PROJECT REPORT

1. **Please list out changes in directions of your project if the final project is different from your original proposal (based on your stage 1 proposal submission).**

   As proposed in Stage 1, the direction of the project remains the same. The main features include giving users a platform to communicate with doctors, schedule and monitor appointments, find nearby hospitals, reserve beds, and donate blood, which is extremely helpful for students who need it right away. This initiative aims to reduce patient treatment delays and enhance functional performance, quality of life, and survival. Moreover, doctors can take appointments with patients and approve/dismiss their immunization records. The university is provided with an overview of the entire system where it can monitor each student or students which are highly affected by some medical condition or are in immediate need of medical attention. They can locate nearest hospitals in proximity to them as well.

2. **Discuss what you think your application achieved or failed to achieve regarding its usefulness.**

   The website aims to assist users and students in receiving the immediate/necessary medical care as quickly as possible. Users can utilize the portal to change their personal information and schedule appointments for a specific time and day. Students may also include recent symptoms and vaccination histories. Additionally, students can request and list a specific doctor as a preference, and the doctor will approve the request. Furthermore, records of students are accessible to universities. This makes it easier to keep track of immunization records, as well as their severity, appointments, and amount of importance for student health. Doctors can also approve or reject a patient's immunization records and schedule visits with them. Universities can use this information provided by the students to understand any severe cases present or where there is a high infection rate of a disease within students.

3. **Discuss if you change the schema or source of the data for your application**

   The Schema of the data was not changed. However, the sources of data were slightly different from what was mentioned earlier.

Since the data found on kaggle, datahub.io and several other sources mentioned earlier did not match our schema, we had to scope further to find the datasets.

While, for a few relations like Doctor, University and Hospital we found data on sites like healthdata.gov and data.world. Some other relations like Students and BloodBanks were generated by us. These replicated real world data.

4. **Discuss what you change to your ER diagram and/or your table implementations. What are some differences between the original design and the final design? Why? What do you think is a more suitable design?**

   For our final design, we decided to remove the health insurance relation and medical store relation as these relations were expected to be a third party entity which requires additional complexities to the proposed system and cannot be incorporated into the system currently. The concept of health insurance, which requires additional understanding of claims and benefits along with charges is quite complex to incorporate in the system. On the other hand, the Medical store relation would consist of stores affiliated with the university and third party too, which added a complexity factor of incorporating them with the Doctors, University and Hospitals. Also, there were some minor changes in some relations, such as, the Hospital relation, we added a column named 'miles' which shows the distance of each hospital (using distance metric miles) for a particular hospital. Also, we added a severity index column for the student's table which displays the severity of the medical conditions provided by the student. The rest of the design and relations remain the same. The current design is quite efficient and does not require any third party entity involvement in the system. The current system establishes an appropriate network of university affiliated entities interacting with each other with ease and efficiency.

5. **Discuss what functionalities you added or removed. Why?**

   A: We added the part where students can fill their vaccination records directly via our portal. Once they have entered the details, we make use of NLP to verify their records. At the same time, we also add human touch by having the doctors verify the vaccination records to ensure high accuracy of the entire system. We initially used

Bag of words and later optimized our algorithm using Cosine similarity to find and calculate similarity scores within a vaccination record.

B: We also added the part where a university can see whether a student is flagged as "additional_assistance_required". This would basically indicate whether a particular student needs additional help to solve the problem that they are facing. This is implemented using NLP as well and the key factor driving this flag is the symptom that the student faces.

C: We have also added an urgent tab on the university login. This tab would basically show all the hospitals within a 5km radius around the university. The main usage of this feature is that whenever a student needs to book a book during an emergency, the university can directly have access to our tab and book a bed immediately for the student.

D: We have also added an analysis tab on the university login. This tab would show all the users that suffer from acute and severe diseases along with the number of people that suffer from a disease. This would help the university understand rising cases of any particular disease and they can take action accordingly.

E: Lastly, we have also added a patient condition flag on the doctor side that could take values such as "Mild", "Severe" & "Intermediate". This would help the university as well as the doctor understand the current situation of the patient before and after any appointments that they might have gone through.

