Covid Thunder System

Team Member: YuanLong Zhang, Bryan Jia, Yiran Lu, XiaoFan He

Project Background

Cleveland Clinic and University Hospital are two of the most famous and advanced hospitals in the United States. And these two hospitals also have the most convicent database system to provide hospital staff to have better services for the patients. However, due to the influence of COVID-19, the original database system is not enough to record covid information. Covid Thunder System is a database hospital system that records covid-19 information of each patient and hospital staff based on the original hospital information system from Cleveland Clinic. After using covid Thunder system, the hospital is able to check the covid information of each patient and hospital staff to monitor their health.

Project Feature

The primary job of a hospital is to treat the patients and take care of them. For a smaller scale hospital, or clinic where they only have a limited small number of patients, it's easier to track the information of the patients and their health care provider. But for a larger hospital, such as University Hospitals and Cleveland Clinic, it's important to have a large and well-defined database to track all the information needed. Who is the healthcare provider of this patient? Did the patient receive COVID testing? Which room is the patient in? All that information needs to be put into a database or the information will be messed up.

Our goal is to build a database that supports such functions which would help a large scale hospital or clinic to accurately get the information needed for healthcare providers, patients, rooms and so on.

Data Description

In this database, there are 7 entities and 8relationships

Entity:

```
Patient(MedicalNumber        int
        pName                String
        dName                String
        PRIMARY KEY(MedicalNumber)
        )
```

This entity records the basic information of each patient. Each patient has their own medical number to identify them.

```
Medical_record(MedicalNumber             int
               disease                   String
               allergen                  String
               PRIMARY KEY(MedicalNumber)
               FOREIGN KEY(MedicalNumber REFERENCING Patient)
              )
```

This entity records the medical record information of each patient. Each patient has their own medical number to identify them. And "MedicalNumber " is the forgeign key referencing entity "Patient".

```
COVID_Test(testNumber      int
           date            String
           result          boolean
           shotNum         int
     PRIMARY KEY(testNumber)
     )
```
This entity records the covid test information of each patient and hospital staff.
```
Department(department      String
           managerDID      int
     PRIMARY KEY(department)
     )
```
This entity record the department
```
Doctor(dID          int
     dname          String
     department     String
     PRIMARY KEY(dID)
     )
```

```
ward(wardNumber            int
     floor                 int
     PRIMARY KEY(wardNumber)
     )
```
This entity records the information of thich floor the ward is on. Each ward has its own wardNumber. wardNumber is the primary key.
```
operating_room(optRoomNumber               int
               department                  String
     PRIMARY KEY(optRoomNumber)
     )
```
This entity records the information of the operating room which is under the department. Each operating room has its own optRoomNumber. optRoomNumber is the primary key.

Relation:
Describes the relationship between Patient living in Ward
```
live_in(medicalNumber    int
     wardNumber          int
     PRIMARY KEY(medicalNumber)
     FOREIGN KEY(medicalNumber REFERRING Patient )
     FOREIGN KEY(wardNumber REFERRING Ward)
)
```

Describes which department the Ward belongs to
```
ward_belongs(wardNumber    int
             department    String
             PRIMARY KEY(wardNumber)
```

