**Documentation of 291 Mini Project**

* Xutong Zhao 1430631
* Ma Yue 1434071
* Yu Zhu 1429028

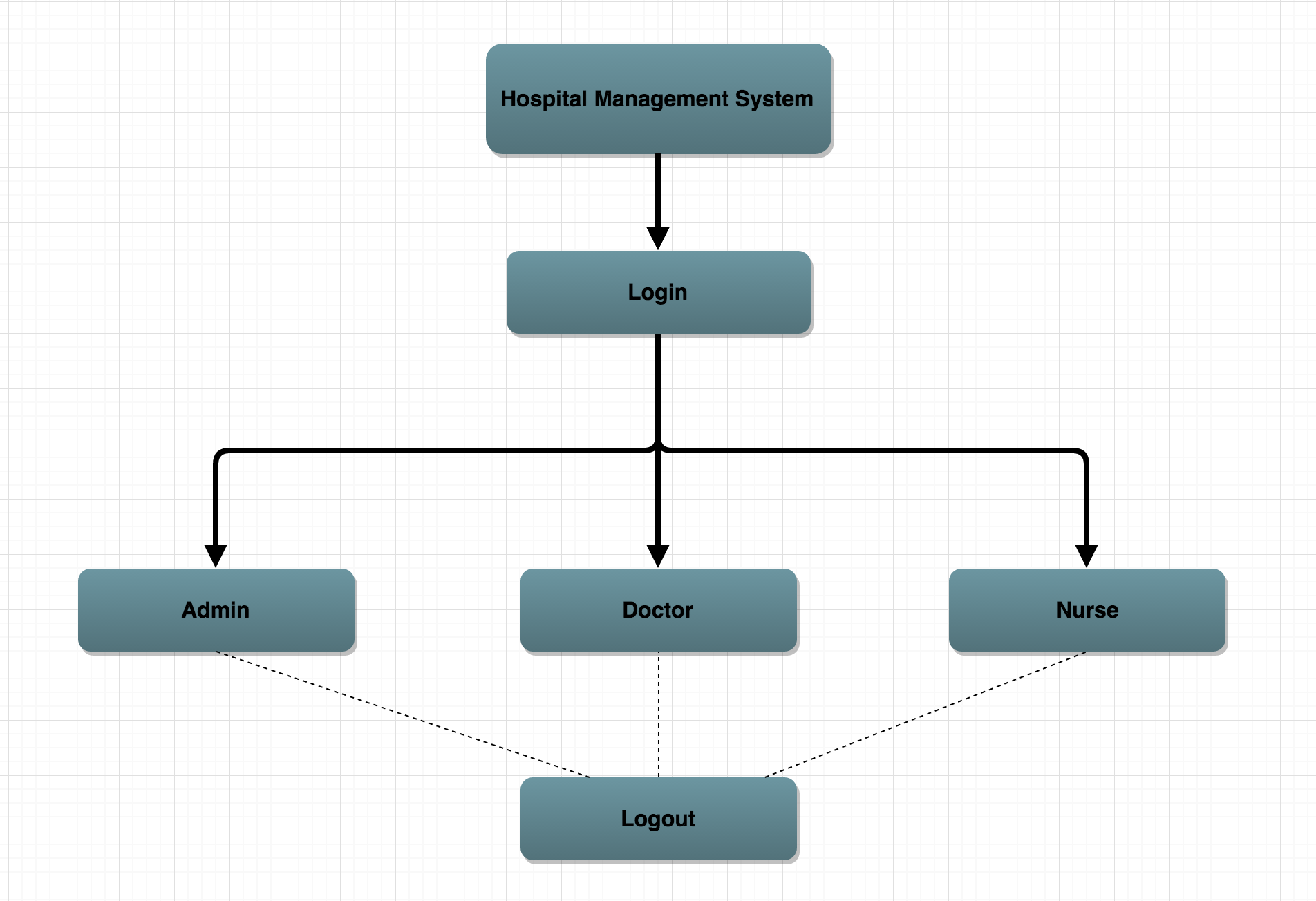
**Part1. Overview:**

This is an application of generating a hospital’s database. All the codes are written in Python and users can run the application through the terminal. For login screen, user can choose to log in or shut down the program. If the user chooses to log in, he should input his role and his password to enter the application. There are three classes of uses. Two of them are *care staffs* (doctors and nurses) who use the system to query and update information about patients. The third one is *administrative staff members who* use the system to create certain types of reports.

Different roles have different accesses to the database. After the user logs in the application, he can choose one of the allowable operations to generate the database.

**Part2. Software Design**

Software Class Design Diagram



Based on the project specification, we created 2 different *.sql* data sets for the purpose of testing the program. One of them is short yet not quite complete since it was simply for testing basic functionality at the beginning. The other one totally conforms to the specification, used for testing all tasks.

In this project, we divided it into several parts. One part is main flow including logging in parts(command.py). Another three part is three different classes which implement different roles’ tasks.

**Command.py**

The main flow of the software is controlled by command.py, including the login, logout, and functionalities of different users. Base on the user's login credential, it would determines user's role and direct them to the correspond user menu. Promoting user to type in the necessary information to fulfill the user task, and then pass over the information to the user class(Doctor, Nurse, and Admin) in order to interact with the Hospital’s Sqlite database.Handing the login and password encryption feature by interacting with the Sqlite database. All user interaction is done through the command.py to keep everything modular and much easier to read.