6. **Explain how you think your advanced database programs complement your application.**

We implemented a stored procedure and a trigger in our application. The basic functionality of the stored procedure was to filter appointment time slots available for a particular doctor for a particular day. This implementation would be very tedious from a general database standpoint because for each doctor and for each day a table would be required to maintain the available and the unavailable time slots which would not be feasible. To avoid that, we required a temporary table which could store the available slots for a particular day for a particular doctor. This could only be

implemented using a stored procedure because it also required looping through the appointments table to check the booked slots for a particular doctor for a particular day. Hence using the stored procedure was extremely essential in our application.

The trigger implemented in the application involved updating the severity index dynamically based on the updates in the medical history. It was essential to implement triggers for this functionality since the value of the history count and the severity index needed to be updated as soon as the medical history values were updated by the user on the frontend side. These values are essential to be updated immediately so that the university can have the correct information about the students to perform accurate analysis.

7. **Each team member should describe one technical challenge that the team encountered. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or where to maintain your project.**

   Preet (ps72): While implementing the NLP module, Javascript does not provide a direct library or any method to calculate the vector form of a given word. Due to this issue, calculating the dot product of different words and their magnitudes was a cumbersome task. This problem was partially solved using tf-idf word embeddings and later used simple cosine conversion of the words to calculate the similarity score. In future, implementing soft cosine similarity in javascript will be very difficult as well. An alternative solution to this would be to set up a flask server and make an independent component for NLP processing.

   Sristi (sristii2): I first attempted to gather data for our application from the internet. I ended up manually entering data for the application because the relations and the characteristics were particularly created by us for our project. At first, I had basic triggers in mind, such as updating the current date. But the CRUD commands handled those circumstances. Later, following brainstorming, the idea of adding a severity index to student data, where the trigger is implemented, was developed.

   Siddharth (ss184) : While implementing the stored procedure to filter the available appointments, the values being compared were strings, and to return the appointments

in a sorted order it was important to compare an integer value. However, since the time slots are a range, sorting would have become very tedious. Hence, only the first 2 values of the string were retrieved and converted to an integer value for comparison. Moreover an extra condition had to be implemented to disable the same user from booking multiple appointments on the same day with the same doctor. For that we returned the earliest appointment booked by the user for the doctor on that particular day.

Kevin (kevind8) : Implementation of two advanced queries was a challenge, as most of the work can be done without using a query or just using the programming language. The usefulness and faster result of the advanced queries was essential to the system. The queries need to be implemented keeping in mind the overall scope and direction of the project.

8. **Are there other things that changed comparing the final application with the original proposal?**

   The only change made as compared to the original proposal is the fact that we are not providing assistance with health insurance and medical stores that come outside of the university. The reason behind dropping this part was the fact that we do not have enough data and information about how health insurances work and how McKinley representatives would allow or disallow a particular health insurance. Apart from this, remaining application from the original proposal has been successfully implemented.

9. **Describe future work that you think, other than the interface, that the application can improve on**

   A: One thing that can be improved is that we can make use of Soft Cosine Similarity instead of Cosine similarity for the NLP part. This would help us calculate scores for students that have diseases or symptoms that are unseen before.

   B: The entire NLP component should be moved to run on a separate flask server as calculation of vectors is slow and cumbersome in Javascript.

C. The inclusion of Health Insurance, where students can view their insurance if taken by the university or a third party. Based on the insurance plan, the benefits, payment plan and other claims can be taken care of while requiring medical attention. This can help students reduce unnecessary bills or extra payments which can be covered under an appropriate health-insurance plan.

D. Medical Stores, where students with the help of a doctor's prescription, can order and buy medicines. The medical stores affiliated with the university as well as third party stores can also be incorporated into the system.

10. Describe the final division of labor and how well you managed teamwork.

Preet (ps72):

1. Created logins for doctor, university and student.
2. Added NLP component to calculate cosine similarity scores to flag users.
3. Worked on CRUD operations on the university & student portal (GET, PUT, POST, DELETE) for blood banks, urgent, analysis & doctors tab.
4. Wrote advanced query for the analysis and urgent tabs on the university portal login and created indexes for the same.
5. Implemented custom hooks to simplify CRUD & NLP-based operations.

Siddharth (ss184):

1. Worked on creating the stored procedure for the project involving the filtering of appointments for a doctor.
2. Worked on the advanced query to find the nearest hospital to a particular university with beds available.
3. Worked on the backend api for the doctor portal involving the appointments and vaccination form approval by the doctor.
4. Created the frontend design and the profile page, appointment page and history page for the user and doctor.