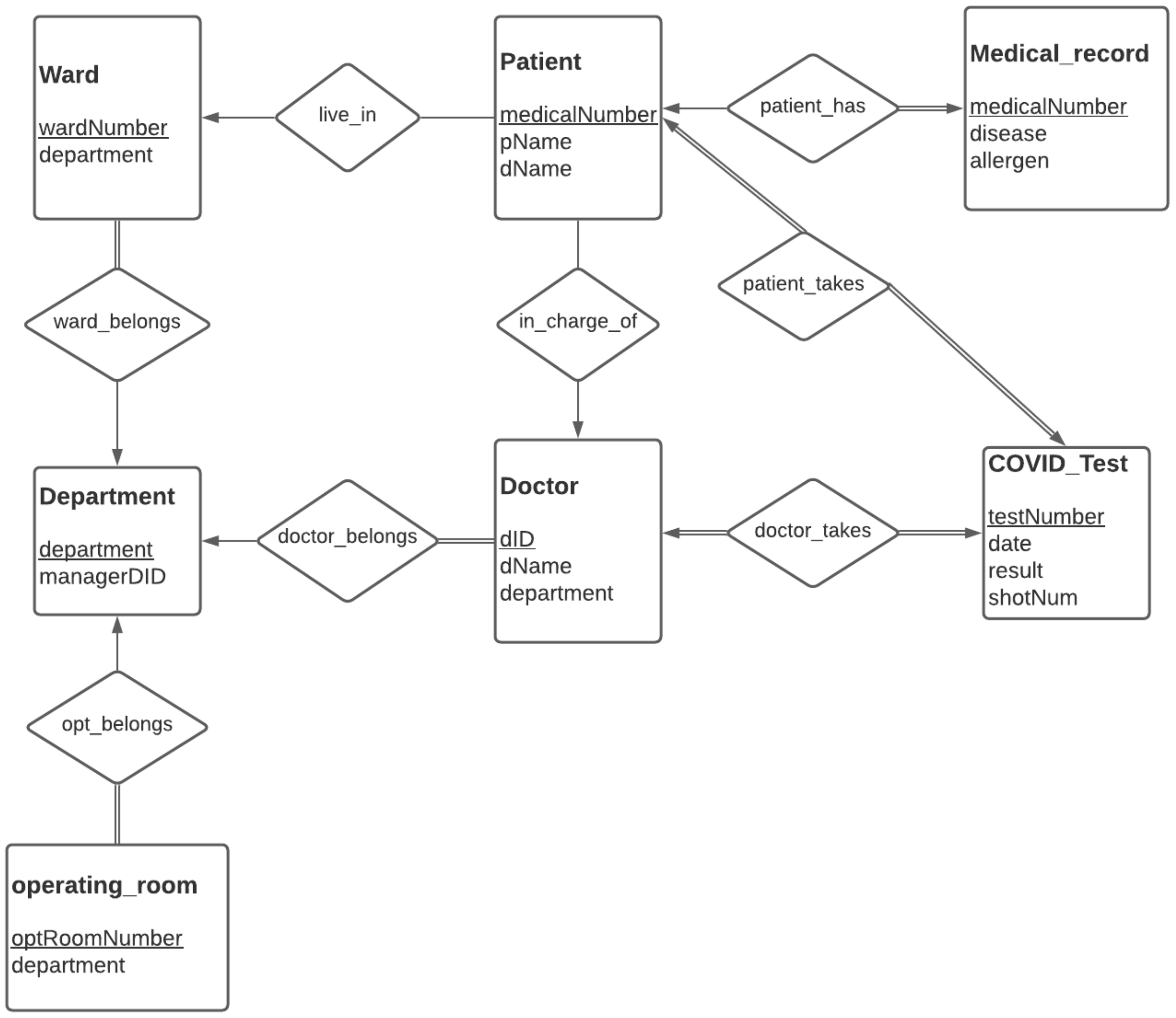
```
                    FOREIGN KEY(department REFERRING Department)
                    FOREIGN KEY(wardNumber REFERRING Ward)
)
```
Describes which department the Operation_room belongs to
```
opt_belongs( optRoomNumber    int
                    department       String
                    PRIMARY KEY(optRoomNumber)
                    FOREIGN KEY(department REFERRING Department)
                    FOREIGN KEY(optRoomNumber REFERRING Operating_Room)
)
```
Describes which department the Doctor belongs to
```
doctor_belongs(dID    int
                    department       String
                    PRIMARY KEY(dID)
                    FOREIGN KEY(department REFERRING Department)
                    FOREIGN KEY(dID REFERRING Doctor)
)
```

Describes a Patient having a Medical_record
```
patient_has(    medicalNumber         int
                    medicalNumber      int
                    PRIMARY KEY(medicalNumber)
                    FOREIGN KEY(medicalNumber REFERRING Patient)
                    FOREIGN KEY(medicalNumber REFERRING Medical_record)
)
```
Describes a Doctor being in charge of a Patient
```
in_charge_of(medicalNumber    int
                    dID                   int
                    PRIMARY KEY(medicalNumber)
                    FOREIGN KEY(medicalNumber REFERRING Patient)
                    FOREIGN KEY(dID REFERRING Doctor)
)
```
Describes a Patient taking covid_test
```
patient_takes(medicalNumber         int
                    testNumber              int
                    PRIMARY KEY(medicalNumber)
                    FOREIGN KEY(medicalNumber REFERRING Patient)
                    FOREIGN KEY(testNumber REFERRING COVID_Test)
)
```

Describes a Doctor taking covid_test
```
doctor_takes(  dID                  int
                    testNumber       int
                    PRIMARY KEY(dID)
```

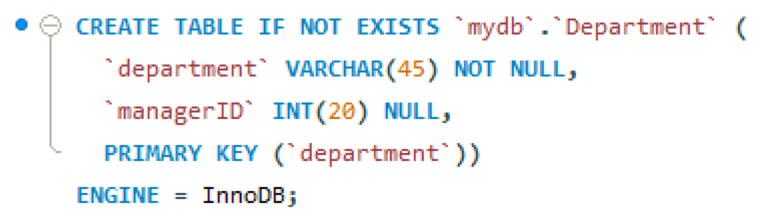
```
                FOREIGN KEY(dID REFERRING Doctor)
                FOREIGN KEY(testNumber REFERRING COVID_Test)
)
```

