# COVID TEST MANAGEMENT SYSTEM
# -Data in the time of Covid-19

PROJECT REVIEW 3 REPORT

Submitted by


**Nakul Jadeja (19BCE0660)**

**Sayan Saha (19BCE0510)**

**Vardhan Khara (19BCE0833)**

**Swayam Shashwat Sharma (19BCE0523)**


Prepared For

**Database Management Systems (CSE2004)- PROJECT COMPONENT**

Submitted to


**Dr. Vellingiri Jayagopal**

**Assistant Professor (Senior)**

**School of Information Technology**

**VIT University**

# *ACKNOWLEDGEMENT*

*We are thankful to the Department because of whom, we have gained confidence in Innovative Thinking and it also enhanced our professional skills as to become competent in this field.*

*In performing our project, we had to take the help and guideline of some respected persons, who deserve our greatest gratitude. The completion of this project gives us much Pleasure. We would like to show our gratitude to* ***Prof. Vellingiri Jayagopal, SITE VIT University*** *for giving us a good guideline for project throughout numerous consultations. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in this project.*

*Thank You*

# CONTENTS

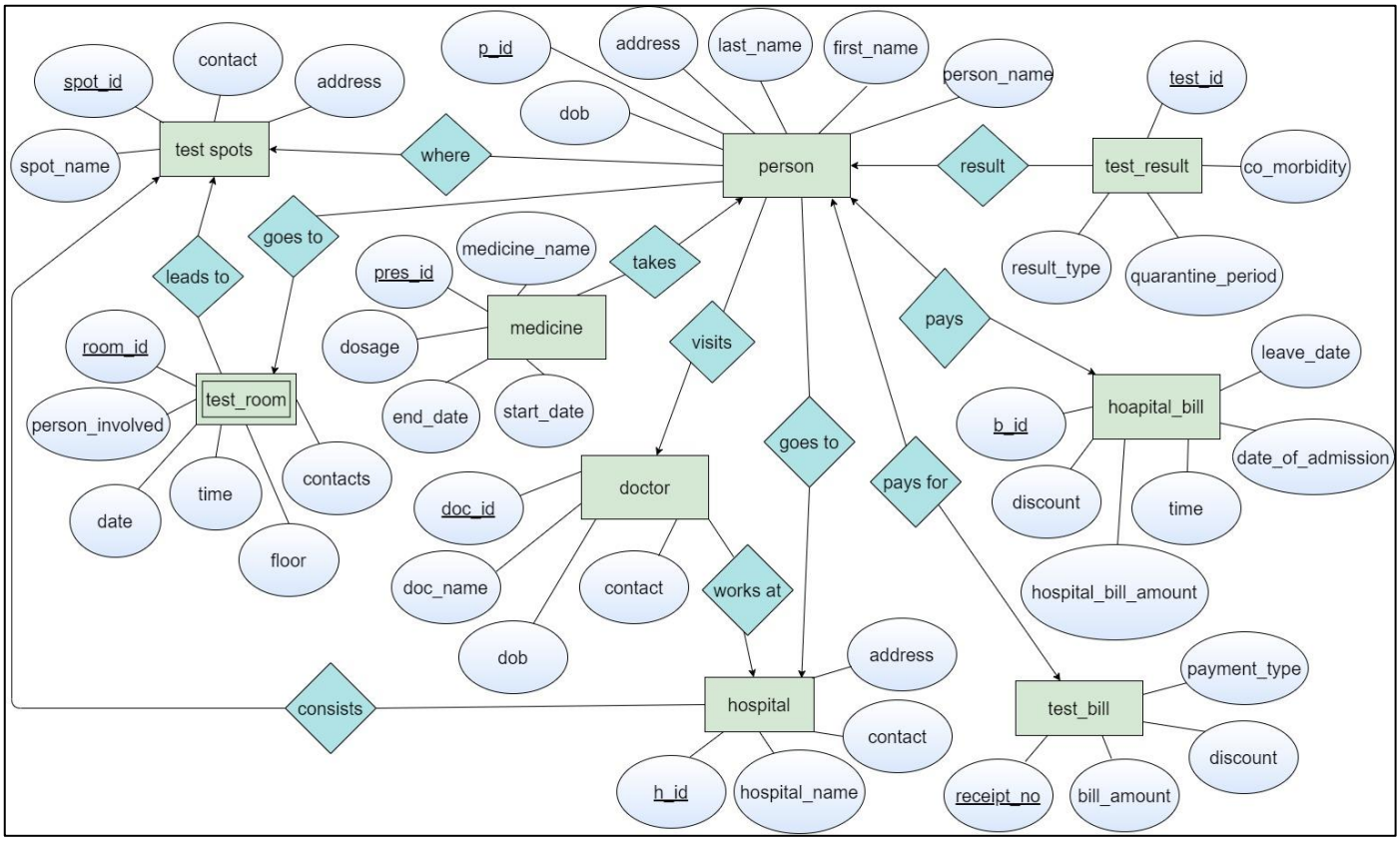
## 1. Abstract

As the COVID-19 crisis endures and the virus continues to spread globally, the need for collecting epidemiological data and patient information also grows exponentially. The race against the clock to find a cure and a vaccine to the disease means researchers require storage of increasingly large and diverse types of information; for doctors following patients, recording symptoms and reactions to treatments, the need for storage flexibility is only surpassed by the necessity of storage security. The volume, variety, and variability of COVID-19 patient data requires storage in SQL database management systems (DBMSs). But with a multitude of existing of SQL DBMSs, there is no straightforward way for institutions to select the most appropriate. And more importantly, they suffer from security flaws that would render them inappropriate for the storage of confidential patient data.

## 2. Introduction

This project develops an innovative solution to remedy the aforementioned shortcomings. COVID-19 patients, as well as medical professionals, could be subjected to privacy-related risks, from abuse of their data to community bullying regarding their medical condition. Thus, in addition to being appropriately stored and analyzed, their data must imperatively be highly protected against misuse.

