Jonah Dubbs-Nadeau
Quinn Wilkins

URL: http://34.68.24.30/ (make sure to use http and NOT https)

Feedback from Peers & Fixes Based on Feedback

One major issue that our peers brought to our attention was that, upon creating new DOCTORS and LOCATIONS, the newly-inserted records would not show up on the View Doctors and View Locations pages. We realized that this was because the query that selected the DOCTORS would only select DOCTORS who had LOCATIONS assigned to them, and likewise the query that selected the LOCATIONS would only select LOCATIONS with DOCTORS on-site. This was an issue because the create logic for DOCTORS and LOCATIONS was not inserting these relationships into the database backend. Now, both aspects of this issue have been fixed: the create logic for DOCTORS and LOCATIONS now successfully inserts the relationship between the two entities into the database, and the query for selecting DOCTORS and LOCATIONS has been modified to a left join rather than a full inner join.

Another very helpful piece of feedback that one of our peers had was to look into JavaScript's async/await functionality, rather than nesting callback after callback. After exploring how we could use async/await in our application, we decided to use it, and it has been enormously helpful in getting our queries to work, particularly in cases where a variable number of queries is required.

All in all, there were no changes to our outline and schema. The ability to schedule, reschedule, and cancel APPOINTMENTS has been added to our site since last week, along with full CRUD functionality.

Up-to-Date Outline

Project Title
Oregon Family Medical Appointment Tracker

Overview
Oregon Family Medical is a chain of doctor's offices that serves over 100,000 patients annually in the state of Oregon. A database-driven website will record **APPOINTMENTS** scheduled between **DOCTORS** and **PATIENTS** at various **LOCATIONS** across the state. The website will allow for the creation of new **PATIENTS**, who will choose their primary **DOCTOR** and primary **LOCATION** upon creating their profile. Once a **PATIENT** profile is created, the **PATIENT** can then book open **APPOINTMENTS** with their primary **DOCTOR** at their primary **LOCATION**. They can also reschedule and cancel existing booked **APPOINTMENTS**. Open **APPOINTMENT** slots can be filtered by month, day of the week, and time. The website also allows for the

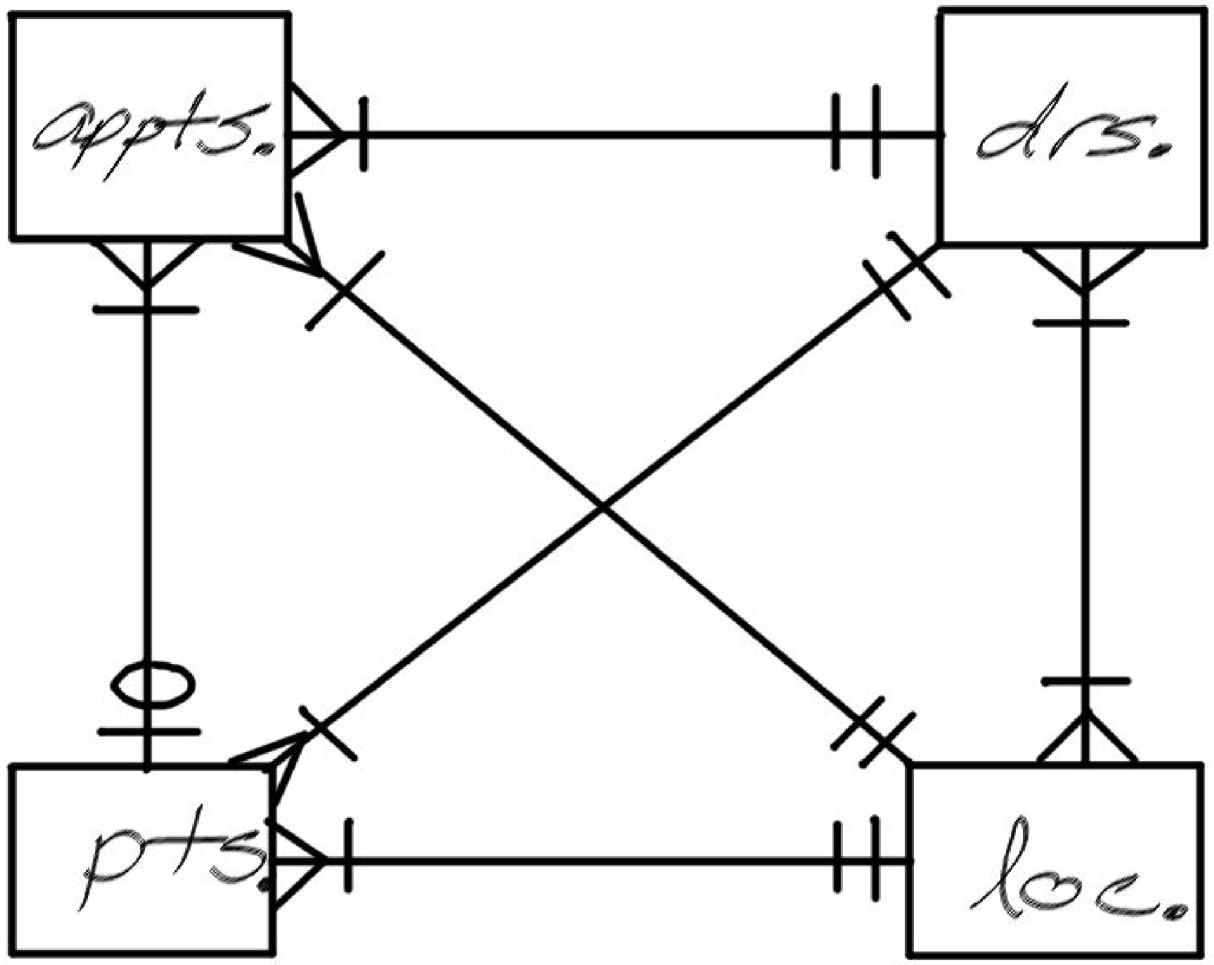
viewing of all **DOCTORS** and **LOCATIONS**, as well as creation, update, and deletion of **DOCTORS** and **LOCATIONS** by the site administrator(s). Each **LOCATION** has multiple **DOCTORS** on-site, and each **DOCTOR** can serve at multiple **LOCATIONS**.