Database ER Diagram



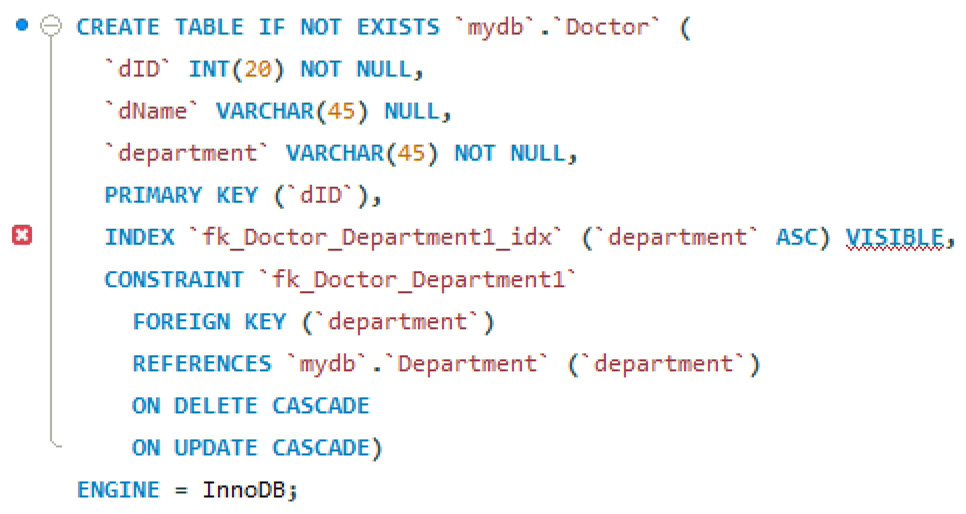
Database Schema:

Department



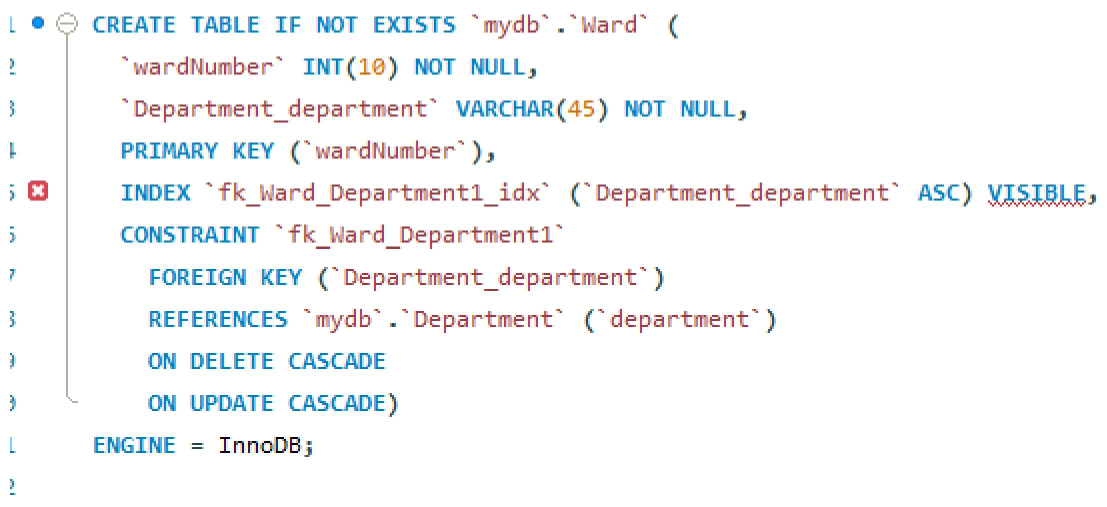
```sql
CREATE TABLE IF NOT EXISTS `mydb`.`Department` (
  `department` VARCHAR(45) NOT NULL,
  `managerID` INT(20) NULL,
  PRIMARY KEY (`department`))
ENGINE = InnoDB;
```

