Project Final Report
COVID-19

Group Member Contribution
We are a group of two. Rahul contributed 60% and Unnat contributed 40% to the project (consisting of the last four deliverables). This is considering the consequences of the ongoing epidemic which caused issues with communication, primarily due to the factor of time zone difference of 10 hours.

README
Installation Instructions for Running this Project (MacOS)

I.
For backend verification from the Database:
- MySQL workbench latest version 8.0.19
- MySQL community server latest version 8.0.19
- Download and establish a connection to server
  - Keep track of username and password
- Open on MySQL workbench: covid19_tracker_dump
  - Run the dump file
  - Refresh your schema
  - Open a new query tab and type: USE `covid19_tracker`
    - Queries to view tables
      - SELECT * FROM person;
      - SELECT * FROM hospital;
      - SELECT * FROM infected;
      - SELECT * FROM recovered;
      - SELECT * FROM dead;
      - SELECT * FROM region;
      - SELECT * FROM hospital_visit;
      - SELECT * FROM test;

II.
For running the client interface application:
- Open Mac Terminal application
- Download Python latest version (Python3). Two methods:
  - (1) Download

- Download latest version from: https://www.python.org/downloads/
- Run the python installer to install Python on your Mac
- Can find python in the Applications directory of your Mac

- (2) Homebrew (longer method)
  - Open Mac terminal
  - Enter in terminal: xcode-select --install
  - Enter in terminal: /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
  - Enter in terminal: brew install python3
- Type the following in the Mac Terminal verify version: python --version (should be 3.X.X)

- Download the following by entering in your Mac terminal (may need to enter your password to allow)

  These enable the application to run and includes libraries used in the application.
  - sudo easy_install pip
  - PyMySQL: pip install PyMySQL
  - Tabulate: pip install tabulate
- In Mac terminal, navigate to the folder containing the python files
- To begin application, type: python run.py
- Login with MySQL username and password

Data Domain Description

This project focuses on the new strain of coronavirus, formally referred to as COVID-19, which broke out in China's Hubei province (Wuhan) and has since then infected over 100,000 people worldwide. The project has artificial data created to mimic real-world data, on a smaller scale. The choice to use artificial data is in part due to privacy concerns, as data including names is not available. The time period represented by this project is near the end of January 2020 to present, since this project explores the *global* outbreak of COVID-19 (the virus was contained in China prior to mid-January). Data in this project includes regions of outbreak, people who have contracted the virus, hospitals in which people are being treated, as well as the status of people infected as indicated by tests: infected, recovered, or dead. The database also stores the date of infection and recovery, as indicated by tests, and that of death.

This project allows users to add regions and associated hospitals. The user will be able to add people who are infected along with their basic demographic information and region of infection and residence. If the region does not yet exist in the database upon inserting a person, the user will be notified and can proceed to add a region. The person will be stored as infected by the database by default, with an associated test. However, the project will allow the user to modify the status to recovered or dead; this will automatically remove them from the infected table and into the appropriate recovered/dead table (with dates). Records are stored in the form of hospital visits and (diagnostic) tests. The risk level of the region ranges from 0 to 3, where 0 is no people infected, while 3 is a high-level of infection region. This number is calculated by comparing the percentage of people infected in the region to the average percentage of people infected across all regions. Apart from viewing, inserting, updating and deleting data, the program is also capable of finding and displaying the top 5 most infected regions, filtering regions by risk level, and viewing actively infected people who were infected before or after a user-specified date.

Entities, Relationships, and Constraints

COVID-19 (coronavirus) has broken out in China and has started infecting people in regions across the world. A region has an id, province, country, latitude, longitude, lockdown status, total population, risk level (derived attribute with default 0 and updated up to 3 based on relative number of infections), percentage of people infected (automatically determined), and may have many people and hospitals. A person has an id, first and last name, date of birth, sex, home region, and current region and may be one of infected, recovered, or dead. The database tracks the infection, recovery, and death date as well. A person may visit a hospital in order to get treated, potentially multiple times (multiple infections/recoveries are possible, death only once); their admit and discharge date are tracked. A person may go to many hospitals as they can be infected many times, but limited to a hospital in the region they are currently infected in (to limit exposure/spread). A recovery can be made without going to a hospital. The database tracks each visit to the hospital, with the associated admit and discharge date as well as result (recovered, died, or infected if discharge date is null). A hospital has an id, name, and is situated in one region and can have multiple people. Each person is diagnosed once by an individual test. However, a person may have many tests over time (to track multiple infections/recoveries). A test has an id, diagnosis, and date.

