5/18/2021

Database Web Application

Covid Vaccine System

Final Project Part II – Web & Database Design
Principles of Database Systems (CS-6083)
Prof. Torsten Suel

Prepared By: Iffat Rahman (ir543) & Dipak Patel (dp3148)

TABLE OF CONTENTS

PROJECT PART 2 – COVID SYSTEM DESIGN

1. INTRODUCTION

The aim of this project is to build a web-based system for signing up people for COVID-19 vaccinations. The system contains three types of participants namely Patients, Providers, and Administrators. Patients can sign up in the system and provide personal information and preferences to assist with offering them vaccination appointments. Providers are places such as pharmacies, doctor's offices, governments, and others., that provide vaccinations. Providers need to sign up with their information which will allow them to upload available vaccination appointments to the system. Finally, administrators of the database system will be able to define priority groups, assign patients to these groups, make sure that vaccination slots are allocated to patients based on their priority group and time preferences. The following report seeks to outline in detail how the system will work based on the database and front end design. It will also provide some sample test data for queries and frontend UI screenshots to observe the system in action.

2. TECHNOLOGY USAGE

The system will utilize MYSQL database for the backend design, and PHP, HTML, Javascript for the frontend UI design.

To keep our database safe from the SQL Injection Attacks and the cross site scripting, we've applied some of these main prevention methods:

1) Using Prepared Statements (with Parameterized Queries)
Using Prepared Statements is one of the best ways to prevent SQL injection. It's also simple to write and easier to understand than dynamic SQL queries. This is where the SQL Command uses a parameter instead of inserting the values directly into the command, thus preventing the backend from running malicious queries that are harmful to the database.

2) Using Stored Procedures
Stored Procedures adds an extra security layer to our database beside using Prepared Statements. It performs the escaping required so that the app treats input as data to be operated on rather than SQL code to be executed. The difference between prepared statements and stored procedures is that the SQL code for a stored procedure is written and stored in the database server, and then called from the web app. If user access to the database is only ever permitted via stored procedures, permission for users to directly access data doesn't need to be explicitly granted on any database table. This way, our database is still safe.

3) Validating user input
We do an input validation first to make sure the value is of the accepted type, length, format, etc. Only the input which passed the validation can be processed to the database. It's like checking who is at the door of your house before you open it and let them in. The system utilizes JavaScript and the HTML input pattern checks to ensure malformed inputs are not past to the server and the database.

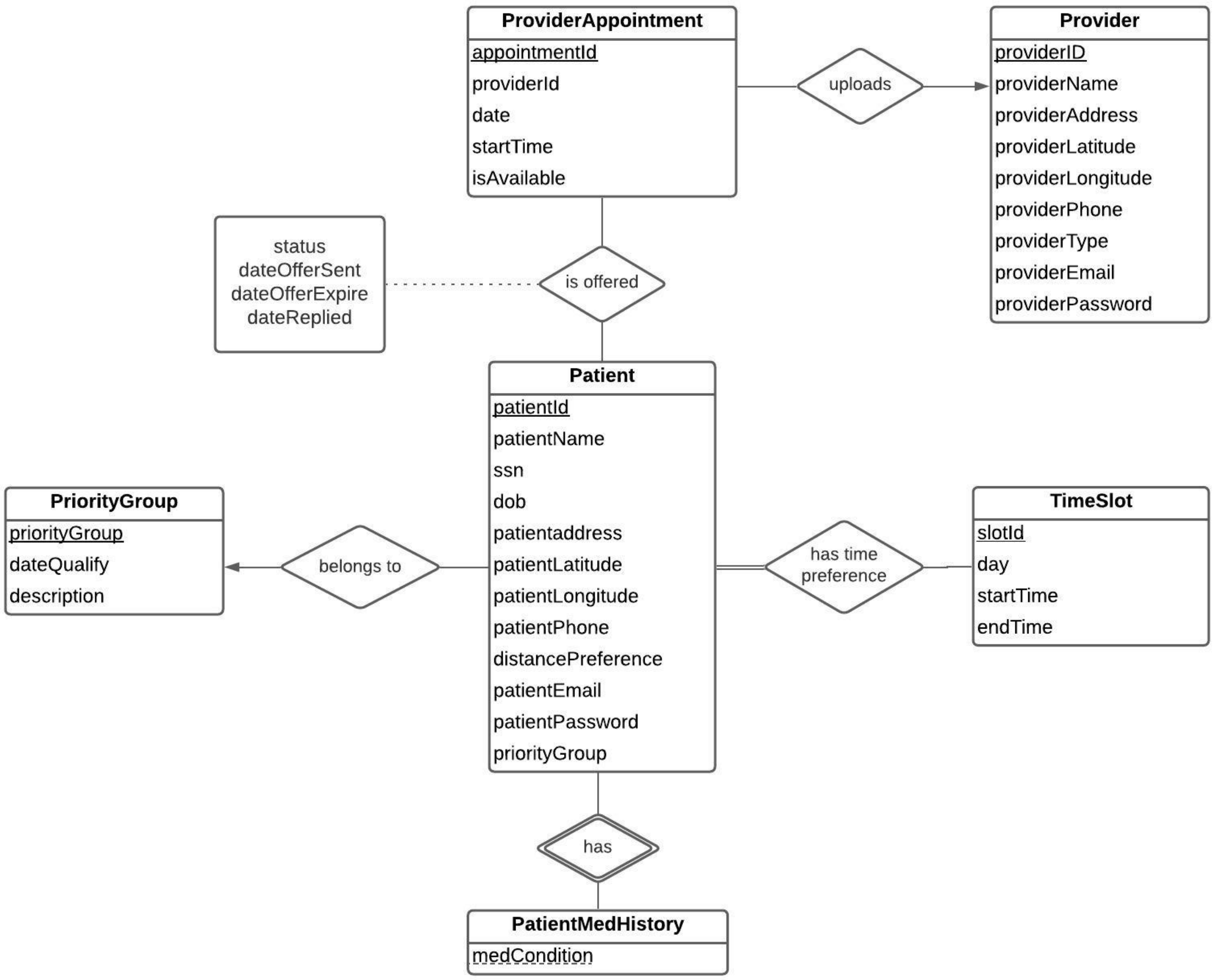
3. DATABASE DESIGN

The database will be designed to capture provider detail, appointments uploaded by each provider, patient detail with their medical conditions and weekly availability, active priority groups released based on governments guidelines, patient assignment to priority groups, and appointment offers sent to users, along with their statuses. Administrators will not be captured in the database design as they are in charge of the maintenance of the database and running of scripts related to the system functionality.

The rest of this section will cover different aspects of the overall design by laying out the Entity Relationship Diagram, Relational Schema and any assumptions/explanation for the design choices made for the project.

3.1. ENTITY RELATIONSHIP DIAGRAM

This section illustrates the ER Diagram with the Entities and Relationships for the Covid Vaccine System.



<u>Strong Entities:</u>

- **Provider** represents providers who are registered with the system and are uniquely identified by providerId
- **Patient** represents patients who are registered with the system and are uniquely identified by the patientId
- **ProviderAppointment** represents the appointments uploaded by the Providers in the system and are each identified by appointmentId
- **PriorityGroup** represents the priority groups for vaccine eligibility and are continuously added based on new government guidelines and requirements. Patients are added to priority groups if they meet the requirements associated with the respective group. So, there can be patients in the system who do not belong to a priority group.
- **TimeSlot** represents 3 time slots per day corresponding to morning, afternoon and evening; so, there will be 21 slots (7day x 3 slots/day) that patients can use to indicate their time availability.

<u>Weak Entities:</u>

- **PatientMedHistory** represents medical conditions of Patients if they are diagnosed with one.

<u>Explanation of the Relationships:</u>

- **Patient** and **PatientMedHistory:** A patient can have many medCondition and a medCondition can have many patients since the assumption is a patient should be able to list out their medical conditions, if needed.
- **Patient** and **TimeSlot:** A patient can select many slotIds and a slotId can be selected by many patients. Also, it is total participation for Patient Table since the assumption is every patient must select a time slot.
- **Patient** and **PriorityGroup:** A patient may only belong to one priority group, but a priorityGroup can have many patients. A patient might not have a priorityGroup when they are registered into the system or if they don't match the eligibility criteria for the priorityGroup.
- **Provider** and **ProviderAppointment:** A poriver can upload many appointments, but an appointment will have only one provider.
- **Patient and ProviderAppointment:** A patient can be offered multiple appointments and an appointment can be offered to multiple patients. The assumption here is a patient is offered one appointment at any given time, and if the appointment offer expires or is declined/cancelled, it can be offered to another patient if there is still time. In the preliminary system, the assumption is that the patient will have 24hrs to respond to an appointment offer.

3.2. RELATIONAL SCHEMA

This section covers the design of the Relational Schema for the COVID Vaccine System.

SCHEMA

Note: Table Names are bolded. Primary Keys are underlined.

Patient (<u>patientId</u>, patientName, ssn, dob, patientAddress, patientLatitude, patientLongitude, patientPhone, distancePreference, priortyGroup, patientEmail, patientPassword)
Assumptions/Explanation:

- patientId uniquely identifies the patients.
- patientLatitude and patientLongitude represents latitude and longitude of the patient’s address, which will be calculated using Google Map or Open Map API.
- distancePreference indicates the maximum distance in miles a patient would be willing to travel to a vaccination appointment.
- patientEmail and patientPassword will be used to store credentials for patients to login to the website. The credentials will be created when a patient registers in the system.
- priortyGroup will be initially NULL after a patient registers and will be assigned a priority group every night at midnight using a scheduled event that calls a priority group assignment function. If a patient does not meet all the criteria for any of the priority groups in the system at the time of assignment logic execution, their priorityGroup will remain NULL.

PatientMedHistory (<u>patientId</u>, <u>medCondition</u>)
Foreign Key: *patientId* references *patientId* in *Patient*
Assumptions/Explanation:

- medCondition attributes will capture medical conditions such as diabetes, asthma, high cholesterol, cancer and so forth.

TimeSlot (<u>slotId</u>, day, startTime, endTime)
Assumptions/Explanation:

- slotId defines specific time slots of startTime and endTime.
- There are 3-time blocks in a day, and they are {(startTime: 8 AM, endTime: 12 PM), (startTime: 12 PM, endTime: 4 PM), (startTime: 4 PM, endTime: 8 PM)}, which will correspond to morning, afternoon and evening.
- There are 7 possible values for day: {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}

PatientTimePreference (<u>patientId</u>, <u>slotId</u>)
Foreign Key:

- *patientId* references *patientId* in *Patient*

- *slotId* references *slotId* in *TimeSlot*

Assumptions/Explanation:

- A patient can have multiple time preferences.

PriorityGroup (priorityGroup, dateQualify, description)

Assumptions & Explanation:

- priorityGroup is a number from 1 to N which identifies the priority group a patient is assigned with priorityGroup 1 being the highest priority.
- dateQualify is the date when the patient's priority group becomes eligible for the covid vaccination.
- description captures the criteria/requirement to be eligible for the priority group.
- Priority groups are added dynamically based on the eligibility guidelines from the government.

Provider (providerId, providerName, provideAddress, providerLatitude, providerLongitude, providePhone, providerType, providerEmail, providerPassword)

Assumptions/Explanation:

- values of providerType: {'pharmacies', 'doctor's offices', 'governments', 'other'}.
- providerLatitude and providerLongitude represent the latitude and longitude of the provider's address, which will be calculated using Google Map API.
- providerEmail and providerPassword will be used to store credentials for providers to login to the website to upload appointments and update the status of appointments in specific scenarios. The credentials will be created and made available to the provider when they call/email to register with the system.

ProviderAppointment (appointmentId, providerId, date, startTime, isAvailable)

Foreign Key: *providerId* references *providerId* in *Provider*

Assumptions/Explanation:

- appointmentId uniquely identifies each appointment.
- isAvailable attribute is dynamically updated using triggers when a record is inserted into PatientAppointmentOffer table based on the value of in the 'status' column.

PatientInAppointmentOffer (appointmentId, patientId, status, dateOfferSent, dateOfferExpire, dateReplied)

Foreign Key:

- *appointmentId* references *appointmentId* in *ProviderAppointment*
- *patientId* references *patientId* in *Patient*

Assumptions/Explanation:

- When a Patient is matched with an appointment, a record will be inserted into PatientInAppointmentOffer.
  The patient to appointment matching logic is based on priority group, patients schedule, and distance preference and available appointments. Appointment will be offered to a patient one at a time.
  Note: This patient to appointment matching is a Python Script that will be set up as a CRON Job see details later.
- status stores the status of appointment at any given point of time and can have one of the following values: {notified, accepted, declined, expired, cancelled, vaccinated, noshow}.
- dateOfferSent stores the date and time of when the appointment offer was sent.
- dateOfferExpire is a calculated attribute based on the following calculation (dateOfferSent + 2 Day).
- dateReplied captures the date the patient replied to the original appointment offer.
- Patients can accept or decline offers until dateOfferExpire, and if they do not reply the status is updated to expired.
- They can also later cancel, or may not show up, or may successfully get the shot, and in each scenario the status will be updated accordingly.

Note: A general assumption for the Database design is that the system is built to be used in the United States, which will be taken into consideration when choosing the data type and size of country specific attributes, like phone number, ssn, etc... However, the system can be scaled to other countries by modifying the country specific attributes.

3.3. STORED PROCEDURES

This section captures the procedures utilized in the system during data retrieval and insertion from the front-end, as well as the Appointment to Patient Matching Script.

3.3.1. Stored Procedures for Data Retrieval

1. The `getUserInfo` stored procedure retrieves user data based their user type using patientId or providerId, which is call in the front-end when a user logs in to the system:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getUserInfo`(IN id INT, type VARCHAR(10))
BEGIN
IF type = "patient"
THEN
    SELECT * FROM CovidVaccineSystem.Patient
    WHERE patientId = id;
ELSEIF type = "provider"
THEN
    SELECT * FROM CovidVaccineSystem.Provider
    WHERE providerId = id;
END IF;
END
```

2. The `getPatientTimePreference` stored procedure retrieves patient specific Time Preference/Availability information from the database and is called by the patient preference page in the front-end on load and data refresh.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getPatientTimePreference`(IN id INT)
BEGIN
    SELECT patientId, slotId, day, startTime, endTime FROM PatientTimePreference pt
    NATURAL JOIN TimeSlot ts
    WHERE patientId = id
    ORDER BY slotId;
END
```

3. The `getPatientDistancePreference` stored procedure retrieves distance preference, i.e. the maximum distance a patient is willing to travel to get their vaccine, from the database and it is called by the patient preference page in the front-end on load and data refresh.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getPatientDistancePreference`(IN id INT)
BEGIN
    SELECT distancePreference FROM Patient
    WHERE patientId = id;
END
```

4. The `getpatientoffers` stored procedure retrieves any appointment that has been offered to a patient, along with the appointment details. This is called upon the logging into the front end by the patient page.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getpatientoffers`(IN id INT)
BEGIN
    SELECT pao.appointmentId, pao.patientId, p.providerName, p.providerAddress, pa.date,
pa.startTime, pao.dateOfferExpire,
       ROUND((ST_Distance_Sphere(point(pat.patientLatitude, pat.patientLongitude),
point(p.providerLatitude, p.providerLongitude))* 0.00062137),2) AS distance
       FROM PatientAppointmentOffer pao
       INNER JOIN ProviderAppointment pa
       ON pao.appointmentId = pa.appointmentId
       INNER JOIN Provider p
```

```
        ON p.providerId = pa.providerId
        INNER JOIN Patient pat
        ON pat.patientId = pao.patientId
        WHERE pao.patientId = id AND pao.status = "notified";
END
```

5. The `GetPatientMedicalHistory` stored procedure retrieves any medical history, if it exists, for a given patient, and it is called by the Patient Medical History Page in the front-end on page load and data refresh.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetPatientMedicalHistory`(IN id INT)
BEGIN
    SELECT * FROM PatientMedHistory
        WHERE patientId = id;
END
```

6. The stored procedure returns the count of all future scheduled appointments, scheduled appointments for the current day, cancelled appointments for the current day, or no shows for the current day based on the *type* parameter for a given provider. It is called provider homepage on load and on data refresh to display some preliminary statistics on the dashboard.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getCountProviderStat`( IN id INT, type
VARCHAR(55))
BEGIN
    IF type = 'total_vaccinated' THEN
        SELECT count(*) AS count FROM PatientAppointmentOffer
        NATURAL JOIN ProviderAppointment
        WHERE providerId = id AND status = 'vaccinated';
    ELSEIF type = 'total_scheduled'  THEN
        SELECT count(*) AS count FROM PatientAppointmentOffer
        NATURAL JOIN ProviderAppointment
        WHERE providerId = id AND status = 'accepted' AND date >= date(now());
    ELSEIF type = 'total_scheduled_today' THEN
        SELECT count(*) AS count FROM PatientAppointmentOffer
        NATURAL JOIN ProviderAppointment WHERE providerId = id
       AND (status = 'accepted' or status='vaccinated') AND date = date(now());

    ELSEIF type = 'total_cancelled_today'THEN
        SELECT count(*) AS count FROM PatientAppointmentOffer
        NATURAL JOIN ProviderAppointment
        WHERE providerId = id AND status = 'cancelled' AND date = date(now())
        AND appointmentId NOT IN (SELECT appointmentId FROM PatientAppointmentOffer
                                    WHERE status = 'notified' or status = 'accepted');
    ELSEIF type = 'total_noshows' THEN
        SELECT count(*) AS count FROM PatientAppointmentOffer
        NATURAL JOIN ProviderAppointment
        WHERE providerId = id AND status = 'noshow';
        END IF;
        END
```

7. The `getProviderAppointment` stored procedure retrieves future available appointments or appointments that have been confirmed by the user, along with no shows and vaccinated, based on the ask for a given provider. It is called when by ‘Add Appointment’ page or ‘Scheduled Appointment’ upon load and data update/refresh.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getProviderAppointment`(IN providerId INT, cond
VARCHAR(20))
BEGIN
IF cond = "future" THEN
```

```
    SELECT appointmentId, providerId, date, startTime, isAvailable
    FROM ProviderAppointment
    WHERE providerId = providerId
          AND date >= now()
          AND isAvailable = 'Y';

  ELSEIF cond = "confirmed" THEN
    SELECT pa.appointmentId, pa.providerId, pa.date, pa.isAvailable, pa.startTime, pao.status,
    pao.patientId, p.patientName, p.patientEmail, p.patientPhone, pao.dateOfferSent
    FROM ProviderAppointment pa
    LEFT JOIN PatientAppointmentOffer pao ON pao.appointmentId = pa.appointmentId
    LEFT JOIN Patient p ON p.patientId = pao.patientId
    WHERE pa.providerId = providerId AND (pao.status = 'accepted' OR pao.status = 'cancelled'
          OR pao.status = 'noshow' OR pao.status = 'vaccinated')
   ORDER BY pa.appointmentId, pao.dateOfferSent;
  END IF;
  END
```

8. The `findAllApptMatches` stored procedure retrieves all patients along with their matching appointments sorted in the order of their distance preference. It also ensures a patient is not matched with an appointment they have declined before by checking against the appointment offer table. This stored procedure is called by the “patient to appointment” matching algorithm file called matchingAlgo.py when it runs.