## 3. **Entity Relationship Diagram**




## Relationship-set:

- person-test_spot(many to one)
- person-medicine(one to many)
- person-doctor(many to one)
- person-test_result(one to many)
- person-test_bill(one to one)
- person-hospital_bill(one to one)
- person-hospital(many to one)
- doctor-hospital(many to one)

## 4. Entity Relationship Diagram to Schema conversion

- Person(p_id, first_name, last_name, gender, DOB, contact, address, spot_id, h_id, doc_id)
- Test_spots(spot_id, spot_name, address,contact)
- Test_room(room_id, person_involved, contacts, floor, time, date, spot_id)
- Test_results(test_id, result_type, co-morbidity, quarantine_period, p_id)
- Test_bill(p_id, receipt_no, bill_amount, discount, payment_type)
- Hospital(h_id, hospital_name, address, contact)
- Doctor(doc_id, doc_name, DOB, contact, department, h_id)
- Medicine(Pres_id, medicine_name, dosage, start_date, end_date, p_id)
- Hospital_bill(b_id, date_of_admission, leave_date, time, hospital_bill_amount, discount, p_id)

## 5. Creation of Relational Tables

Person:

```
mysql> desc Person;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| p_id       | varchar(25) | NO   | PRI | NULL    |       |
| first_name | varchar(25) | NO   |     | NULL    |       |
| last_name  | varchar(25) | NO   |     | NULL    |       |
| gender     | varchar(25) | NO   |     | NULL    |       |
| DOB        | date        | NO   |     | NULL    |       |
| contact    | varchar(25) | NO   |     | NULL    |       |
| address    | varchar(25) | NO   |     | NULL    |       |
| spot_id    | varchar(25) | YES  | MUL | NULL    |       |
| h_id       | varchar(25) | YES  | MUL | NULL    |       |
| doc_id     | varchar(25) | YES  | MUL | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

Test Spots:

```
mysql> desc test_spots;
+-----------+-------------+------+-----+---------+-------+
| Field     | Type        | Null | Key | Default | Extra |
+-----------+-------------+------+-----+---------+-------+
| spot_id   | varchar(25) | NO   | PRI | NULL    |       |
| spot_name | varchar(25) | NO   |     | NULL    |       |
| address   | varchar(25) | NO   |     | NULL    |       |
| contact   | varchar(25) | NO   |     | NULL    |       |
+-----------+-------------+------+-----+---------+-------+
```

Test Room:

```
mysql> desc Test_room;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| room_id         | varchar(25) | NO   | PRI | NULL    |       |
| person_involved | varchar(25) | NO   |     | NULL    |       |
| contacts        | varchar(25) | NO   |     | NULL    |       |
| floor           | int         | NO   |     | NULL    |       |
| time            | varchar(25) | NO   |     | NULL    |       |
| Date            | date        | NO   |     | NULL    |       |
| sp_id           | varchar(25) | NO   |     | NULL    |       |
| spot_id         | varchar(25) | YES  | MUL | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
```

Test Results:

```
mysql> desc Test_results;
+------------------+-------------+------+-----+---------+-------+
| Field            | Type        | Null | Key | Default | Extra |
+------------------+-------------+------+-----+---------+-------+
| test_id          | varchar(25) | NO   | PRI | NULL    |       |
| result_type      | varchar(25) | NO   |     | NULL    |       |
| co_morbidity     | varchar(25) | NO   |     | NULL    |       |
| quarantine_period | varchar(25) | NO   |     | NULL    |       |
| p_id             | varchar(25) | YES  | MUL | NULL    |       |
+------------------+-------------+------+-----+---------+-------+
```

Test Bill:

```
mysql> desc Test_bill;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| p_id         | varchar(25) | NO   | MUL | NULL    |       |
| receipt_no   | varchar(25) | NO   | PRI | NULL    |       |
| bill_amount  | int         | NO   |     | NULL    |       |
| discount     | int         | YES  |     | NULL    |       |
| payment_type | varchar(25) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
```

Hospital:

```
mysql> desc Hospital;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| h_id          | varchar(25) | NO   | PRI | NULL    |       |
| hospital_name | varchar(25) | NO   |     | NULL    |       |
| address       | varchar(25) | NO   |     | NULL    |       |
| contact       | varchar(25) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

Doctor:

```
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| doc_id     | varchar(25) | NO   | PRI | NULL    |       |
| doc_name   | varchar(25) | NO   |     | NULL    |       |
| DOB        | date        | YES  |     | NULL    |       |
| contact    | varchar(25) | NO   |     | NULL    |       |
| department | varchar(25) | NO   |     | NULL    |       |
| h_id       | varchar(25) | NO   | MUL | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

Medicine:

```
mysql> desc Medicine;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| Pres_id       | varchar(25) | NO   | PRI | NULL    |       |
| medicine_name | varchar(25) | NO   |     | NULL    |       |
| dosage        | varchar(25) | NO   |     | NULL    |       |
| start_date    | date        | NO   |     | NULL    |       |
| end_date      | date        | NO   |     | NULL    |       |
| p_id          | varchar(25) | YES  | MUL | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

Hospital Bill:

```
mysql> desc Bill_hospital;
+----------------------+-------------+------+-----+---------+-------+
| Field                | Type        | Null | Key | Default | Extra |
+----------------------+-------------+------+-----+---------+-------+
| b_id                 | varchar(25) | NO   | PRI | NULL    |       |
| date_of_admission    | date        | NO   |     | NULL    |       |
| leave_date           | date        | NO   |     | NULL    |       |
| time                 | varchar(25) | YES  |     | NULL    |       |
| hospital_bill_amount | int         | NO   |     | NULL    |       |
| discount             | int         | YES  |     | NULL    |       |
| p_id                 | varchar(25) | NO   | MUL | NULL    |       |
+----------------------+-------------+------+-----+---------+-------+
```

## 6. Normalisation of Tables:

Entity sets:

1. Test_bill(receipt_no,bill_amount,discount,payment_type,p_id)

| Receipt_no | Bill_amount | discount | Payment_type | p_id |
|---|---|---|---|---|
| R0001 | 2000 | 60 | Card | P0003 |
| R0002 | 2500 | 50 | Cash | P0003 |
| R0003 | 3000 | 20 | Card | P0001 |
| R0004 | 4000 | null | Card | P0002 |

Functional dependency:

Receipt_no--->p_id,bill_amount,discount,payment_type

(receipt_no)+ = receipt_no, p_id, bill_amount, discount, payment_type)

Hence receipt_no is the candidate key.

- Checking for 1NF

| Receipt_no | Bill_amount | discount | Payment_type | p_id |
|---|---|---|---|---|
| R0001 | 2000 | 60 | Card | P0003 |
| R0002 | 2500 | 50 | Cash | P0003 |
| R0003 | 3000 | 20 | Card | P0001 |
| R0004 | 4000 | null | Card | P0002 |

Conditions:

1. All the attributes are atomic in the above table, they cannot be further divided.
2. All the attributes are single valued.

   As the conditions for 1NF are met, hence we can conclude the table is in 1NF.

- Checking for 2NF

| Receipt_no | Bill_amount | discount | Payment_type | p_id |
|---|---|---|---|---|
| R0001 | 2000 | 60 | Card | P0003 |
| R0002 | 2500 | 50 | Cash | P0003 |
| R0003 | 3000 | 20 | Card | P0001 |
| R0004 | 4000 | null | Card | P0002 |

Conditons:

1. The table is in 1NF already.
2. There is no partial functional dependency present as receipt_no is the only candidate key present in the table.

   Hence the table test_bill is in 2NF.