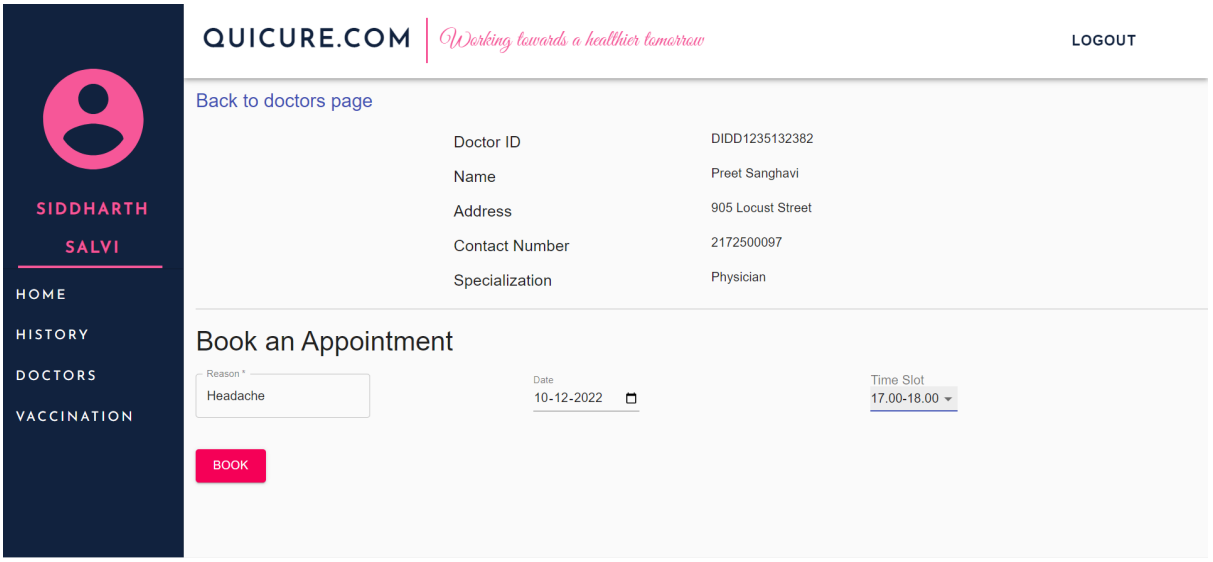
Kevin (kevind8):

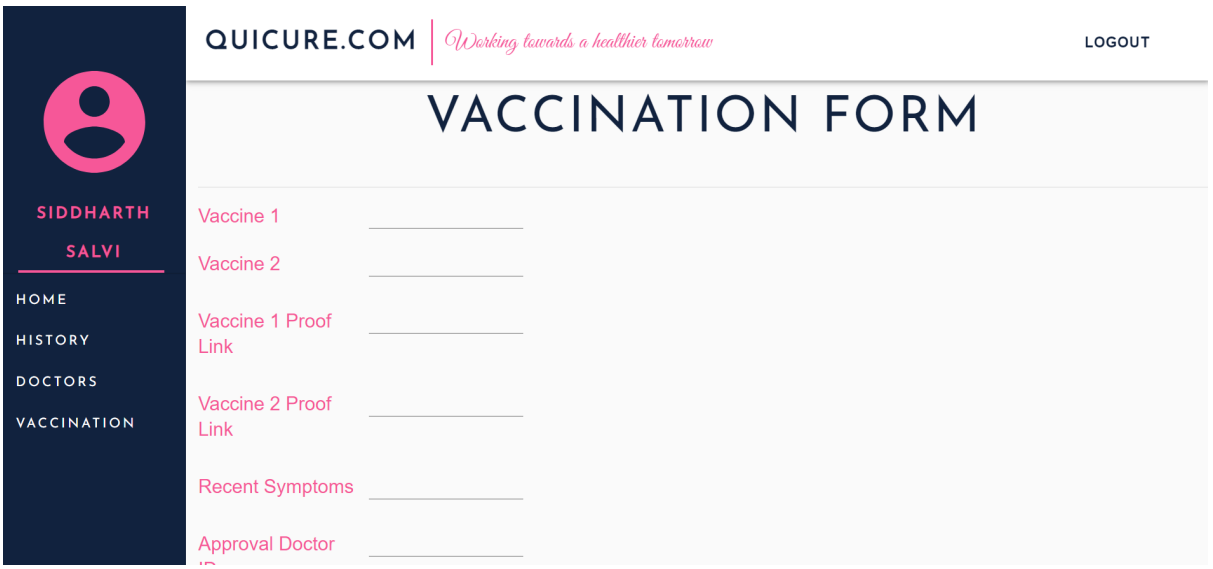
1. Worked on creating the database relations, collecting data and preprocessing data for the relations.
2. Worked on the indexing and analysis for the advanced queries provided in the urgent and hospital tabs.
3. Implemented the trigger for the university portal where the heath severity index was provided for each student.
4. Implemented the keyword search part to find hospitals or students using a faster keyword search.
5. Designed the UML diagram, created the CRUD commands for the database relations.