Doctor



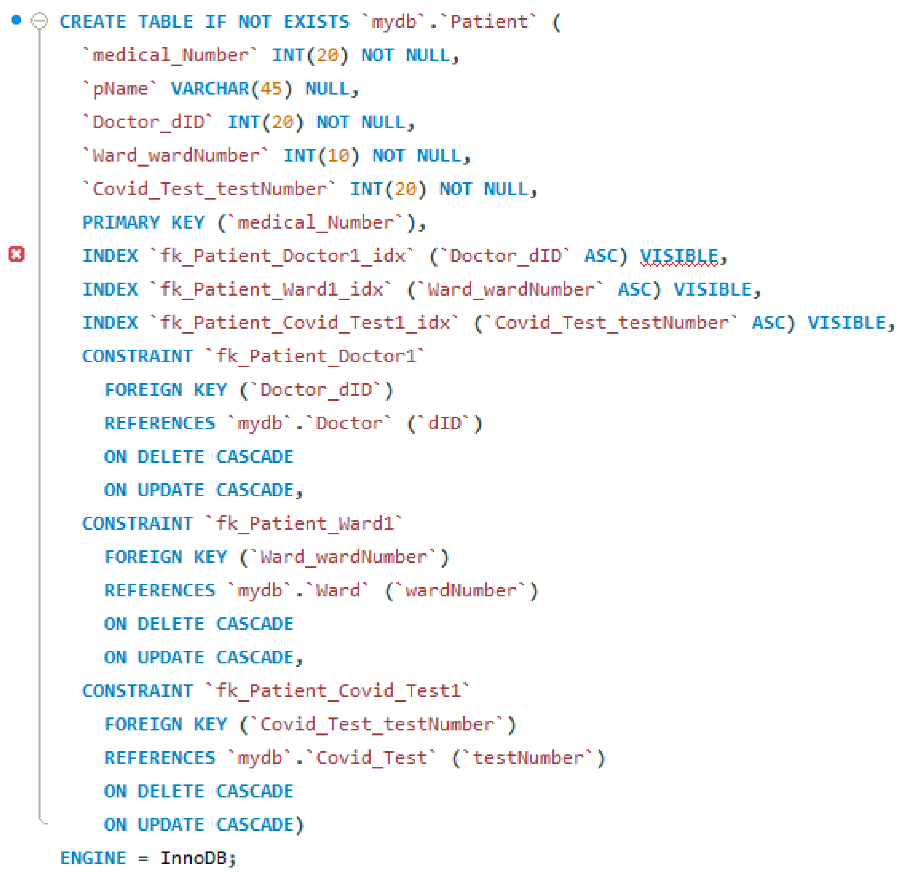
```sql
CREATE TABLE IF NOT EXISTS `mydb`.`Doctor` (
  `dID` INT(20) NOT NULL,
  `dName` VARCHAR(45) NULL,
  `department` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`dID`),
  INDEX `fk_Doctor_Department1_idx` (`department` ASC) VISIBLE,
  CONSTRAINT `fk_Doctor_Department1`
    FOREIGN KEY (`department`)
    REFERENCES `mydb`.`Department` (`department`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Ward



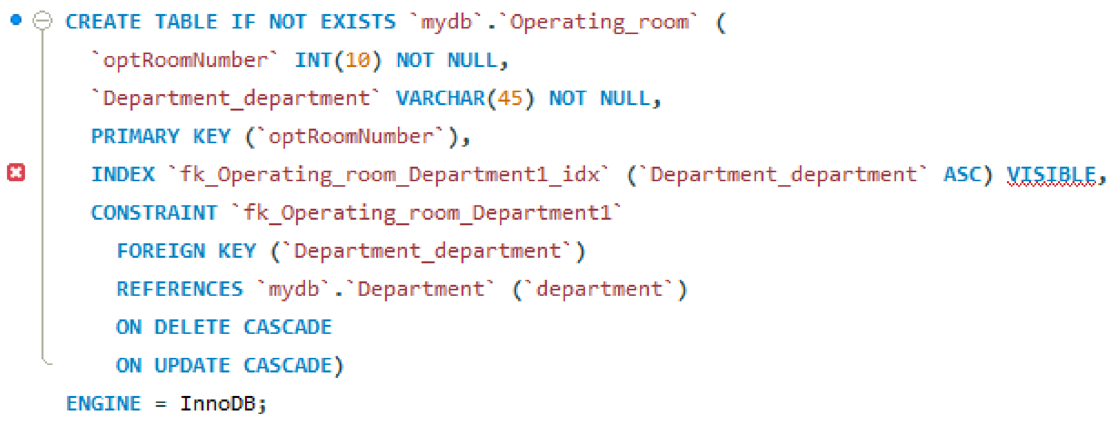
```sql
CREATE TABLE IF NOT EXISTS `mydb`.`Ward` (
  `wardNumber` INT(10) NOT NULL,
  `Department_department` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`wardNumber`),
  INDEX `fk_Ward_Department1_idx` (`Department_department` ASC) VISIBLE,
  CONSTRAINT `fk_Ward_Department1`
    FOREIGN KEY (`Department_department`)
    REFERENCES `mydb`.`Department` (`department`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Patient