- Checking for 3NF:

| Receipt_no | Bill_amount | discount | Payment_type | p_id |
|---|---|---|---|---|
| R0001 | 2000 | 60 | Card | P0003 |
| R0002 | 2500 | 50 | Cash | P0003 |
| R0003 | 3000 | 20 | Card | P0001 |
| R0004 | 4000 | null | Card | P0002 |

Conditions for 3NF:

1. The above table is already is 2NF.
2. There are no non-key attributes which determine other non-key attributes ,the key is receipt_no only and hence no possibility of transitivity.

So we can say the above table is in 3NF.

- Checking for BCNF

| Receipt_no | Bill_amount | discount | Payment_type | p_id |
|---|---|---|---|---|
| R0001 | 2000 | 60 | Card | P0003 |
| R0002 | 2500 | 50 | Cash | P0003 |
| R0003 | 3000 | 20 | Card | P0001 |
| R0004 | 4000 | null | Card | P0002 |

Conditions:

1. The table is in 3NF
2. Here receipt_no is the only super key which uniquely determines all the other attributes present in the test_bill table.

   Hence the table is in BCNF.

2.Doctor(doc_id,doc_name,department,contact,address,DOB,h_id)

| Doc_id | Doc_name | department | contact | address | DOB | H_id |
|---|---|---|---|---|---|---|
| D0001 | manu | surgeon | 9327555811 | gujarat | 2004-09-12 | H0001 |
| D0003 | nitesh | cardio | 9328555611 | Madhya pradesh | 2004-09-14 | H0001 |
| D0002 | arjun | paedia | 9925008589 | rajasthan | 2004-02-27 | H0001 |
| D0005 | chavi | paedia | 9375499974 | Andhra pradesh | 2004-01-1 | H0001 |

Functional dependency defined as:

- doc_id→department
- doc_id,doc_name→department,contact,address,DOB,h_id

Finding candidate key by finding closure:

- (doc_id)+ =doc_id,department != Doctor
  Hence it is not a candidate key

- (doc_id,doc_name)+=doc_id,department,doc_name,contact,address,DOB ,h_id = Doctor

Hence (doc_id,doc_name) is together a candidate key.

Checking for all normalisations for the table doctor:

- Checking for 1NF

| Doc_id | Doc_name | department | contact | address | DOB | H_id |
|---|---|---|---|---|---|---|
| D0001 | manu | surgeon | 9327555811 | gujarat | 2004-09-12 | H0001 |
| D0003 | nitesh | cardio | 9328555611 | Madhya pradesh | 2004-09-14 | H0001 |
| D0002 | arjun | paedia | 9925008589 | rajasthan | 2004-02-27 | H0001 |
| D0005 | chavi | paedia | 9375499974 | Andhra pradesh | 2004-01-1 | H0001 |

Conditions:

1. All the attributes are atomic in the above table ,they cannot be further divided.
2. All the attributes are single valued.

As the conditions for 1NF are met ,hence we can conclude the table is in 1NF.


| Doc_id | Doc_name | department | contact | address | DOB | H_id |
|---|---|---|---|---|---|---|
| D0001 | manu | surgeon | 9327555811 | gujarat | 2004-09-12 | H0001 |
| D0003 | nitesh | cardio | 9328555611 | Madhya pradesh | 2004-09-14 | H0001 |
| D0002 | arjun | paedia | 9925008589 | rajasthan | 2004-02-27 | H0001 |
| D0005 | chavi | paedia | 9375499974 | Andhra pradesh | 2004-01-1 | H0001 |


- Checking for 2NF:

Conditions:
1.the table is already in 1NF.

2.This table consist of a partial key dependency as doc_id is a non-key attribute here,hence the table is not in 2NF

Doc_id→department

doc_id,doc_name→department,contact,address,DOB,h_id

To decompose the above table into 2NF
1) D1(doc_id,department)
2) D2(doc_id,doc_name,contact,address,DOB,h_id)

Now the above table is in 2NF.

- Checking for 3NF:

1) D1(doc_id,department)
2) D2(doc_id,doc_name,contact,address,DOB,h_id)

Conditions:

1.The table already is in 2NF.

2.No non-key attributes are forming key and hence no transitivity is seen.

Hence the table is in 3NF.

- Checking for BCNF:

1) D1(doc_id,department)
2) D2(doc_id,doc_name,contact,address,DOB,h_id)

Conditions:
1)The table is already in 3NF form.

2)Each determinant has a key

D1 relation has doc_id as the key
D2 relation has doc_id,doc_name as key

Hence the above decomposition is in BCNF .

- Test_results(test_id, result_type, co-morbidity, quarantine_period, p_id )

  Functional dependency defined as:

  test_id->result_type,co-morbidity,quarantine_period,p_id

(test_id)+ =test_id, result_type, co-morbidity, quarantine_period, p_id =test_results

As every determinant declared above has a super key,in this case test_id,hence the table is already in BCNF

- Hospital(h_id ,hospital_name ,address ,contact )

  Functional dependency defined as:

h_id->hospital_name,address,contact

(h_id)+ =h_id, hospital_name,address,contact =hospital

As every determinant declared above has a super key,in this case h_id,hence the table is already in BCNF

- Medicine(Pres _id, medicine_name, dosage, start_date , end_date date, p_id)

  Functional dependency defined as:

Pres_id->medicine_name, dosage,start_date, end_date, p_id

(pres_id)+ =pres_id, medicine_name,dosage,start_date,end_date,p_id =medicine

As every determinant declared above has a super key,in this case pres_id,hence the table is already in BCNF

- Bill_hospital(b_id, date_of_admission, leave_date, time, hospital_bill_amount, p_id )

Functional dependency defined as:

b_id ->date_of_admission, leave_date, time, hospital_bill_amount, p_id

(b_id)+ =b_id, date_of_admission, leave_date, time, hospital_bill_amount, p_id = bill_hospital

As every determinant declared above has a super key,in this case b_id,hence the table is already in BCNF

- Test_room(spot_id, room_id, person_involved, contacts, floor, time, Date )

  Spot_id,room_id->Date,time,person_involved,floor,contacts

  (spot_id,room_id)+ =spot_id, room_id, Date,time, person_involved, floor, contacts =test_room

  As test_room is a weak entity set room_id ,spot_id together form the composite key

  As every determinant declared above has a super key,in this case (room_id,spot_id),hence the table is already in BCNF.