SQL Storage

We are using SQL storage because it is a better choice for performing fast queries, and we do not require the higher scalability of NoSQL storage due to the relatively small size of our project. Additionally, we do not expect the structure of our model to change.

Technical Specifications

Backend

- MySQL Workbench (v. 8.0.19)
- MySQL Community Server (8.0.19)

Frontend

- MacOS Terminal (user interface)
- Python 3.7.3 (client code language)
    - Pip (package management)
    - Tabulate (library)
    - PyMySQL (connect to database

Languages, Libraries and Framework

ORIGINAL

Our backend is in SQL, we are using MySQL for our database and Python for the frontend. We are using PyMySql for the connection. We are also using a basic command line interface with Tabulate to make it look better.

We plan to use Express and NodeJS for the backend (Javascript), MySQL for our database, and React.js for the frontend. These libraries/frameworks work on Windows, MacOS, and Linux (no machine restrictions). NodeJS will be used to run Javascript code on the server. Express will handle routing and process requests from the client. React.js will load the data from the backend, as well as provide the *interface* to modify data in the backend.

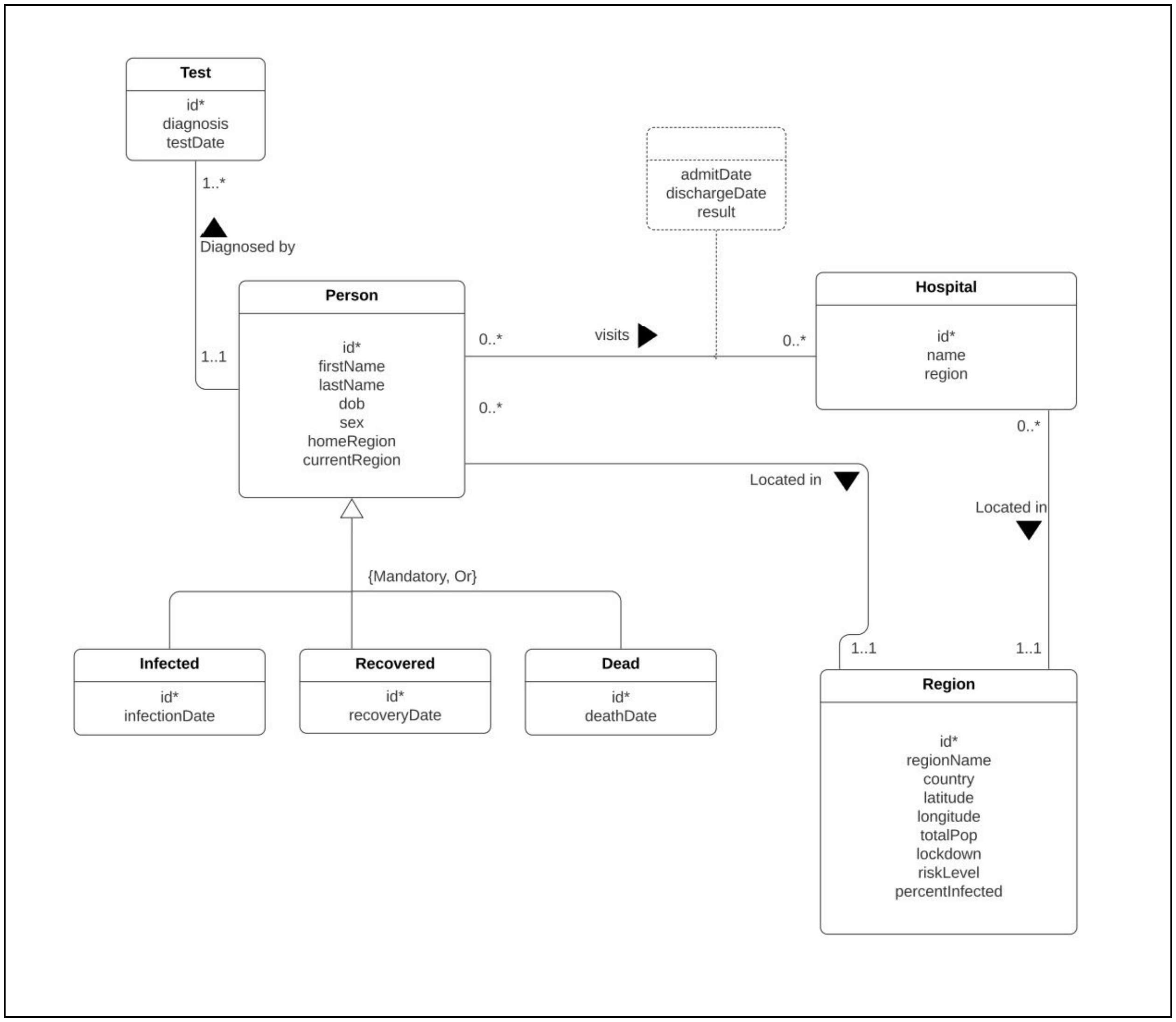
CURRENT

We have since shifted to a command line interface run using Python script. The reason for this is that our application's features are sufficiently met through a command line interface. Additionally, we did not have much expertise on interfacing a backend with a database, and transmitting http responses to convey error messages. We are still using MySQL for our database.