```
 CREATE DEFINER=`root`@`localhost` PROCEDURE `findAllApptMatches`()
 BEGIN
 WITH
   EXCLUDED_PATIENTS AS (
     SELECT p.patientId  FROM Patient p
     INNER JOIN PatientAppointmentOffer pao ON pao.patientId = p.patientId
     WHERE pao.status IN ('accepted', 'notified', 'vaccinated')),

   PATIENT_AVAILABILITY AS(
     SELECT p.patientId, p.patientLatitude, p.patientLongitude, p.distancePreference, t.day,
            t.startTime, t.endTime, pg.dateQualify, p.priorityGroup
     FROM `CovidVaccineSystem`.`Patient` p
     INNER JOIN `CovidVaccineSystem`.`PatientTimePreference` pt ON pt.patientId = p.patientId
     INNER JOIN `CovidVaccineSystem`.`TimeSlot` t ON t.slotId = pt.slotId
     LEFT JOIN `CovidVaccineSystem`.`PriorityGroup` pg ON pg.priorityGroup = p.priorityGroup
     WHERE p.patientId NOT IN ( SELECT patientId FROM EXCLUDED_PATIENTS)),

   ALL_MATCHING_APPOINTMENTS AS (
     SELECT pa.patientId, a.appointmentId, pa.priorityGroup,
            ST_Distance_Sphere(point(pa.patientLatitude, pa.patientLongitude),
            point(p.providerLatitude, p.providerLongitude))* 0.00062137 as distAppt,
            ROW_NUMBER() OVER (PARTITION BY pa.patientId ORDER BY
            ST_Distance_Sphere(point(pa.patientLatitude, pa.patientLongitude),
            point(p.providerLatitude, p.providerLongitude))* 0.00062137) ASC) AS RowRank
     FROM `CovidVaccineSystem`.`ProviderAppointment` a
     CROSS JOIN PATIENT_AVAILABILITY pa
     INNER JOIN `CovidVaccineSystem`.`Provider` p ON p.providerId = a.providerId
     WHERE a.isAvailable = 'Y'
     AND pa.dateQualify IS NOT NULL
     AND a.date >= pa.dateQualify
     AND a.date >= DATE(now() + INTERVAL 2 DAY)
     AND DAYNAME(a.date) = pa.day
```

```
        AND a.startTime >= pa.startTime AND a.startTime < pa.endTime
        AND ST_Distance_Sphere(point(pa.patientLatitude, pa.patientLongitude),
            point(p.providerLatitude, p.providerLongitude))* 0.00062137 <= pa.distancePreference
        ORDER BY pa.patientId, RowRank ASC)

      SELECT DISTINCT(pa.appointmentId), pa.patientId, pa.priorityGroup, pa.distAppt, pa.RowRank
      FROM ALL_MATCHING_APPOINTMENTS pa
      LEFT JOIN  `CovidVaccineSystem`.`PatientAppointmentOffer` pao
      ON pa.patientId = pao.patientId
      WHERE (pa.appointmentId <> pao.appointmentId OR pao.appointmentId IS NULL);
    END
```

3.3.2. Stored Procedures for Data Insertion/Update

1. The UpdateDistancePreference stored procedure updates the distancePreference column in the Patient table for a given patient, and it is called when a user updates the distance preference on the Patient Preference page in the front-end.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `UpdateDistancePreference`(IN id INT, distance
INT)
BEGIN
    UPDATE Patient
    SET distancePreference = distance
    WHERE patientId = id;
END
```

2. The UpdateApptStatus stored procedure updates the status column in the PatientAppointmentOffer table for an appointmentId and patientId combination based on the passed status parameter. This stored procedure is called when a patient accepts, declines or cancels an appointment in the front-end and when a provider updates the status of an appointment to vaccinated or no show on the day of a scheduled appointment.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `UpdateApptStatus`(IN id INT, apptId INT, status
VARCHAR(10))
BEGIN
    UPDATE PatientAppointmentOffer
    SET status = status
    WHERE appointmentId = apptId AND patientId = id;
END
```

3. The updatePassword stored procedure updates the password in the database for a patient or a provider, based on the parameters passed. This is called when a patient or a provider resets their password on the respective profile pages.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `updatePassword`(IN id INT, type VARCHAR(10),
password VARCHAR(255))
BEGIN
IF type = "patient" THEN
    UPDATE Patient
    SET patientPassword = password
    WHERE patientId = id;
ELSEIF type = "provider" THEN
    UPDATE Provider
    SET providerPassword = password
    WHERE providerId = id;
END IF;
END
```

4. The updatePatientProfile stored procedure is called when a patient updates their email, date of birth, phone number or address, along with the latitude and longitude associated with the address. The latitude and longitude are automatically calculated in the front-end using GOOGLE Maps API.
   Note: Patients are not allowed to update their name and ssn directed on the UI, instead they are instructed to reach our support team.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `updatePatientProfile`(IN id INT, address
VARCHAR(255),email VARCHAR(255), dob DATE, phone CHAR(10), longitude DECIMAL(11,8), latitude
DECIMAL(11,8))
BEGIN
UPDATE Patient
SET patientAddress = address, patientEmail = email,
    dob = dob, patientPhone = phone,
    patientLongitude = longitude, patientLatitude = latitude
WHERE patientId = id;
END
```

5. The UpdateProviderProfile stored procedure is called when a provider updates their email, phone number or address, along with the latitude and longitude associated with the address. The latitude and longitude are automatically calculated in the front-end using GOOGLE Maps API.
   Note: Providers are not allowed to update their name directly on the UI, instead they are instructed to reach our support team.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `UpdateProviderProfile`(IN id INT, address
VARCHAR(255),email VARCHAR(255), phone CHAR(10), longitude DECIMAL(11,8), latitude
DECIMAL(11,8))
BEGIN
UPDATE Provider
SET providerAddress = address, providerEmail = email, providerPhone = phone,
    providerLongitude = longitude, providerLatitude = latitude
    WHERE providerId = id;
END
```

6. The stored procedure UpdateApptAvailability updates the isAvailable column in the ProviderAppoinment table based on the status passed as parameter. This is called by 2 triggers associated with the PatientAppointmentOffer table when the status for the appointment offer is updated. (Reference Section 3.4 for the detail of the triggers)

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `UpdateApptAvailability`(IN apptID INT, status
CHAR(50))
BEGIN
    IF status = 'vaccinated' OR  status = 'notified' OR  status = 'accepted' OR status =
'noshow'
    THEN
        UPDATE `CovidVaccineSystem`.`ProviderAppointment`
        SET `isAvailable` = 'N'
        WHERE `appointmentId` = apptID;
    ELSE
        UPDATE `CovidVaccineSystem`.`ProviderAppointment`
        SET `isAvailable` = 'Y'
        WHERE `appointmentId` = apptID;
    END IF;
END
```

3.4. TRIGGERS

This section captures the triggers that have been used in the database for table maintenance

1. Patients are uniquely identified in the system using patientId instead of ssn due to privacy constraints, so in order to ensure a patient can create only one account, a trigger is used to check if a patient with the same ssn already exists in the database prior to insertion.

```
-- Trigger to prevent duplicate account creation
DELIMITER $$
CREATE TRIGGER CheckDuplicateAccount
BEFORE INSERT
ON `CovidVaccineSystem`.`Patient`  FOR EACH ROW
BEGIN
    IF
      NEW.ssn IN (SELECT ssn FROM `CovidVaccineSystem`.`Patient` WHERE ssn = NEW.ssn)
    THEN
       SIGNAL SQLSTATE '45000' SET message_text = 'An Account with the SSN already exists.';
    END IF;
END $$
DELIMITER ;
```

2. The `CallUpdateApptAvailOnInsert` and `UpdateApptAvailOnUpadte` triggers will keep the isAvailable column in ProviderAppointment table always up to date. The first trigger is fired upon insertion of a new record into the PatientAppointmentOffer table and the second trigger is fired upon the update of the status column of PatientAppointmentOffer table. Both of these triggers call the stored procedure UpdateApptAvailability referenced in section 3.3.

```
/*
   Trigger to call UpdateApptAvailability stored procedure upon insert into
   PatientAppointmentOffer Table
*/
DELIMITER $$
CREATE TRIGGER CallUpdateApptAvailOnInsert
AFTER INSERT ON `CovidVaccineSystem`.`PatientAppointmentOffer` FOR EACH ROW
BEGIN
    CALL UpdateApptAvailability(NEW.appointmentId, NEW.status);
END $$
DELIMITER ;

 /*
    Trigger to call UpdateApptAvailability stored procedure upon update of
    PatientAppointmentOffer Table
 */
 DELIMITER $$
 CREATE TRIGGER UpdateApptAvailOnUpadte
 AFTER UPDATE ON `CovidVaccineSystem`.`PatientAppointmentOffer` FOR EACH ROW
 BEGIN
     CALL UpdateApptAvailability(NEW.appointmentId, NEW.status);
 END $$
 DELIMITER ;
```

3.5. PRIORITY GROUP ASSIGNMENT

3.5.1. Functions

The findPriorityGroup function returns the priority group of a patient based on their age and medical condition. This uses the requirements associated with each priority group and can be updated as new priority groups are added

```
CREATE DEFINER=`root`@`localhost` FUNCTION `findPriorityGroup`(id INT) RETURNS int
       DETERMINISTIC
BEGIN
    DECLARE medCondCount INT;
    DECLARE age INT;

    SELECT count(*) INTO medCondCount FROM PatientMedHistory
    WHERE patientId = id
    GROUP BY patientId;

    SELECT TIMESTAMPDIFF(YEAR, dob, now()) INTO age FROM Patient
    WHERE patientId = id;

    IF medCondCount > 0
        THEN RETURN 1;
    ELSEIF age >= 65
        THEN RETURN 2;
    ELSEIF age >= 45 AND age < 65
        THEN RETURN 3;
    ELSEIF age >= 16 AND age < 45
        THEN RETURN 4;
    ELSE
        RETURN NULL;
    END IF;
END
```

3.5.2. Events

The Assign_Priority_Group event is created to run the findPriorityGroup function on all patients that have a NULL priority group in the database. This event runs everyday at midnight.

```
CREATE EVENT Assign_Priority_Group
ON SCHEDULE
       EVERY 1 DAY
       STARTS str_to_date( date_format(now(), '%Y%m%d 0000'), '%Y%m%d %H%i' ) +
INTERVAL 1 DAY
DO
    WITH Patient_NULL_Priority AS (
    SELECT patientId FROM Patient WHERE priorityGroup IS NULL
    )

    UPDATE Patient
        SET priorityGroup = findPriorityGroup(patientId)
        WHERE patientId IN (SELECT patientId FROM Patient_NULL_Priority);
```

3.6. APPOINTMENT MATCHING ALGORITHM

The application currently uses a naive patient appointment matching algorithm where the goal is to maximize the appointment offers belonging to the groups with top priority. The system has 4 priority groups in the system, following are the group numbers in order of descending priority - 1, 2, 3 and 4.

This algorithm loops through priority group 1 and tries to give each patient their first preference based on distance, while keeping track of any appointment that has already been offered to a patient. If a patient’s first preference is not available, the algorithm loops through the preference in order until til finds an available appointment. In the scenario where a patient's preferred appointment list is exhausted without a matching appointment the patient will not have an offer.

The algorithm retrieves all the patients to appointment matching using the findAllMatchingAppt stored procedure. This stored procedure also ensures that it doesn't send a patient and available appointment combination, where the patient has already been offered that appointment by checking against the existing Appointment offers in the PatientAppointmentOffer table. This script is automated using CRON Job to run every day at midnight. Below is the python script:

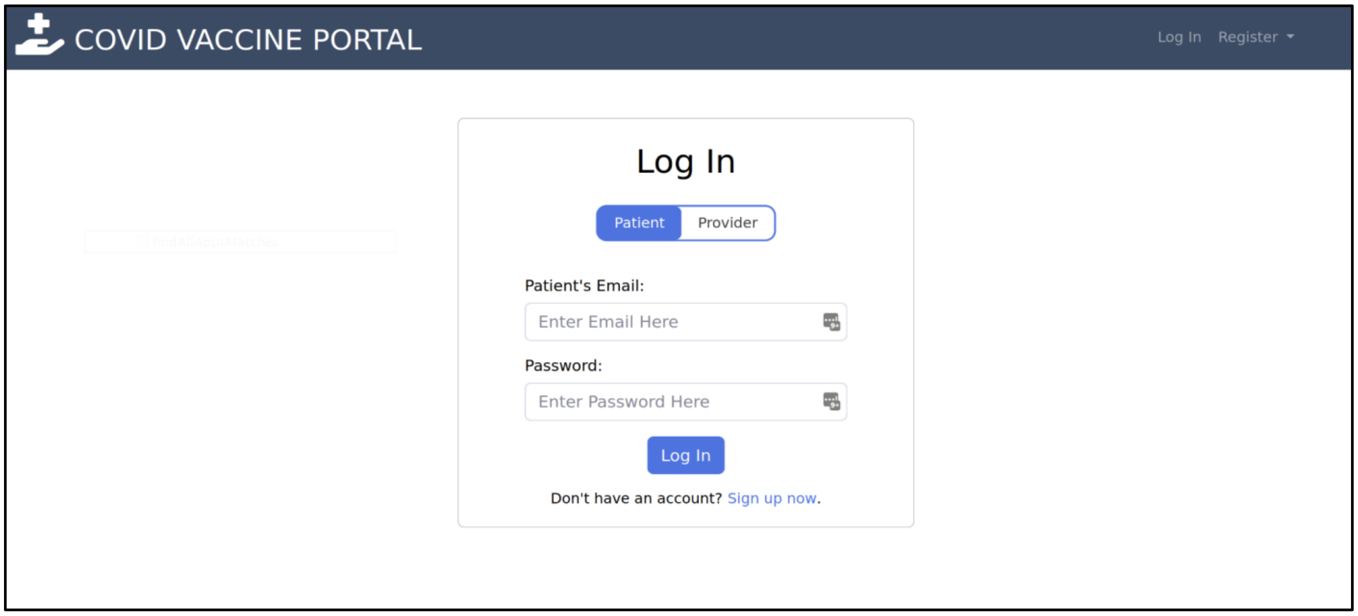
```
import mysql.connector
from mysql.connector import Error
from datetime import datetime

def getAppt():
   try:
       connection = mysql.connector.connect(host='localhost',
                                            database='CovidVaccineSystem',
                                            user='root',
                                            password='Fiffat123!')
       cursor = connection.cursor()
       query2 = ("CALL findAllApptMatches()")
       cursor.execute(query2)
       pg_1 = {}
       pg_2 = {}
       pg_3 = {}
       pg_4 = {}

       seen = set()
       matched = []

       for apptId, patientId, pg, dist, rowrank in cursor:
           if (pg == '1'):
               if patientId not in pg_1:
                   pg_1[patientId] = {apptId: int(rowrank)}
               elif(len(pg_1[patientId]) == 5 ):
                   continue
               else:
                   pg_1[patientId][apptId] = int(rowrank)
           if (pg == '2'):
               if patientId not in pg_2:
                   pg_2[patientId] = {apptId: int(rowrank)}
               elif(len(pg_2[patientId]) == 3):
                   continue
               else:
                   pg_2[patientId][apptId] = int(rowrank)
           if (pg == '3'):
               if patientId not in pg_3:
                   pg_3[patientId] = {apptId: int(rowrank)}
               elif(len(pg_3[patientId]) == 3):
                   continue
               else:
                   pg_3[patientId][apptId] = int(rowrank)
           if (pg == '4'):
               if patientId not in pg_4:
                   pg_4[patientId] = {apptId: int(rowrank)}
               elif(len(pg_4[patientId]) == 3):
                   continue
               else:
                   pg_4[patientId][apptId] = int(rowrank)
        for key, val in pg_1.items():
           for k, v in val.items():
               if k not in seen:
                   matched.append([key,k,v])
                   seen.add(k)
                   break
       for key, val in pg_2.items():
           for k, v in val.items():
               if k not in seen:
                   matched.append([key,k,v])
                   seen.add(k)
                   break
       for key, val in pg_3.items():
           for k, v in val.items():
               if k not in seen:
                   matched.append([key,k,v])
                   seen.add(k)
                   break
       for key, val in pg_4.items():
           for k, v in val.items():
               if k not in seen:
                   matched.append([key,k,v])
                   seen.add(k)
                   break
   except Error as e:
       print("Error while connecting to MySQL", e)
   finally:
       if connection.is_connected():
           cursor.close()
           connection.close()
           print("MySQL connection is closed")
       return matched

if __name__ == "__main__":
   matchings = getAppt()
   if len(matchings) > 0:
       try:
           connection = mysql.connector.connect(host='localhost',
                                               database='CovidVaccineSystem',
                                               user='root',
                                               password='Fiffat123!')
           cursor = connection.cursor()
           query = "INSERT INTO PatientAppointmentOffer(appointmentId, patientId, status,
dateOfferSent) VALUES"
            now = datetime.now()
           formatted_date = now.strftime('%Y-%m-%d %H:%M:%S')

           for i in range(len(matchings)):
               if i < len(matchings)-1:
                   query = query + "('"+str(matchings[i][1])+"', '"+ str(matchings[i][0]) + "',
'notified', '"+ formatted_date +"'),"
               else:
                   query = query + "('"+str(matchings[i][1])+"', '"+ str(matchings[i][0]) + "',
'notified', '"+ formatted_date +"');"
```

```
    print(query)