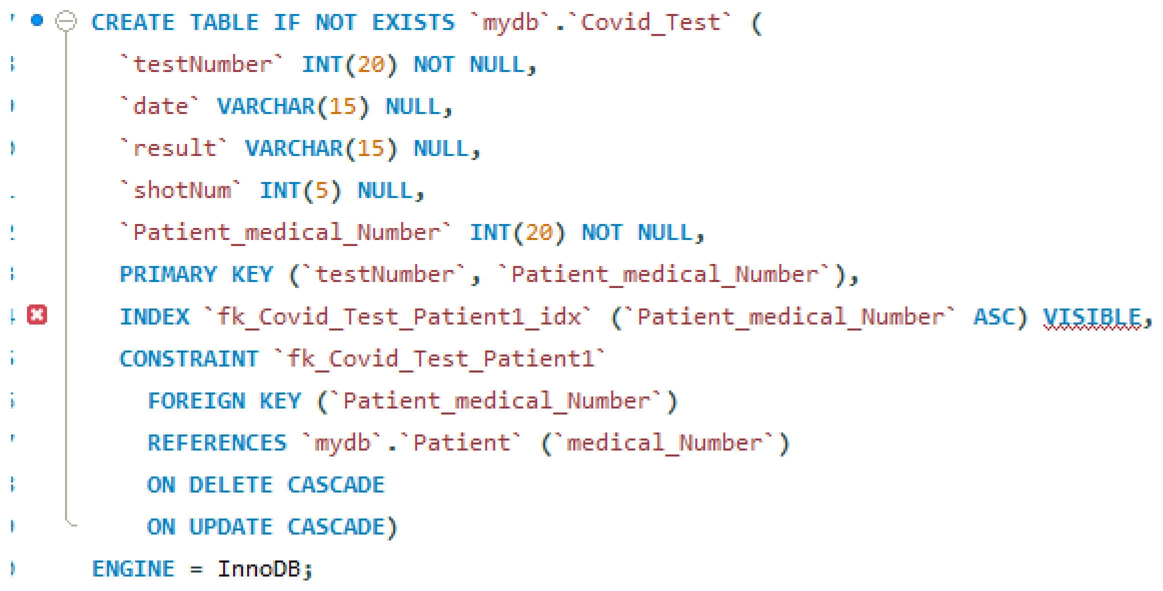
```
CREATE TABLE IF NOT EXISTS `mydb`.`Patient` (
  `medical_Number` INT(20) NOT NULL,
  `pName` VARCHAR(45) NULL,
  `Doctor_dID` INT(20) NOT NULL,
  `Ward_wardNumber` INT(10) NOT NULL,
  `Covid_Test_testNumber` INT(20) NOT NULL,
  PRIMARY KEY (`medical_Number`),
  INDEX `fk_Patient_Doctor1_idx` (`Doctor_dID` ASC) VISIBLE,
  INDEX `fk_Patient_Ward1_idx` (`Ward_wardNumber` ASC) VISIBLE,
  INDEX `fk_Patient_Covid_Test1_idx` (`Covid_Test_testNumber` ASC) VISIBLE,
  CONSTRAINT `fk_Patient_Doctor1`
    FOREIGN KEY (`Doctor_dID`)
    REFERENCES `mydb`.`Doctor` (`dID`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Patient_Ward1`
    FOREIGN KEY (`Ward_wardNumber`)
    REFERENCES `mydb`.`Ward` (`wardNumber`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `fk_Patient_Covid_Test1`
    FOREIGN KEY (`Covid_Test_testNumber`)
    REFERENCES `mydb`.`Covid_Test` (`testNumber`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

OperatingRoom



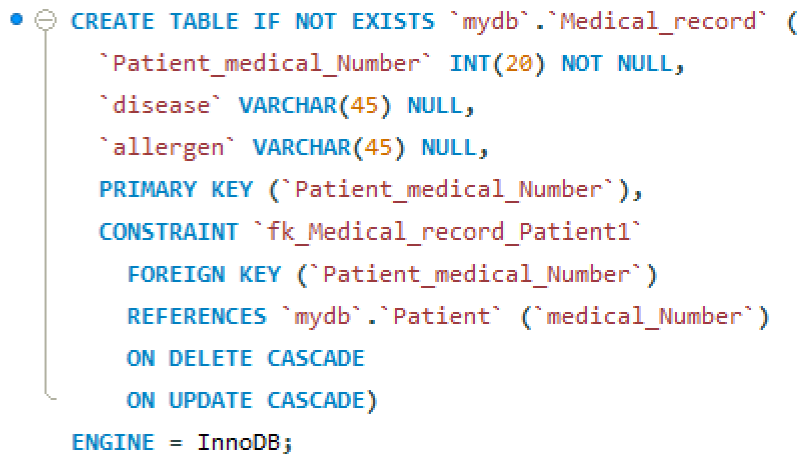
```