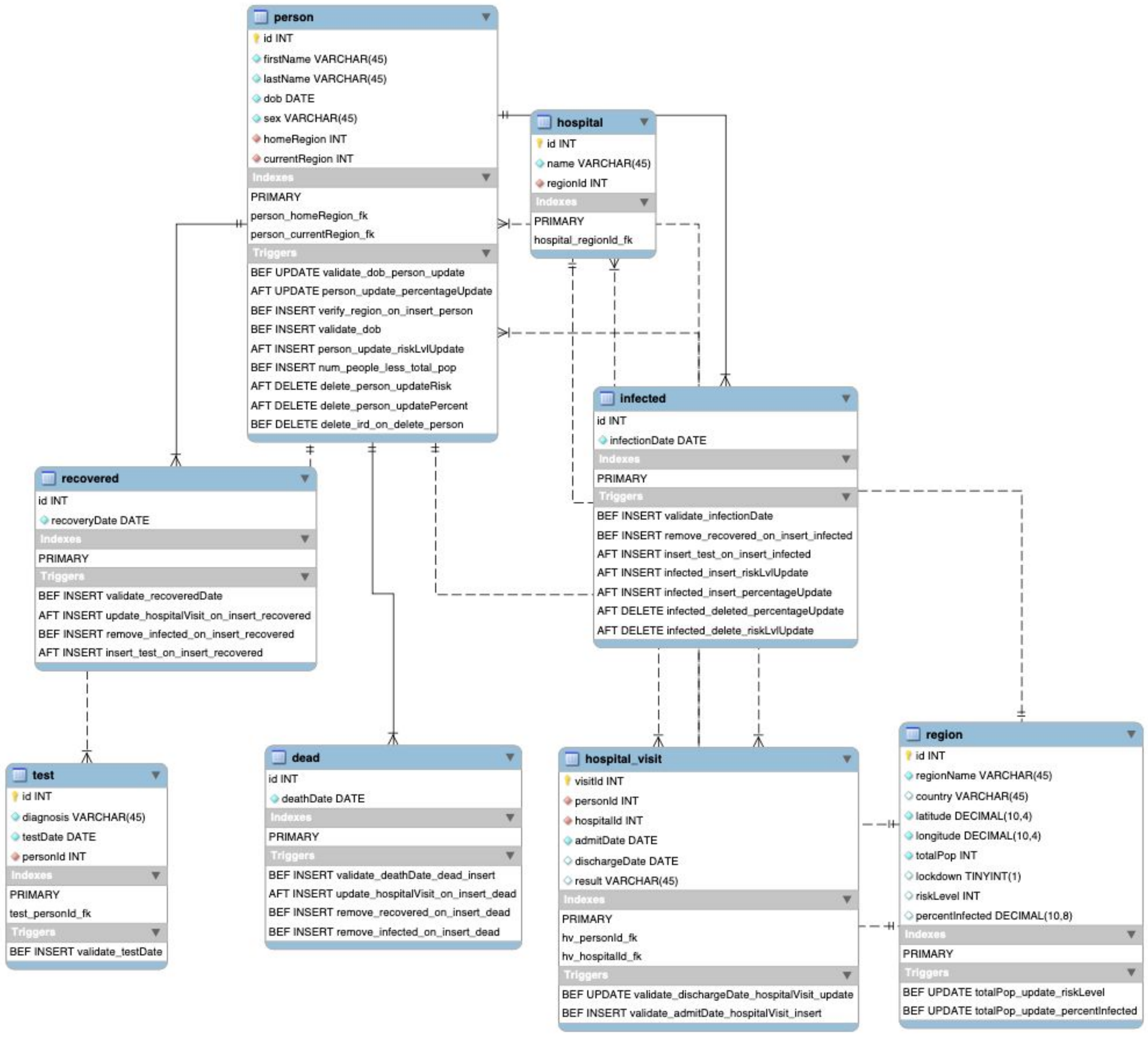
Reason for Interest

We are interested in this model because we are currently experiencing the global outbreak of coronavirus, an epidemic. The virus has spread quickly and testing of people is frequently done to monitor the virus's mortality rate. This model provides functionality to mark regions as low to high risk, as well as indicate whether a region is at risk based on the warning level of nearby regions. This informative model represents a simplified version/facet of the robust tools that health officials of the CDC use to monitor the progress of the virus. This enables things like warnings to work from home and travel restrictions to be enforced, in order to slow or prevent the spread of the virus, subsequently minimizing the death toll.