    cursor.execute(query)
    connection.commit()
except Error as e:
    print("Error while connecting to MySQL", e)
finally:
    if connection.is_connected():
        cursor.close()
        connection.close()
        print("MySQL connection is closed")
```

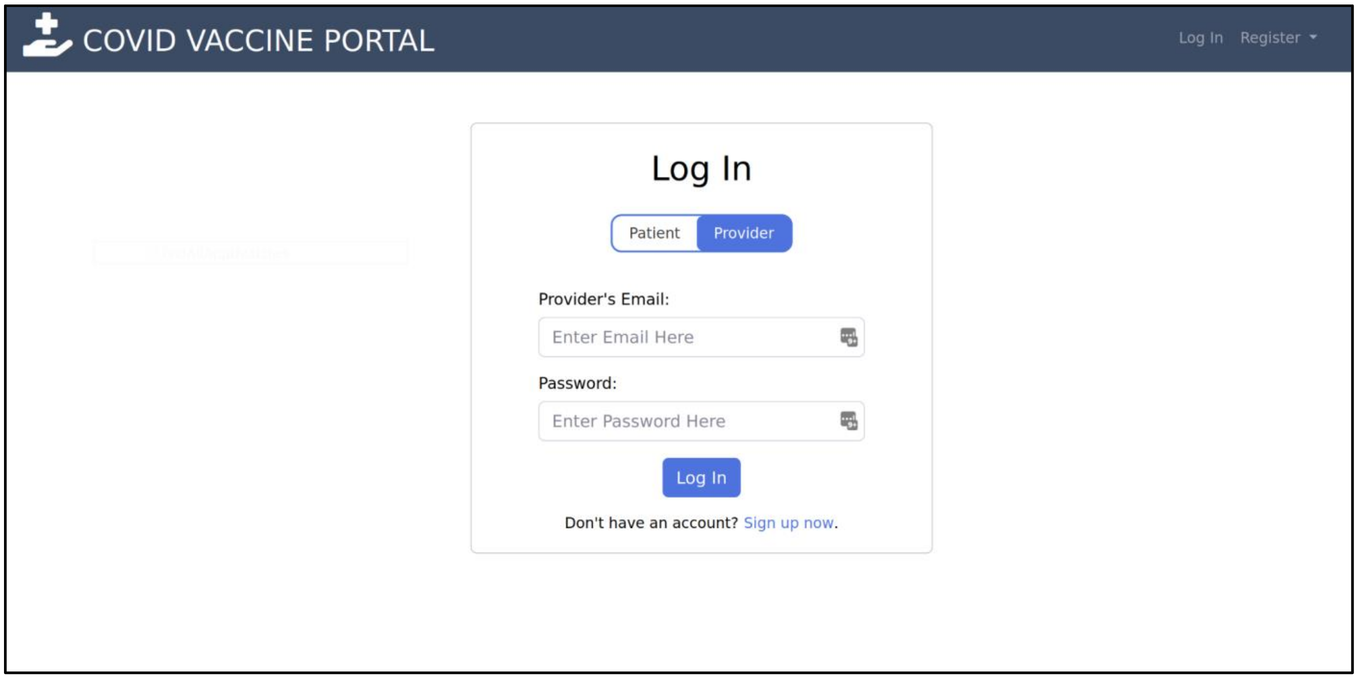
4. FRONT END DESIGN

1. **Login Page (Patient & Provider): The frontend login page uses dynamic pill tabs to differentiate between provider and patient login**

   Patient Login Tab



Provider Login Tab:



Log In

Patient Provider

Provider's Email:

Enter Email Here

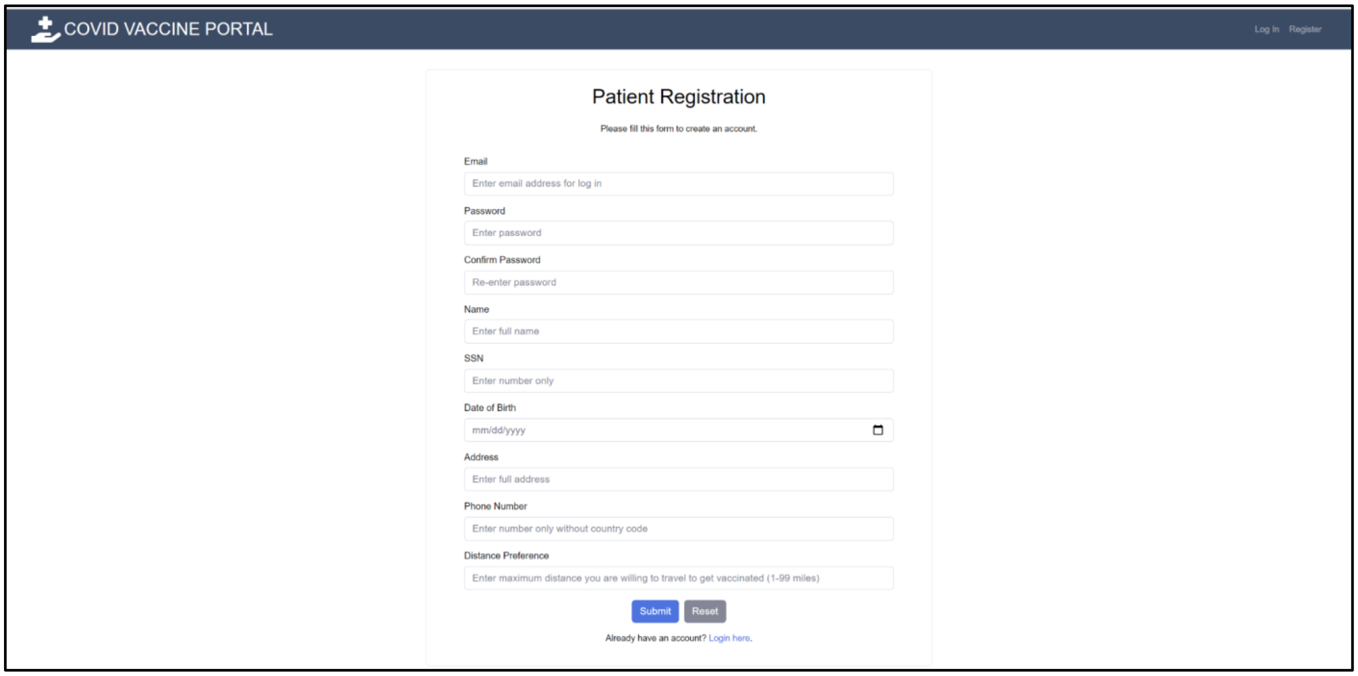
Password:

Enter Password Here

Log In

Don't have an account? Sign up now.

2. **Patient Registration Page: Patients can register with the system using Patient Registration Page shown below**



Patient Registration

Please fill this form to create an account.

Email

Enter email address for log in

Password

Enter password

Confirm Password

Re-enter password

Name

Enter full name

SSN

Enter number only

Date of Birth

mm/dd/yyyy

Address

Enter full address

Phone Number

Enter number only without country code

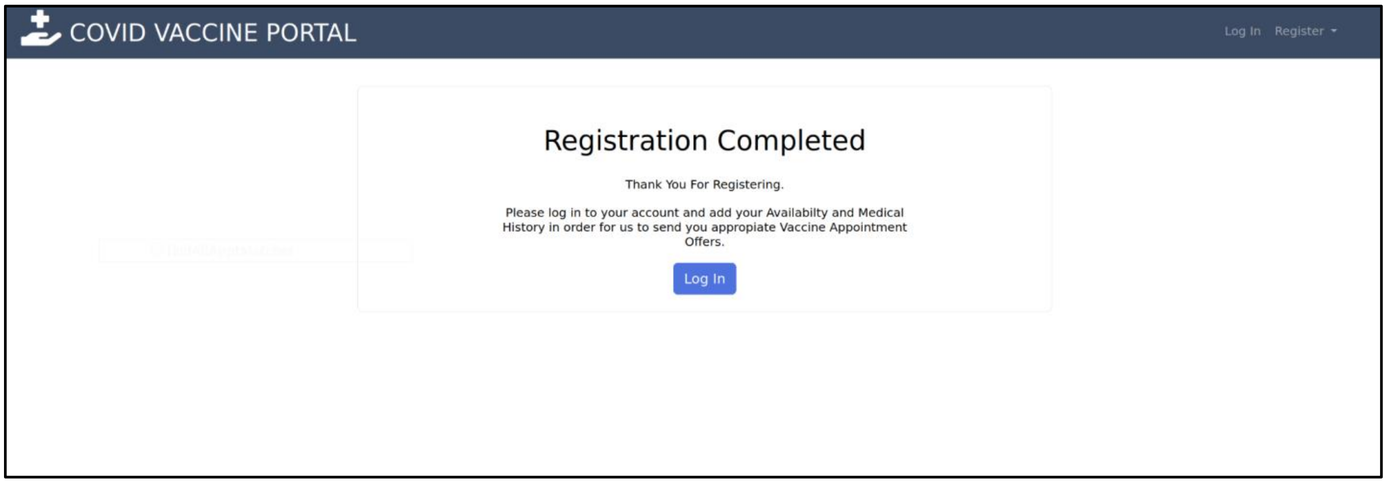
Distance Preference

Enter maximum distance you are willing to travel to get vaccinated (1-99 miles)

Submit Reset

Already have an account? Login here.

3. Patient Registration Confirmation Page

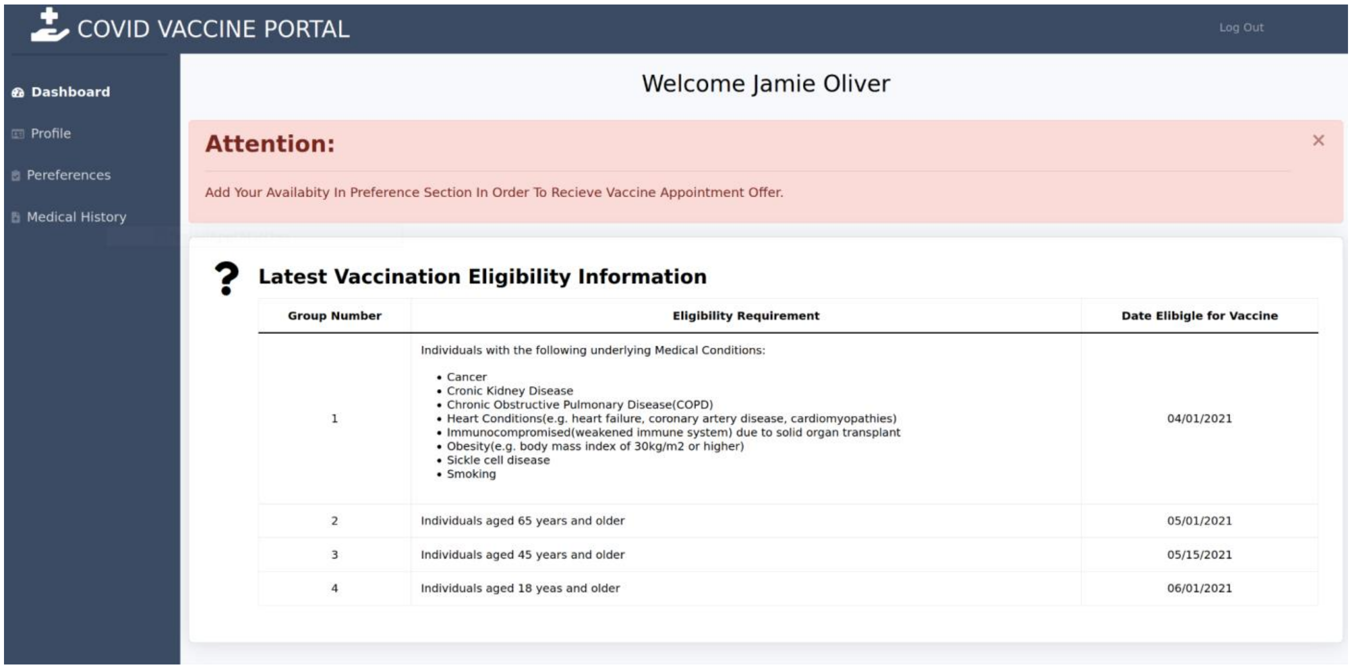


Registration Completed

Thank You For Registering.

Please log in to your account and add your Availabilty and Medical History in order for us to send you appropiate Vaccine Appointment Offers.

4. Patient Home/Dashboard after Logging in for the first time. Patients are asked to fill in their time preference/availability in the Preference Page in order for the system to start sending out appointment offers.



Welcome Jamie Oliver

Attention:

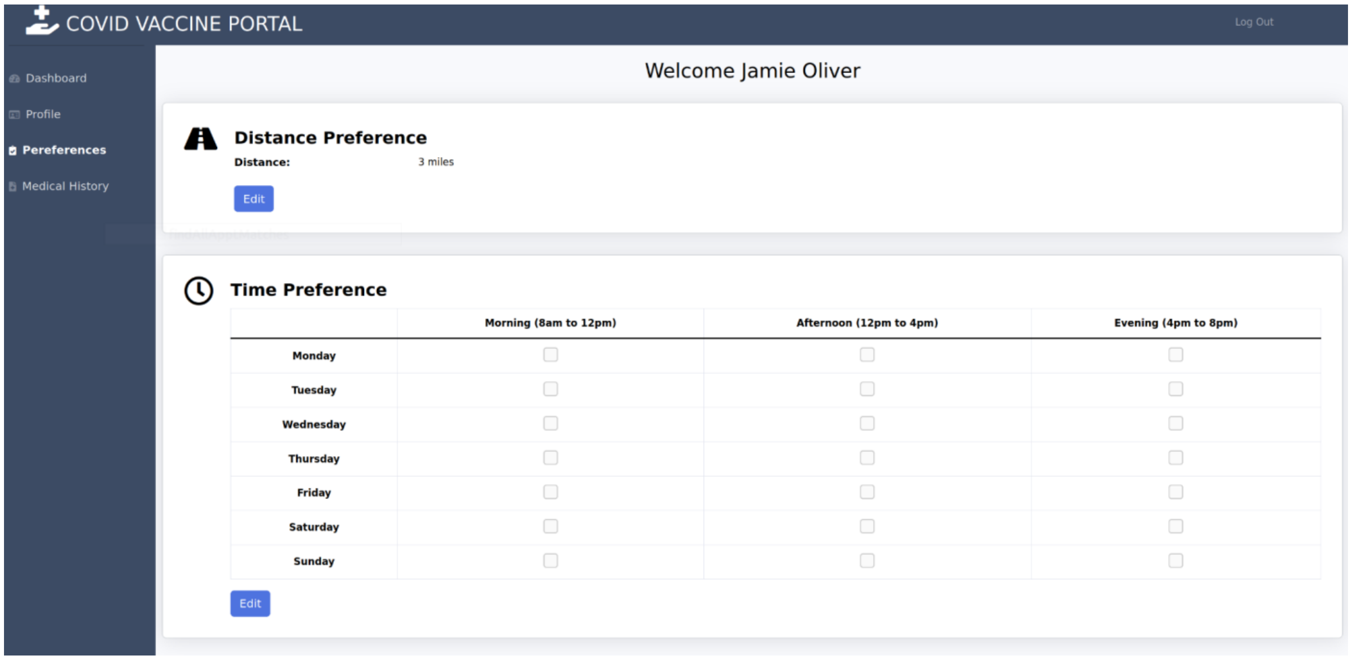
Add Your Availabity In Preference Section In Order To Recieve Vaccine Appointment Offer.

Latest Vaccination Eligibility Information

Group Number	Eligibility Requirement	Date Elibigle for Vaccine
1	Individuals with the following underlying Medical Conditions: • Cancer • Cronic Kidney Disease • Chronic Obstructive Pulmonary Disease(COPD) • Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies) • Immunocompromised(weakened immune system) due to solid organ transplant • Obesity(e.g. body mass index of 30kg/m2 or higher) • Sickle cell disease • Smoking	04/01/2021
2	Individuals aged 65 years and older	05/01/2021
3	Individuals aged 45 years and older	05/15/2021
4	Individuals aged 18 yeas and older	06/01/2021

5. Patient Preference Page where patients can update Distance Preference and Time Preference/Availability. Note: Time Preference Checkboxes disabled in view mode.

Preference Page in View Mode:



Welcome Jamie Oliver

Distance Preference

Distance: 3 miles

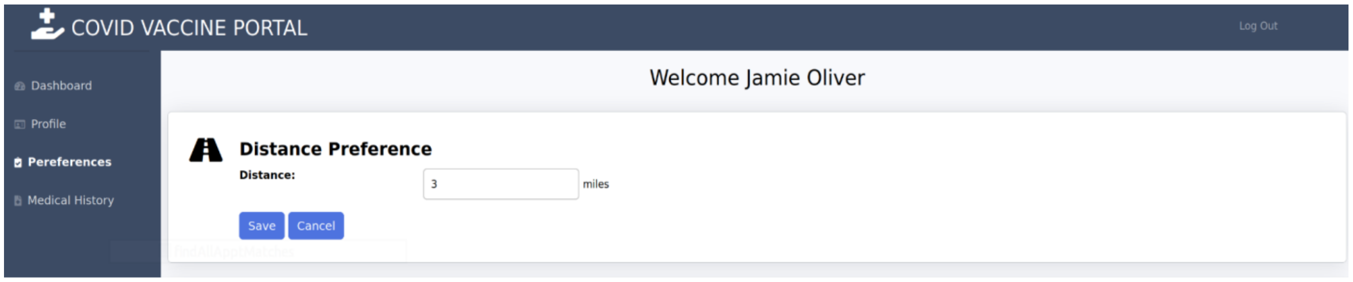
Edit

Time Preference

	Morning (8am to 12pm)	Afternoon (12pm to 4pm)	Evening (4pm to 8pm)
Monday	☐	☐	☐
Tuesday	☐	☐	☐
Wednesday	☐	☐	☐
Thursday	☐	☐	☐
Friday	☐	☐	☐
Saturday	☐	☐	☐
Sunday	☐	☐	☐

Edit

Editing Distance Preference:



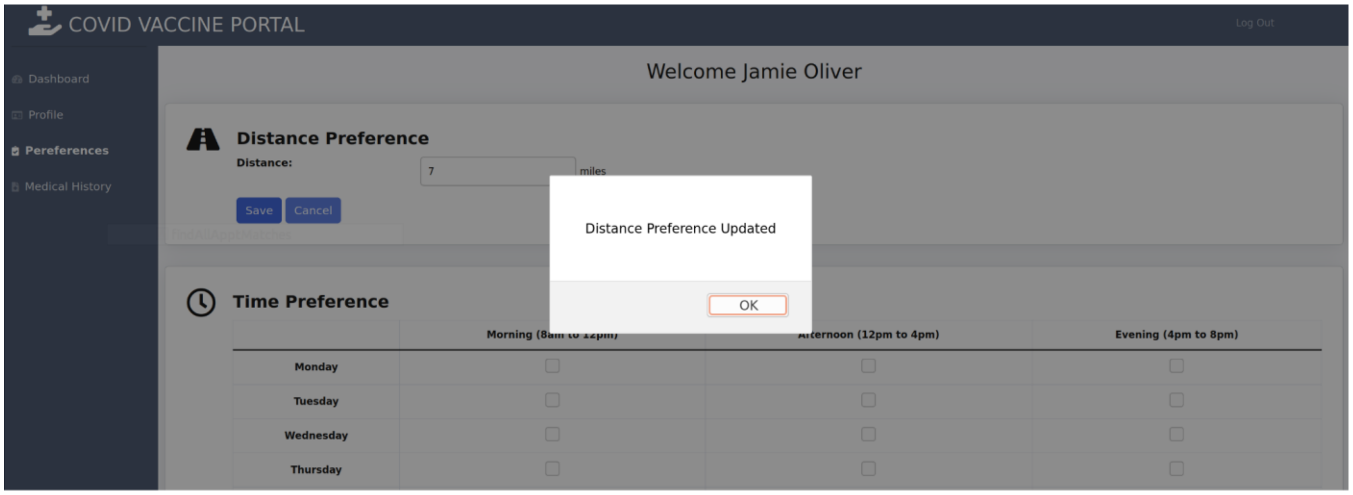
Welcome Jamie Oliver

Distance Preference

Distance: 3 miles

Save Cancel

Changing and Saving New Distance Preference:



Welcome Jamie Oliver

Distance Preference

Distance: 7 miles

Save Cancel

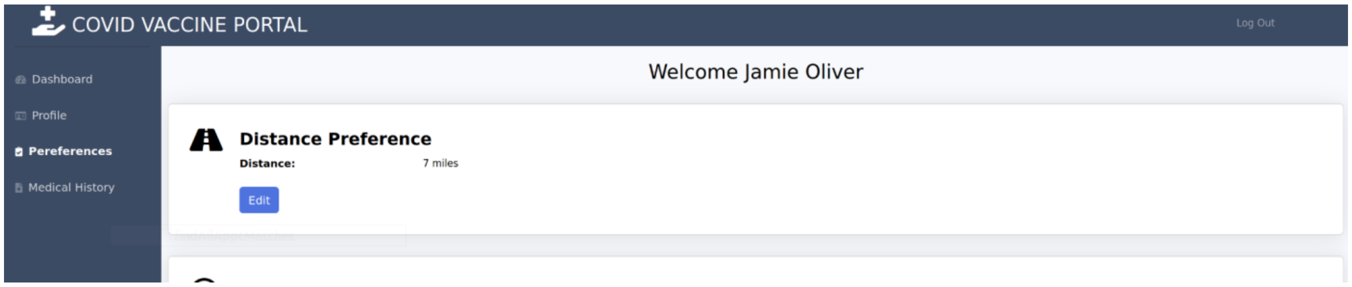
Distance Preference Updated

OK

Time Preference

	Morning (8am to 12pm)	Afternoon (12pm to 4pm)	Evening (4pm to 8pm)
Monday	☐	☐	☐
Tuesday	☐	☐	☐
Wednesday	☐	☐	☐
Thursday	☐	☐	☐

Updated Distance Preference:

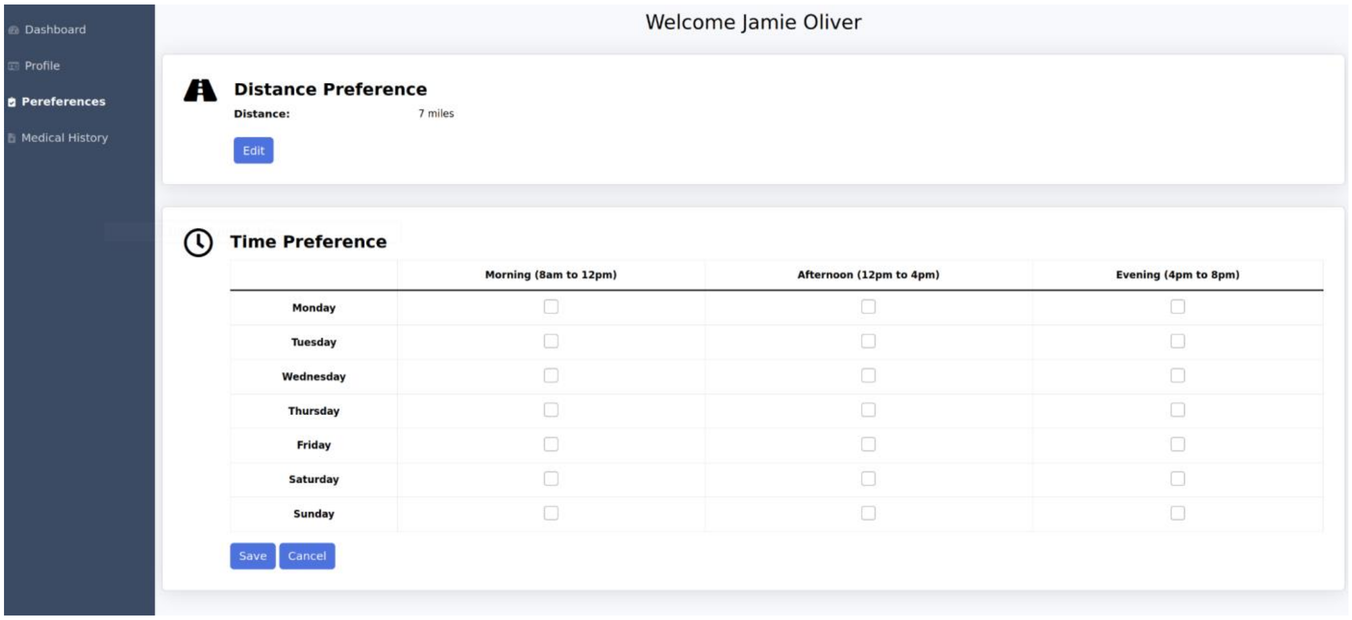


Welcome Jamie Oliver

Distance Preference

Distance:	7 miles

Editing Time Preference:



Welcome Jamie Oliver

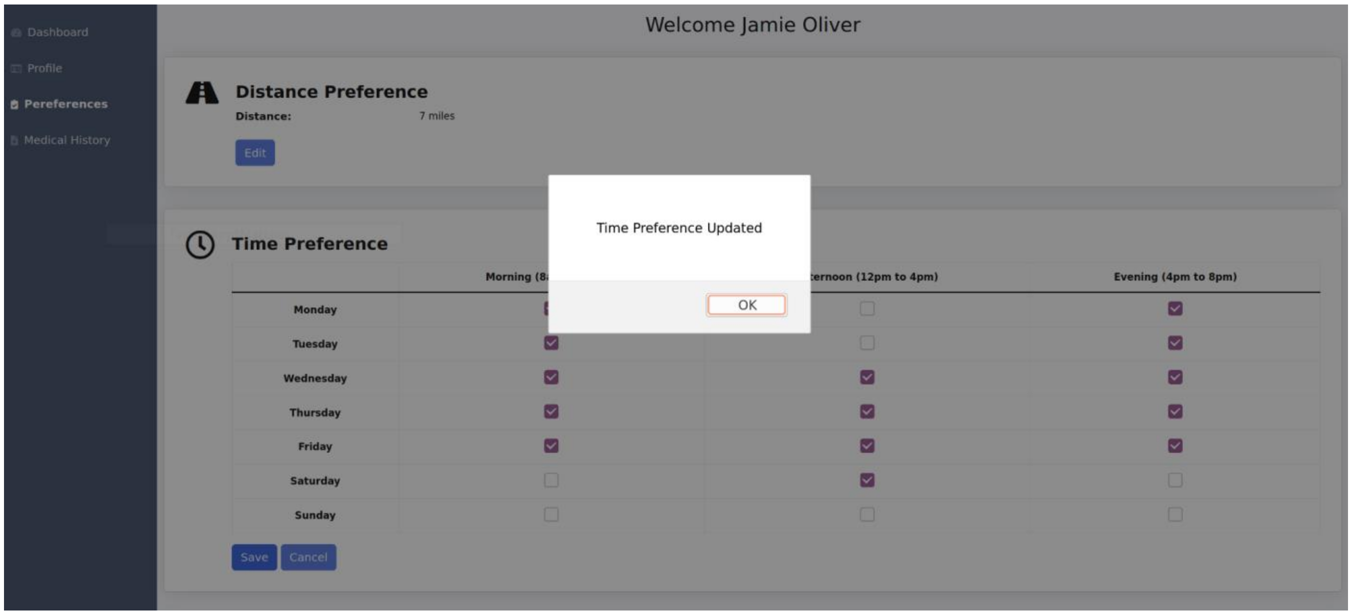
Distance Preference

Distance:	7 miles

Time Preference

	Morning (8am to 12pm)	Afternoon (12pm to 4pm)	Evening (4pm to 8pm)
Monday	☐	☐	☐
Tuesday	☐	☐	☐
Wednesday	☐	☐	☐
Thursday	☐	☐	☐
Friday	☐	☐	☐
Saturday	☐	☐	☐
Sunday	☐	☐	☐

Changing and Saving Time Preference:



Welcome Jamie Oliver

Distance Preference

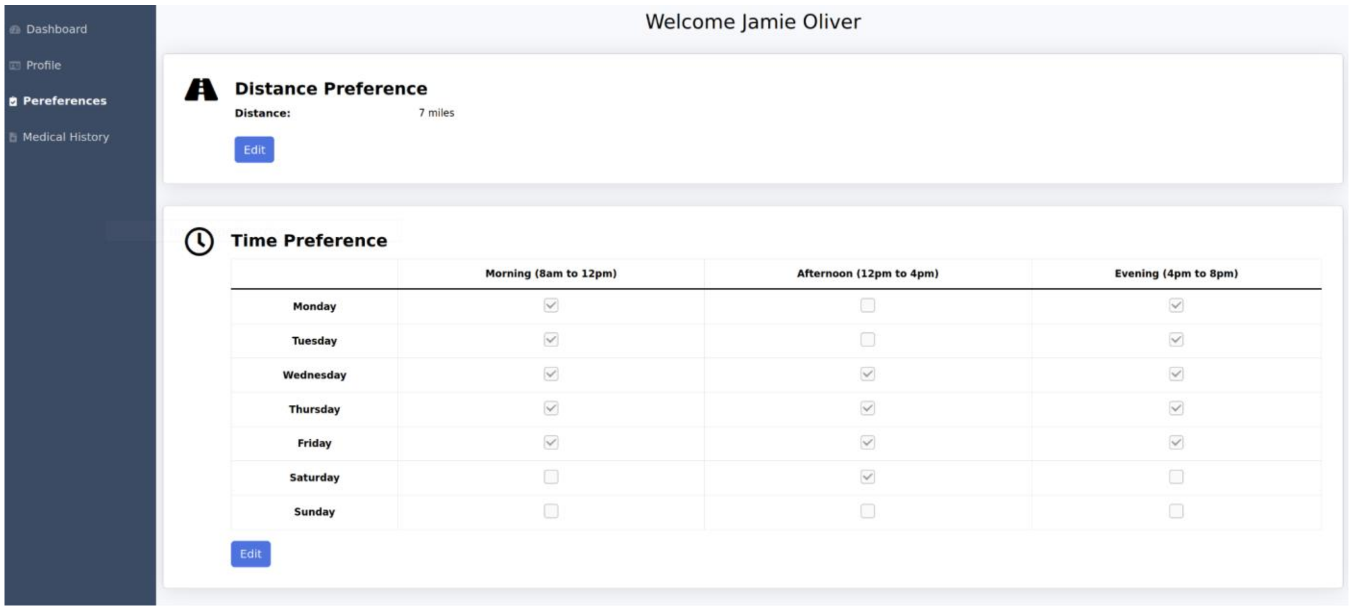
Distance:	7 miles

Time Preference Updated

Time Preference

	Morning (8	ternoon (12pm to 4pm)	Evening (4pm to 8pm)
Monday	✓	☐	✓
Tuesday	✓	☐	✓
Wednesday	✓	✓	✓
Thursday	✓	✓	✓
Friday	✓	✓	✓
Saturday	☐	✓	☐
Sunday	☐	☐	☐

Updated Time Preference:



Welcome Jamie Oliver

Distance Preference

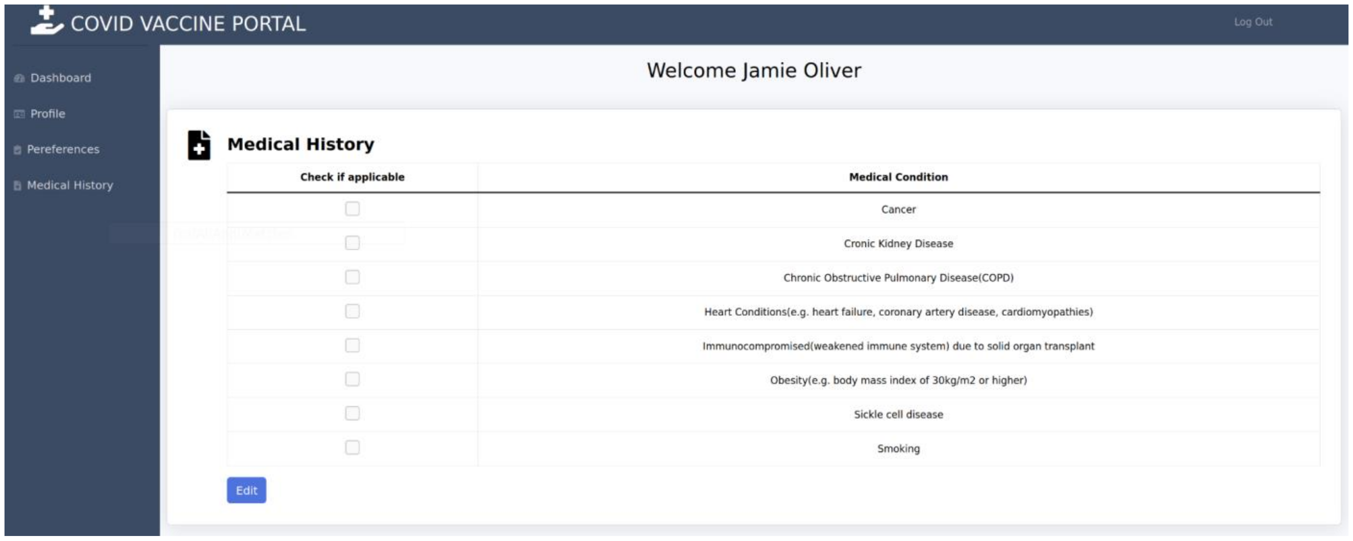
Distance: 7 miles

Time Preference

	Morning (8am to 12pm)	Afternoon (12pm to 4pm)	Evening (4pm to 8pm)
Monday	☑	☐	☑
Tuesday	☑	☐	☑
Wednesday	☑	☑	☑
Thursday	☑	☑	☑
Friday	☑	☑	☑
Saturday	☐	☑	☐
Sunday	☐	☐	☐

6. **Patient Medical History Page where Patient can indicate if they have any of the Medical condition associated with Priority Group 1**

   Patient Medical History Page in View Mode:

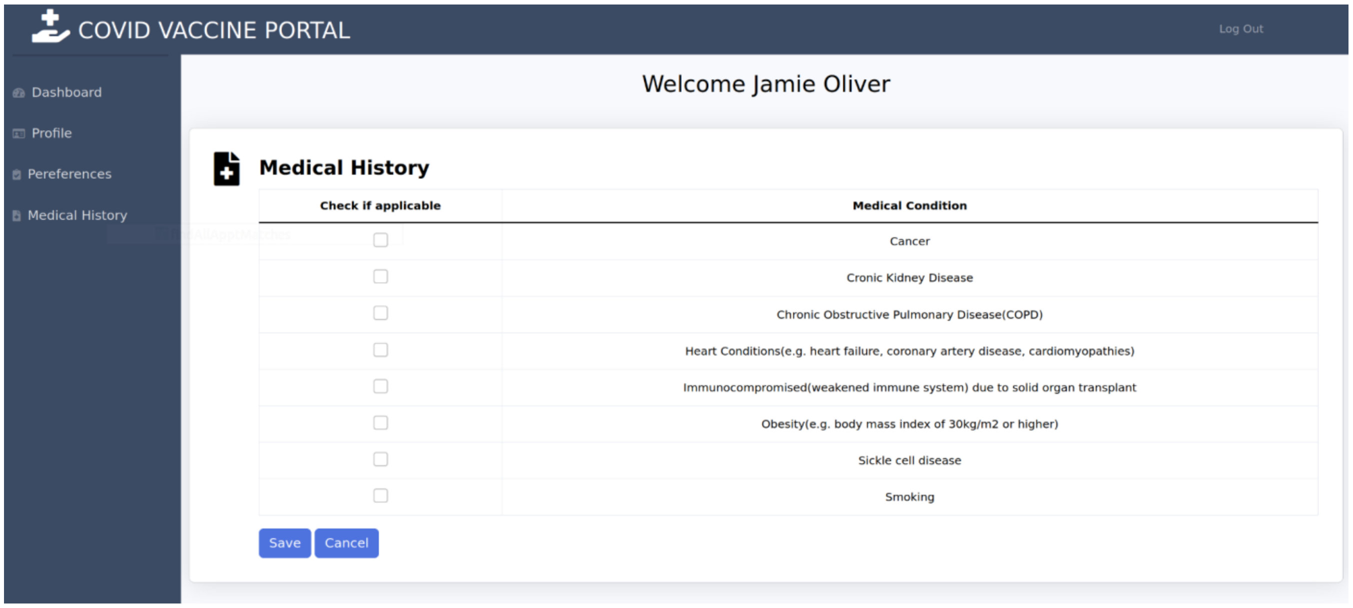


Welcome Jamie Oliver

Medical History

Check if applicable	Medical Condition
☐	Cancer
☐	Cronic Kidney Disease
☐	Chronic Obstructive Pulmonary Disease(COPD)
☐	Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies)
☐	Immunocompromised(weakened immune system) due to solid organ transplant
☐	Obesity(e.g. body mass index of 30kg/m2 or higher)
☐	Sickle cell disease
☐	Smoking

Patient Medical History Page in Edit Mode:

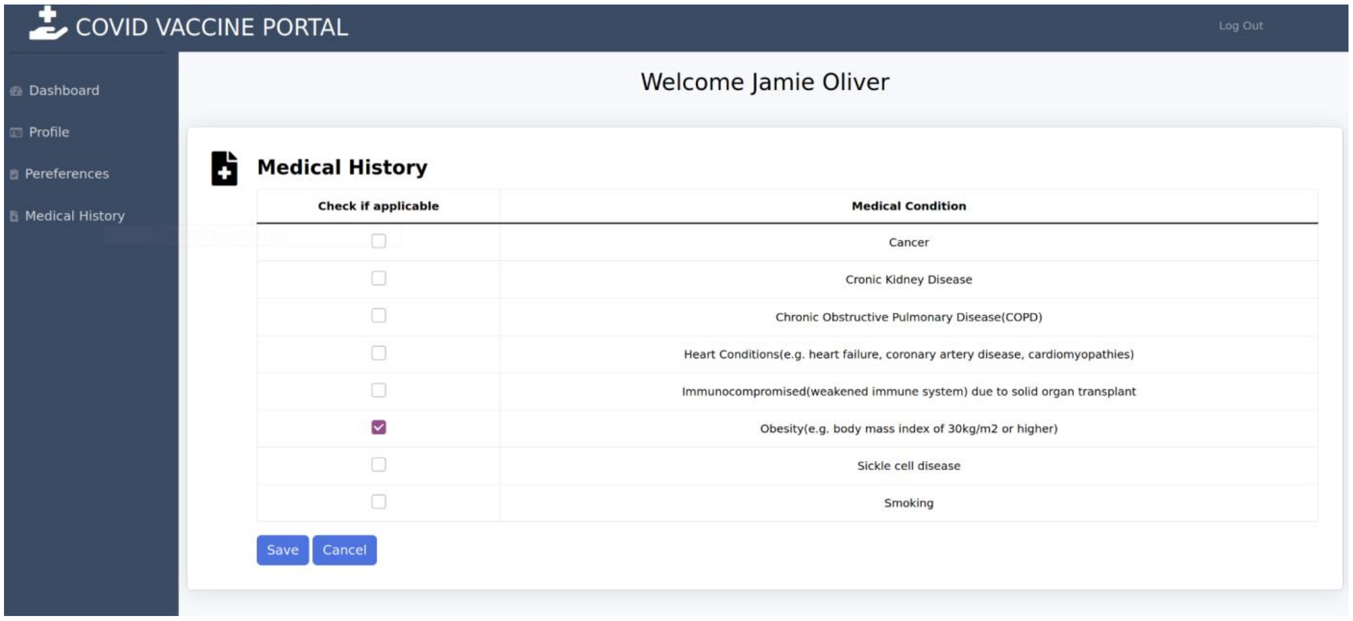


Welcome Jamie Oliver

Medical History

Check if applicable	Medical Condition
☐	Cancer
☐	Cronic Kidney Disease
☐	Chronic Obstructive Pulmonary Disease(COPD)
☐	Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies)
☐	Immunocompromised(weakened immune system) due to solid organ transplant
☐	Obesity(e.g. body mass index of 30kg/m2 or higher)
☐	Sickle cell disease
☐	Smoking

Patient Medical History Page While Editing Mode:

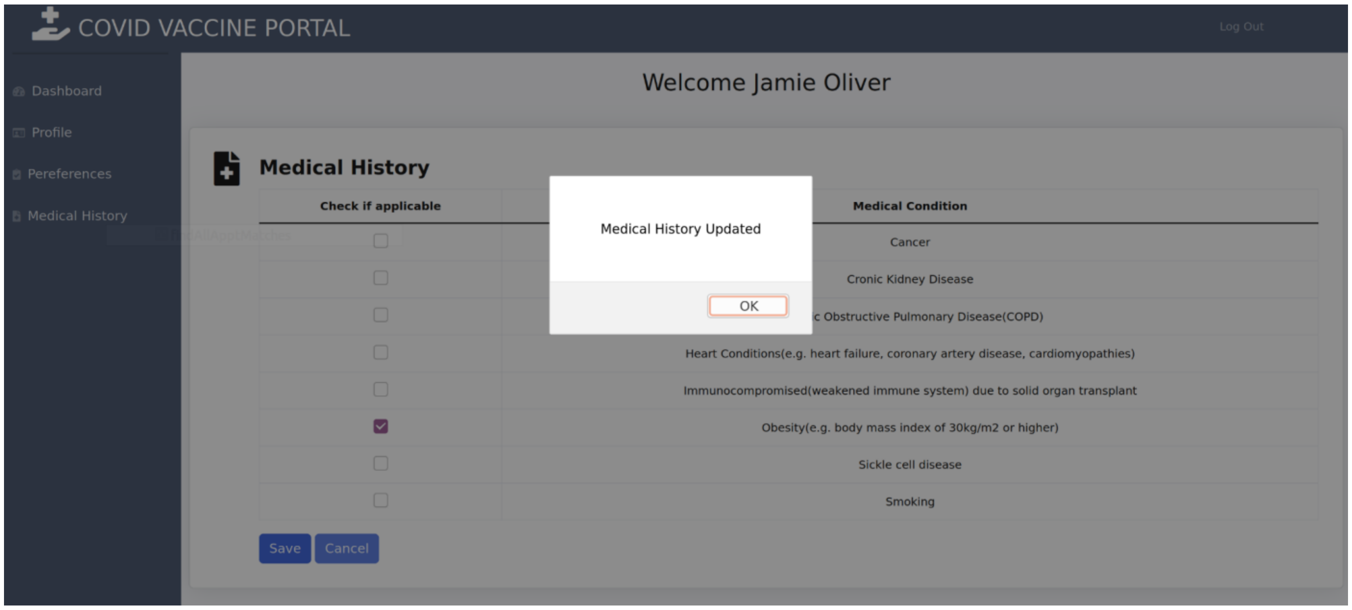


Welcome Jamie Oliver

Medical History

Check if applicable	Medical Condition
☐	Cancer
☐	Cronic Kidney Disease
☐	Chronic Obstructive Pulmonary Disease(COPD)
☐	Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies)
☐	Immunocompromised(weakened immune system) due to solid organ transplant
✓	Obesity(e.g. body mass index of 30kg/m2 or higher)
☐	Sickle cell disease
☐	Smoking

Patient Medical History Page Upon Saving:



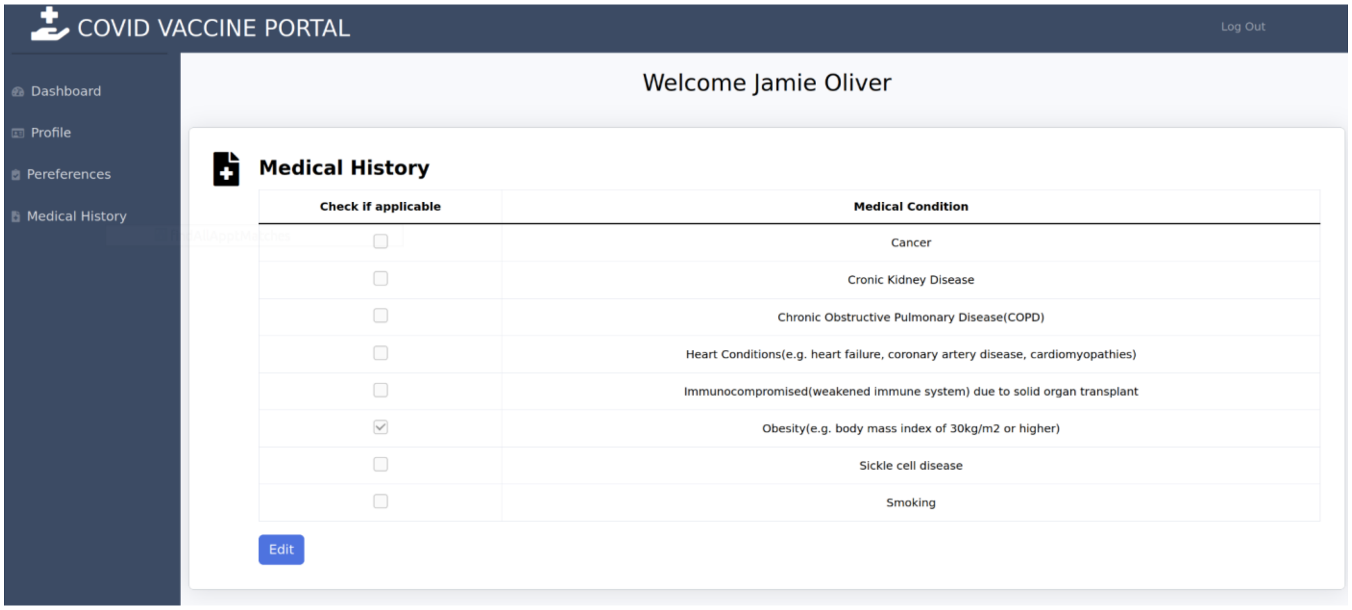
Welcome Jamie Oliver

Medical History

Check if applicable	Medical Condition
☐	Cancer
☐	Cronic Kidney Disease
☐	c Obstructive Pulmonary Disease(COPD)
☐	Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies)
☐	Immunocompromised(weakened immune system) due to solid organ transplant
✓	Obesity(e.g. body mass index of 30kg/m2 or higher)
☐	Sickle cell disease
☐	Smoking

Medical History Updated

Patient Medical History Page Upon Update:



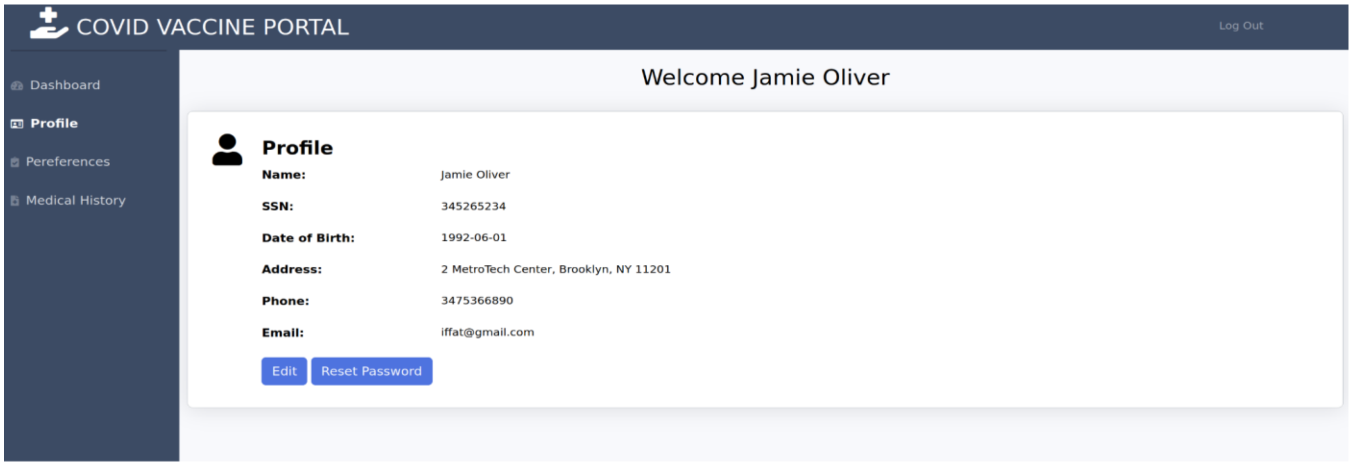
Welcome Jamie Oliver

Medical History

Check if applicable	Medical Condition
☐	Cancer
☐	Cronic Kidney Disease
☐	Chronic Obstructive Pulmonary Disease(COPD)
☐	Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies)
☐	Immunocompromised(weakened immune system) due to solid organ transplant
✓	Obesity(e.g. body mass index of 30kg/m2 or higher)
☐	Sickle cell disease
☐	Smoking

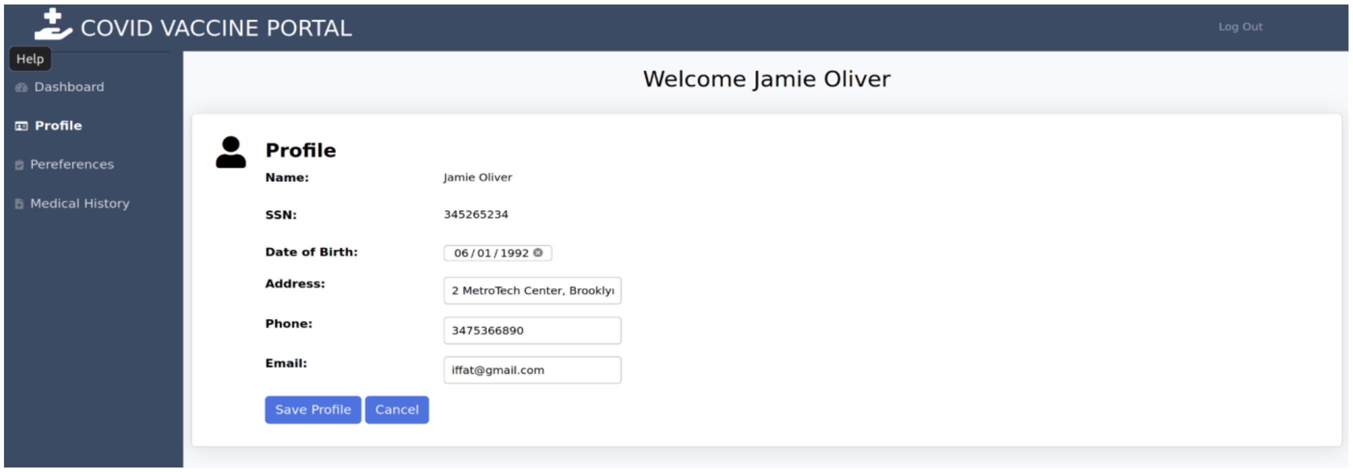
7. **Patient Profile Page is where Patients can update their Profile information.**
Note: SSN and Patient Name cannot be updated on this page; patients must contact customer@support.com

Patient Profile Page in View Mode



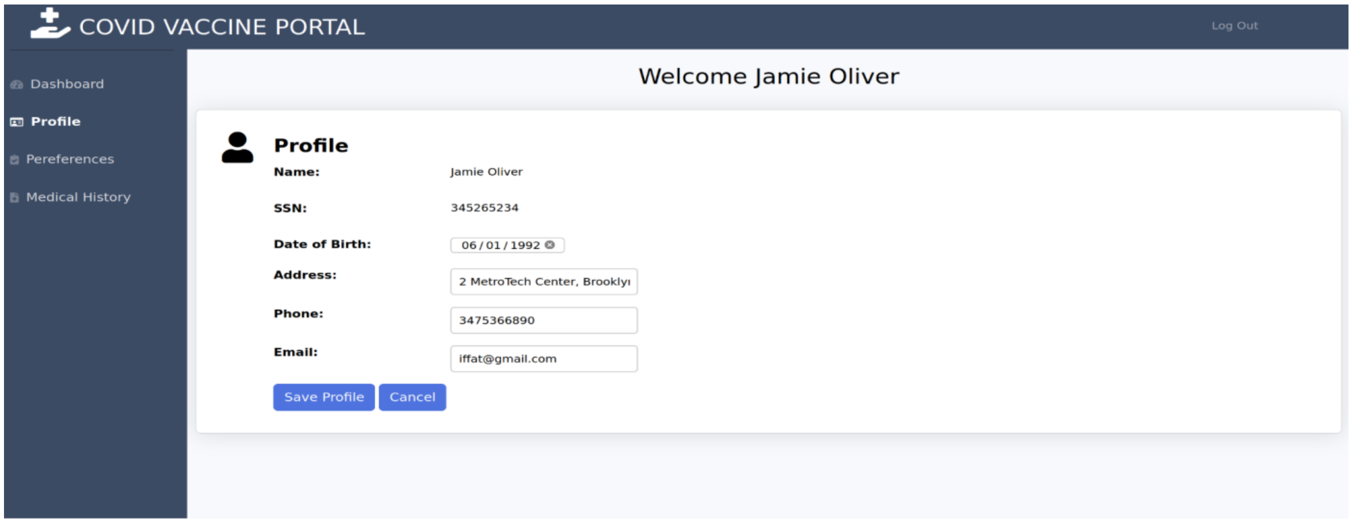
Name:	Jamie Oliver
SSN:	345265234
Date of Birth:	1992-06-01
Address:	2 MetroTech Center, Brooklyn, NY 11201
Phone:	3475366890
Email:	iffat@gmail.com

Patient Profile Page in Edit Mode



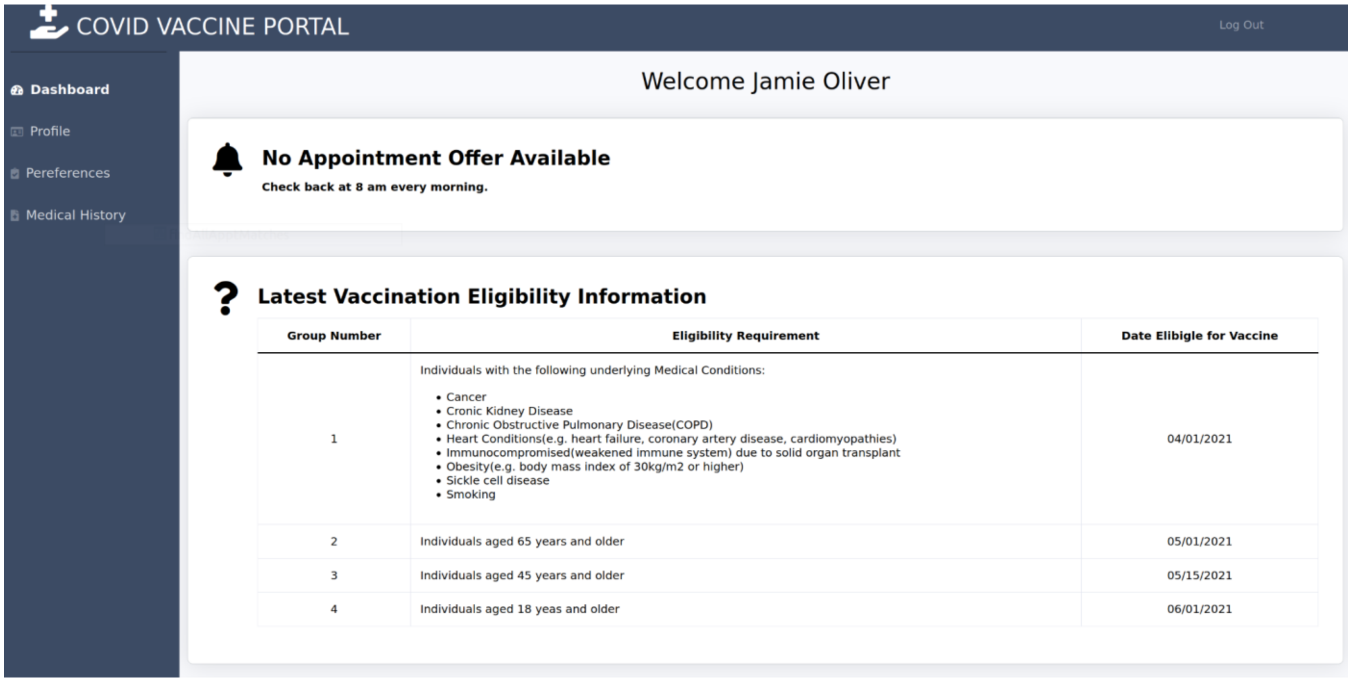
Name:	Jamie Oliver
SSN:	345265234
Date of Birth:	06 / 01 / 1992
Address:	2 MetroTech Center, Brooklyı
Phone:	3475366890
Email:	iffat@gmail.com

Patient Profile Page in Edit Mode for Reset Password



Name:	Jamie Oliver
SSN:	345265234
Date of Birth:	06 / 01 / 1992
Address:	2 MetroTech Center, Brooklyı
Phone:	3475366890
Email:	iffat@gmail.com

8. **Patient Home/Dashboard after Patient fills out the time preference/availability in the Preference Page and before an appointment is offered to the Patient.**
Note: The alert banner is no longer shown.



Welcome Jamie Oliver

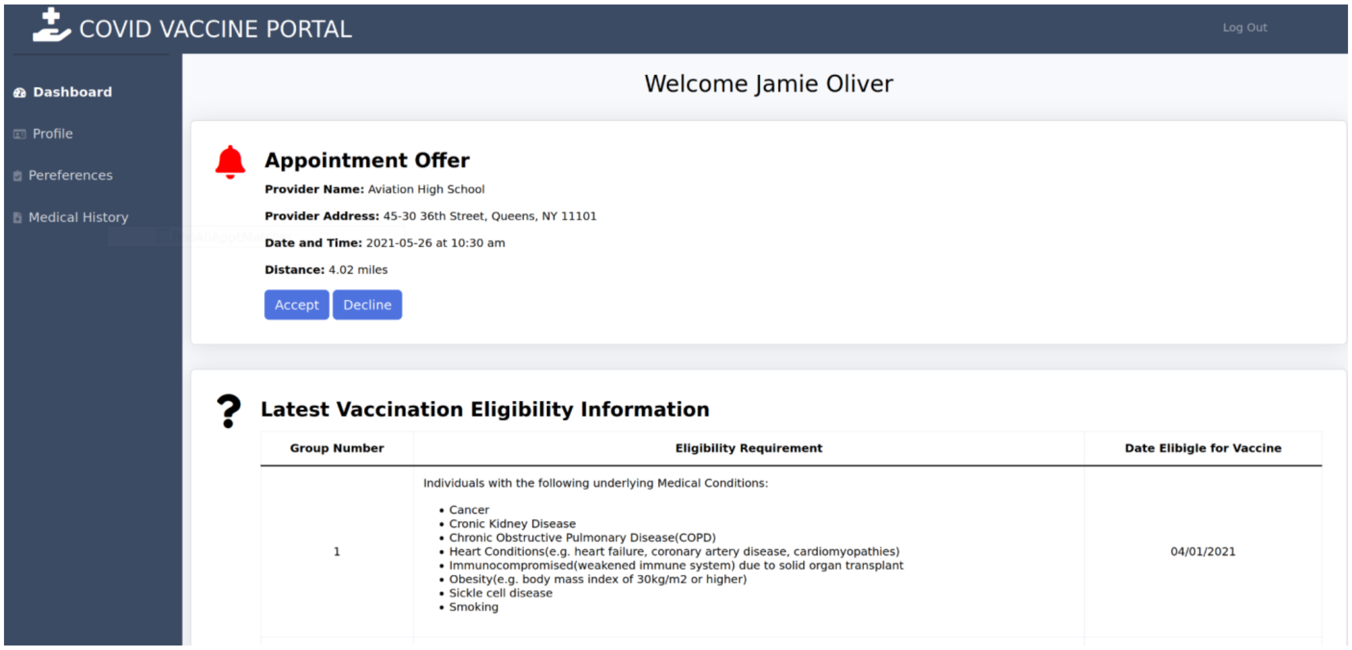
No Appointment Offer Available

Check back at 8 am every morning.

Latest Vaccination Eligibility Information

Group Number	Eligibility Requirement	Date Elibigle for Vaccine
1	Individuals with the following underlying Medical Conditions: • Cancer • Cronic Kidney Disease • Chronic Obstructive Pulmonary Disease(COPD) • Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies) • Immunocompromised(weakened immune system) due to solid organ transplant • Obesity(e.g. body mass index of 30kg/m2 or higher) • Sickle cell disease • Smoking	04/01/2021
2	Individuals aged 65 years and older	05/01/2021
3	Individuals aged 45 years and older	05/15/2021
4	Individuals aged 18 yeas and older	06/01/2021

9. **Patient Home/Dashboard after the Patient fills out the time preference/availability in the Preference Page and an appointment is offered to the patient.**



Welcome Jamie Oliver

Appointment Offer

Provider Name: Aviation High School

Provider Address: 45-30 36th Street, Queens, NY 11101

Date and Time: 2021-05-26 at 10:30 am

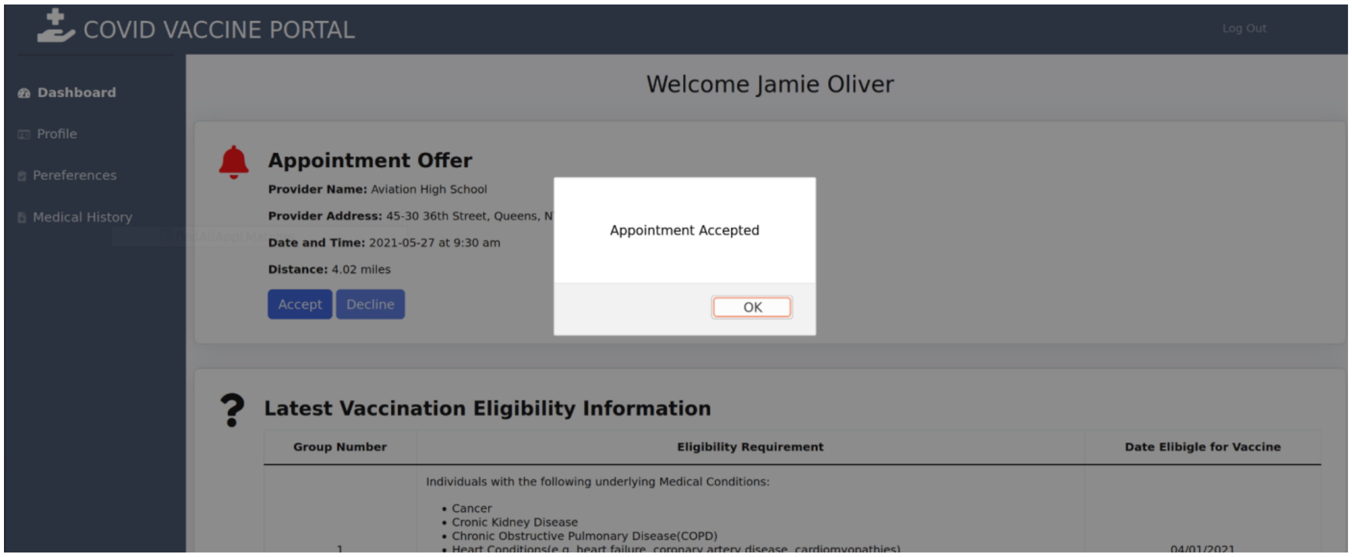
Distance: 4.02 miles

Latest Vaccination Eligibility Information

Group Number	Eligibility Requirement	Date Elibigle for Vaccine
1	Individuals with the following underlying Medical Conditions: • Cancer • Cronic Kidney Disease • Chronic Obstructive Pulmonary Disease(COPD) • Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies) • Immunocompromised(weakened immune system) due to solid organ transplant • Obesity(e.g. body mass index of 30kg/m2 or higher) • Sickle cell disease • Smoking	04/01/2021

10. Patient Home/Dashboard after an appointment is accepted by the Patient.

Home Page on Accepting Appointment Offer



Welcome Jamie Oliver

Appointment Offer

Provider Name: Aviation High School

Provider Address: 45-30 36th Street, Queens, N

Date and Time: 2021-05-27 at 9:30 am

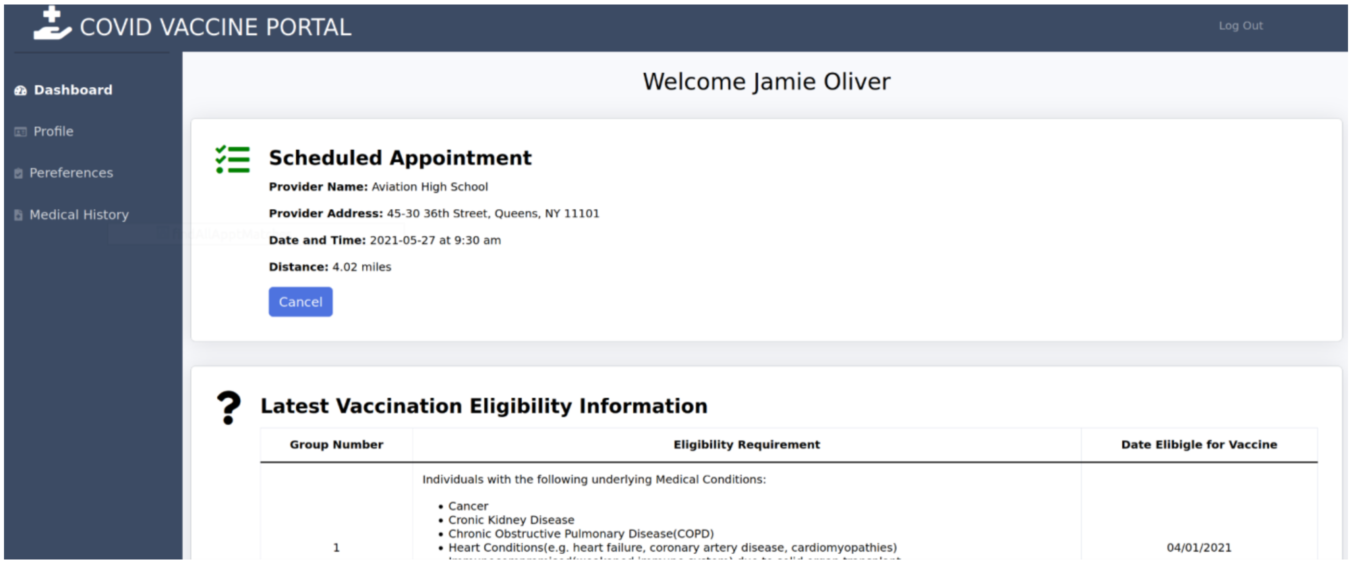
Distance: 4.02 miles

Appointment Accepted

Latest Vaccination Eligibility Information

Group Number	Eligibility Requirement	Date Elibigle for Vaccine
1	Individuals with the following underlying Medical Conditions: • Cancer • Cronic Kidney Disease • Chronic Obstructive Pulmonary Disease(COPD) • Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies)	04/01/2021

Home Page After Accepting Appointment Offer



Welcome Jamie Oliver

Scheduled Appointment

Provider Name: Aviation High School

Provider Address: 45-30 36th Street, Queens, NY 11101

Date and Time: 2021-05-27 at 9:30 am

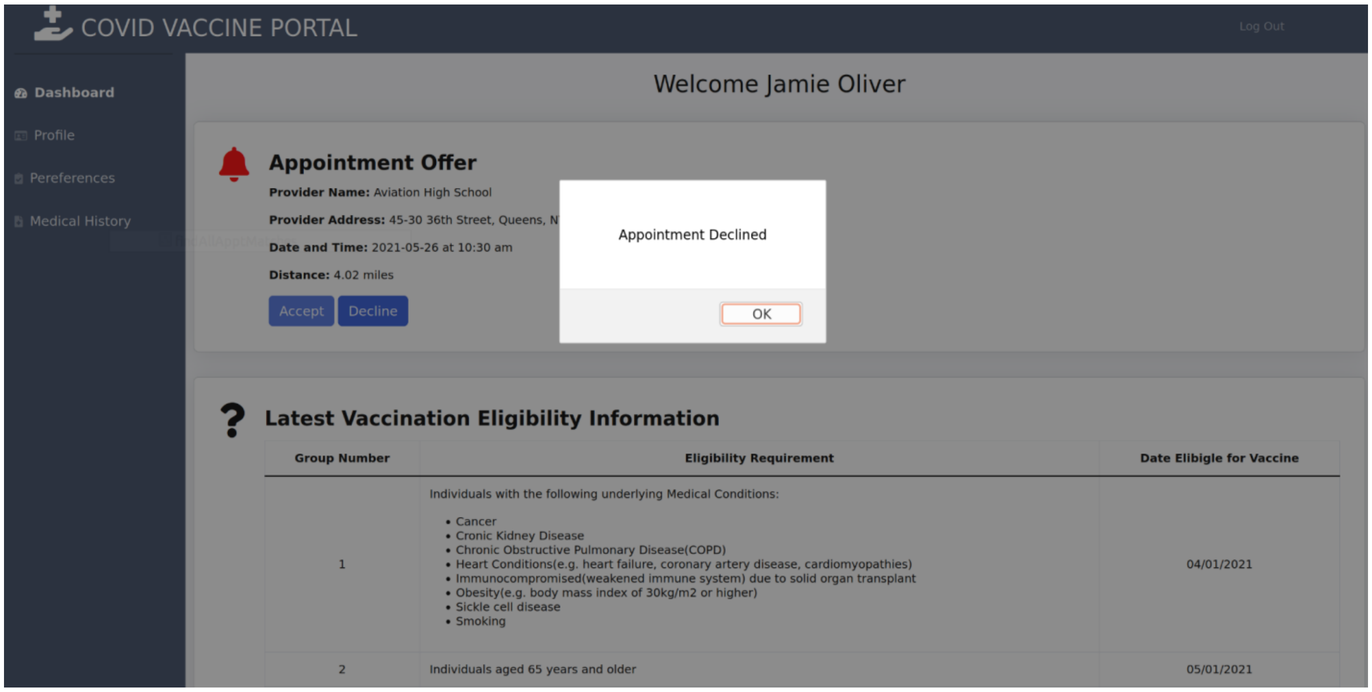
Distance: 4.02 miles

Latest Vaccination Eligibility Information

Group Number	Eligibility Requirement	Date Elibigle for Vaccine
1	Individuals with the following underlying Medical Conditions: • Cancer • Cronic Kidney Disease • Chronic Obstructive Pulmonary Disease(COPD) • Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies)	04/01/2021

11. Patient Home/Dashboard after an appointment is declined by the Patient.

Home Page on Declining Appointment Offer



Welcome Jamie Oliver

Appointment Offer

Provider Name: Aviation High School

Provider Address: 45-30 36th Street, Queens, N

Date and Time: 2021-05-26 at 10:30 am

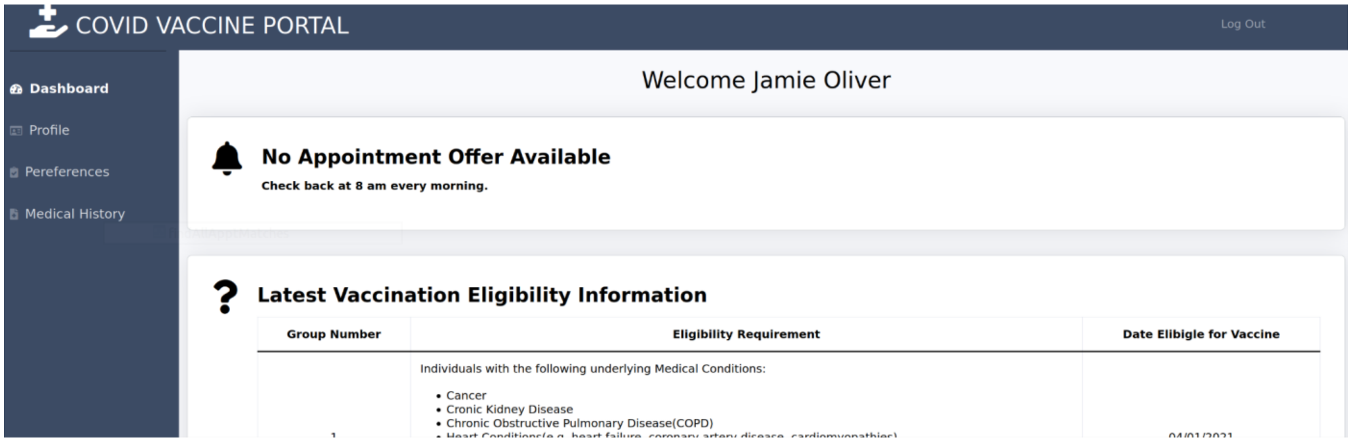
Distance: 4.02 miles

Appointment Declined

Latest Vaccination Eligibility Information

Group Number	Eligibility Requirement	Date Elibigle for Vaccine
1	Individuals with the following underlying Medical Conditions: • Cancer • Cronic Kidney Disease • Chronic Obstructive Pulmonary Disease(COPD) • Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies) • Immunocompromised(weakened immune system) due to solid organ transplant • Obesity(e.g. body mass index of 30kg/m2 or higher) • Sickle cell disease • Smoking	04/01/2021
2	Individuals aged 65 years and older	05/01/2021

Home Page after Patient Declines Appointment Offer



Welcome Jamie Oliver

No Appointment Offer Available

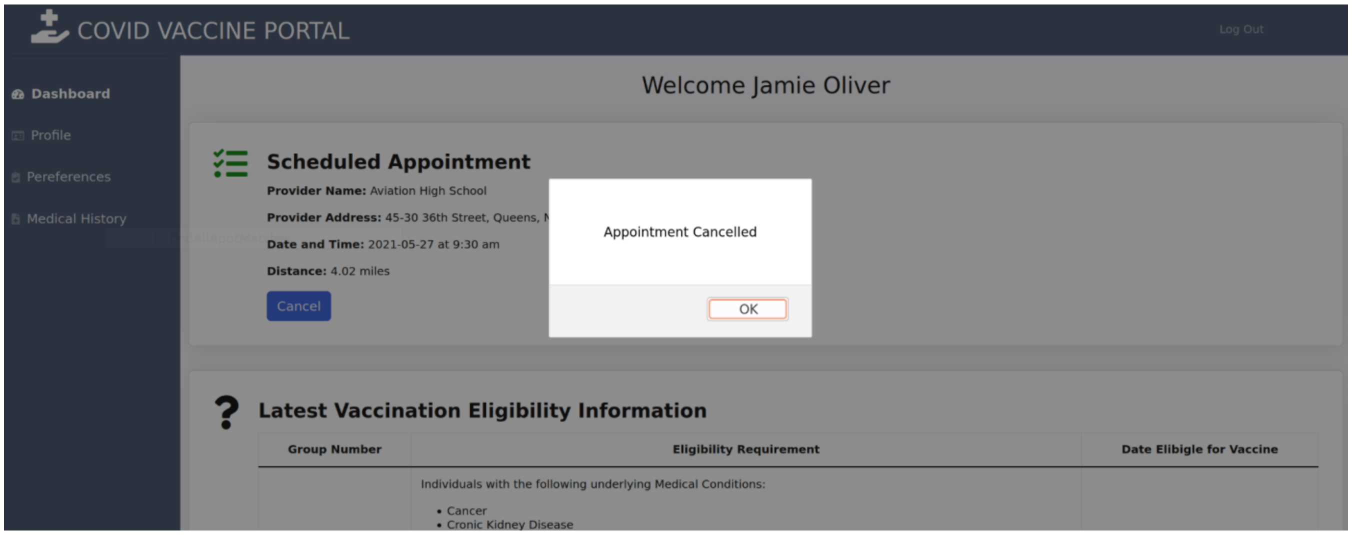
Check back at 8 am every morning.

Latest Vaccination Eligibility Information

Group Number	Eligibility Requirement	Date Elibigle for Vaccine
1	Individuals with the following underlying Medical Conditions: • Cancer • Cronic Kidney Disease • Chronic Obstructive Pulmonary Disease(COPD) • Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies)	04/01/2021

12. Patient Home/Dashboard an appointment is Cancelled by the Patient.

Home Page on Declining Appointment Offer



COVID VACCINE PORTAL

Log Out

Dashboard
Profile
Pereferences
Medical History

Welcome Jamie Oliver

Scheduled Appointment

