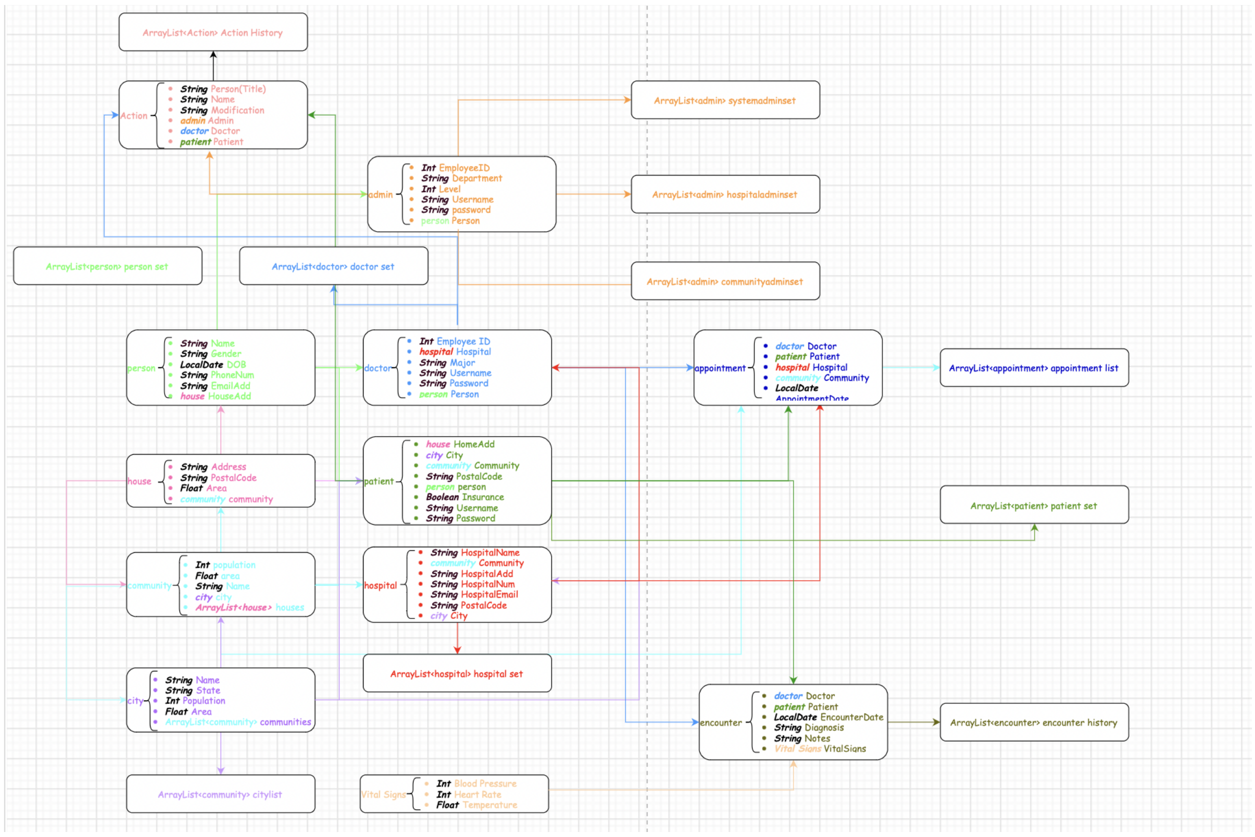
Classes

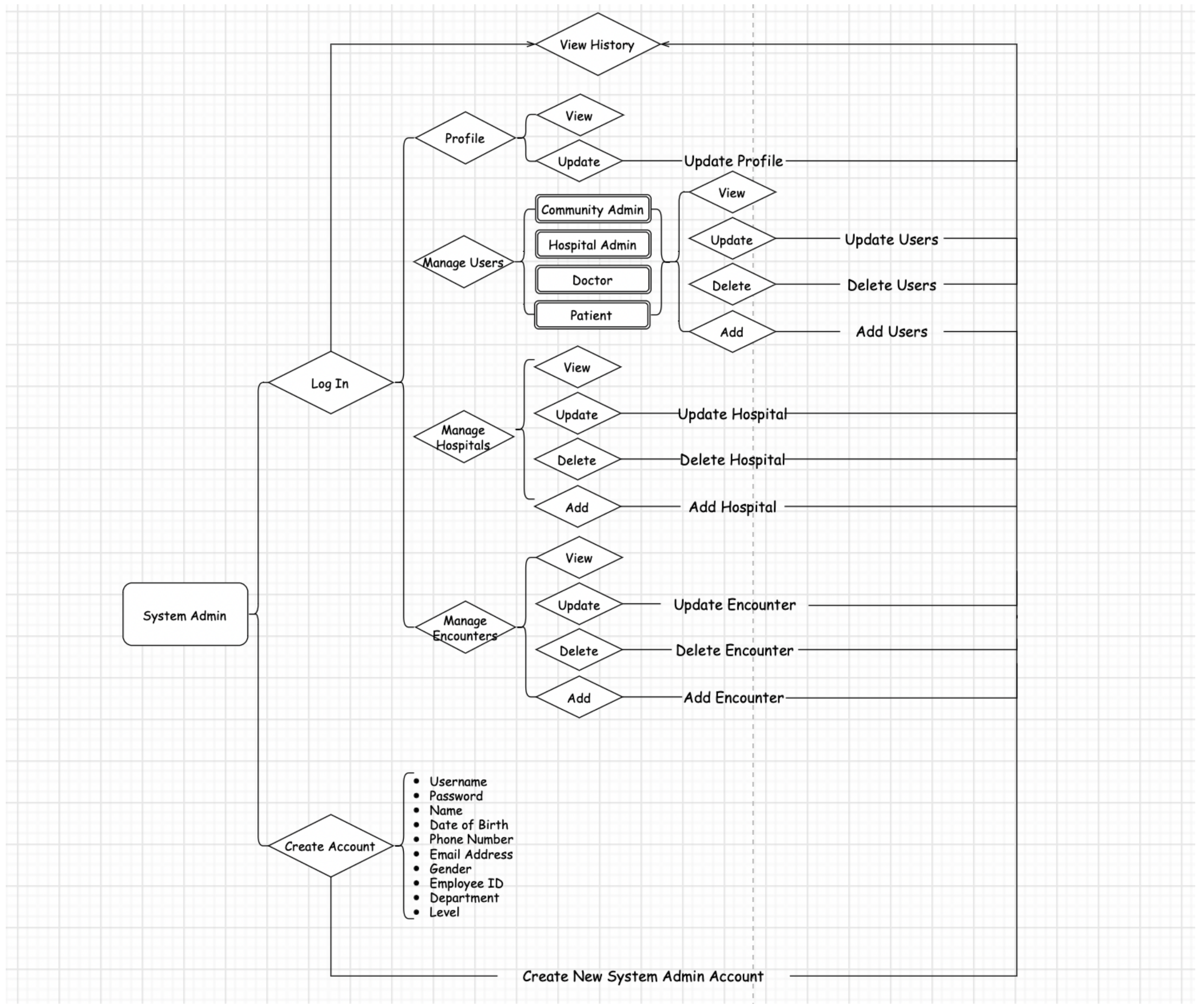
As shown in the demo class diagram, a well-organized structure of classes could help a lot while designing an application. Setting and getting values from corresponding classes will be easy and intuitive. Based on the basic classes, I have designed a more comprehensive class-net for the project. The attributes and how they're linked together can be seen from the following diagram.



Relevant classes are connected and annotated by the same color. For example, A *patient* class has attributes like *"house", "city", "community", "person"* and some primitive data types like String and Boolean. *Patients, doctors and administrators* are all humans. Therefore, they can all inherit some attributes from the class *person*, like Name, Gender, DOB, etc. By defining a class *person* and connecting *patients, doctors and administrators* to the class person could save us from defining the same attributes for each of them.

