NYU Tandon School of Engineering
Computer Science and Engineering
CS6083, Spring 2021

April 12, 2021

Project #1 (due 4/26)

The goal of our course project is to build a web-based system for signing up people for COVID-19 vaccinations. This is an extension of the schema from one of the midterm exams, with additional features such as preferred timeslots, travel distance limitations, and different priority levels. As before, we assume single-shot vaccinations.

Problem Outline: In the completed system, there will be three types of participants: Patients, providers, and administrators. Patients can sign up in the system, and provide necessary information such as name, SSN, date of birth, address, phone, and email, and maybe additional information that can be used to determine the priority group the patient belongs to. Patients may also indicate preferred times during the week when they would like to be scheduled for the vaccination. Providers are places such as hospitals, doctor's offices, pharmacies, schools, etc., that provide vaccinations. Providers need to sign up with information such as their name, phone number, and address. Registered providers can then upload available vaccination slots to the system, usually at least a few days or weeks in advance. You may assume that all vaccinations by one provider happen at the same location – if a hospital or pharmacy has several locations, they need to register as different providers. The third type of participant is the administrator of the database system, who defines priority groups, assigns patients to these groups, makes sure that vaccination slots are allocated to patients based on priority and time preferences, and messages patients and providers about appointments.

For simplicity, we omit any issues of payments or insurance, and assume vaccinations are free. We also assume that the administrator has some way to assign people to the correct priority group; e.g., assume that patients can upload additional documents as PDF or image files that the administrator can use, either manually or algorithmically, in combination with age, to determine the priority group. In this first project, you also do not have to come up with any algorithm for matching patients to available appointments – for now, assume that the system periodically (say, once every few hour) runs an algorithm that matches currently available appointments (new appointments, or appointments not confirmed, rejected, or canceled by other patients) with eligible patients. Finally, you do not have to model different types of providers, such as hospitals or doctors, beyond maybe an attribute ***providertype*** for each provider.

To describe the process in more detail, patients sign up by providing their information, and receive a username (say, their email) and password. Their priority group is initially set to NULL, until the administrator assigns them to a group. Priority groups are numbered from 1 to some number N, so that group #1 has the highest priority, #2 the next highest, and so on. The administrator also maintains for each group a date when they qualify for vaccination, say January 1 for group #1, January 15 for group #2, etc. Patients also provide a "calendar" of their availability during the week. It is recommended to keep this feature simple – e.g., you could split each day into 4-hour blocks, and patients can select blocks where they are most likely to be available or that are convenient for them. Thus, a patient might select every weekday 8am-noon, or Tuesday noon-4pm and Friday 8am-noon, or any time on the weekend. You should assume that the schedule is the same every week, though patients could later decide to update their schedule. Patients also provide a maximum distance they are willing to travel to get to the appointment, e.g., 30 miles.

Providers that have signed up periodically upload available appointments to the system. Note that for larger providers, there may be several appointments at the same time. The administrator periodically runs an algorithm that tries to match available appointments with patients, in a way that takes eligibility, priority, availability, and distance into account, though the details of this algorithm will only become relevant in the second part of the project. Afterwards, selected patients will be notified to offer them the appointment, and the patients will have some limited amount of time to either accept or decline the appointment. If the patient accepts, the provider needs to be notified about the patient having accepted the appointment. An appointment may also be cancelled later, or a patient may fail to show up. All this information about what offers were sent to patients, whether they accepted or declined, if they cancelled later or did not show up, should also be stored in the database. (For this part of the project, you do not have to worry about how the notifications are done, whether via email, or text message, etc.)

Project Guidelines: In this first part of the course project, you should focus on designing a suitable relational schema that can be used to store data and perform queries for this system - including information such as patients, providers, priority groups, appointments, and proposed matches between patients and appointments and their outcomes (accepted, declined, cancelled, did not show up). In the second part of the project, you will need to build a web-accessible front-end interface, with a back-end server interacting with your database, which should implement the functions mentioned and allow patients and providers to access the service through a web browser.

You should use your own database system on your laptop or an internet-accessible server. Use a system that supports full text search operators similar to "contains". Most of you will probably use mySQL, SQL Server, Postgres, or Oracle - if you want to use another system, please ask the instructor for permission.

Please note that you need to implement user system and content access control inside your web application logic, which means that there should not be a separate DBMS account for each user! Instead, the web application itself will log into the database using an account created for it. Thus, the system you implement can see all the content, but has to make sure at the application level that each logged-in user is identified through the use of cookies in the second part of the project. Both parts of the project may be done individually or in teams of two students. You will receive an announcement from the TAs asking you to sign up as a group or an individual. The second part of the project will be due around May 10. The second project builds on top of this one, so you cannot skip this project.

Project Steps: Following is a list of steps for this part of the project. Note that in this first part, you will only deal with the database side of this project, and a suitable web interface will be designed and implemented in the second part. However, you should already envision and plan the interface that you intend to implement.

(a) Design, justify, and create an appropriate relational schema for the above scenario. Make sure your schema avoids redundancies. Show an E-R diagram of your design, and a translation into relational format. Identify keys and foreign key constraints. Note that you may have to revisit your design if it turns out later that the design is not suitable.

(b) Use a database system to create the database schema, together with key, foreign key, and other constraints.

(c) Write SQL queries (or sequences of SQL queries or scripting language statements) for the following tasks. You may use suitable placeholder values in the statements.

(1) Create a new patient account, together with email, password, name, date of birth, etc.
(2) Insert a new appointment offered by provider.
(3) Write a query that, for a given patient, finds all available (not currently assigned) appointments that satisfy the constraints on the patient's weekly schedule, sorted by increasing distance from the user's home address.
(4) For each priority group, list the number of patients that have already received the vaccination, the number of patients currently scheduled for an appointment, and the number of patients still waiting for an appointment.
(5) For each patient, output the name and the date when the patient becomes eligible for vaccination.
(6) Output all patients that have cancelled at least 3 appointments, or that did not show up for at least two confirmed appointments that they did not cancel.
(7) Output the name of the provider that has performed the largest number of vaccinations.

(d) Populate your database with some sample data, and test the queries you have written in part (c). Make sure to input interesting and meaningful data. Limit yourself to a few patients, providers, priority classes, etc., but make sure there is enough data to generate interesting test cases for the above queries. It is suggested that you design your test data very carefully. Draw and submit a little picture of your tables that fits on one or two pages and that illustrates your test data! (Do not submit a long list of insert statements, but show the resulting tables.) Print out and submit your testing.

(e) Document and log your design and testing appropriately. Submit a properly documented description and justification of your entire design, including E-R diagrams, tables, constraints, queries, procedures, and tests on sample data, and a few pages of description. This should be a paper of say 10-12 pages with introduction, explanations, E-R and other diagrams, etc., which you will then revise and expand in the second part.