Provider Name: Aviation High School

Provider Address: 45-30 36th Street, Queens, N

Date and Time: 2021-05-27 at 9:30 am

Distance: 4.02 miles

Cancel

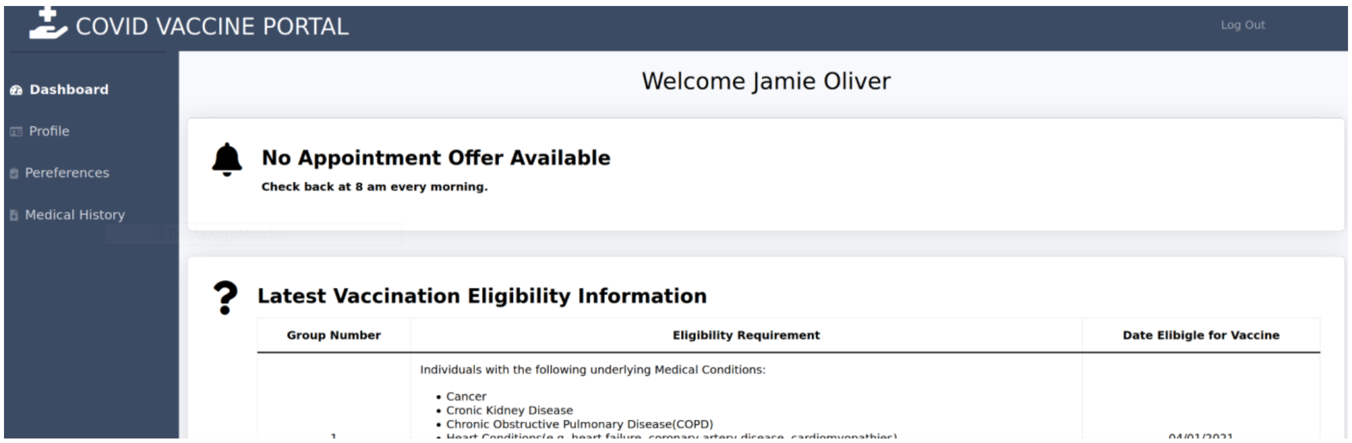
Appointment Cancelled

OK

Latest Vaccination Eligibility Information

Group Number	Eligibility Requirement	Date Elibigle for Vaccine
	Individuals with the following underlying Medical Conditions: • Cancer • Cronic Kidney Disease	

Home Page after Patient Cancels Appointment Offer



COVID VACCINE PORTAL

Log Out

Dashboard
Profile
Pereferences
Medical History

Welcome Jamie Oliver

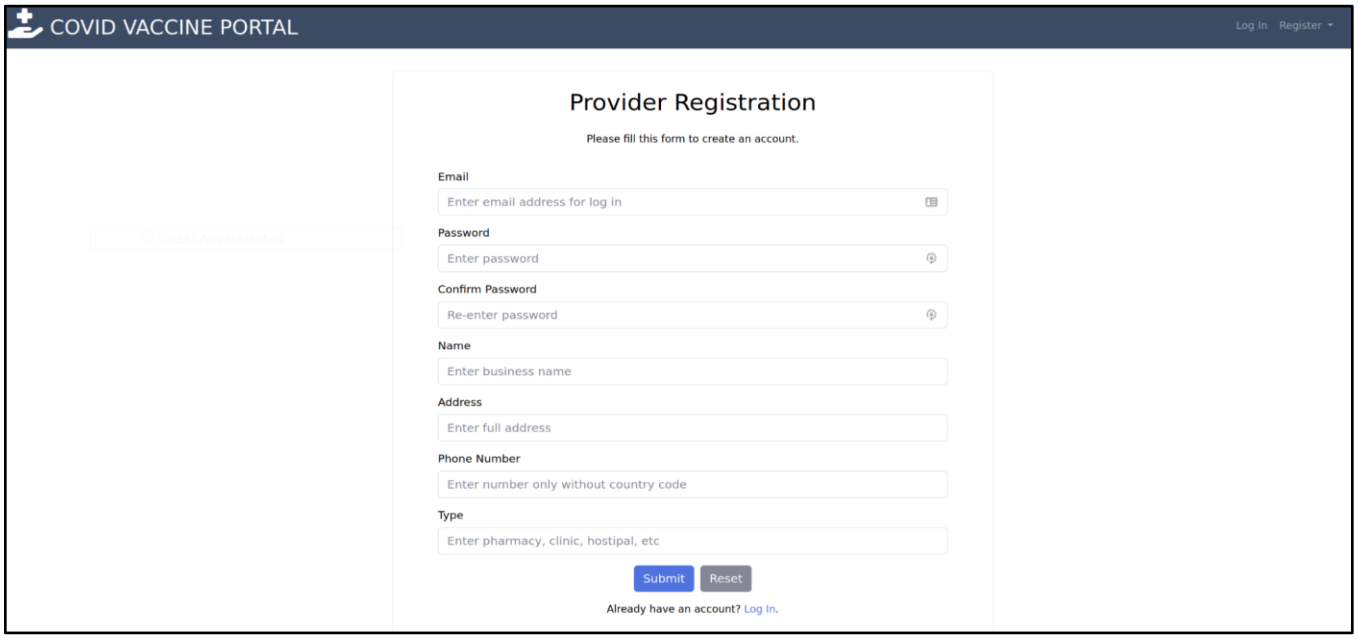
No Appointment Offer Available

Check back at 8 am every morning.

Latest Vaccination Eligibility Information

Group Number	Eligibility Requirement	Date Elibigle for Vaccine
1	Individuals with the following underlying Medical Conditions: • Cancer • Cronic Kidney Disease • Chronic Obstructive Pulmonary Disease(COPD) • Heart Conditions(e.g. heart failure, coronary artery disease, cardiomyopathies)	04/01/2021

13. **Provider Registration Page: Provider can register with the system using Patient Registration Page shown below**



COVID VACCINE PORTAL

Log In Register

Provider Registration

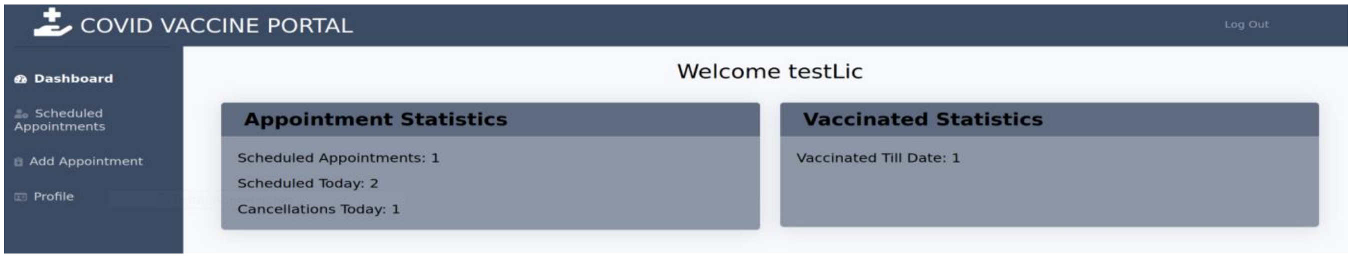
Please fill this form to create an account.

Email	Enter email address for log in
Password	Enter password
Confirm Password	Re-enter password
Name	Enter business name
Address	Enter full address
Phone Number	Enter number only without country code
Type	Enter pharmacy, clinic, hostipal, etc

Submit Reset

Already have an account? Log In.

14. **Provider Registration Home Page/Dashboard with some statistics displayed once they log in. Newly Registered Provider will have all statistics as 0.**



COVID VACCINE PORTAL

Log Out

Dashboard
Scheduled Appointments
Add Appointment
Profile

Welcome testLic

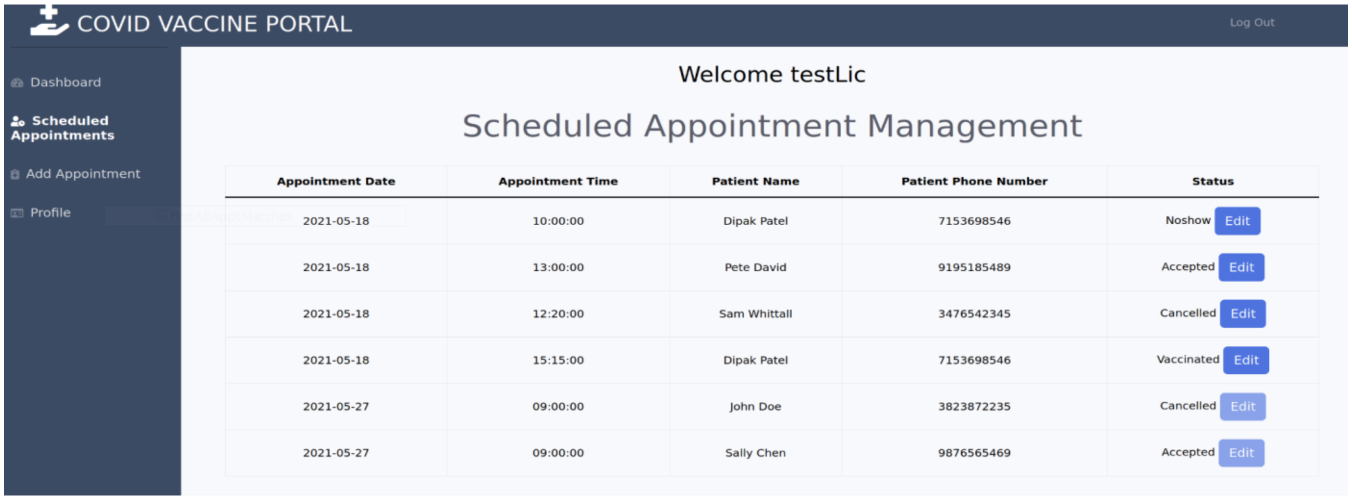
Appointment Statistics

Scheduled Appointments: 1
Scheduled Today: 2
Cancellations Today: 1

Vaccinated Statistics

Vaccinated Till Date: 1

15. **Appointment Page for Provider where they can see all scheduled, vaccinated, cancelled and no show appointments**
Note: The system only allows provider to Edit appointments on the day of the appointments, as seen below the EDIT Button is disabled



COVID VACCINE PORTAL

Log Out

Dashboard
Scheduled Appointments
Add Appointment
Profile

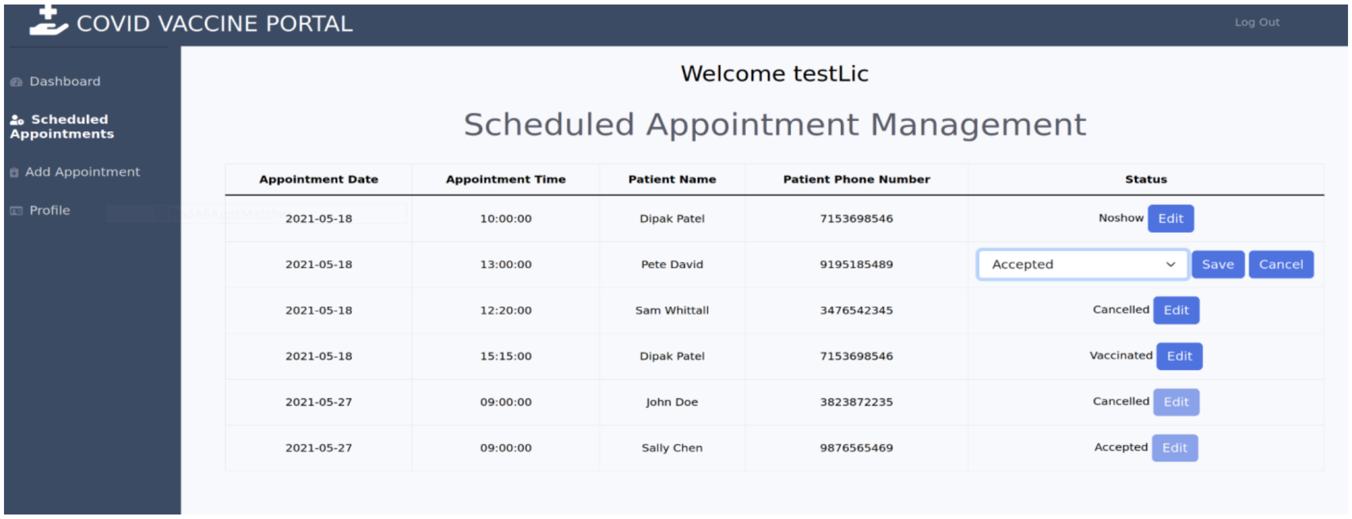
Welcome testLic

Scheduled Appointment Management

Appointment Date	Appointment Time	Patient Name	Patient Phone Number	Status
2021-05-18	10:00:00	Dipak Patel	7153698546	Noshow Edit
2021-05-18	13:00:00	Pete David	9195185489	Accepted Edit
2021-05-18	12:20:00	Sam Whittall	3476542345	Cancelled Edit
2021-05-18	15:15:00	Dipak Patel	7153698546	Vaccinated Edit
2021-05-27	09:00:00	John Doe	3823872235	Cancelled Edit
2021-05-27	09:00:00	Sally Chen	9876565469	Accepted Edit

16. Providers can update the status of appointments on the Scheduled Appointment Page

Provider when editing the status of an appointment for a patient


COVID VACCINE PORTAL

Log Out

Dashboard

Scheduled Appointments

Add Appointment

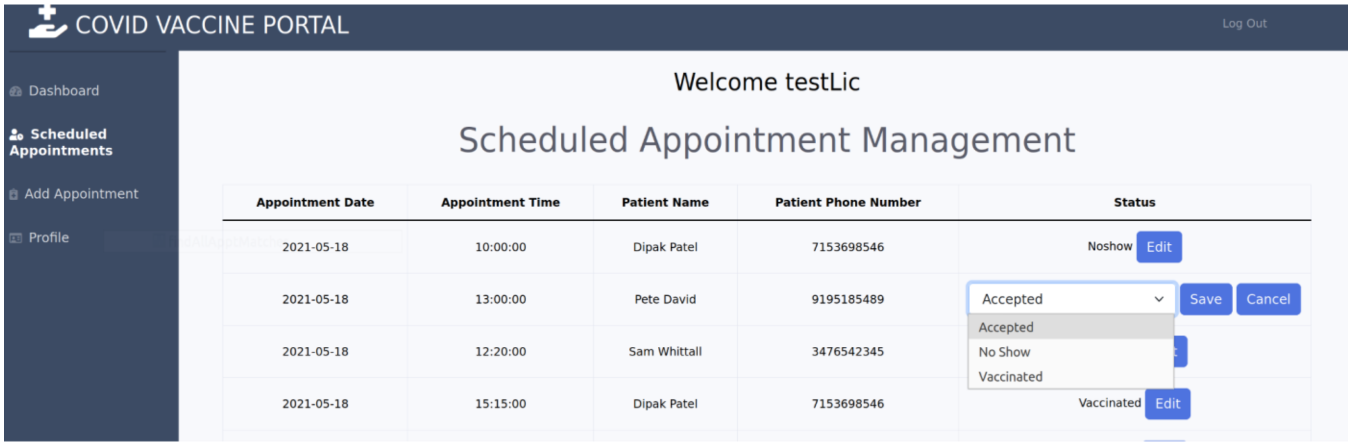
Profile

Welcome testLic

Scheduled Appointment Management

Appointment Date	Appointment Time	Patient Name	Patient Phone Number	Status
2021-05-18	10:00:00	Dipak Patel	7153698546	Noshow Edit
2021-05-18	13:00:00	Pete David	9195185489	Accepted Save Cancel
2021-05-18	12:20:00	Sam Whittall	3476542345	Cancelled Edit
2021-05-18	15:15:00	Dipak Patel	7153698546	Vaccinated Edit
2021-05-27	09:00:00	John Doe	3823872235	Cancelled Edit
2021-05-27	09:00:00	Sally Chen	9876565469	Accepted Edit

Provider dropdown options for status


COVID VACCINE PORTAL

Log Out

Dashboard

Scheduled Appointments

Add Appointment

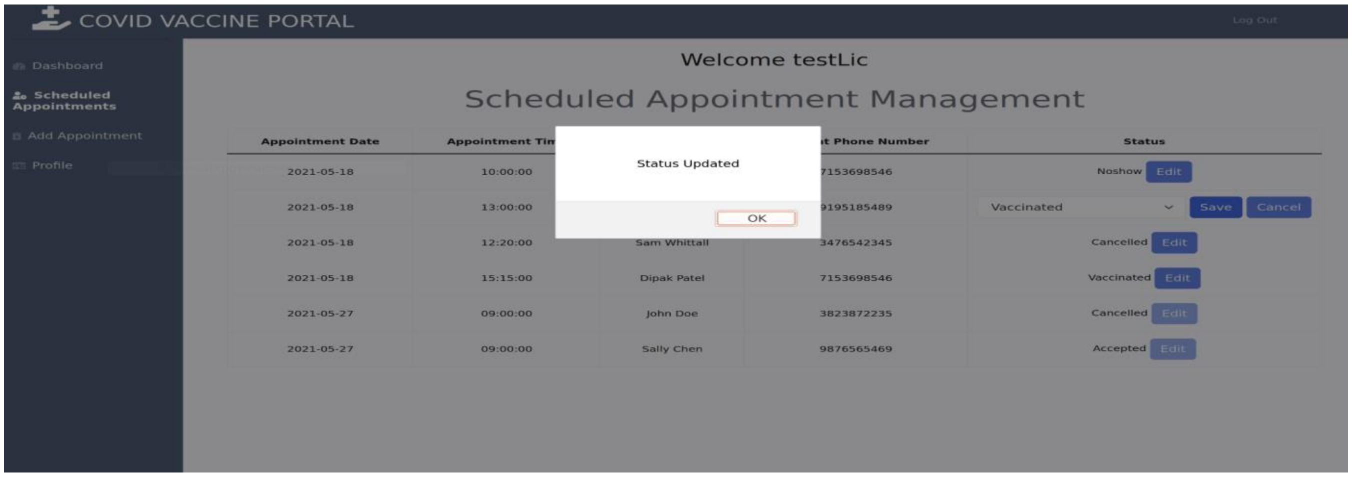
Profile

Welcome testLic

Scheduled Appointment Management

Appointment Date	Appointment Time	Patient Name	Patient Phone Number	Status
2021-05-18	10:00:00	Dipak Patel	7153698546	Noshow Edit
2021-05-18	13:00:00	Pete David	9195185489	Accepted Save Cancel
2021-05-18	12:20:00	Sam Whittall	3476542345	
2021-05-18	15:15:00	Dipak Patel	7153698546	Vaccinated Edit

Scheduled appointment Page upon saving the updated status


COVID VACCINE PORTAL

Log Out

Dashboard

Scheduled Appointments

Add Appointment

Profile

Welcome testLic

Scheduled Appointment Management

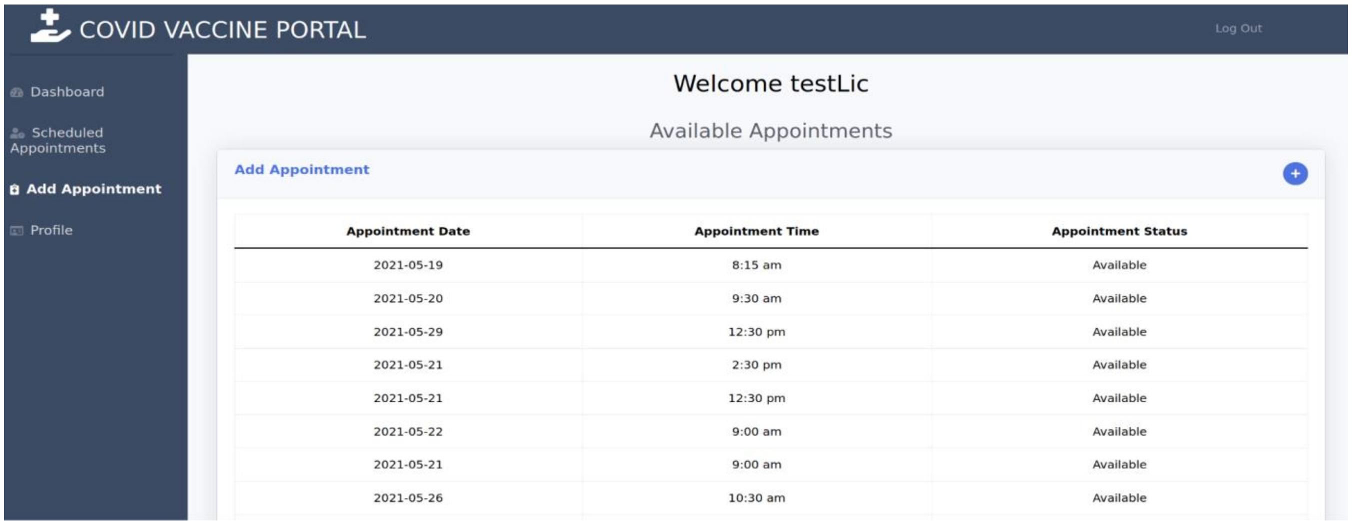
Appointment Date	Appointment Tim		t Phone Number	Status
2021-05-18	10:00:00		7153698546	Noshow Edit
2021-05-18	13:00:00		9195185489	Vaccinated Save Cancel
2021-05-18	12:20:00	Sam Whittall	3476542345	Cancelled Edit
2021-05-18	15:15:00	Dipak Patel	7153698546	Vaccinated Edit
2021-05-27	09:00:00	John Doe	3823872235	Cancelled Edit
2021-05-27	09:00:00	Sally Chen	9876565469	Accepted Edit

Status Updated

OK

17. Providers can add New appointment on the Add Appointment Page

All future appointments that are still available is displayed on the Add Appointment Page



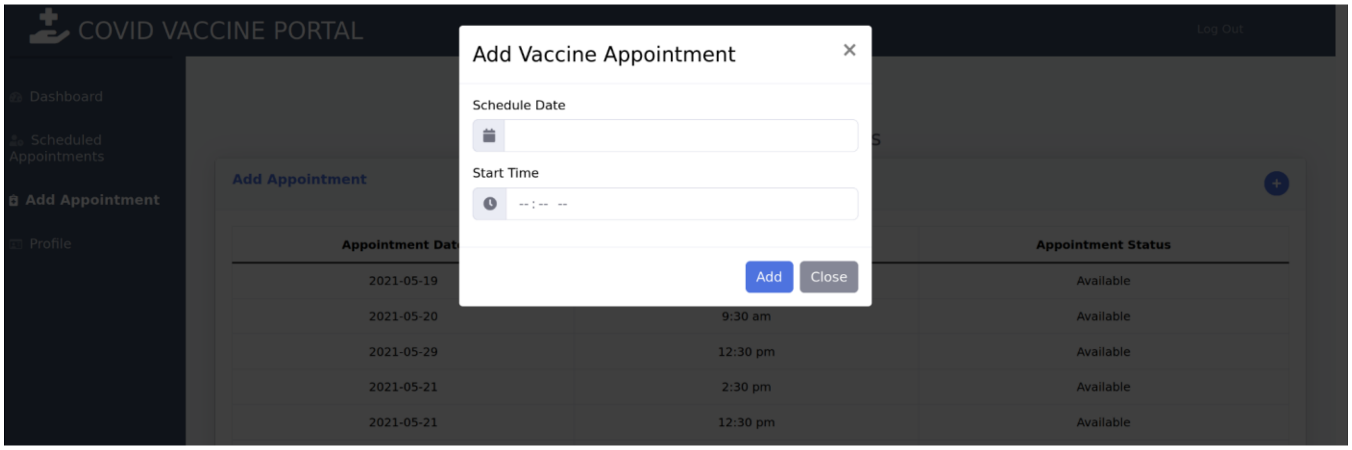
Welcome testLic

Available Appointments

Add Appointment

Appointment Date	Appointment Time	Appointment Status
2021-05-19	8:15 am	Available
2021-05-20	9:30 am	Available
2021-05-29	12:30 pm	Available
2021-05-21	2:30 pm	Available
2021-05-21	12:30 pm	Available
2021-05-22	9:00 am	Available
2021-05-21	9:00 am	Available
2021-05-26	10:30 am	Available

Provider can add appointments by clicking on the + icon



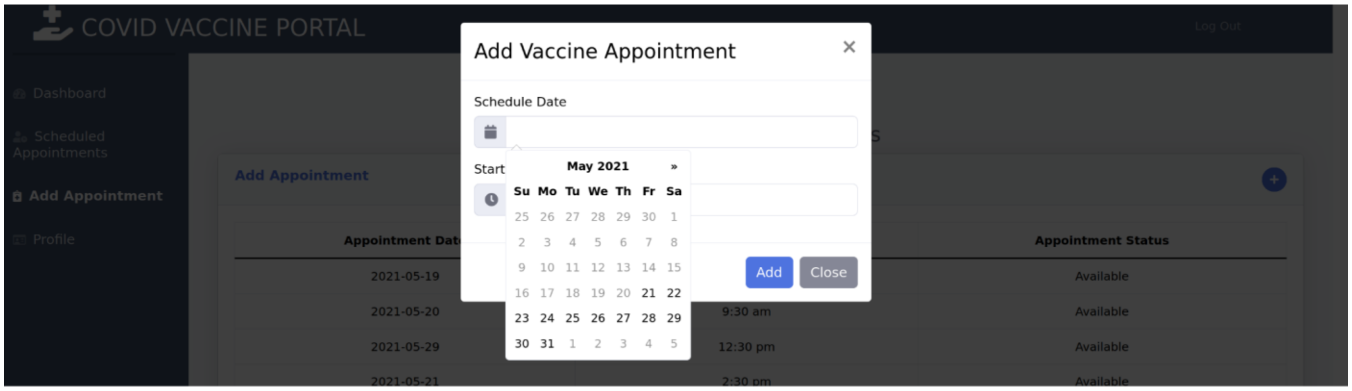
Add Vaccine Appointment

Schedule Date

Start Time

--:-- --

Date of Appointment can only be set to 3 days in the future to allow time for offering the appointment to the Patient and time is restricted between 8 AM to 8 PM. Date of Screenshot: 05/18/2021



Add Vaccine Appointment

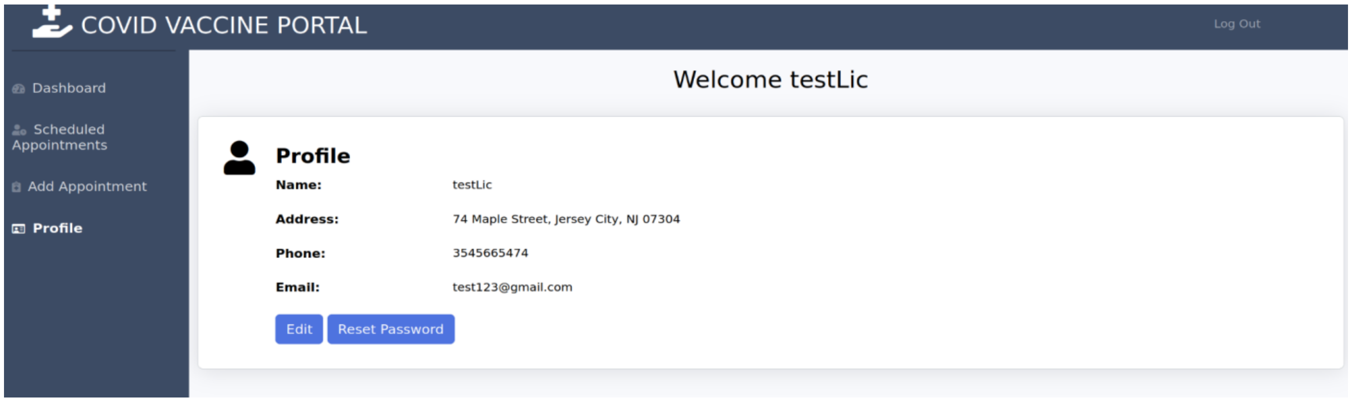
Schedule Date

May 2021 »

Su	Mo	Tu	We	Th	Fr	Sa
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

18. Provider Profile Page is where Provider can update their Profile information.
Note: Provider Name cannot be updated on this page; provider must contact customer@support.com

Provider Profile Page in View Mode

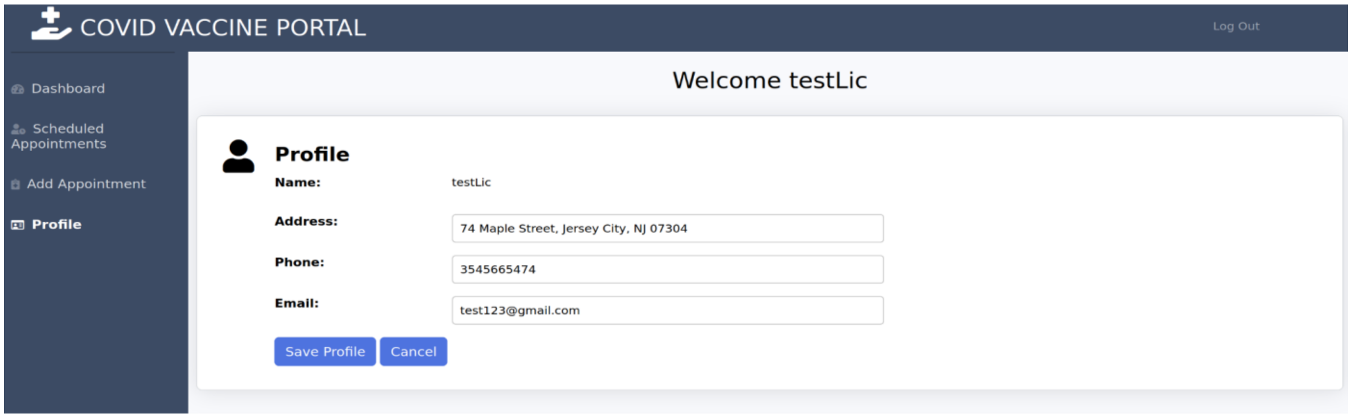


Welcome testLic

Profile

Name:	testLic
Address:	74 Maple Street, Jersey City, NJ 07304
Phone:	3545665474
Email:	test123@gmail.com

Provider Profile Page in Edit Mode

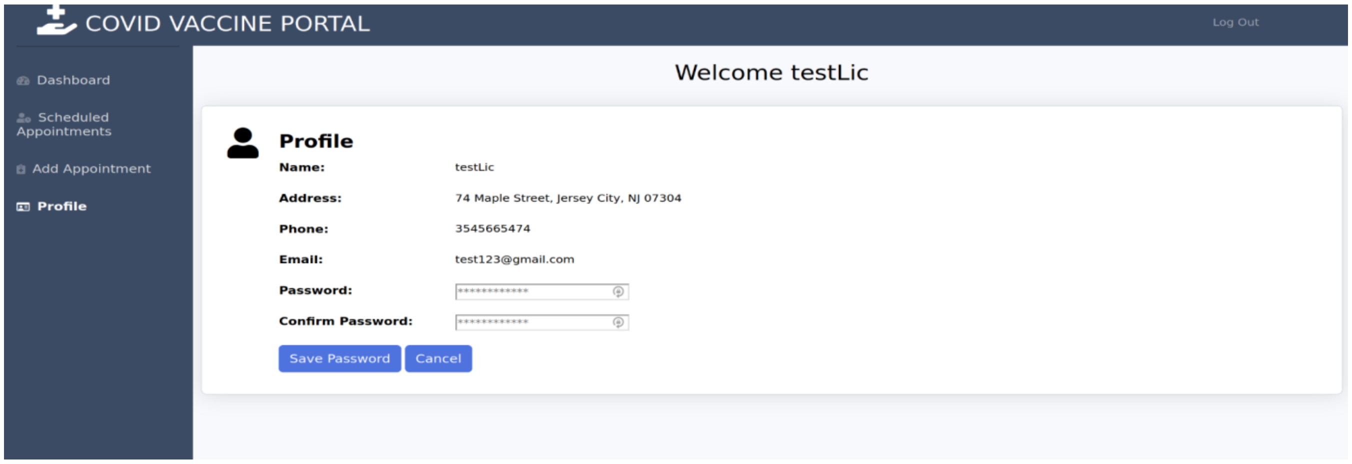


Welcome testLic

Profile

Name:	testLic
Address:	74 Maple Street, Jersey City, NJ 07304
Phone:	3545665474
Email:	test123@gmail.com

Prover Profile Page Password Reset in Edit Mode



Welcome testLic

Profile

Name:	testLic
Address:	74 Maple Street, Jersey City, NJ 07304
Phone:	3545665474
Email:	test123@gmail.com
Password:	************
Confirm Password:	************

5. TEST

This section covers some test scenarios that were conducted to validate the design of the database discussed in section 2.

5.1. TEST DATA

In order to test the database design, some test data were inserted into all the tables. Listed below are the dataset inserted into the tables:

1. **Patient:**

patientId	patientName	ssn	dob	patientAddress	patientLatitude	patientLongitude	patientPhone	distancePreference	priorityGroup	patientEmail	patientPassword
1	Iffat Rahman	32567897	1992-01-20	74 Maple Street, Jersey City, NJ 07304	40.7113052	-74.0579327	2342257476	10	4	iffat.rahman92@gmail.com	5cf97fd358ebf8faed92691b73513ed0
2	Dipak Patel	123456789	1991-02-01	21 Nelson Avenue, Jersey City, NJ 07307	40.755322	-74.051056	7153698546	15	4	dipak@dipak.com	2eb445320ff4adfad4693cfa56c08790
3	Faran Jessani	12365478	1988-03-24	74 Maple Street, Jersey City, NJ 07304	40.7113052	-74.0579327	7165344585	4	4	faran@fj.com	7c221e30c569c7e168461ae8a9057ee4
4	Sally Chen	454897216	1994-07-01	2590 41st Street, Astoria, NY 11103	40.765568	-73.912866	9876565469	5	1	sally@chen.com	c7be7e4cd9579f55e605b3e08313880f
5	Fatema Akhter	546931529	1964-01-25	3926 68th Street, Woodside, NY 11377	40.74620653	-73.89727672	7187563568	10	1	fa@gmail.com	a85c7e3915f1463322cc0548095200fa
6	Morshed Alam	314589725	1972-11-25	4817 48th Street, Woodside, NY 11377	40.73831986	-73.91770151	3458967878	15	3	malam@alam.com	baaefcf79e68c4154953c3a3193a2c16
7	Jane Yon	516936528	2006-09-27	4611 Broadway, Astoria, NY 11103	40.756759	-73.913626	9175236489	7	NULL	jyon@gmail.com	867c5feddc36a0725b6dafd53a2af03f
8	Pete David	35691756	1956-05-12	163-15 35th Ave, Flushing, NY 11358	40.76549	-73.801791	9195185489	8	2	wb@basher.com	83ec84dcde5fc7759350616f33f71f9f
9	Jiyan Hu	256963452	1987-06-11	71 Coles Street, Jersey City, NJ 07302	40.724447	-74.047538	7459632569	20	1	jhu@hu.com	b7cabe95e4e4dda4317631c4199af4b1

2. **PatientMedHistory:**

patientId	medCondition
4	Chronic Obstructive Pulmonary Disease(COPD)

5	Heart Conditions
5	High Blood Pressure
9	Sickle Cell Disease

Note: document column is of BLOB type and stores any supporting pdf file for the patient's medical condition if they upload it.

3. TimeSlot:

slotId	day	startTime	endTime
1	Monday	8:00:00	11:59:59
2	Monday	12:00:00	15:59:59
3	Monday	16:00:00	17:59:59
4	Tuesday	8:00:00	11:59:59
5	Tuesday	12:00:00	15:59:59
6	Tuesday	16:00:00	17:59:59
7	Wednesday	8:00:00	11:59:59
8	Wednesday	12:00:00	15:59:59
9	Wednesday	16:00:00	17:59:59
10	Thursday	8:00:00	11:59:59
11	Thursday	12:00:00	15:59:59
12	Thursday	16:00:00	17:59:59
13	Friday	8:00:00	11:59:59
14	Friday	12:00:00	15:59:59
15	Friday	16:00:00	17:59:59
16	Saturday	8:00:00	11:59:59
17	Saturday	12:00:00	15:59:59
18	Saturday	16:00:00	17:59:59
19	Sunday	8:00:00	11:59:59
20	Sunday	12:00:00	15:59:59
21	Sunday	16:00:00	17:59:59

4. PatientTimePreference:

patientId	slotId
1	1
1	5
2	16
2	17
3	4
3	7
3	10
3	11
4	7
4	16
4	17
5	1
5	3
5	5
6	15
6	18
6	21
7	7
7	10
7	13
8	16
8	17
8	18
8	19
8	20
8	21
9	1
9	10
9	16
9	17

5. Provider:

providerId	providerName	providerAddress	providerLatitude	providerLongitude	providerPhone	providerType	providerEmail	providerPassword
1	Rite Aid	2859-61 John F. Kennedy Blvd, Jersey City, NJ 07306	40.731972	-74.065037	2014332826	pharmacy	jcriteaid@raid.com	542221ab663c77f4ec6b388c9f2f13f0
2	Hudson County COVID-19 Vaccination	110 Hackensack Ave, Kearny, NJ 07032	40.7234755	-74.1105453	2014332826	government	hudsoncounty@covid.com	542221ab663c77f4ec6b388c9f2f13f0
3	CVS @ 49th St	49-2 Queens Blvd, Woodside, NY 11377	40.742537	-73.9158	7182050550	pharmacy	woosidecvsd@cvs.com	da2f12f340fb8cf06f74b308d4af93ed
4	Aviation High School	45-30 36th Street, Queens, NY 11101	40.743703	-73.929288	2012615899	government	vaccineahs@covid.com	e029c735597aa766d94efb13b51abb1b
5	Duane Reade Pharmacy	2858 Steinway St,Queens, NY 11103	40.764294	-73.915166	7182781402	pharmacy	drpastoria@duane.com	841a61688324d42bbdc577874cd560e1

6. ProviderAppointment:

appointmentId	providerId	date	startTime	isAvailable
1	1	2021-04-01	8:15:00	Y
2	1	2021-04-07	8:15:00	N
3	1	2021-04-01	9:30:00	Y
4	1	2021-04-01	12:30:00	Y
5	1	2021-04-01	14:30:00	Y
6	2	2021-04-02	12:30:00	Y
7	4	2021-04-02	9:00:00	Y
8	4	2021-04-02	9:00:00	Y
9	4	2021-04-02	10:30:00	Y
10	4	2021-04-02	14:00:00	Y
11	4	2021-04-05	17:30:00	N
12	3	2021-04-05	9:30:00	Y
13	3	2021-04-08	12:30:00	Y
14	3	2021-04-05	14:30:00	Y
15	3	2021-04-06	9:30:00	Y

16	3	2021-04-17	14:30:00	N
17	3	2021-04-25	9:30:00	N
18	3	2021-04-25	9:30:00	Y
19	3	2021-04-25	14:30:00	Y
20	3	2021-04-25	17:30:00	Y
21	3	2021-04-25	17:45:00	N
22	3	2021-04-01	9:30:00	Y
23	3	2021-05-01	9:30:00	N
24	3	2021-05-01	12:30:00	Y
25	3	2021-05-01	14:30:00	Y
26	3	2021-05-01	14:30:00	Y
27	1	2021-05-02	9:30:00	Y
28	1	2021-05-02	9:30:00	Y
29	1	2021-05-02	12:30:00	Y
30	1	2021-05-02	14:30:00	Y
31	1	2021-05-02	14:30:00	Y
32	4	2021-05-02	9:30:00	Y
33	4	2021-05-02	9:30:00	Y
34	4	2021-05-02	12:30:00	Y
35	4	2021-05-02	14:30:00	Y
36	4	2021-05-02	14:30:00	Y
37	2	2021-05-05	10:00:00	N

7. PatientAppointmentOffer:

appointmentId	patientId	status	dateOfferSent	dateOfferExpire	dateReplied
2	4	noshow	2021-03-31 0:00:00	2021-04-01 0:00:00	2021-03-31 9:24:43
9	9	cancelled	2021-03-24 0:00:00	2021-03-25 0:00:00	2021-04-24 10:32:01
11	5	vaccinated	2021-03-27 0:00:00	2021-03-28 0:00:00	2021-03-27 8:15:21
12	5	declined	2021-03-25 0:00:00	2021-03-26 0:00:00	2021-03-25 12:15:51
12	9	cancelled	2021-03-30 0:00:00	2021-03-31 0:00:00	2021-04-30 9:12:01
13	9	cancelled	2021-04-01 0:00:00	2021-04-02 0:00:00	2021-04-01 17:15:22
16	4	noshow	2021-04-10 0:00:00	2021-04-11 0:00:00	2021-04-10 14:45:34

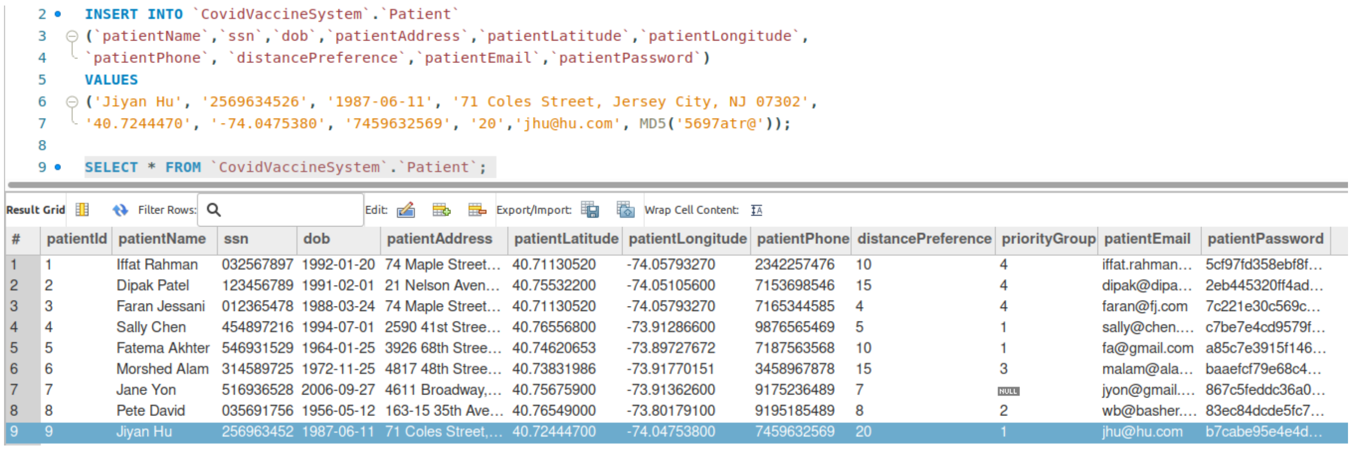
17	8	vaccinated	2021-04-18 0:00:00	2021-04-19 0:00:00	2021-04-18 10:32:01
21	6	vaccinated	2021-04-18 0:00:00	2021-04-19 0:00:00	2021-04-18 14:12:01
23	2	accepted	2021-04-25 0:00:00	2021-04-26 0:00:00	2021-04-25 2:12:01
37	3	notified	2021-04-25 0:00:00	2021-04-26 0:00:00	NULL

5.2. TEST QUERIES

This section will layout the queries that were executed on the test data set inserted into the database tables.

1. **Create a new patient account, together with email, password, name, date of birth, etc.**