- Person(p_id , first_name, last_name, gender, DOB, contact, address, spot_id, doc_id, h_id)

  p_id ->first_name , last_name ,gender, DOB, contact, address , spot_id, doc_id, h_id

  (p_id)+ =p_id, first_name , last_name ,gender, DOB, contact, address, spot_id, doc_id, h_id =person

Now as here contact is a multi-valued attribute in the person table hence we need to decompose it

Schema1: (p_id, contact)

Schema2: (p_id,first_name, last_name, gender, DOB, address, spot_id, doc_id, h_id)

Now As every determinant has a super key in both the above tables  it is p_id,hence the table is now in BCNF

- Test_spots(spot_id ,spot_name ,address,contact)

  spot_id ->spot_name ,address,contact

(spot_id)+ =spot_id, spot_name ,address,contact

As every determinant declared above has a super key,in this case it is spot_id, hence the table is already in BCNF

# 7. Code for Database and Front-end connectivity

## 7.1 Code for Database

```
create database dbms;
use dbms;
create table employee(name varchar(25),age int);
insert into employee values("dhamnn",25);
select *from employee;
show tables;
use dbms;
create table users(name varchar(25) NOT NULL primary key,email varchar(25));
desc users;
use dbms;
select * from users;
use dbms;
create table test_spot(spot_id varchar(25) NOT NULL primary key,spot_name varchar(25) NOT NULL,
address varchar(25) NOT NULL,contact varchar(25) NOT NULL);

create table hospital(h_id varchar(25) NOT NULL UNIQUE primary key,hospital_name varchar(25) NOT
NULL
,address varchar(25) NOT NULL,contact varchar(25)
, spot_id varchar(25) NOT NULL,FOREIGN KEY(spot_id) REFERENCES test_spot(spot_id));

create table doctor(doc_id varchar(25) NOT NULL UNIQUE primary key,doc_name varchar(25) NOT
NULL
,DOB date,contact varchar(25) NOT NULL,h_id varchar(25) NOT NULL ,
FOREIGN KEY(h_id) REFERENCES hospital(h_id));

create table doc_dept(doc_id varchar(25) NOT NULL primary key,department varchar(25));

create table person(p_id varchar(25) NOT NULL UNIQUE primary key,first_name varchar(25) NOT
NULL,
last_name varchar(25) NOT NULL,gender varchar(25) NOT NULL,DOB date NOT NULL,
address varchar(25) NOT NULL,spot_id varchar(25)
,h_id varchar(25) ,doc_id varchar(25) ,FOREIGN KEY(spot_id) REFERENCES test_spot(spot_id),
FOREIGN KEY(h_id) REFERENCES hospital(h_id),FOREIGN KEY(doc_id) REFERENCES
doctor(doc_id));

create table person_contacts(p_id varchar(25) NOT NULL,contacts varchar(55) NOT null,
FOREIGN KEY(p_id) REFERENCES person(p_id));
```

```
create table test_room(room_id varchar(25) NOT NULL UNIQUE primary key,person_involved
varchar(25) NOT NULL
,contacts varchar(25) NOT NULL,floor int NOT NULL
,time varchar(25) NOT NULL,Date date NOT NULL,spot_id varchar(25) NOT NULL,p_id varchar(25)
NOT NULL,
FOREIGN KEY(p_id) REFERENCES person(p_id),FOREIGN KEY(spot_id) REFERENCES
person(spot_id));

create table test_results(test_id varchar(25) NOT NULL primary key,
result_type varchar(25) NOT NULL,comorbidity varchar(25) NOT NULL,
quarantine_period varchar(25) NOT NULL,room_id varchar(25) ,
FOREIGN KEY(room_id) references test_room(room_id));

create table test_bill(p_id varchar(25) NOT NULL,receipt_no varchar(25) NOT NULL UNIQUE,
bill_amount int NOT NULL,discount int ,payment_type varchar(25) NOT NULL,FOREIGN KEY(p_id)
references person(p_id),
PRIMARY KEY(receipt_no));

create table medicine(Pres_id varchar(25) NOT NULL UNIQUE,medicine_name varchar(25) NOT NULL
,dosage varchar(25) NOT NULL,start_date date NOT NULL,
end_date date NOT NULL,PRIMARY KEY(Pres_id));

create table takes(Pres_id varchar(25) NOT NULL,p_id varchar(25) NOT NULL,foreign key(Pres_id)
references
medicine(Pres_id),foreign key(p_id) references person(p_id));

create table bill_hospital(b_id varchar(25) NOT NULL UNIQUE,date_of_admission date NOT NULL,
leave_date date NOT NULL,time varchar(25),hospital_bill_amount int NOT NULL,
discount int ,p_id varchar(25) NOT NULL,
FOREIGN KEY(p_id) references person(p_id),PRIMARY KEY(b_id));

show tables;

desc person;
use dbms;
desc person;
use dbms;
show tables;
desc person;
desc test_spot;
insert into test_spot values('SP0001','Paldi','Ahmedabad','9393939393');
insert into test_spot values('SP0002','thaltej','Gandhinagar','943939493');
insert into test_spot values('SP0003','prahladnagar','Ahmedabad','9593959397');
insert into test_spot values('SP0004','naroda','Ahmedabad','9593939396');
select *from test_spot;
desc person;
insert into person values('P0001','shilp','patel','male','2001-10-26','Ahmedabad','SP0003',null,null);
select *from person;
select spot_name from test_spot,person where first_name='shilp' and person.spot_id=test_spot.spot_id;
use dbms;
desc doctor;
desc hospital;
insert into hospital values('H0001','Sanjivni','Ahmedabad','9925008589','SP0002');
insert into hospital values('H0002','Zydus','Gandhinagar','9925008584','SP0002');
insert into hospital values('H0003','Sal','Ahmedabad','9925058589','SP0001');
insert into hospital values('H0004','Apollo','Ahmedabad','9925408589','SP0003');
```

```
insert into hospital values('H0005','Bodyline','Ahmedabad','9925008789','SP0004');
select * from hospital;
show tables;
desc doc_dept;
drop table doc_dept;
create table doc_dept(doc_id varchar(25) NOT NULL ,department varchar(25) NOT NULL,foreign
key(doc_id)
references doctor(doc_id));
desc doc_dept;
select *from person;
update person set h_id='H0004' where p_id='P0001';
select *from person;
desc doctor;
insert into doctor values('D0001','Nitish kumar','1990-12-10','9375499974','H0005');
insert into doctor values('D0002','muskan singh','1990-11-09','9375499874','H0002');
insert into doctor values('D0003','Aashal shah','1990-01-10','9345499974','H0005');
insert into doctor values('D0004','Prakhya Chopra','1991-12-10','9375497974','H0001');
insert into doctor values('D0005','Sanjana mathur','1990-12-25','8375499974','H0003');
insert into doctor values('D0006','Shilp patel','1990-02-10','9385499974','H0004');
select * from  doctor;
desc doc_dept;
insert into doc_dept values('D0001','surgeon');
insert into doc_dept values('D0002','paediatrician');
insert into doc_dept values('D0003','paediatrician');
insert into doc_dept values('D0004','paediatrician');
insert into doc_dept values('D0005','paediatrician');
insert into doc_dept values('D0004','paediatrician');
select *from doc_dept;
delete from doc_dept where doc_id='D0004';
select *from doc_dept;
insert into doc_dept values('D0004','paediatrician');
insert into doc_dept values('D0006','paediatrician');
select*from doc_dept;
use dbms;
desc person;
show tables;
desc person_contacts;
desc person;
select *from person;
select *from person_contacts;
use dbms;
desc test_spot;
select *from test_spot;
select *from person;
desc test_spot;
desc person;
insert into person values('P0006','dj','nigam','trans','2001-10-26','kanpur',null,null,null);
use dbms;
insert into person values('P0006','dj','nigam','trans','2001-10-26','kanpur',null,null,null);
select *from person_contacts;
select *from person;
desc hospital;
insert into hospital values('null','null','null','null','null');
desc test_spot;
insert into test_spot values('null','null','null','null');
desc doctor;
```

```
select *from person;
select *from hospital;
desc doctor;
insert into doctor values('null','null',null,'null','null');
select *from doctor;
select *from person;

use dbms;
select *from medicine;
show tables;
desc test_bill;
select * from person;
select *from person_contacts;
desc test_bill;
alter table test_bill drop constraint test_bill_ibfk_1;
alter table test_bill drop column p_id;
alter table test_bill add(p_id varchar(25),foreign key(p_id) references person(p_id));
desc test_bill;
select *from test_bill;
insert into test_bill values('RE0001',1000,null,'card','P0004');
insert into test_bill values('RE0002',8000,null,'card','P0007');
insert into test_bill values('RE0003',7000,500,'cash','P0001');
insert into test_bill values('RE0004',10000,null,'sodex','P0002');
insert into test_bill values('RE0005',5000,500,'cash','P0008');
select *from test_bill;
show tables;
desc bill_hospital;
insert into bill_hospital values('BH0001','2020-02-15','2020-02-29','12pm',10000,null,'P0007');
insert into bill_hospital values('BH0002','2020-02-20','2020-03-15','12:30pm',90000,null,'P0004');
insert into bill_hospital values('BH0003','2020-02-17','2020-03-01','12pm',95000,null,'P0002');
insert into bill_hospital values('BH0004','2020-02-15','2020-02-29','12pm',80000,null,'P0008');
insert into bill_hospital values('BH0005','2020-03-15','2020-03-30','1pm',100000,null,'P0001');
select *from bill_hospital;
desc hospital;
select first_name,last_name,hospital_name from person,hospital where p_id='P0007' and
person.h_id=hospital.h_id;
use dbms;
show tables;
select *from test_spot;
select *from person;
select *from test_bill;
select bill_amount from test_bill,person where first_name='harsh' and test_bill.p_id=person.p_id;
select *from test_results;
use dbms;
show tables;
select *from hospital;
select *from person order by DOB;
select *from person order by first_name;
select *from hospital;
update person set spot_id='SP0003' where p_id='P00012';
update person set h_id='H0004' where p_id='P00012';
select *from hospital;
select *from doctor;
update person set doc_id='D0006' where p_id='P00012';
select *from person;
select *from hospital;
select *from hospital;
```

```
alter table hospital drop column spot_id;
alter table hospital drop constraint hospital_ibfk_1;
alter table hospital drop column spot_id;
alter table person drop column room_id;
alter table person drop constraint person_ibfk_4;
select *from person;
select *from doctor;
select *from hospital;
use dbms
```