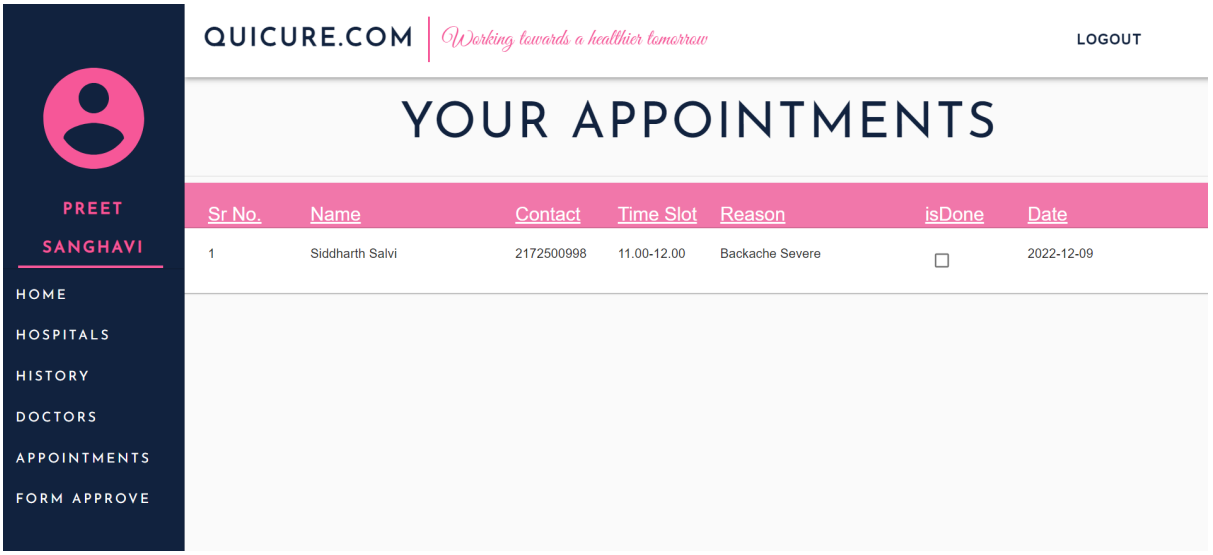
Sristi (sristii2)