### **Doctor Class**

Doctor menu is used to manage the logic of nurse class and functionalities of nurse. The task of doctor includes the following:

1. For a given patient, list all charts in the system ordered by *adate* (indicating also whether they are closed or open). The user should be given the option to select a chart. Once a chart is selected, all entries (symptoms, diagnoses, and medications) associated with that chart must be listed, and the result must be ordered by the date of the entries.
2. For a given patient and an open chart of the patient add an entry for symptoms. The date *obs\_date* should be set to the current date and time.
3. For a given patient and an open chart of the patient add an entry for diagnosis. The date *ddate* should be set to the current date and time.
4. For a given patient and an open chart of the patient add an entry for medications. The date mdate should be set to the current date and time. Additional checks should be performed before adding the entry: (1) if the prescribed amount for the patient is larger than the recommended amount sug\_amount for that drug and the patient's age group, a warning should be issued that contains the information about recommended amount for a patient for that age group, and the doctor should be given the choice to confirm his prescription or to change the amount. (2) If the patient *could* be allergic to the prescribed drug *drug\_name*, a warning should be issued that contains the information about the reported allergy; the warning should display the name of the drug that the patient reported being allergic to, and, if that is not directly *drug\_name*, the name of the drug D should be displayed, which the patient reported being allergic to and from which it can be "inferred" that the patient may also be allergic to *drug\_name*.

### **Nurse Class**

Nurse menu is used to manage the logic of nurse class and functionalities of nurse. The task of nurse includes the following:

1. Create a new chart for a patient at the time of admission to the hospital. At that point in time, the *adate* is filled with the current date and time, and the *edate* is filled with a NULL value, indicating an "open" chart. Before creating a new chart, the system should check whether there is already an open chart for that patient, and if so, provide the options to either close this chart before creating a new one, or not creating a new one. When creating a new chart, the system also must provide the functionality to add the patient information, if the patient information is not already in the system (from a previous stay in the hospital).
2. Close a chart when a patient is dismissed from the hospital. At that point in time, the *edate* is filled with the current date and time, indicating that the chart is closed.
3. Same as 1. for the doctors.
4. Same as 2. for the doctors.

### **Admin Class**

Admin menu is used to manage the logic of admin class and functionalities of admin. The task of admin includes the following:

1. creating a report, that lists for each doctor the name and the total amount of each drug that the doctor prescribed in a specified period of time. (Drugs that he did not prescribe in that period of time should not be listed.)
2. For each category of drugs, list the total amount prescribed for each drug in that category in a specified period of time. The report should also contain a total for each category.
3. List for a given diagnosis all possible medications that have been prescribed over time after that diagnosis (over all charts). The list should be ordered by the frequency of the medication for the given diagnosis.
4. List for a given drug all the diagnoses that have been made before prescribing the drug (over all charts). The list should be ordered by the average amount of the drug prescribed for the diagnoses.

**Part3. Testing Strategy**

Since our project was implemented via Object-Oriented Programming, each classes were tested individually. After all partial tests succeeded, we completed the main function and tested again.

Nurse class testing strategy began with a test file named *NurseTest.py*. For the first task of a nurse, we create a lot of patients and charts. During the test we found that the *chart\_id* shouldn’t be repeated, so we changed the nurse class and checked whether the *chart\_id* had already existed. For the second task of a nurse, there was a bug founded during the test. If a patient had several charts, we should only fill up the *edate* which is *NULL* and shouldn’t change all edate of that patient. The third and the fourth tasks were tested below doctor class.

The general testing strategy for the Doctor class is similar to that for the Nurse class. In a separate test file, we run the primary methods with different doctor *stuff\_id*s. Each time a row(s) is inserted into a table, we compare the updated database with initial data. Similarly, each time we select rows, we check their correctness by manually extracting them from the database. By doing so, we did find and fix a few issues/bugs. For instance, the description for the first task indicates that selecting a chart is an option, instead of a required step. Hence, we added an additional condition to ensure that it is up to the user’s choice. Moreover, when displaying entries associated with a chart, each entry is supposed to be listed separately, otherwise the program will result in all combinations of them.

The general testing strategy for the Doctor class is similar to that for the Nurse and Doctor class. It is done by observing the comparison between manually selected data and query selected data by running separate test file.

**Part4. Group Work Strategy**

There are three people working on this project. This project have four code files and a database file. One file named command.py which acts as a main function including logging in part is mainly done by Yu(1 hr). The other three code files are three classes. Doctor class is done by Xutong (3 hr). Nurse class is done by Yue(3 hr). Administration class is done by Yu (3 hr). The database is mainly generated by Xutong (2 hr). For the documentation, overview and group work strategy are written by Yue(1 hr). Software design is done by Xutong and Yu (1 hr). Testing strategy is done by Yue, Yu, and Xutong(2 hr). Every person has his own task and we also help with each other to complete the project.

To keep the project on track, we met every second day. We also chatted and shared codes with each other through the internet.