## 7.2 Code for Front-end connectivity

```
from flask import Flask,render_template,request
from flask_mysqldb import MySQL
import json
with open('config.json','r') as c:
    params=json.load(c)["params"]
app=Flask(__name__,template_folder='template')
app.config['MYSQL_HOST']=params['mysql_host']
app.config['MYSQL_USER']=params['mysql_user']
app.config['MYSQL_PASSWORD']=params['mysql_password']
app.config['MYSQL_DB']=params['mysql_db']
mysql=MySQL(app)
@app.route('/',methods=['GET','POST'])
def home():
    if request.method=='POST':
        p_id=request.form['p_id']
        first_name=request.form['first_name']
        last_name = request.form['last_name']
        Gender = request.form['Gender']
        DOB = request.form['DOB']
        Address = request.form['Address']
        contacts=request.form['contacts']
        spot_id=request.form['spot_id']
        h_id=request.form['h_id']
        doc_id=request.form['doc_id']
        cur=mysql.connection.cursor()
        cur.execute("insert into
person(p_id,first_name,last_name,gender,DOB,Address,spot_id,h_id,doc_id)values(%s,%s,%s,%s,%s,%s,
%s,%s,%s)",(p_id,first_name,last_name,Gender,DOB,Address,spot_id,h_id,doc_id))
        cur.execute("insert into person_contacts(p_id,contacts) values(%s,%s)", (p_id, contacts))
        mysql.connection.commit()
        cur.close()
        return "success"
    return render_template('index.html')
@app.route('/places',methods=['GET','POST'])
def places():
    cur=mysql.connection.cursor()
    res=cur.execute("select *from test_spot")
    spot_details=cur.fetchall()
    return render_template('places.html',spot_details=spot_details)
@app.route('/findspot',methods=['GET','POST'])
def findspot():
```

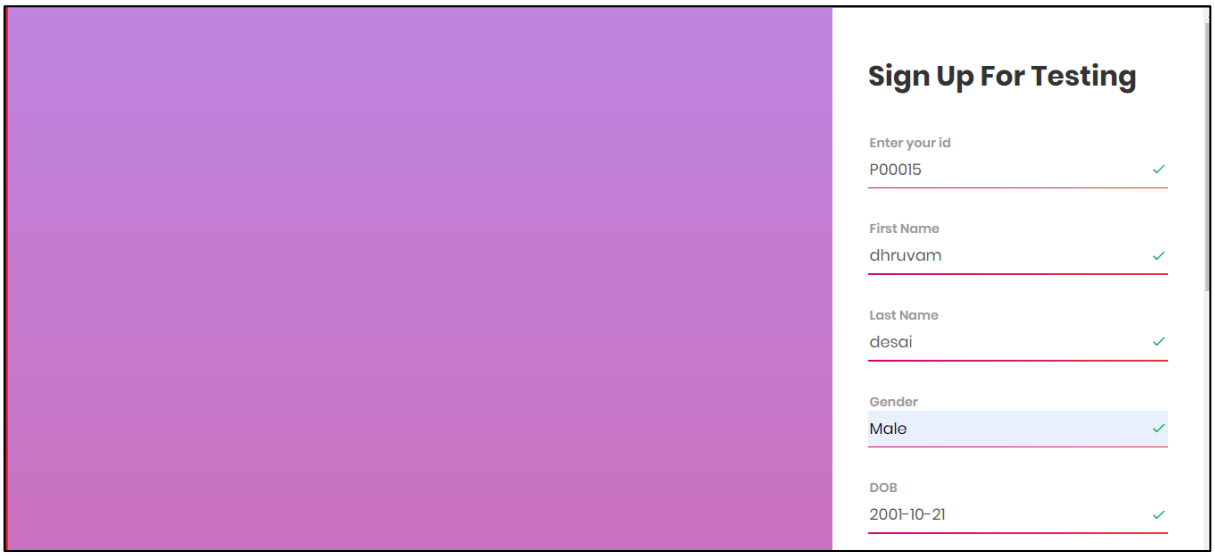
```
    if request.method=='POST':
        first_name = request.form['first_name']
        p_id=request.form['p_id']
        cur=mysql.connection.cursor()
        res=cur.execute("select receipt_no,bill_amount,payment_type from test_bill,person where
first_name='"+first_name+"' and test_bill.p_id=person.p_id")
        details=cur.fetchall()
        cur.execute("select spot_name from person,test_spot where person.spot_id=test_spot.spot_id and
first_name='"+first_name+"'")
        que1=cur.fetchall()
        cur.execute("select result_type from test_results,person where person.p_id=test_results.p_id and
first_name='"+first_name+"'")
        que2=cur.fetchall()
        return render_template('spot.html',details=details,que1=que1,que2=que2)
    return render_template('spot.html')
@app.route('/hospital_details',methods=['GET','POST'])
def hospital_details():
    if request.method=='POST':
        first_name=request.form['first_name']
        p_id=request.form['p_id']
        cur=mysql.connection.cursor()
        res=cur.execute("select hospital_name from hospital,person where p_id='"+p_id+"' and
hospital.h_id=person.h_id")
        details=cur.fetchall()
        cur.execute("select doc_name from doctor,person where p_id='" + p_id + "' and
doctor.doc_id=person.doc_id")
        doc_details=cur.fetchall()
        return render_template('hospital.html',details=details,doc_details=doc_details)
    return render_template('hospital.html')
app.run(debug=True)
```