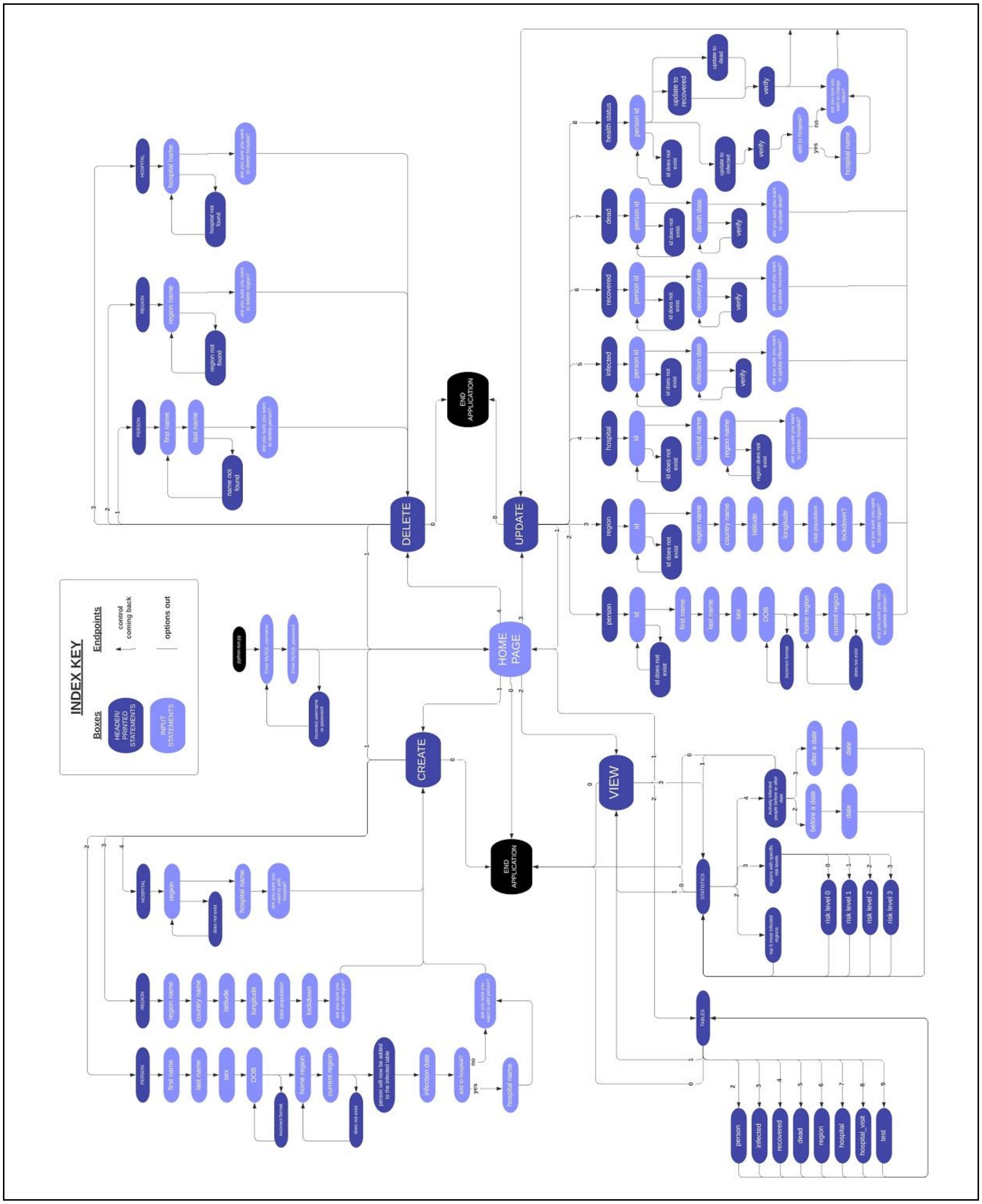
UML Diagram - Conceptual Design



EER Model- Logical Design


person
id INT
firstName VARCHAR(45)
lastName VARCHAR(45)
dob DATE
sex VARCHAR(45)
homeRegion INT
currentRegion INT
Indexes
PRIMARY
person_homeRegion_fk
person_currentRegion_fk
Triggers
BEF UPDATE validate_dob_person_update
AFT UPDATE person_update_percentageUpdate
BEF INSERT verify_region_on_insert_person
BEF INSERT validate_dob
AFT INSERT person_update_riskLvlUpdate
BEF INSERT num_people_less_total_pop
AFT DELETE delete_person_updateRisk
AFT DELETE delete_person_updatePercent
BEF DELETE delete_ird_on_delete_person
hospital
id INT
name VARCHAR(45)
regionId INT
Indexes
PRIMARY
hospital_regionId_fk
infected
id INT
infectionDate DATE
Indexes
PRIMARY
Triggers
BEF INSERT validate_infectionDate
BEF INSERT remove_recovered_on_insert_infected
AFT INSERT insert_test_on_insert_infected
AFT INSERT infected_insert_riskLvlUpdate
AFT INSERT infected_insert_percentageUpdate
AFT DELETE infected_deleted_percentageUpdate
AFT DELETE infected_delete_riskLvlUpdate
recovered
id INT
recoveryDate DATE
Indexes