CREATE TABLE IF NOT EXISTS `mydb`.`Operating_room` (
  `optRoomNumber` INT(10) NOT NULL,
  `Department_department` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`optRoomNumber`),
  INDEX `fk_Operating_room_Department1_idx` (`Department_department` ASC) VISIBLE,
  CONSTRAINT `fk_Operating_room_Department1`
    FOREIGN KEY (`Department_department`)
    REFERENCES `mydb`.`Department` (`department`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Covid Test



```
CREATE TABLE IF NOT EXISTS `mydb`.`Covid_Test` (
  `testNumber` INT(20) NOT NULL,
  `date` VARCHAR(15) NULL,
  `result` VARCHAR(15) NULL,
  `shotNum` INT(5) NULL,
  `Patient_medical_Number` INT(20) NOT NULL,
  PRIMARY KEY (`testNumber`, `Patient_medical_Number`),
  INDEX `fk_Covid_Test_Patient1_idx` (`Patient_medical_Number` ASC) VISIBLE,
  CONSTRAINT `fk_Covid_Test_Patient1`
    FOREIGN KEY (`Patient_medical_Number`)
    REFERENCES `mydb`.`Patient` (`medical_Number`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

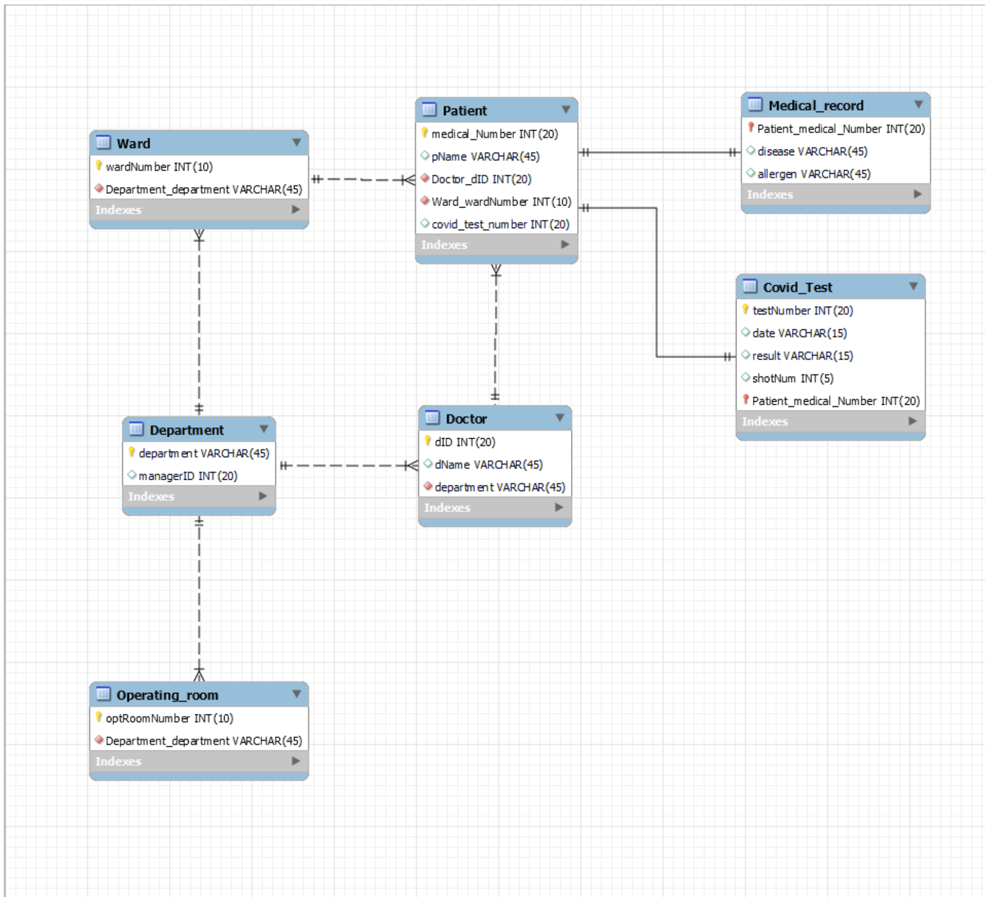
Medical Record