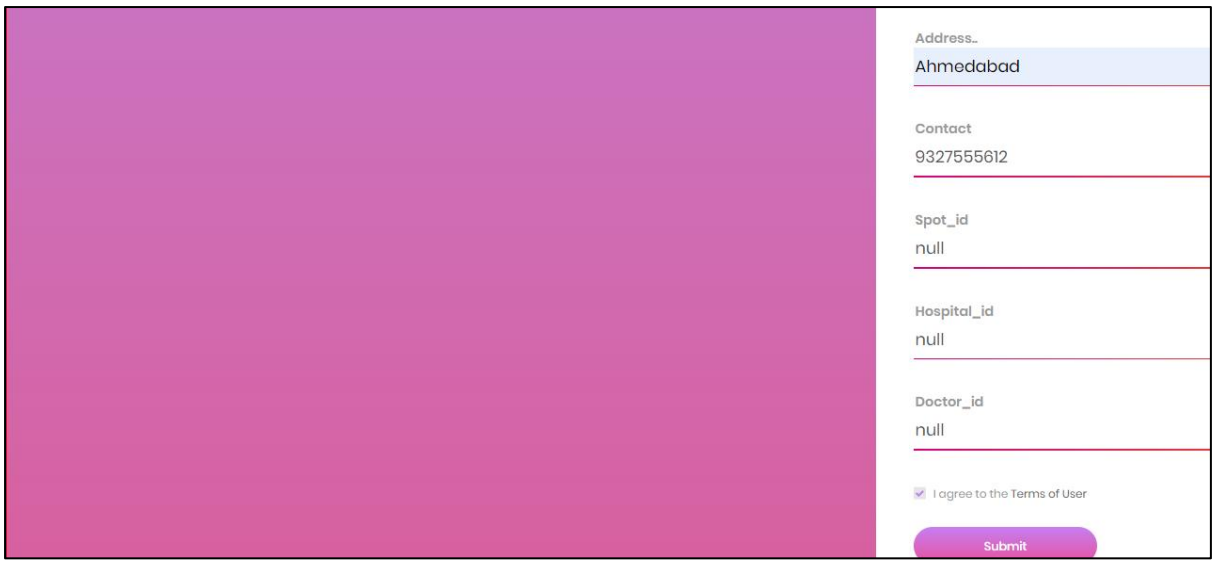
## 8. Manipulation at front end:

Once we run the file project.py we will get the URL, and when we click at the URL, we are redirected to the login page of our website.

```
project
C:\Users\USER\Downloads\python.exe C:/Users/USER/PycharmProjects/projectdbms(main)/project.py
 * Serving Flask app "project" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 561-142-740
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

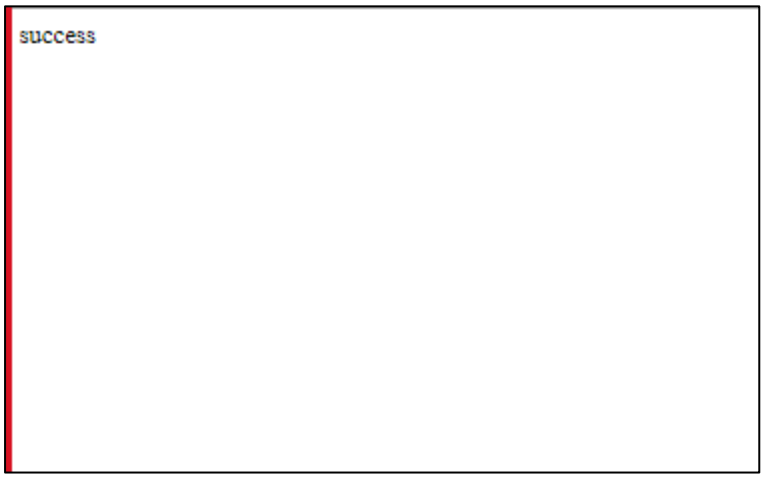
User posts at the frontend and inputs his data for covid testing.






As the user posts the data he clicks on submit and the data goes into the database.

Then if we get success page then our data is successfully registered into the database. Method=’Post’ is necessary for the user.



Therefore, we can see that data posted by user is visible in our person table.

```
mysql> select *from person;
+--------+------------+-----------+--------+------------+---------------+---------+-------+--------+
| p_id   | first_name | last_name | gender | DOB        | address       | spot_id | h_id  | doc_id |
+--------+------------+-----------+--------+------------+---------------+---------+-------+--------+
| P0001  | shilp      | patel     | male   | 2001-10-26 | Ahmedabad     | SP0003  | H0004 | NULL   |
| P00012 | vardhan    | khara     | male   | 2001-10-27 | Ahmedabad,Sbr | SP0003  | H0004 | D0006  |
| P00013 | simran     | singh     | Female | 2001-10-27 | delhi         | SP0001  | null  | null   |
| P00015 | dhruvam    | desai     | Male   | 2001-10-21 | Ahmedabad     | SP0004  | H0003 | D0005  |
| P0002  | Shilp      | Patel     | Male   | 2001-10-26 | Ahmedabad     | SP0001  | H0001 | D0001  |
| P0004  | harsh      | pandey    | Male   | 2001-10-26 | kanpur        | SP0002  | H0002 | D0002  |
| P0007  | chavi      | pathak    | female | 2001-10-27 | bhopal        | null    | null  | null   |
| P0008  | shilp      | patel     | male   | 2001-10-26 | abad          | SP0003  | null  | null   |
| P0009  | niraj      | patel     | Male   | 2001-10-27 | Ahmedabad     | null    | null  | null   |
+--------+------------+-----------+--------+------------+---------------+---------+-------+--------+
9 rows in set (0.00 sec)
```

Now we will update the spot_id for our user whose first name is dhruvam.