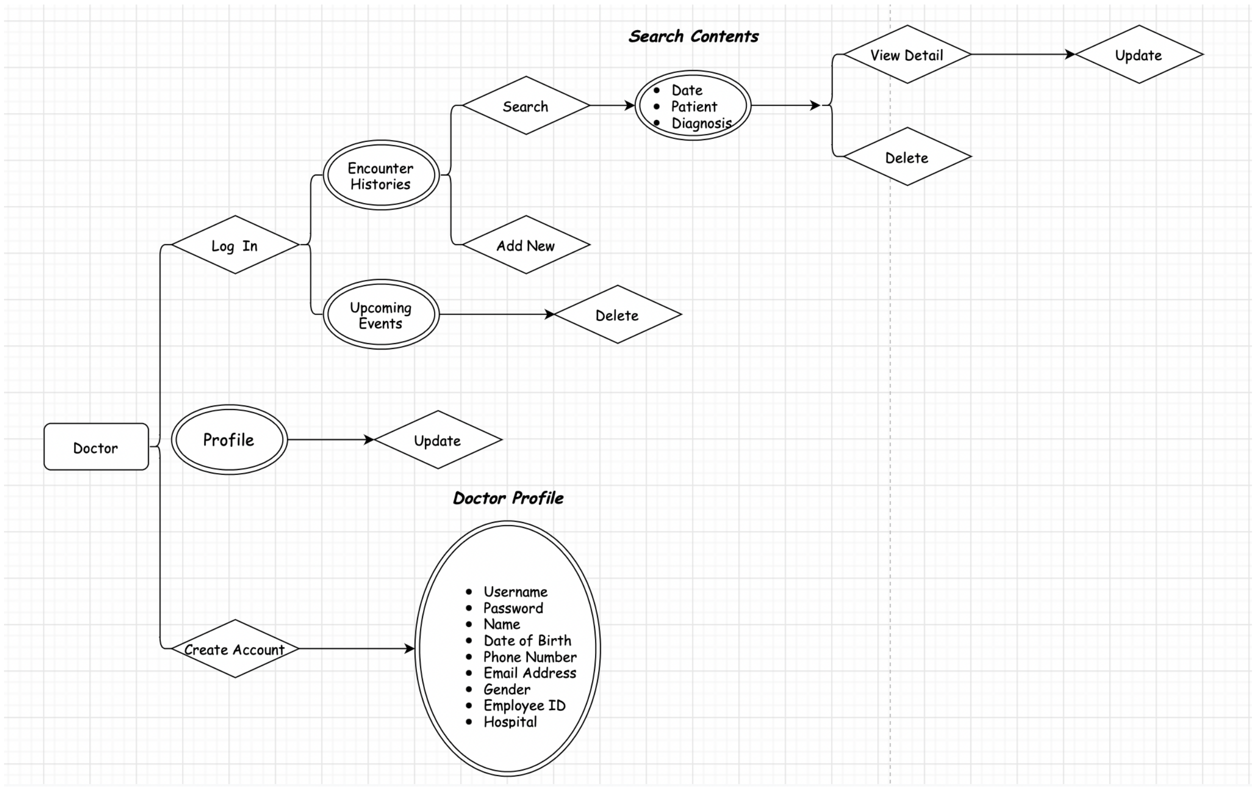
Functions

Being aware of different roles, their access to different data and what operations they could perform could help designers gain an overview of the whole application before actually diving into design. From the perspective of a system administrator. The work flow can be shown as the following diagram.