```
CREATE TABLE IF NOT EXISTS `mydb`.`Medical_record` (
  `Patient_medical_Number` INT(20) NOT NULL,
  `disease` VARCHAR(45) NULL,
  `allergen` VARCHAR(45) NULL,
  PRIMARY KEY (`Patient_medical_Number`),
  CONSTRAINT `fk_Medical_record_Patient1`
    FOREIGN KEY (`Patient_medical_Number`)
    REFERENCES `mydb`.`Patient` (`medical_Number`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

Overall Structure of Tables:

We use MySQL for our database.This is the table we created through MySQL



Sample Queries

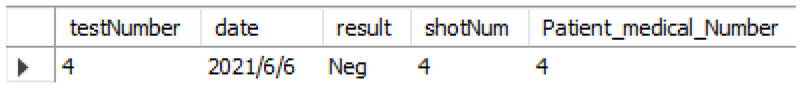
Query1:

Select the covid test report of the patient who has doctor id 101112:

SQL:

```
select * from covid_test
        where Patient_medical_number = (
        select medical_number from Patient where Doctor_dID = 101112)
```

Result:



	testNumber	date	result	shotNum	Patient_medical_Number
▶	4	2021/6/6	Neg	4	4

Query2:

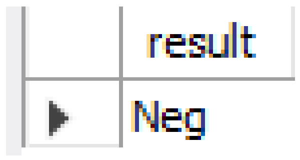
Select the result of the covid test of the patient who has doctor id 101112:

SQL:

```
select result from covid_test
```

```
where Patient_medical_number = (
select medical_number from Patient where Doctor_dID = 101112)
```

Result:



	result
▶	Neg

Implementation

Tech Stack for this project:
Front End: JFrame
Programing Language: Java
DBMS: MySQL

Source Code:

Hospital_Database.sql	SQL corresponding to covidthurder.java
covidthurder.java	implementation of the front end.

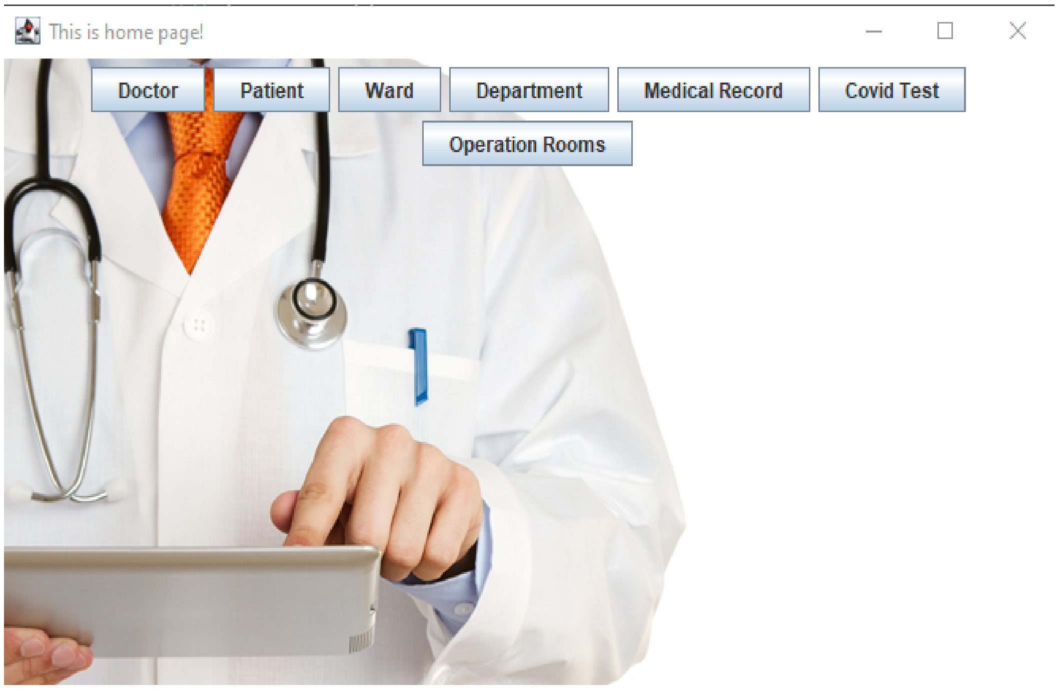
Requirements:
Java 17 installed

User Guide

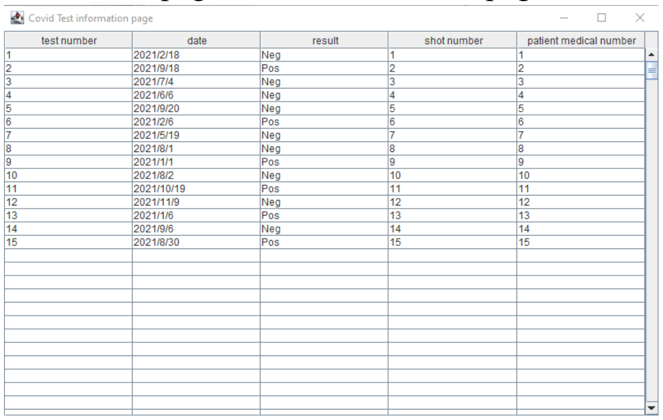
1. Open the java17 to run the program
2. Wait for the program to collect data and show up
3. After the database connects to the front end, the homepage will display and you can view the information of the hospital system.

