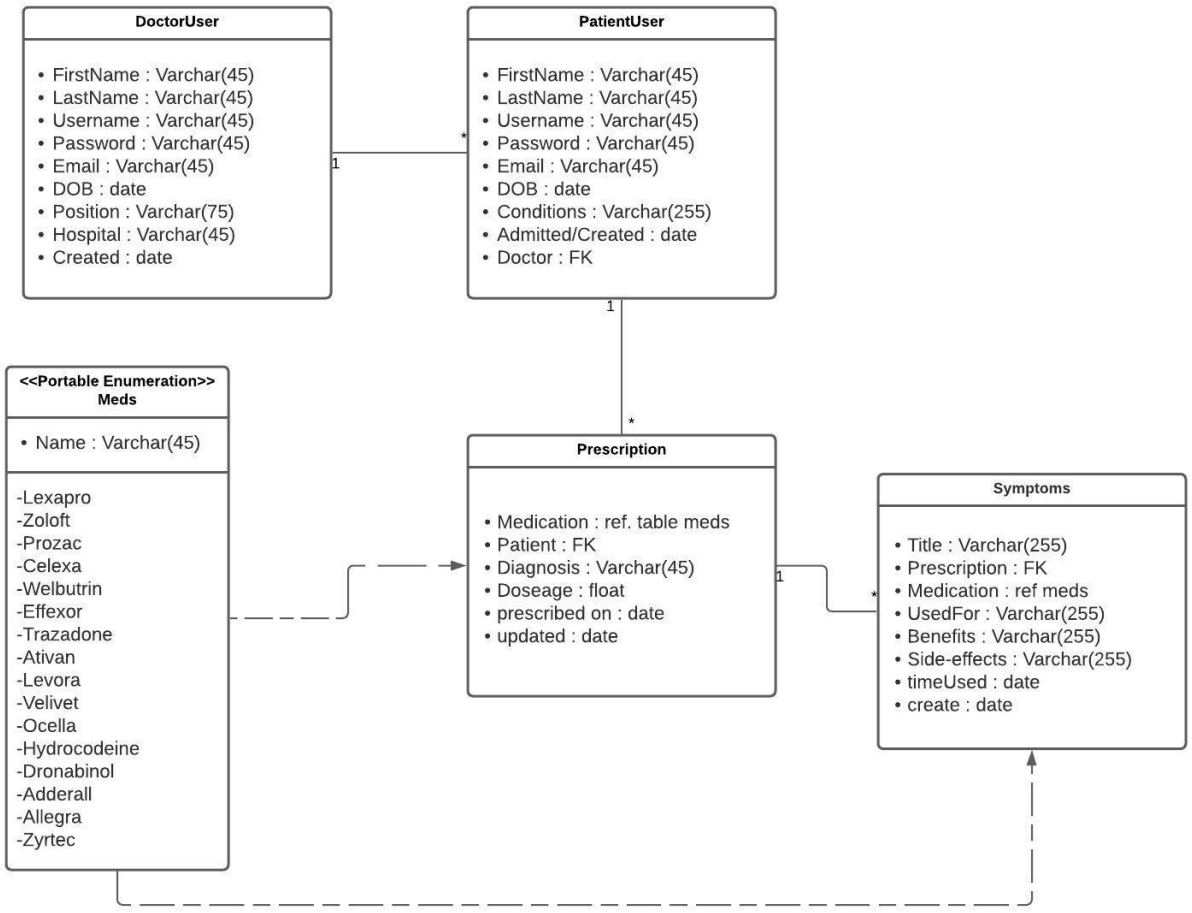
PROJECT NAME: Doctor to Patient Medication and Symptom Database

Project Team: Group 34

Member: Noah Evantash

After getting my first covid vaccine, I was given a pamphlet to access a patient portal to report symptoms or problems following the vaccine. This inspired my idea for the Database project; a server for doctors to track their patients' medications and those patients' symptoms. The design of the database is straight forward, with two types of users (doctors and patients), a domain object for the patients' prescriptions, and another domain object for those prescription's symptoms. The relationship of the tables is between doctors to patients, patients to prescriptions, and prescriptions to symptoms. The following UML diagram depicts the data model for this database:



There will be two types of users – doctors and patients. Under both are the server user requirements (firstname, lastname, username, password, email, DOB).