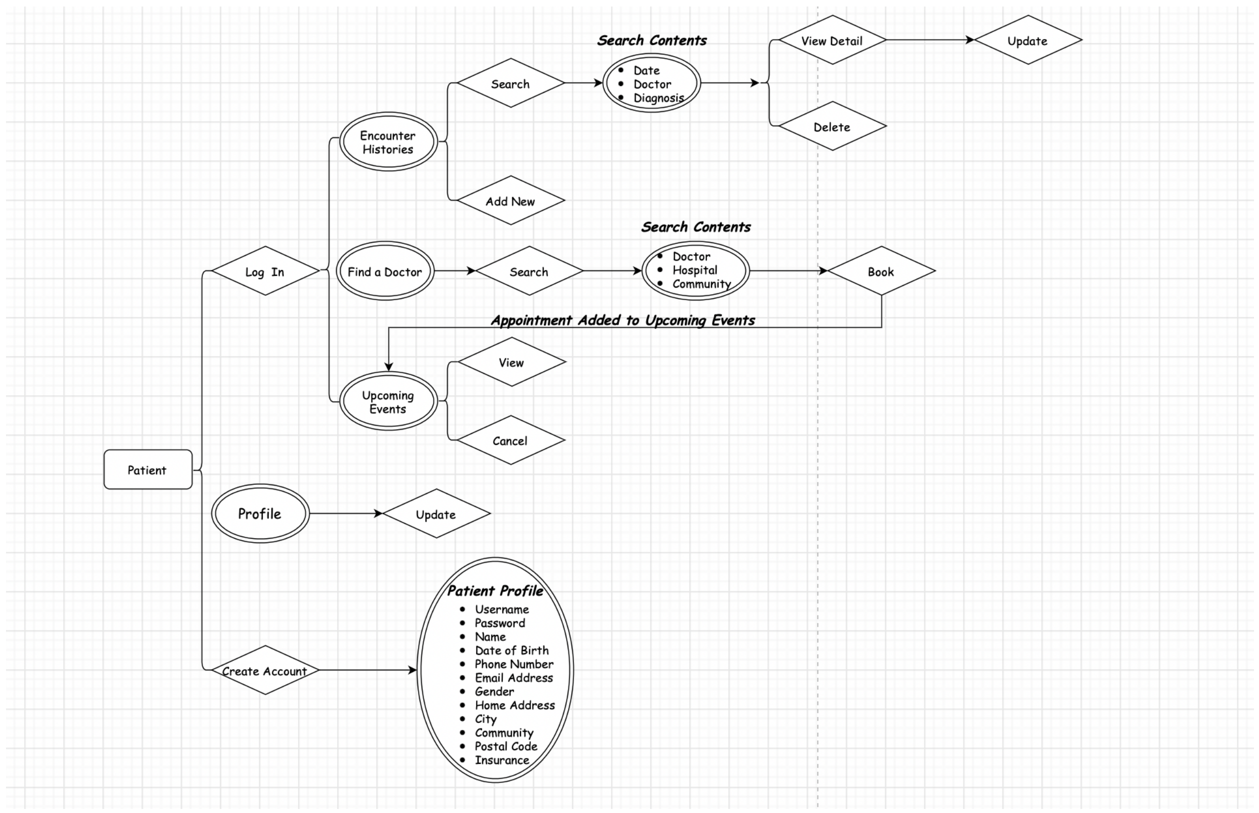
In the log in page, an older user should be able to log in directly. New Users should be able to create a new account by entering the information required to define a new administrator. After logging in, the system administrator should be able to manage his own profile, including view of update information. Also, system administrators should be able to manage all the other users: community administrators, hospital administrators, doctors and patients by CRUD their accounts. Similarly, system administrators should also be able to manage the hospitals and encounters. Any changes will be added into the arraylist of action history. In the meantime, system administrators could view all of the change histories in a panel.

From the perspective of a doctor, his workflow is as follows:



Compared to a system administrator, a doctor can access much less data and do less operations. Apart from some basic functions, like updating profiles and creating accounts mentioned earlier in the system administrator workflow, a doctor's work is mainly about managing the encounter history with his patients. A doctor can view, add, edit and delete an encounter. However, unlike system administrators, a doctor can only access the encounter histories of his own patients. Additionally, a doctor could cancel an appointment assigned to him if any circumstance happens.

As for a patient, what is unique is he could find a doctor or a hospital he likes and make an appointment with a doctor. Also, he could use the search function to find a doctor or hospital nearest to him. The workflow is shown below.



One thing that is worth noting is that by booking an appointment with a doctor, the appointment will then be assigned to the doctor and shown in both the patient and doctor's upcoming events table.

Run the Application

Basically, after downloading the project and open it in NetBeans, clicking on the "RUN" button, the application should work, and feel free to play with the buttons and see what happens.