```sql
INSERT INTO `CovidVaccineSystem`.`Patient`
(`patientName`,`ssn`,`dob`,`patientAddress`,`patientLatitude`,`patientLongitude`,
`patientPhone`, `distancePreference`,`patientEmail`,`patientPassword`)
VALUES
('Jiyan Hu', '2569634526', '1987-06-11', '71 Coles Street, Jersey City, NJ 07302',
'40.7244470', '-74.0475380', '7459632569', '20','jhu@hu.com', MD5('5697atr@'));
```



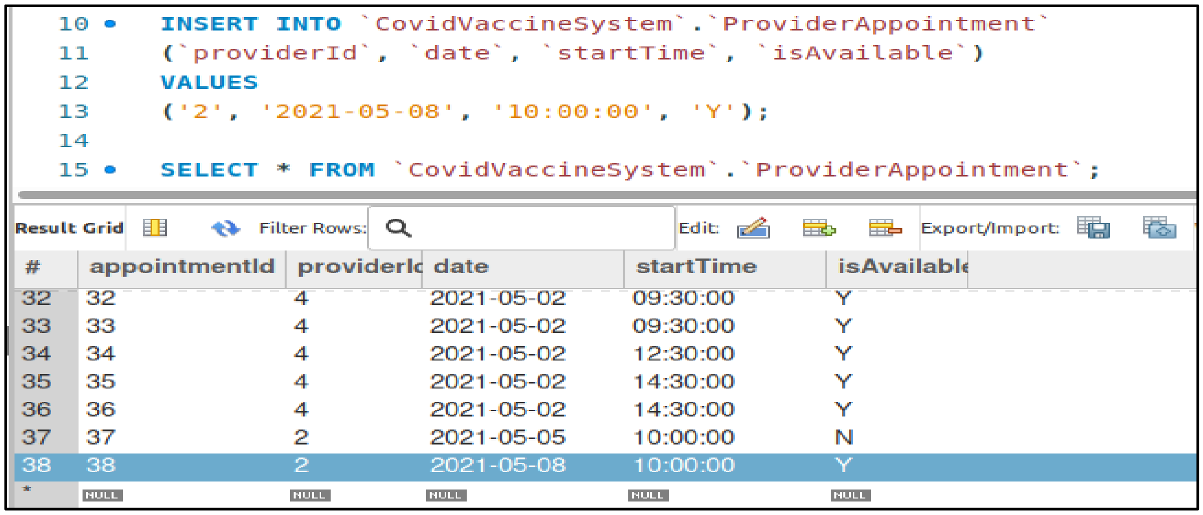
```sql
INSERT INTO `CovidVaccineSystem`.`Patient`
(`patientName`,`ssn`,`dob`,`patientAddress`,`patientLatitude`,`patientLongitude`,
`patientPhone`, `distancePreference`,`patientEmail`,`patientPassword`)
VALUES
('Jiyan Hu', '2569634526', '1987-06-11', '71 Coles Street, Jersey City, NJ 07302',
'40.7244470', '-74.0475380', '7459632569', '20','jhu@hu.com', MD5('5697atr@'));

