**Design Document for Hospital Management System**

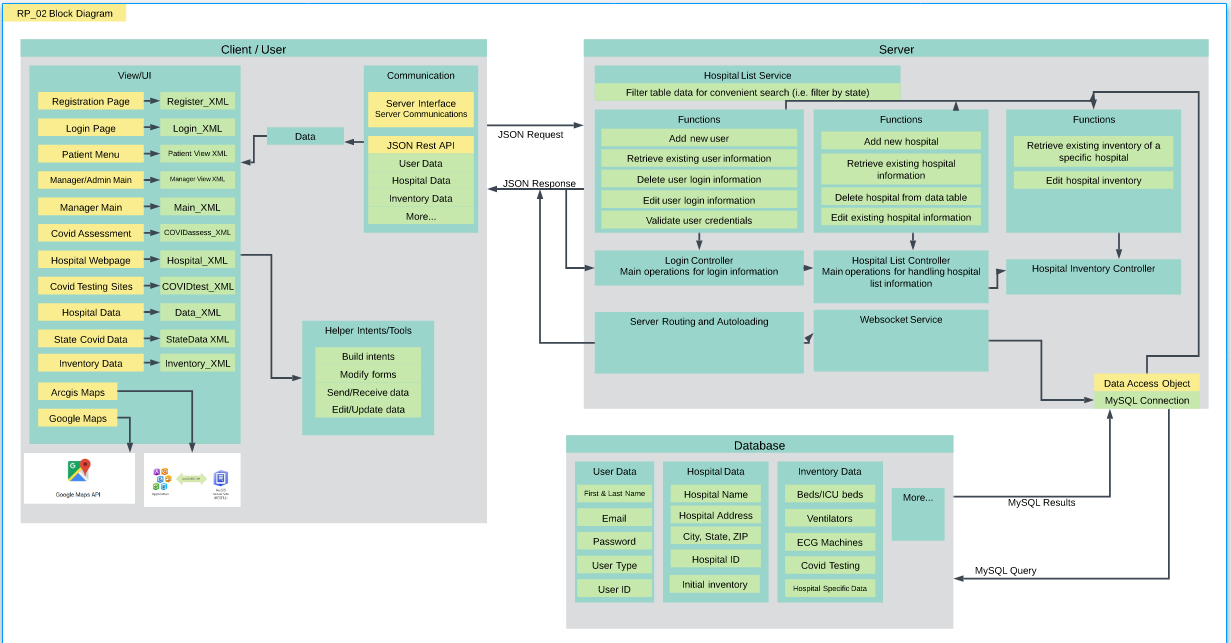
Group RP-02

Amber Chin: 25 % contribution

Vance Kaw: 25% contribution

Daniel King: 25% contribution

Yuin Choon Ng: 25% contribution

PUT THE BLOCK DIAGRAM PICTURE ON THIS PAGE! (Create the picture using lucid chart)

Use this third page to describe complex parts of your design.

The client side for our project, Hospital Management, uses android studio as a base. Our application has 13 activity classes which are listed in the View UI section and each activity has its corresponding xml file which provides the view for the users in terms of page styles and format. Within our app, there are different activity classes for each individual app page. Hence, our app has helper intent and tools that facilitates communication between components and or activities. The helper intents and tools mostly help us to edit and update data as well as send and receive data in the frontend. For our application, we have a communication interface with the client. The interface handles the JSON commands from the user that is being sent to the server. All the activities will send and get data through the server interface. In our application, we dedicated a page which includes the statistics of the available hospitals and their related information in each state in the United States. Almost all the inventory data can be accessed from the server; covid updates are being run with a script that accesses the data from a csv. From a user perspective, the users of the Hospital Management application are able to make changes to the personal information through the personal information page. After users have confirmed to make changes and push the edit button, our application will send a json request to the login server, and edit the elements of user data.

The database for this project includes all the tables and information we need to run the app. The tables include a user information table and a hospital list table. The user information stores individual user information, including first name, last name, email address, password, and user type. The user type option allows users to be designated as admins, managers, doctors, or patients, while allowing the app the flexibility to expand this list of users in the future. In addition, the hospital lists database includes the data we need to make a significant portion of the live data updates for the app, including total beds, ICU beds, ventilators, ECG machines, Covid testing, and more. The hospital lists database also allows us to populate the app with important individual data about each hospital, including hospital name, address, coordinate locations, and more. The tables are configured to allow us to make additional changes as necessary.

The server part of this project serves as the backbone of storing, editing and retrieving data. At some point, information will need to be stored in a specific location so it could easily be retrieved anytime by the frontend team. Inside the server are several tables that serve as like a list of information pushed by the frontend team. Our team created a table called login information which basically stores the credentials of every user for easy access and convenience for everyone. The login controller has the ability to add a new user, change the information such as the email address of an existing user, view the user’s credentials, validate the login credentials, and lastly delete any existing user. Next, we have another table dedicated to storing the important information of every hospital in the US such as the name, county, address, latitude and longitude for easy access to maps. The backend team also created a filter feature to filter out hospitals by state so it’s much easier to view. Then, we have a table relationship that maps every hospital to many inventory meaning every hospital will have a unique set/number of hospital equipment/inventory. In the hospital inventory table, we have the similar ability/function to change the existing numbers to match the current inventory that a hospital has. For frontend or client communication, the server/spring will send a JSON object and receive the same object too. We also used MySQL Workbench which offers a cleaner and more organized look at the tables present.

PUT THE TABLE RELATIONSHIPS DIAGRAM on this fourth page! (Create the picture using MySQLWorkbench)