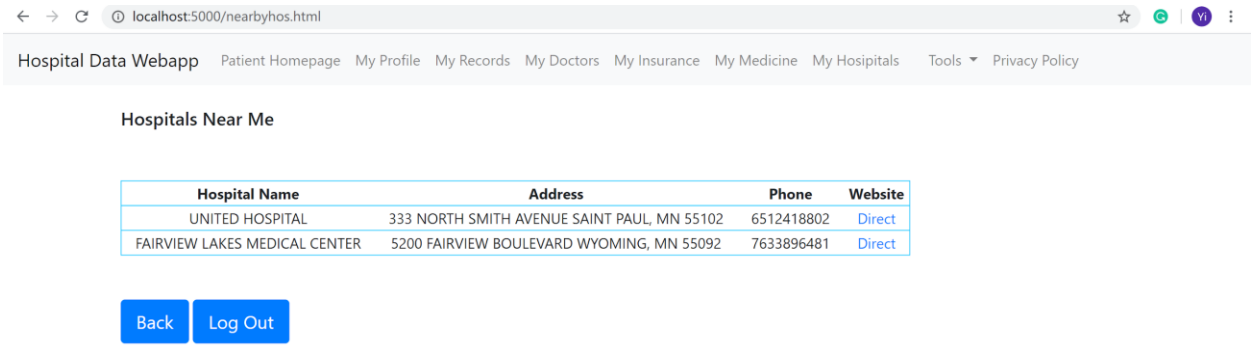
This page displays the medicine name and quantity prescribed to the patient. Hospital, doctor, date of prescription are also being displayed.

- Patient Hospital Page



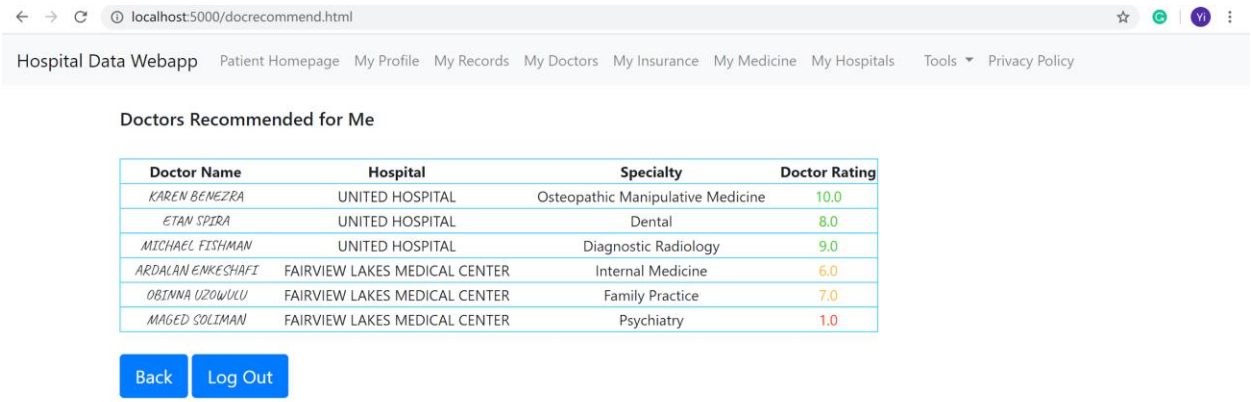
This page displays all hospitals that a patient has visited in the past, the address, phone number as well as the link for websites. The hospitals were ordered by its Zipcode on this page.

- Hospitals Nearby Page (under the Tools dropdown)



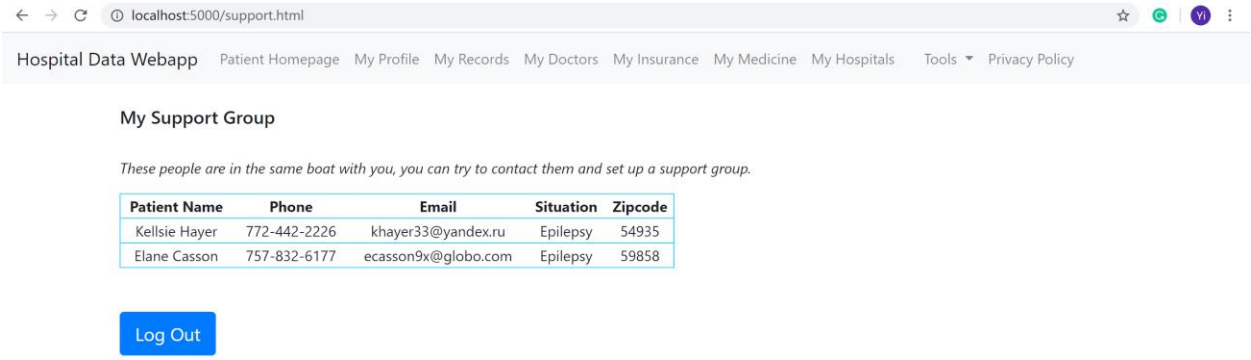
This page displays names and addresses of hospitals that accept patient's insurance near the patient. Phone numbers and websites are also included on this page.