SELECT * FROM `CovidVaccineSystem`.`Patient`;
```

#	patientId	patientName	ssn	dob	patientAddress	patientLatitude	patientLongitude	patientPhone	distancePreference	priorityGroup	patientEmail	patientPassword
1	1	Iffat Rahman	032567897	1992-01-20	74 Maple Street...	40.71130520	-74.05793270	2342257476	10	4	iffat.rahman...	5cf97fd358ebf8f...
2	2	Dipak Patel	123456789	1991-02-01	21 Nelson Aven...	40.75532200	-74.05105600	7153698546	15	4	dipak@dipa...	2eb445320ff4ad...
3	3	Faran Jessani	012365478	1988-03-24	74 Maple Street...	40.71130520	-74.05793270	7165344585	4	4	faran@fj.com	7c221e30c569c...
4	4	Sally Chen	454897216	1994-07-01	2590 41st Stree...	40.76556800	-73.91286600	9876565469	5	1	sally@chen....	c7be7e4cd9579f...
5	5	Fatema Akhter	546931529	1964-01-25	3926 68th Stree...	40.74620653	-73.89727672	7187563568	10	1	fa@gmail.com	a85c7e3915f146...
6	6	Morshed Alam	314589725	1972-11-25	4817 48th Stree...	40.73831986	-73.91770151	3458967878	15	3	malam@ala...	baaefcf79e68c4...
7	7	Jane Yon	516936528	2006-09-27	4611 Broadway,...	40.75675900	-73.91362600	9175236489	7	NULL	jyon@gmail....	867c5feddc36a0...
8	8	Pete David	035691756	1956-05-12	163-15 35th Ave...	40.76549000	-73.80179100	9195185489	8	2	wb@basher....	83ec84dcde5fc7...
9	9	Jiyan Hu	256963452	1987-06-11	71 Coles Street,...	40.72444700	-74.04753800	7459632569	20	1	jhu@hu.com	b7cabe95e4e4d...

Note: patientId is auto-incremented and the default value for priorityGroup is NULL.
Once the patientId is created, we will use that to insert a row into the PatientTimePreference table to add the patient’s time preferences.

2. **Insert a new appointment offered by a provider.**