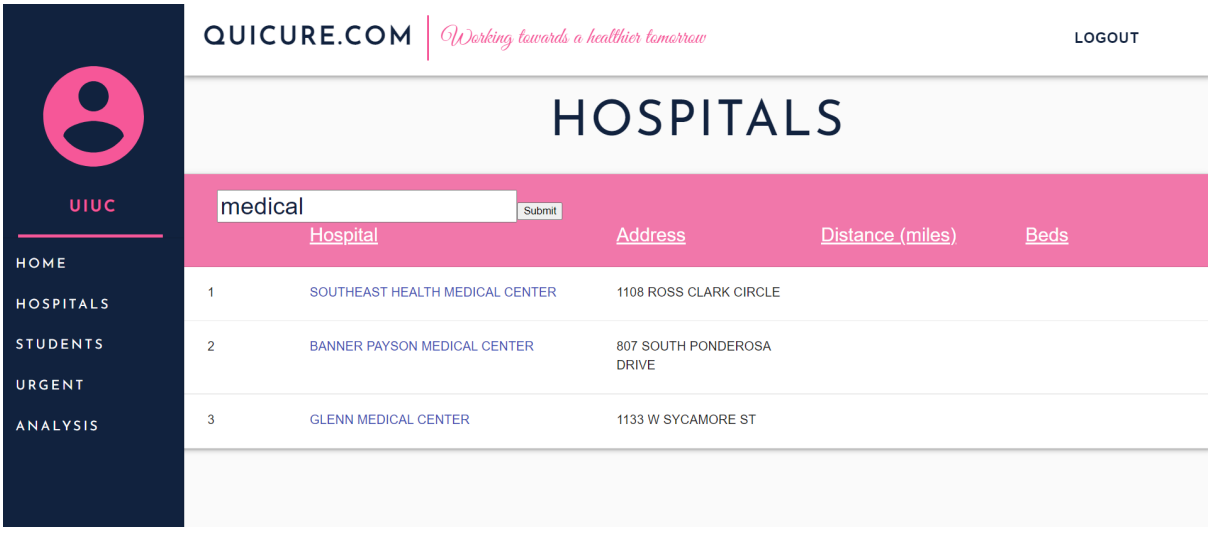
1. Designed the UML diagram, created the insert, delete, update modify commands for the database relations.
2. Worked on the trigger for the university portal in order to provide the health severity index of students.
3. Created the database relations, collected data and preprocessing data for the relations.
4. Worked on the backend part for the doctor portal to add appointments and vaccination form approval by the doctor.

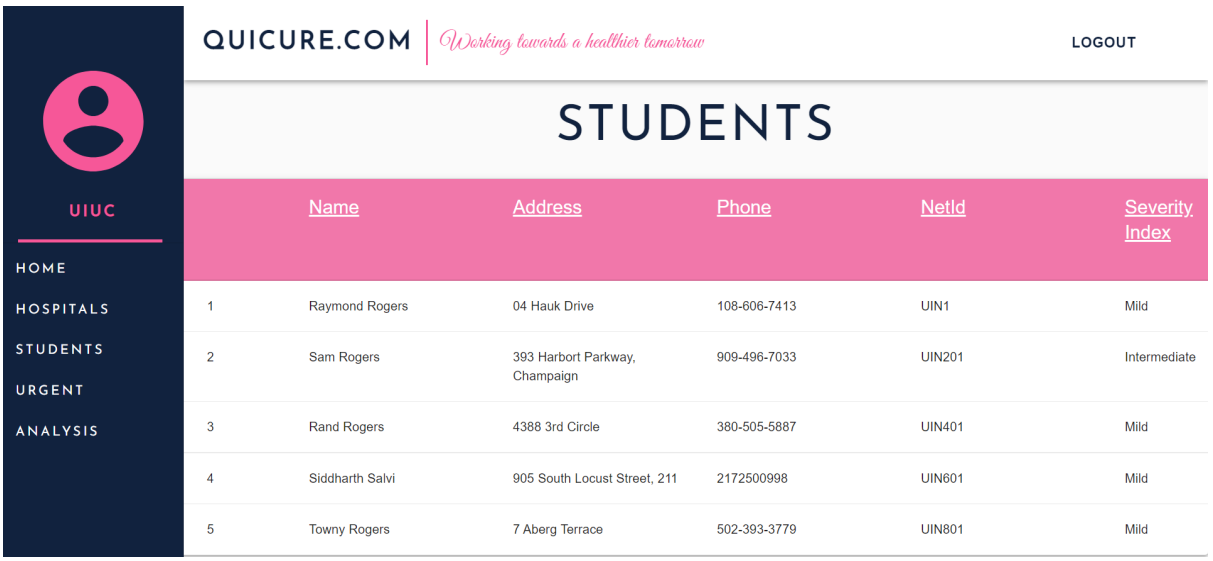
11. SCREENSHOTS:




QUICURE.COM
Working towards a healthier tomorrow
LOGOUT
SIDDHARTH SALVI
HOME
HISTORY
DOCTORS
VACCINATION
VACCINATION FORM
Vaccine 1
Vaccine 2
Vaccine 1 Proof Link
Vaccine 2 Proof Link
Recent Symptoms
Approval Doctor