PRIMARY
Triggers
BEF INSERT validate_recoveredDate
AFT INSERT update_hospitalVisit_on_insert_recovered
BEF INSERT remove_infected_on_insert_recovered
AFT INSERT insert_test_on_insert_recovered
test
id INT
diagnosis VARCHAR(45)
testDate DATE
personId INT
Indexes
PRIMARY
test_personId_fk
Triggers
BEF INSERT validate_testDate
dead
id INT
deathDate DATE
Indexes
PRIMARY
Triggers
BEF INSERT validate_deathDate_dead_insert
AFT INSERT update_hospitalVisit_on_insert_dead
BEF INSERT remove_recovered_on_insert_dead
BEF INSERT remove_infected_on_insert_dead
hospital_visit
visitId INT
personId INT
hospitalId INT
admitDate DATE
dischargeDate DATE
result VARCHAR(45)
Indexes
PRIMARY
hv_personId_fk
hv_hospitalId_fk
Triggers
BEF UPDATE validate_dischargeDate_hospitalVisit_update
BEF INSERT validate_admitDate_hospitalVisit_insert
region
id INT
regionName VARCHAR(45)
country VARCHAR(45)
latitude DECIMAL(10,4)
longitude DECIMAL(10,4)
totalPop INT
lockdown TINYINT(1)
riskLevel INT
percentInfected DECIMAL(10,8)
Indexes
PRIMARY
Triggers
BEF UPDATE totalPop_update_riskLevel
BEF UPDATE totalPop_update_percentInfected