```sql
INSERT INTO `CovidVaccineSystem`.`ProviderAppointment`
(`providerId`, `date`, `startTime`, `isAvailable`)
VALUES
('2', '2021-05-08', '10:00:00', 'Y');
```

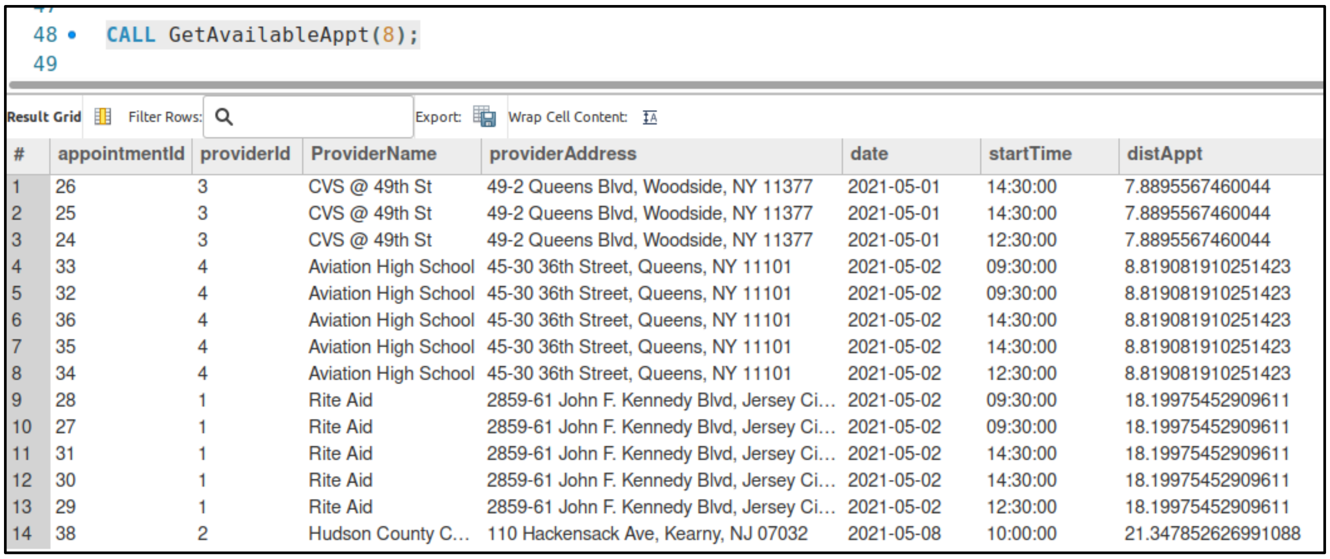