QUICURE.COM
Working towards a healthier tomorrow
LOGOUT
PREET SANGHAVI
HOME
HOSPITALS
HISTORY
DOCTORS
APPOINTMENTS
FORM APPROVE
YOUR APPOINTMENTS
Sr No. Name Contact Time Slot Reason isDone Date
1 Siddharth Salvi 2172500998 11.00-12.00 Backache Severe 2022-12-09


QUICURE.COM
Working towards a healthier tomorrow
LOGOUT
UIUC
HOME
HOSPITALS
STUDENTS
URGENT
ANALYSIS
HOSPITALS
medical
Submit
Hospital Address Distance (miles) Beds
1 SOUTHEAST HEALTH MEDICAL CENTER 1108 ROSS CLARK CIRCLE
2 BANNER PAYSON MEDICAL CENTER 807 SOUTH PONDEROSA DRIVE
3 GLENN MEDICAL CENTER 1133 W SYCAMORE ST



	Name	Address	Phone	NetId	Severity Index
1	Raymond Rogers	04 Hauk Drive	108-606-7413	UIN1	Mild
2	Sam Rogers	393 Harbort Parkway, Champaign	909-496-7033	UIN201	Intermediate
3	Rand Rogers	4388 3rd Circle	380-505-5887	UIN401	Mild
4	Siddharth Salvi	905 South Locust Street, 211	2172500998	UIN601	Mild
5	Towny Rogers	7 Aberg Terrace	502-393-3779	UIN801	Mild

TRIGGER:

```
use dataproj;
DELIMITER $$
CREATE TRIGGER med_history_trigger2
BEFORE UPDATE ON Users
FOR EACH ROW
BEGIN

SET @History_Count =  (SELECT (LENGTH(Medical_History) - LENGTH(REPLACE(Medical_History,",",""))) + 1) from Users where New.UserID= UserID);

IF @History_Count < 3 THEN
SET New.Severity_Index = "Mild";
SET New.History_Count = @History_Count;
END IF;

IF @History_Count > 6 THEN
SET New.Severity_Index = "Severe";
SET New.History_Count = @History_Count;
END IF;

IF @History_Count <= 6 AND @History_Count > 3 THEN
SET New.Severity_Index = "Intermediate";
SET New.History_Count = @History_Count;
END IF;

END $$

DELIMITER ;
```

STORED PROCEDURE

```
DELIMITER $$
CREATE PROCEDURE Check_Slots3(IN doc_id VARCHAR(100), date_val VARCHAR(100))
BEGIN
    DECLARE varAppNum INT;
```

```
        DECLARE varSlot VARCHAR(100);
    DECLARE varIsDone VARCHAR(100);
    DECLARE varAid INT;
    DECLARE varpid VARCHAR(100);
    DECLARE exit_loop BOOLEAN DEFAULT FALSE;
     DECLARE custCur CURSOR FOR(SELECT p_uid, MIN(CAST(SUBSTRING(timeslot,
1, 2) AS UNSIGNED)) AS MIN_TIMESLOT
                                                    FROM APPOINTMENT INNER
JOIN users ON APPOINTMENT.p_uid = users.UserID
                                                    WHERE  d_uid  =  doc_id  AND
doa = date_val
                                                    GROUP BY p_uid
                                                    ORDER BY p_uid
                                                );

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET exit_loop = TRUE;
        DROP TABLE IF EXISTS App_Slots;
    CREATE TABLE App_Slots(
    start_val INT PRIMARY KEY,
        time_slot VARCHAR(100),
        isDone VARCHAR(100)
        );
    INSERT INTO App_Slots
        VALUES
        (10,"10.00-11.00","No"),
        (11,"11.00-12.00","No"),
        (17,"17.00-18.00","No"),
        (18,"18.00-19.00","No"),
        (19,"19.00-20.00","No"),
        (20,"20.00-21.00","No");
        OPEN custCur;
    cloop: LOOP
                FETCH custCur INTO varpid, varSlot;
       IF(exit_loop) THEN
```

```
            LEAVE cloop;
        END IF;
        IF (varSlot IN (SELECT start_val FROM App_Slots)) THEN
        UPDATE App_Slots SET isDone = "Yes" WHERE  start_val = varSlot;
        END IF;


    END LOOP cloop;
    CLOSE custCur;
    SELECT time_slot FROM App_Slots WHERE isDone = "No";
END$$
DELIMITER ;
```