User Flow Diagram


INDEX KEY
Boxes
HEADER/ PRINTED STATEMENTS
INPUT STATEMENTS
Endpoints
control coming back
options out
python ent.py
Enter MySQL username
Enter MySQL password
Incorrect username or password
HOME PAGE
CREATE
VIEW
UPDATE
DELETE
END APPLICATION
PERSON
first name
last name
sex
DOB
incorrect format
home region
current region
does not exist
person will now be added to the infected table
infection date
add to hospital?
yes
no
hospital name
are you sure you want to add person?
REGION
region name
country name
latitude
longitude
total population
lockdown
are you sure you want to add region?
HOSPITAL
region
hospital name
are you sure you want to add hospital?
STATISTICS
top 5 most infected regions
regions with specific risk levels
risk level 0
risk level 1
risk level 2
risk level 3
Actively infected people before or after date
before a date
after a date
date
TABLES
person
infected
recovered
dead
region
hospital
hospital_visit
test
id
id does not exist
region does not exist
lockdown?
are you sure you want to update person?
are you sure you want to update region?
are you sure you want to update hospital?
infection date
verify
are you sure you want to update infected?
recovery date
are you sure you want to update recovered?
death date
are you sure you want to update dead?
health status
person id
update to infected
update to recovered
update to dead
add to hospital?
are you sure you want to change status?
name not found
are you sure you want to delete person?
region not found
are you sure you want to delete region?
hospital not found
are you sure you want to delete hospital?

Lessons Learned

1. Technical Expertise

   Both of us had a very basic knowledge of Python prior to this project. This project not only expanded our Python knowledge, but we also learned how to use it to interface with a database. Before we took this class, we had never written anything in SQL. After completing this project, we now feel comfortable with SQL code, concepts such as integrity constraints, table relationships, etc. We also learned how to use the connectors to connect sql with python/java. This project also helped us acquire knowledge about different libraries such as Tabulate and how it is used in a command line interface. For the first time, we built a complete database with all the functionality using Stored Procedures, functions and triggers.
2. Group work insights, time management insights, data domain insights etc.

   Working from two different countries, both on different ends of the globe, certainly did pose some problems. We worked individually and communicated with each other in the common time during the day. This helped us put together and evaluate the day's work and plan the work ahead. Usually when it was night in India (Unnat), it was morning in the US (Rahul), so both of us would either stay up late or would wake up early to communicate. We used GitHub, Zoom and Whatsapp extensively. We decided on our topic much before COVID-19 became a pandemic, and ironically the topic of our project has put these communication constraints on us. Working on it since then, we have acquired immense knowledge on the demographics and statistics. This project has certainly motivated us to keep up to date as some of our design choices are grounded in the evolving situation (e.g. people can get infected multiple times).
3. Realized or contemplated alternative design / approaches to the project

   We had multiple discussions about how we should design a particular object. Initially, when we were designing the schema, we were not sure how to represent the person, infected, recovered and dead tables. We were thinking of having just 1 table for person and adding a field called status but then later we realized it makes more sense to use inheritance and make the infected, recovered and dead entities children of the parent class (person). We also debated on the structure for tests and hospitals but finally we came to a conclusion. Later, when we were starting work on the client code, we were faced with the dilemma of choosing between Java and Python for the frontend. We ended up with Python since it offers the flexibility of dynamic programming.
4. Document any code not working in this section

   We had a few issues with the same code working only for one of us. For example, while taking input in Python, 'input()' worked for one of us (Python 3.X) but it didn't for the other (Python 2.X)- 'raw_input' instead. Nevertheless, we got everything to work. All of the functionality implemented in the application works as per our testing.

Future Work

1. Planned uses of the database

   We plan on using this project to derive insights such as recovery rates based on hospital, age, etc. It can also serve as a tool for health offices and hospitals.
2. Potential areas for added functionality

   The largest area for more functionality is statistics. There are innumerous things we can add. We can expand by first adding more fields in the person table. More information about the person infected will make predictions more accurate and it will open the way for a vast number of insights. For example, if we record information such as past diseases and conditions, we can predict if a person with a particular disease/infection has a higher chance of contracting the virus. Adding fields for hospitals and visits will also give us better insights on them. We will be able to say how many people contracted the virus in the ICU or how many nurses/doctors contracted the virus after treating a patient. The scope of adding functionality is limitless.