```
INSERT INTO `CovidVaccineSystem`.`ProviderAppointment`
(`providerId`, `date`, `startTime`, `isAvailable`)
VALUES
('2', '2021-05-08', '10:00:00', 'Y');

SELECT * FROM `CovidVaccineSystem`.`ProviderAppointment`;
```

#	appointmentId	providerId	date	startTime	isAvailable
32	32	4	2021-05-02	09:30:00	Y
33	33	4	2021-05-02	09:30:00	Y
34	34	4	2021-05-02	12:30:00	Y
35	35	4	2021-05-02	14:30:00	Y
36	36	4	2021-05-02	14:30:00	Y
37	37	2	2021-05-05	10:00:00	N
38	38	2	2021-05-08	10:00:00	Y
*	NULL	NULL	NULL	NULL	NULL

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.**

```
DELIMITER //
CREATE PROCEDURE GetAvailableAppt(IN id INT)
BEGIN
     WITH PATIENT_AVAILABILITY AS(
          SELECT p.patientId, p.patientLatitude, p.patientLongitude,
                 p.distancePreference, t.day, t.startTime, t.endTime
          FROM `CovidVaccineSystem`.`Patient` p
          INNER JOIN `CovidVaccineSystem`.`PatientTimePreference` pt ON pt.patientId =
p.patientId
          INNER JOIN `CovidVaccineSystem`.`TimeSlot` t ON t.slotId = pt.slotId
          WHERE p.patientId = id)

    SELECT a.appointmentId, p.providerId, p.ProviderName, p.providerAddress, a.date,
a.startTime, ST_Distance_Sphere(point(pa.patientLatitude, pa.patientLongitude),
              point(p.providerLatitude, p.providerLongitude))* 0.00062137 AS distAppt
    FROM `CovidVaccineSystem`.`ProviderAppointment` a
    CROSS JOIN PATIENT_AVAILABILITY pa
    INNER JOIN `CovidVaccineSystem`.`Provider` p ON p.providerId = a.providerId
    WHERE a.isAvailable = 'Y'
    AND a.date >= date(now())
    AND DAYNAME(a.date) = pa.day
    AND a.startTime >= pa.startTime AND a.startTime < pa.endTime
    ORDER BY distAppt ASC;
END //
DELIMITER ;

CALL GetAvailableAppt(8);
```



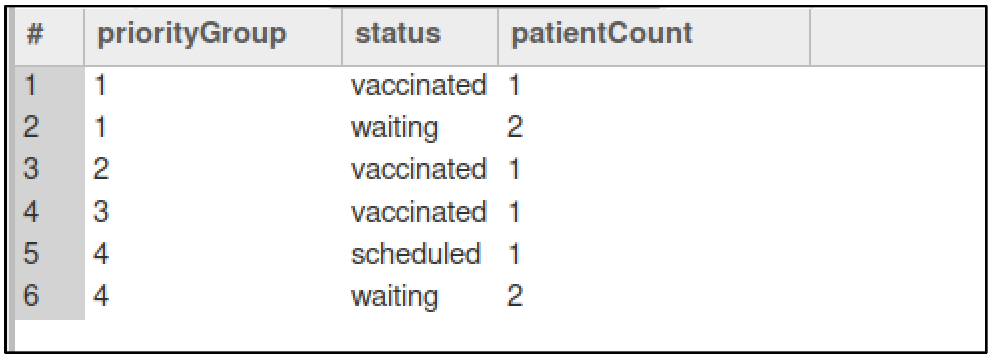
```
CALL GetAvailableAppt(8);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

#	appointmentId	providerId	ProviderName	providerAddress	date	startTime	distAppt
1	26	3	CVS @ 49th St	49-2 Queens Blvd, Woodside, NY 11377	2021-05-01	14:30:00	7.8895567460044
2	25	3	CVS @ 49th St	49-2 Queens Blvd, Woodside, NY 11377	2021-05-01	14:30:00	7.8895567460044
3	24	3	CVS @ 49th St	49-2 Queens Blvd, Woodside, NY 11377	2021-05-01	12:30:00	7.8895567460044
4	33	4	Aviation High School	45-30 36th Street, Queens, NY 11101	2021-05-02	09:30:00	8.819081910251423
5	32	4	Aviation High School	45-30 36th Street, Queens, NY 11101	2021-05-02	09:30:00	8.819081910251423
6	36	4	Aviation High School	45-30 36th Street, Queens, NY 11101	2021-05-02	14:30:00	8.819081910251423
7	35	4	Aviation High School	45-30 36th Street, Queens, NY 11101	2021-05-02	14:30:00	8.819081910251423
8	34	4	Aviation High School	45-30 36th Street, Queens, NY 11101	2021-05-02	12:30:00	8.819081910251423
9	28	1	Rite Aid	2859-61 John F. Kennedy Blvd, Jersey Ci...	2021-05-02	09:30:00	18.19975452909611
10	27	1	Rite Aid	2859-61 John F. Kennedy Blvd, Jersey Ci...	2021-05-02	09:30:00	18.19975452909611
11	31	1	Rite Aid	2859-61 John F. Kennedy Blvd, Jersey Ci...	2021-05-02	14:30:00	18.19975452909611
12	30	1	Rite Aid	2859-61 John F. Kennedy Blvd, Jersey Ci...	2021-05-02	14:30:00	18.19975452909611
13	29	1	Rite Aid	2859-61 John F. Kennedy Blvd, Jersey Ci...	2021-05-02	12:30:00	18.19975452909611
14	38	2	Hudson County C...	110 Hackensack Ave, Kearny, NJ 07032	2021-05-08	10:00:00	21.347852626991088

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.**

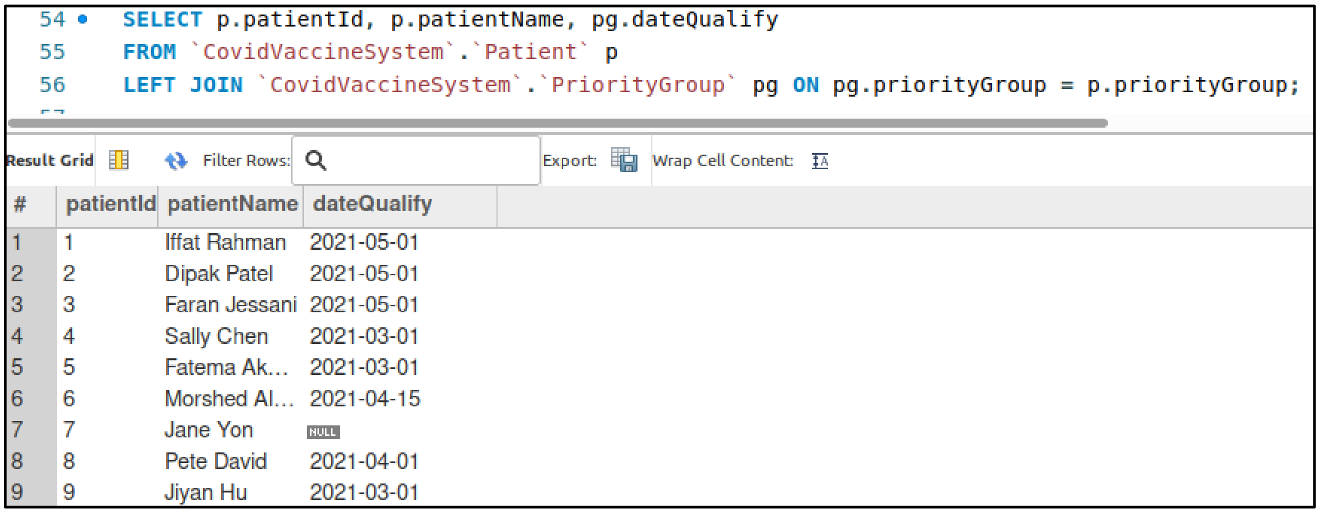
```
WITH PATIENT_APPT_STATUS AS (
	SELECT p.priorityGroup, p.patientId, pg.dateQualify, pao.dateOfferSent, 
pao.status
	FROM `CovidVaccineSystem`.`Patient` p
	INNER JOIN `CovidVaccineSystem`.`PriorityGroup` pg ON pg.priorityGroup = 
p.priorityGroup
	LEFT JOIN `CovidVaccineSystem`.`PatientAppointmentOffer` pao ON pao.patientId 
= p.patientId
),
VACCINATED_SCHEDULED_PATIENT AS (
	SELECT priorityGroup, patientId,
		(CASE WHEN status = 'accepted' THEN 'scheduled' ELSE status END) AS 
status
	FROM PATIENT_APPT_STATUS
	WHERE status = 'vaccinated' OR status = 'accepted'
),
AWAITING_PATIENT AS (
	SELECT priorityGroup, patientId, ('waiting') AS status
	 FROM PATIENT_APPT_STATUS
	 WHERE patientId NOT IN (SELECT patientId FROM VACCINATED_SCHEDULED_PATIENT )),

AGGRAGATED_STATUS AS (
	SELECT * FROM VACCINATED_SCHEDULED_PATIENT
	UNION SELECT * FROM AWAITING_PATIENT)

SELECT priorityGroup, status, count(patientId) as patientCount FROM AGGRAGATED_STATUS 
GROUP BY priorityGroup, status
ORDER BY priorityGroup;
```



#	priorityGroup	status	patientCount
1	1	vaccinated	1
2	1	waiting	2
3	2	vaccinated	1
4	3	vaccinated	1
5	4	scheduled	1
6	4	waiting	2

5. **For each patient, output the ID, name, and date when the patient becomes eligible for vaccination.**

```
-- Note: Query returns all patients including ones who do not have a priority group
assigned yet (they have NULL dateQualify)

SELECT p.patientId, p.patientName, pg.dateQualify
FROM `CovidVaccineSystem`.`Patient` p
LEFT JOIN `CovidVaccineSystem`.`PriorityGroup` pg ON pg.priorityGroup =
p.priorityGroup;
```

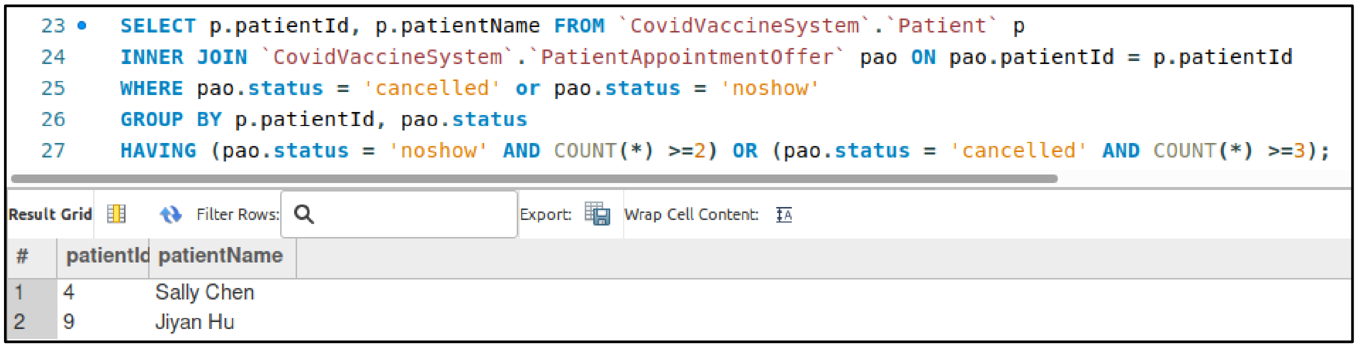


```
SELECT p.patientId, p.patientName, pg.dateQualify
FROM `CovidVaccineSystem`.`Patient` p
LEFT JOIN `CovidVaccineSystem`.`PriorityGroup` pg ON pg.priorityGroup = p.priorityGroup;
```

#	patientId	patientName	dateQualify
1	1	Iffat Rahman	2021-05-01
2	2	Dipak Patel	2021-05-01
3	3	Faran Jessani	2021-05-01
4	4	Sally Chen	2021-03-01
5	5	Fatema Ak...	2021-03-01
6	6	Morshed Al...	2021-04-15
7	7	Jane Yon	NULL
8	8	Pete David	2021-04-01
9	9	Jiyan Hu	2021-03-01

6. **Output the ID and name of 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.**

```
SELECT p.patientId, p.patientName FROM `CovidVaccineSystem`.`Patient` p
INNER JOIN `CovidVaccineSystem`.`PatientAppointmentOffer` pao
ON pao.patientId = p.patientId
WHERE pao.status = 'cancelled' or pao.status = 'noshow'
GROUP BY p.patientId, pao.status
HAVING (pao.status = 'noshow' AND COUNT(*) >=2) OR (pao.status = 'cancelled' AND
COUNT(*) >=3);
```

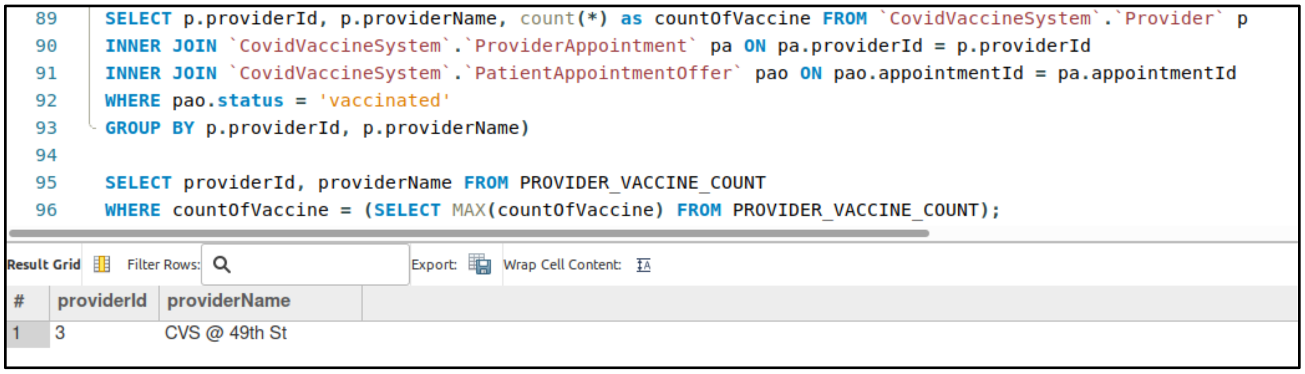


```
SELECT p.patientId, p.patientName FROM `CovidVaccineSystem`.`Patient` p
INNER JOIN `CovidVaccineSystem`.`PatientAppointmentOffer` pao ON pao.patientId = p.patientId
WHERE pao.status = 'cancelled' or pao.status = 'noshow'
GROUP BY p.patientId, pao.status
HAVING (pao.status = 'noshow' AND COUNT(*) >=2) OR (pao.status = 'cancelled' AND COUNT(*) >=3);
```

#	patientId	patientName
1	4	Sally Chen
2	9	Jiyan Hu

7. Output the ID and name of the provider(s) that has performed the largest number of vaccinations.

```
WITH PROVIDER_VACCINE_COUNT AS(
        SELECT p.providerId, p.providerName, count(*) as countOfVaccine
        FROM `CovidVaccineSystem`.`Provider` p
        INNER JOIN `CovidVaccineSystem`.`ProviderAppointment` pa
        ON pa.providerId = p.providerId
        INNER JOIN `CovidVaccineSystem`.`PatientAppointmentOffer` pao
        ON pao.appointmentId = pa.appointmentId
        WHERE pao.status = 'vaccinated'
        GROUP BY p.providerId, p.providerName)

SELECT providerId, providerName FROM PROVIDER_VACCINE_COUNT
WHERE countOfVaccine = (SELECT MAX(countOfVaccine) FROM PROVIDER_VACCINE_COUNT);
```



```
SELECT p.providerId, p.providerName, count(*) as countOfVaccine FROM `CovidVaccineSystem`.`Provider` p
INNER JOIN `CovidVaccineSystem`.`ProviderAppointment` pa ON pa.providerId = p.providerId
INNER JOIN `CovidVaccineSystem`.`PatientAppointmentOffer` pao ON pao.appointmentId = pa.appointmentId
WHERE pao.status = 'vaccinated'
GROUP BY p.providerId, p.providerName)

SELECT providerId, providerName FROM PROVIDER_VACCINE_COUNT
WHERE countOfVaccine = (SELECT MAX(countOfVaccine) FROM PROVIDER_VACCINE_COUNT);
```

#	providerId	providerName
1	3	CVS @ 49th St