- Doctor Recommended Page



This page displays doctors' names, their specialties, their ratings and the hospitals they worked for. These doctors are being recommended because their hospitals are located near the patient and their hospitals support the patient's insurance.

- Support Group Page



This page displays other nearby patients who shared the same affliction with the current patient. The information of those patients were being displayed for the purpose of encouraging them to form up a support group to chat about their experiences or alleviate their stress. We hope that we can add a post or blog thread on this page for users within this list so that they can use fake names or even be anonymous to protect their privacy.

Future Improvement and Plan

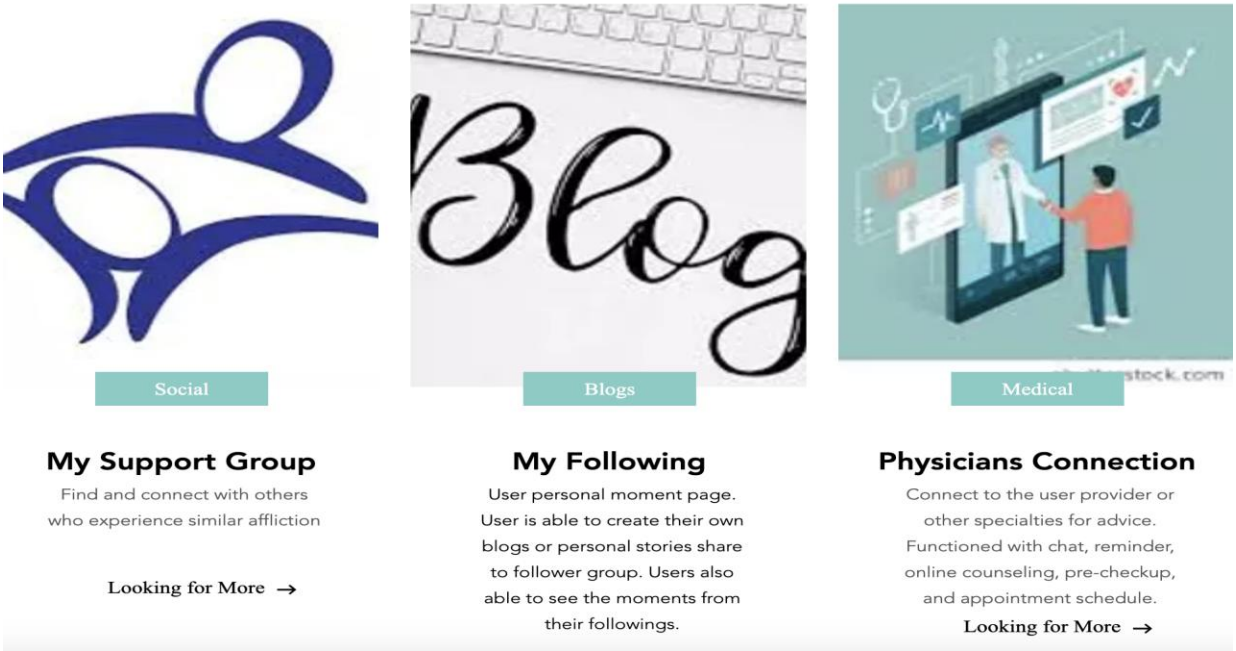
Relating to a realistic situation, like a patient visits a hospital staying for a couple of days and meeting with multiple doctors and nurses, in the future, we will improve our database by expanding the Patient Visit entity. We'll add more columns like days_spent, doctor1, doctor2,... when more doctors and nurses are necessary to be recorded. Besides improving database design with more edge-cases and thorough consideration, we plan to add more functionality in the web app.

- Community expansion:

  Beyond finding patients having similar diseases, people are able to create their own blog page that takes trusted sources or posts personal stories to share publicly. In this way, the community becomes tighter and more helpful internally.

- Physicians-Patients Connection:

  Patients not only expect to communicate with each other, they also look for specialties' advice. For example, it could be professional counseling with a licensed therapist, or a general advice of symptoms before going to hospitals. We will add online appointment options in the future, either paid or free. Also patients have chances to leave reviews and comments for each physician. The purpose is to make professional counseling accessible, affordable, convenient.



By integrating all types of use cases, this webapp will be a medical hub across our society.