```
mysql> update person set spot_id='SP0004' where first_name='dhruvam';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select *from test_results;
+---------+-------------+-------------+--------------------+-------+
| test_id | result_type | comorbidity | quarantine_period  | p_id  |
+---------+-------------+-------------+--------------------+-------+
| TT0001  | positive    | none        | 14 days            | P0001 |
| TT0002  | positive    | diabetic    | 14days             | P0004 |
| TT0003  | negative    | none        | 0days              | P0002 |
| TT0004  | positive    | diabetic    | 14days             | P0008 |
| TT0005  | positive    | thyroid     | 14days             | P0007 |
+---------+-------------+-------------+--------------------+-------+
5 rows in set (0.00 sec)
```

Now we insert values for for test_id, result_type comorbidity, quarantine_period, p_id, for the new user. Then to view the test_results table we use select statement to check if our new values are updated or not.

select*from test_results

```
mysql> insert into test_results values('TT0006','positive','none','14 days','P00015');
Query OK, 1 row affected (0.01 sec)

mysql> select *from test_results;
+---------+-------------+-------------+--------------------+--------+
| test_id | result_type | comorbidity | quarantine_period  | p_id   |
+---------+-------------+-------------+--------------------+--------+
| TT0001  | positive    | none        | 14 days            | P0001  |
| TT0002  | positive    | diabetic    | 14days             | P0004  |
| TT0003  | negative    | none        | 0days              | P0002  |
| TT0004  | positive    | diabetic    | 14days             | P0008  |
| TT0005  | positive    | thyroid     | 14days             | P0007  |
| TT0006  | positive    | none        | 14 days            | P00015 |
+---------+-------------+-------------+--------------------+--------+
6 rows in set (0.00 sec)
```

Now we will update receipt_no, bill_amount, discount, payment_type, p_id to the table test_bill by using insert statement and now we will can view the test_bill table.

```
mysql> insert into test_bill values('RE0007',2000,500,'cash','P00015');
Query OK, 1 row affected (0.01 sec)

mysql> select *from test_bill;
+------------+-------------+----------+--------------+--------+
| receipt_no | bill_amount | discount | payment_type | p_id   |
+------------+-------------+----------+--------------+--------+
| RE0001     |        1000 |     NULL | card         | P0004  |
| RE0002     |        8000 |     NULL | card         | P0007  |
| RE0003     |        7000 |      500 | cash         | P0001  |
| RE0004     |       10000 |     NULL | sodex        | P0002  |
| RE0005     |        5000 |      500 | cash         | P0008  |
| RE0006     |        4000 |     NULL | card         | P00013 |
| RE0007     |        2000 |      500 | cash         | P00015 |
+------------+-------------+----------+--------------+--------+
7 rows in set (0.00 sec)
```

Now we will update the h_id where the first name of user is dhruvam. Therefore, using the select statement we can view the hospital table.

```
mysql> update person set h_id='H0003' where first_name='dhruvam';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select *from hospital;
+-------+---------------+-------------+------------+
| h_id  | hospital_name | address     | contact    |
+-------+---------------+-------------+------------+
| H0001 | Sanjivni      | Ahmedabad   | 9925008589 |
| H0002 | Zydus         | Gandhinagar | 9925008584 |
| H0003 | Sal           | Ahmedabad   | 9925058589 |
| H0004 | Apollo        | Ahmedabad   | 9925408589 |
| H0005 | Bodyline      | Ahmedabad   | 9925008789 |
| null  | null          | null        | null       |
+-------+---------------+-------------+------------+
6 rows in set (0.00 sec)
```

We view the doctor table using select statement select * from doctor. We now update the person table with first_name as dhruvam and set the doctor id for the user.

```
mysql> select *from doctor;
+--------+----------------+------------+------------+-------+
| doc_id | doc_name       | DOB        | contact    | h_id  |
+--------+----------------+------------+------------+-------+
| D0001  | Nitish kumar   | 1990-12-10 | 9375499974 | H0005 |
| D0002  | muskan singh   | 1990-11-09 | 9375499874 | H0002 |
| D0003  | Aashal shah    | 1990-01-10 | 9345499974 | H0005 |
| D0004  | Prakhya Chopra | 1991-12-10 | 9375497974 | H0001 |
| D0005  | Sanjana mathur | 1990-12-25 | 8375499974 | H0003 |
| D0006  | Shilp patel    | 1990-02-10 | 9385499974 | H0004 |
| null   | null           | NULL       | null       | null  |
+--------+----------------+------------+------------+-------+
7 rows in set (0.00 sec)

mysql> update person set doc_id='D0005' where first_name='dhruvam';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