Entity Outline

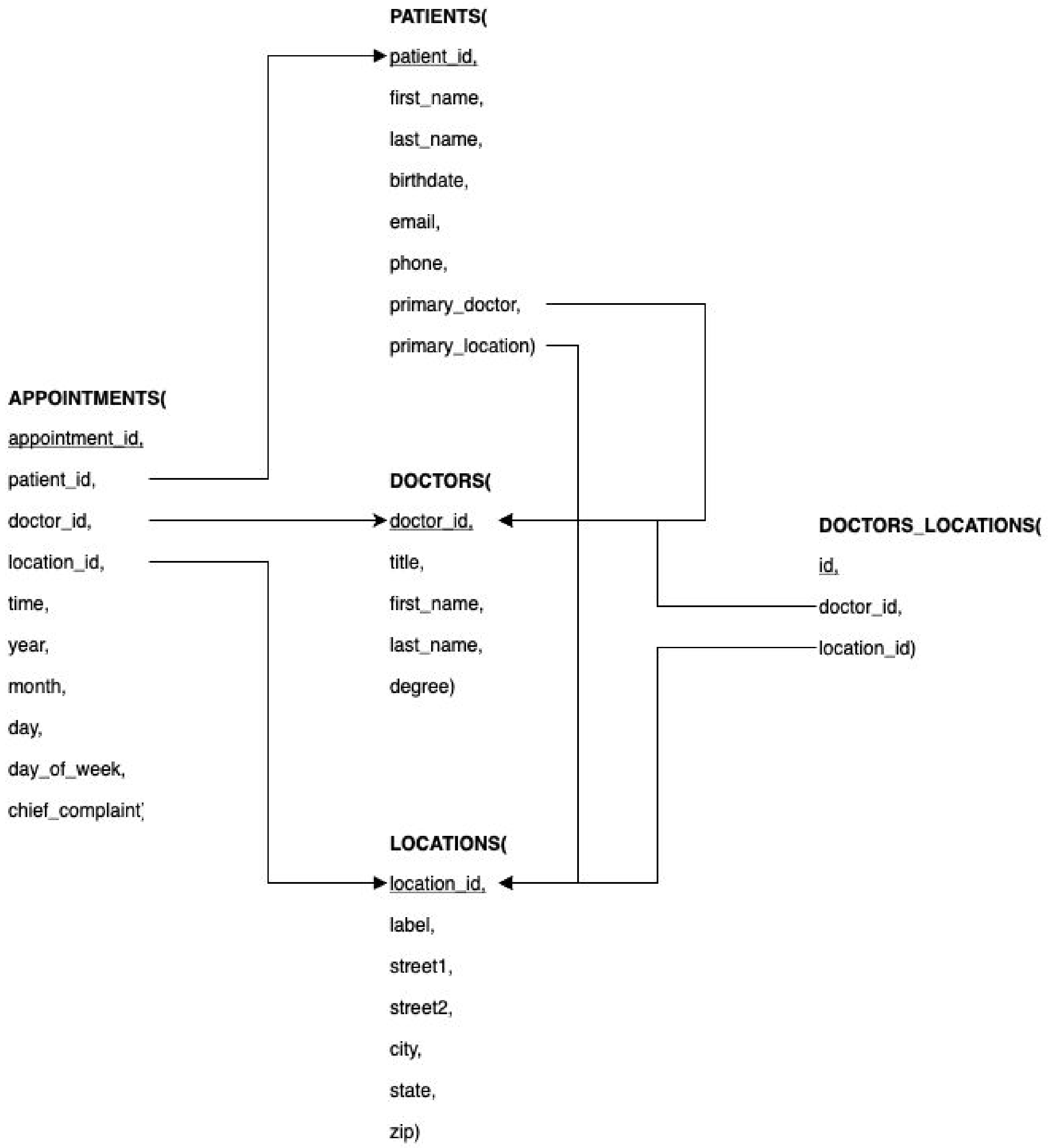
- **PATIENTS**: records the details of the patients who visit Oregon Family Medical
    - Attributes
        - patient_id: int, auto_increment, unique, not NULL, PK
        - first_name: varchar, not NULL
        - last_name: varchar, not NULL
        - birthdate: date, not NULL
        - email: varchar, not NULL
        - phone: varchar, not NULL
        - primary_doctor: int, FK
        - primary_location: int, FK
    - Relationships
        - A 1:M relationship between **PATIENTS** and **APPOINTMENTS** is implemented with patient_id as a FK inside of **APPOINTMENTS**. The FK can be NULL if the **APPOINTMENT** slot has not yet been booked by a **PATIENT**.
        - A 1:1 relationship between **PATIENTS** and **DOCTORS** is implemented with doctor_id as a FK inside of **PATIENTS**.
        - A 1:1 relationship between **PATIENTS** and **LOCATIONS** is implemented with location_id as a FK inside of **PATIENTS**.
- **DOCTORS**: records the details of the doctors who work for Oregon Family Medical
    - Attributes
        - doctor_id: int, auto_increment, unique, not NULL, PK
        - title: varchar, not NULL
        - first_name: varchar, not NULL
        - last_name: varchar, not NULL
        - degree: varchar, not NULL
    - Relationships
        - A 1:M relationship between **DOCTORS** and **APPOINTMENTS** is implemented with doctor_id as a FK inside of **APPOINTMENTS**.
        - A 1:M relationship between **DOCTORS** and **PATIENTS** is implemented with doctor_id as a FK inside of **PATIENTS**.
        - A M:M relationship between **DOCTORS** and **LOCATIONS** is implemented as a separate table containing doctor_id as a FK and location_id as a FK.
- **LOCATIONS:** records the details of the different Oregon Family Medical locations
    - Attributes
        - location_id: int, auto_increment, unique, not NULL, PK
        - label: varchar, unique, not NULL