Doctor User Data Model:

- Firstname : Varchar(45) -> first name of the doctor
- Lastname : Varchar(45) -> last name of the doctor

- Username : Varchar(45) -> chosen username by the doctor
- Password : Varchar(45) -> chosen password by the doctor
- Email : Varchar(45) -> Email of the doctor
- DOB : date -> Date of birth
- Position : Varchar(75) -> Position of the doctor in hospital, description of field they are in
- Hospital : Varchar(45) -> Hospital where the doctor is employed
- Created : date -> timestamp from creation of user

Patient User Data Model:

- FirstName : Varchar(45) -> first name of patient
- LastName : Varchar(45) -> last name of patient
- Username : Varchar(45) -> username chosen by patient
- Password : Varchar(45) -> password chosen by patient
- Email : Varchar(45) -> email of the patient
- DOB : date -> date of birth of the patient
- Conditions : Varchar(255) -> Description of what the patient's conditions are, usually what they are being treated for.
- Admitted/Created : date -> Timestamp from when they were added to the system.
- Doctor : ref(doctor.id) -> a foreign key referencing the doctor's id

Beyond the user models are the prescription and symptoms domain models. The prescriptions domain model stores all the patient's prescriptions by referencing the id of the patient with a foreign key and using a portable enumerator to store which medications are acceptable in the model. The portable enumerator is set up at such:

Medication Portable Enumerator:

- Name : Varchar (45) -> The name of the medication be prescribed

The portable enumerator is populated with a number of samples medications and is referenced by the prescriptions domain object to describe which medicine a patient is prescribed. The prescriptions domain object is set up as such:

Prescriptions Domain Object:

- Medication : ref(medication.name) -> References the portable enumerator to describe which medicine is being prescribed.
- Patient : ref(patients.id) -> References a patient from the Patient User data model to describe which medication they are being prescribed.
- Diagnosis : Varchar(45) -> A description of why the patient is prescribed this medication.
- Doseage : float -> The amount of medicine prescribed, quantified in mg
- prescribed : date -> The timestamp the prescription was created
- updated : date -> the timestamp from when the prescription was last updated (usually a change in dose)

The last table made is the symptoms domain object, which describes symptoms patients have with their prescriptions. This domain object has a one-to-many relationship with the Prescriptions domain object

by referencing the primary key of the prescription. This is because a patient can write multiple symptoms for a prescription but cannot write symptoms for multiple medications together. The setup for the symptoms domain object is such:

Symptoms Domain Object:

- Title : Varchar(255) -> A description or intro the patient's complaint
- Prescription : ref(prescription.id) -> Foreign key relating to the prescription complaint
- Medication : ref (meds.name) -> Refencing the medications
- UsedFor : Varchar(255) -> a description of what the med was used for
- Benefits : Varchar(255) -> a description of the benefits from the medicine
- Side-effects : Varchar(255) -> a description of the side-effects/complaints from the medicine
- timeUsed : date -> the user inputs when they first started noticing these symptoms
- create : date -> when the post was created.

The models are created with the files: ***create_doctors.sql, create_patients.sql, create_prescriptions.sql,*** and ***create_symptoms.sql*****.** The final thing to create was the medications portable enumerator, which had to be populated with several examples of medications. The model is therefore created and populated in the ***create_medications_portable_enum.sql*** file. The rest of the models are populated with the files: ***populate_doctors.sql, populate_patients.sql, populate_prescriptions.sql,*** and ***populate_symptoms.sql.*** These files implement and populate the diagram as such:

- ***Populate_doctors.sql:*** Two doctors (Psychiatrist and General Physician)
- ***Populate_patients.sql:*** Four patients for each doctor (8 total)
- ***Populate_prescriptions:*** Each patient has at least one prescription, 3 patients have 2 prescriptions, and one patient has 3 prescriptions (13 total)
- ***Populate_symptoms:*** Write symptoms for 7 of the 13 prescriptions. Some symptoms can be only positive, some only negative.

This is the final setup of the model diagram. Looking forward, the function of this database can be broken down by each user and what their ability should be extending to creating, editing, updating, and deleting. For doctors, they should be able to add new patients to their care, prescribe them new medication, update their medication/dosage, or delete a medication prescription. A patient should only be able to update and/or create new reports of symptoms for their prescriptions. Additionally, because this is a medical database, confidentiality should be considered in who can view what. For example, a patient should only be limited to view their prescriptions and symptoms, with a reference to their doctor (since patients can only have 1 doctor). The doctor can view more however, but cannot view patients, prescriptions, or symptoms belonging to another doctor. Therefore, the UI will look be set up as such:

- **Doctor service**
    - Create new patient, find patient by id, delete patient, update patient.
    - Create new prescription, find prescription(s) by patient, delete prescription, update prescription.
    - Find symptoms by patient (cannot write/edit symptoms, that is for the patient)
    - ViewAllPatients -> returns all the patients under this doctor care.
    - ViewAllPrescription -> returns all the prescriptions prescribed by this doctor.

    - ViewAllSymptoms -> returns all the symptoms by patients under this doctor's care.
- **Patient service**
    - View their doctor as a reference to the relative patient.
    - Create new symptoms report, update symptoms, delete symptom, view symptoms.
    - ViewPrescriptions -> returns all prescriptions for this patient.
    - ViewSymptoms -> returns all the symptoms this patient reported.