User Interface

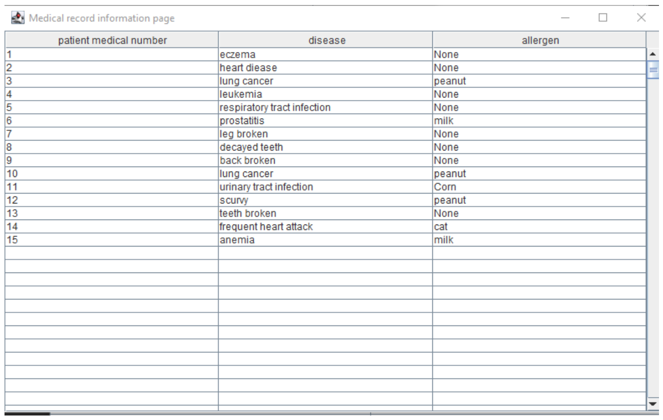
The front end of the Covid Thunder System allows the users to view the information of the patients and hospital staff. And you can track the covid test and vaccine information through the system.
Homepage:



Covid-test page and Medical record page：

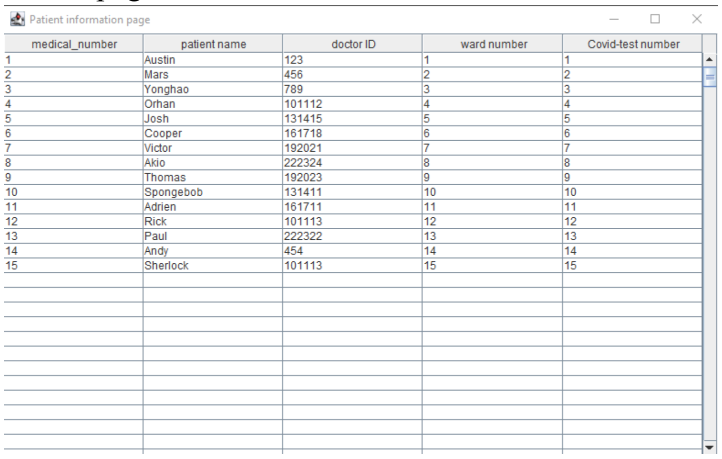


test number	date	result	shot number	patient medical number
1	2021/2/18	Neg	1	1
2	2021/9/18	Pos	2	2
3	2021/7/4	Neg	3	3
4	2021/6/6	Neg	4	4
5	2021/9/20	Neg	5	5
6	2021/2/6	Pos	6	6
7	2021/5/19	Neg	7	7
8	2021/8/1	Neg	8	8
9	2021/1/1	Pos	9	9
10	2021/8/2	Neg	10	10
11	2021/10/19	Pos	11	11
12	2021/11/9	Neg	12	12
13	2021/1/6	Pos	13	13
14	2021/9/6	Neg	14	14
15	2021/8/30	Pos	15	15



patient medical number	disease	allergen
1	eczema	None
2	heart diease	None
3	lung cancer	peanut
4	leukemia	None
5	respiratory tract infection	None
6	prostatitis	milk
7	leg broken	None
8	decayed teeth	None
9	back broken	None
10	lung cancer	peanut
11	urinary tract infection	Corn
12	scurvy	peanut
13	teeth broken	None
14	frequent heart attack	cat
15	anemia	milk

Patient page:



medical_number	patient name	doctor ID	ward number	Covid-test number
1	Austin	123	1	1
2	Mars	456	2	2
3	Yonghao	789	3	3
4	Orhan	101112	4	4
5	Josh	131415	5	5
6	Cooper	161718	6	6
7	Victor	192021	7	7
8	Akio	222324	8	8
9	Thomas	192023	9	9
10	Spongebob	131411	10	10
11	Adrien	161711	11	11
12	Rick	101113	12	12
13	Paul	222322	13	13
14	Andy	454	14	14
15	Sherlock	101113	15	15

What we’ve learnt:

In this project we are learning to use a database application with both front end and back end. We use our in class knowledge to create ER diagrams and focus on what we are going to write about. Then we will use dependency theories to look back at our ER diagrams and start to work on the schemas. After that we start to use MySQL to work on our database and use Eclipse IDE to put our database and java frontend together to finalize our project. By finishing this project we learnt how to use ER diagrams and how to program in MySQL to build our database. Besides that, we also learnt how to link our database to Eclipse IDE to build our frontend application. There are more details that we could edit. For example, we could refine our way of programming our frontend because it’s dangerous to simply link the database to Eclipse IDE because that will make the file not “read only” which will lead to potential hazard to edit the database from the outside. We could also have more functions to our database, for example, add buttons to the page so that we could add patients, doctors and so on to the database from the application.

Contribution

Xiaofan He: Responsible for the front end and part of the SQL
YuanLong Zhang: Responsible for the SQL and ER-Diagram
Yiran Lu: Responsible for the front end and sample queries
Bryan Jia: Responsible for the Schema and ER-Diagram