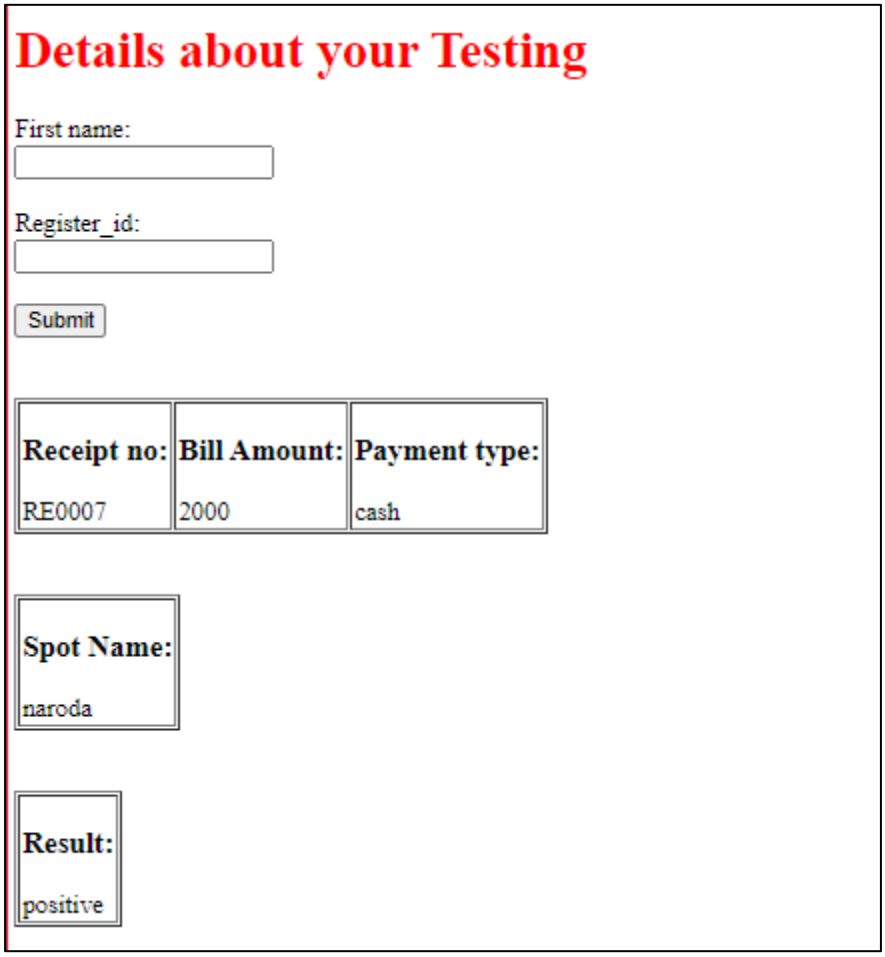
We view the person table.

```
mysql> select *from person;
+--------+------------+-----------+--------+------------+---------------+---------+-------+--------+
| p_id   | first_name | last_name | gender | DOB        | address       | spot_id | h_id  | doc_id |
+--------+------------+-----------+--------+------------+---------------+---------+-------+--------+
| P0001  | shilp      | patel     | male   | 2001-10-26 | Ahmedabad     | SP0003  | H0004 | NULL   |
| P00012 | vardhan    | khara     | male   | 2001-10-27 | Ahmedabad,Sbr | SP0003  | H0004 | D0006  |
| P00013 | simran     | singh     | Female | 2001-10-27 | delhi         | SP0001  | null  | null   |
| P00015 | dhruvam    | desai     | Male   | 2001-10-21 | Ahmedabad     | null    | null  | null   |
| P0002  | Shilp      | Patel     | Male   | 2001-10-26 | Ahmedabad     | SP0001  | H0001 | D0001  |
| P0004  | harsh      | pandey    | Male   | 2001-10-26 | kanpur        | SP0002  | H0002 | D0002  |
| P0007  | chavi      | pathak    | female | 2001-10-27 | bhopal        | null    | null  | null   |
| P0008  | shilp      | patel     | male   | 2001-10-26 | abad          | SP0003  | null  | null   |
| P0009  | niraj      | patel     | Male   | 2001-10-27 | Ahmedabad     | null    | null  | null   |
+--------+------------+-----------+--------+------------+---------------+---------+-------+--------+
9 rows in set (0.00 sec)
```

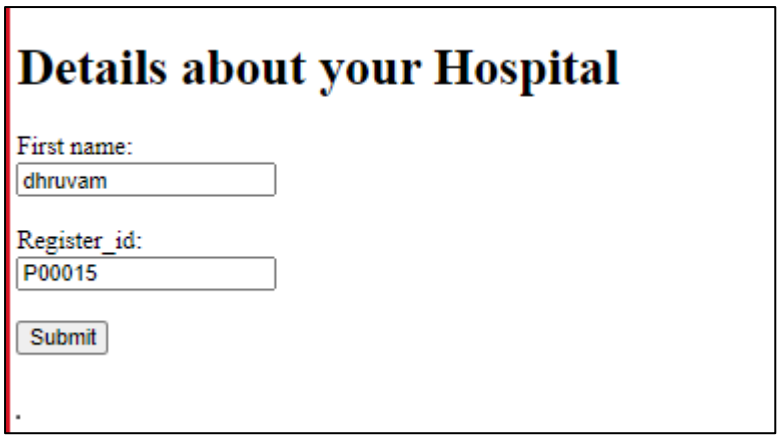
Now we enter details about testing like name and register_id.

We click on submit once we fill all required information in the form.

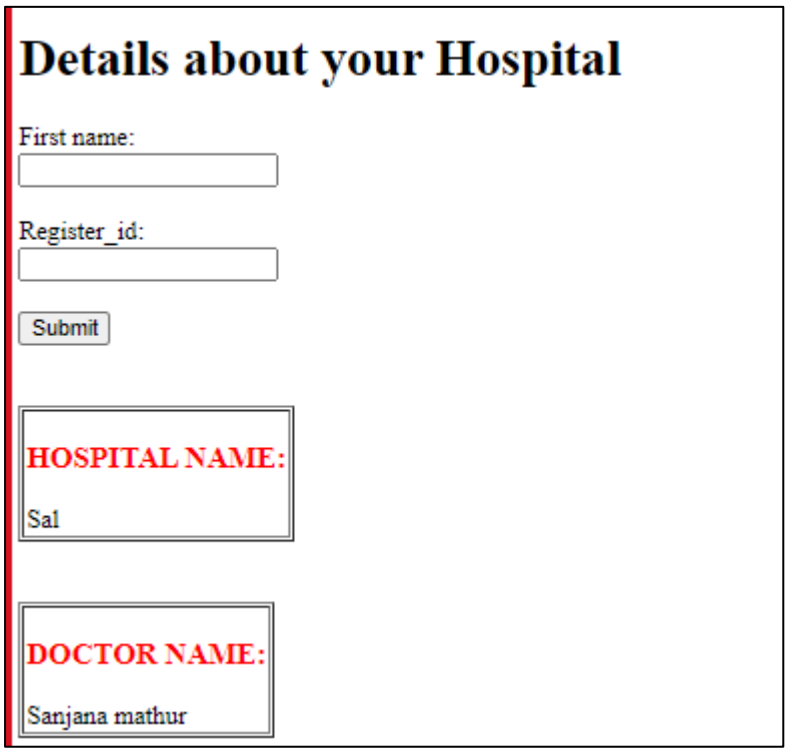

As we click on submit we get all details about the user that is stored in the database. We get Receipt no, Bill amount, Payment_type, Spot name, Result.




Now we enter name and Register_id to get to know about the details about the hospital of the user.Then we click on submit to get the data from the database.




Now we can get all the details regarding hospital i.e hospital Name and Doctor Name.



## 9. Conclusion

This project paves the way for the use of SQL databases to store and protect COVID19 patients' information, removing existing hurdles to their adoption. It does so by understanding the five main categories of SQL databases, offering a comparative study among these categories based upon a set of comparison criteria, namely Performance, Scalability, Flexibility, Complexity, Functionality, and Security issues. DBMSs were discussed from two points of view: the pertained to the information, the second the security analysis. The project Covid-19 Hospital Management System is for computerizing the working in a hospital. The software takes care of all the requirements of an average hospital and is capable to provide easy and effective storage of information related to patients that come up to the hospital. It generates test reports; provide prescription details including various tests, check-up and medicines prescribed to patient and doctor. It also provides the billing facility.

## 10. References


1) https://flask.palletsprojects.com/
2) www.javapoint.com
3) www.w3schools.com
4) www.udemy.com
5) https://getbootstrap.com/
6) www.youtube.com
7) www.geeksforgeeks.org
8) Silberschatz – Database System Concepts Fourth Edition