**Pharmacy Database:** Service for Doctors to Receive Information for Their Patients Medications and Experiences.

**Team**: Noah Evantash

**Problem Statement:** The problem this database design seeks to resolve is patients trying to relay information to their doctor about a new medication they are on. How can they effectively tell their doctor how the medication makes them feel through a web service that records the prescriptions of these patients. Doctors are then able to view their patients, prescriptions, and the symptoms of those prescriptions. The database allows doctors to record which patients are under their care for what conditions, what those patient’s prescriptions are, and the symptoms those patients are experiencing.

**Solution Statement:** The solution created to help track these patients taking new medication is to create a database that keeps records of these patients, their prescriptions, medications, and symptoms. To do this, I must create a database on MySQL that represents this database as done in part 1. Next, I must create a relational mapping of a java model for the database diagram that translates the SQL code into java. Lastly, and unsuccessfully, I wanted to create a UI that allowed doctors to implement new patients and write prescriptions for them.

**Users:** While patients are usually a User class, in the UI it exists as a relational object of the doctors, so it can be treated similarly to a domain object with one-to-many relationships to another domain object prescriptions. Doctors are the primary user implemented in this design and has objects patients that can be created and edited under the patient’s tab of whatever doctor that patients belong’ to. For each doctor, there is an inline-editor script created that allows the doctors conditions to be changed and updated in the main list interface. Under the doctors is a patient’s tab that connects to the patients list of that specific doctor. Those patients can be edited in an inline editor that updates or creates the patients’ parameters. There is a back on that list that returns to the list of all the other objects.

**Domain Objects:** The objects in this database are the prescriptions and the symptoms, as well as the patients described above in relation to the doctor users. For the other domain objects, the doctor user is able to create new prescriptions under the patients tab. The patients have a tab that relays which prescriptions they have and allows for creating and editing of new prescriptions with CCUD (cancel, create, update, delete) editor format. Once the prescription is made, the only updatable parameter should be the dosage. In the inline editor, it would only be allowing changing of the dosage and updating of that parameter (which returns the whole body, but only with the updated dose). This is to keep prescriptions in line and prevent changing the medication which was a difficult parameter to implement due to it being a portable enumeration. Therefore, to mitigate errors of passing an invalid medication, we only want the medication to be changed in the editor with a dropdown list of acceptable medications. If I had more time, I would have included another form for the medications that allows the doctor to create new medication names. This was an issue since I was not sure how to include the new meds in the dropdown list. The final list was the symptoms list which showed all the reactions to the prescription. Under each prescription in the list under the main screen, the user can create new symptoms for the prescription. Each symptom allows the user to relay information about the efficacy, benefits, and/or side effects of the prescription given. Such information can be used by the doctor to access their patient’s information of prescriptions and symptoms.

**Note On functionality:** As of today, the only services that have been working for the design UI is the doctors and patients. There was a litany of unforeseen errors and complications relating to development of the UI that made it unreadable with the index.html. The object relational mapping is working as intended and the user can receive information from the database using localhost:8080/api/… but it does not yet fully extend to the UI implementation. So far, the UI only works up to the patients list trying to get prescriptions but runs into a Reach.createElement error that has not been fixed yet. Therefore, the project is only finished and requires more time to try and correct these problems. Nonetheless, the database is set up to create new doctors, add new patients to those doctors, edit those doctors, or delete those doctors. The patients can be accessed through whichever doctor they are a part of and edited or deleted. This functionality is supposed to be extended to the prescriptions and symptoms models but isn’t able to due to the errors.