        - street1: varchar, unique, not NULL
        - street2: varchar, unique
        - city: varchar, not NULL
        - state: varchar, not NULL
        - zip: int, not NULL
    - Relationships
        - A 1:M relationship between **LOCATIONS** and **APPOINTMENTS** is implemented with location_id as a FK inside of **APPOINTMENTS**.
        - A 1:M relationship between **LOCATIONS** and **PATIENTS** is implemented with location_id as a FK inside of **PATIENTS**.
        - A M:M relationship between **LOCATIONS** and **DOCTORS** is implemented as a separate table containing doctor_id as a FK and location_id as a FK.
- **APPOINTMENTS:** records the details of all appointment slots (booked and unbooked)
    - Attributes
        - appointment_id: int, auto_increment, unique, not NULL, PK
        - patient_id: int, FK
        - doctor_id: int, not NULL, FK
        - location_id: int, not NULL, FK
        - time: int, not NULL
        - year: int, not NULL
        - month: int, not NULL
        - day: int, not NULL
        - day_of_week: int, not NULL
        - chief_complaint: varchar
    - Relationships
        - A 1:1 relationship between **APPOINTMENTS** and **PATIENTS** is implemented with patient_id as a FK inside of **APPOINTMENTS**. The FK can be NULL if the **APPOINTMENT** slot has not yet been booked by a **PATIENT**.
        - A 1:1 relationship between **APPOINTMENTS** and **DOCTORS** is implemented with doctor_id as a FK inside of **APPOINTMENTS**.
        - A 1:1 relationship between **APPOINTMENTS** and **LOCATIONS** is implemented with location_id as a FK inside of **APPOINTMENTS**.

Entity-Relationship Diagram



Schema


PATIENTS(
patient_id,
first_name,
last_name,
birthdate,
email,
phone,
primary_doctor,
primary_location)
APPOINTMENTS(
appointment_id,
patient_id,
doctor_id,
location_id,
time,
year,
month,
day,
day_of_week,
chief_complaint)
DOCTORS(
doctor_id,
title,
first_name,
last_name,
degree)
DOCTORS_LOCATIONS(
id,
doctor_id,
location_id)
LOCATIONS(
location_id,
label,
street1,
street2,
city,
state,
zip)