Appendix

Select Profile:

```
SELECT CONCAT(first_name, ' ', last_name) AS name, phone_number, CONCAT(street, ' ',
city, ', ', state) AS address, zip \
    FROM patients WHERE id = %s", user_id)
```

Select Patient Records:

(Distinct is here because there was some error with our data generation that generated duplicate patient visits)

```
SELECT DISTINCT v.date,
        h.name AS hospital_name,
        CONCAT(p2.first_name,' ', p2.last_name) AS doctor_name,
        CONCAT(n.first_name,' ', n.last_name) AS nurse_name,
        t.activity, d.name
       FROM patients p
        JOIN visits v on p.id = v.patient_id
        JOIN hospitals h on v.hospital_id = h.id
        JOIN physicians p2 on v.doctor_id = p2.id
        JOIN nurses n on v.nurse_id = n.id
        JOIN visit_treatment vt on v.id = vt.visit_id
        JOIN treatment t on vt.treatment_id = t.id
        JOIN diseases d on v.disease_id = d.id
        JOIN drugs d2 on d2.id=t.drug_id
        WHERE p.id=1
        ORDER BY 1 DESC;
```

Get Doctor Info:

```
SELECT DISTINCT CONCAT(p.first_name, ' ', p.last_name), h.name,

        CONCAT(h.street, ' ', h.city, ', ', h.state, ' ', h.zip), specialty, rating FROM patients
JOIN visits v on patients.id = v.patient_id

        JOIN hospitals h on v.hospital_id = h.id

        JOIN physicians p on v.doctor_id = p.id

        WHERE patients.id = 1;
```

Get Insurance Info:

```
SELECT i.id AS insurance_ID, i.name

        FROM patients p

        LEFT JOIN insurance i ON p.insurance_id = i.id

        WHERE p.id=1
```

Get Medicine/Drug Info:

```
SELECT v.date,

    h.name AS hospital_name,

    CONCAT(p2.first_name, ' ', p2.last_name) AS doctor_name,

    d2.name,

    t.quantity,

    t.drug_id

  FROM patients p

  JOIN visits v on p.id = v.patient_id

  JOIN hospitals h on v.hospital_id = h.id

  JOIN physicians p2 on v.doctor_id = p2.id

  JOIN visit_treatment vt on v.id = vt.visit_id

  JOIN treatment t on vt.treatment_id = t.id
```

```
JOIN diseases d on v.disease_id = d.id

JOIN drugs d2 on d2.id=t.drug_id

WHERE p.id=1 AND drug_id<>0

ORDER BY 1 DESC;
```

Get Hospital Info:

```
SELECT DISTINCT h.name,

        CONCAT(h.street, ' ', h.city, ', ', h.state, ' ', h.zip), h.phone_number, h.website FROM patients JOIN visits v on patients.id = v.patient_id

          JOIN hospitals h on v.hospital_id = h.id

          JOIN physicians p on v.doctor_id = p.id

          WHERE patients.id = 1;
```

Get Nearby Hospitals (by state so we get more results in demo, but can be changed to zip code):

```
SELECT h.name AS hospital_name, CONCAT(h.street, ' ', h.city, ', ', h.state, ' ', h.zip) AS address, h.phone_number, h.website

        FROM patients p

        LEFT JOIN insurance i ON p.insurance_id = i.id

        LEFT JOIN insurance_hospital ih ON i.id = ih.insurance_id

        LEFT JOIN hospitals h ON ih.hospital_id=h.id

        WHERE p.id=1 AND h.state = p.state;
```

Get Recommended Doctors (Doctors in nearby hospitals):

```
SELECT CONCAT(d.first_name, ' ', d.last_name), h.name, d.specialty, d.rating

   FROM hospitals h JOIN physicians d ON h.id = d.hospital_id WHERE h.id IN (

          SELECT h.id

          FROM patients p
```

```
        LEFT JOIN insurance i ON p.insurance_id = i.id

        LEFT JOIN insurance_hospital ih ON i.id = ih.insurance_id

        LEFT JOIN hospitals h ON ih.hospital_id=h.id

        WHERE p.id=1 AND h.state = p.state;
```

Get Support Group (people with same disease by zip code):

```
SELECT CONCAT(sub.first_name,' ', sub.last_name) AS name, sub.phone_number, sub.email,
sub.name AS disease_name, sub.zip

   FROM (

    SELECT p.first_name, p.last_name,p.phone_number, p.email, p.zip,v.disease_id,
d.name,SUBSTRING(zip,1,1) AS sub_zip

    FROM patients p

    JOIN visits v on p.id = v.patient_id

    JOIN diseases d on v.disease_id = d.id

    WHERE v.disease_id IN (

      SELECT disease_id

      FROM patients

      JOIN visits v2 on patients.id = v2.patient_id

      WHERE patients.id=1 AND disease_id<>0)

    AND p.id<>1) AS sub

  JOIN

    (SELECT SUBSTRING(zip,1,1) AS sub_zip

    FROM patients p2

    WHERE p2.id=1 ) sub1

    ON sub.sub_zip=sub1.sub_zip;
```