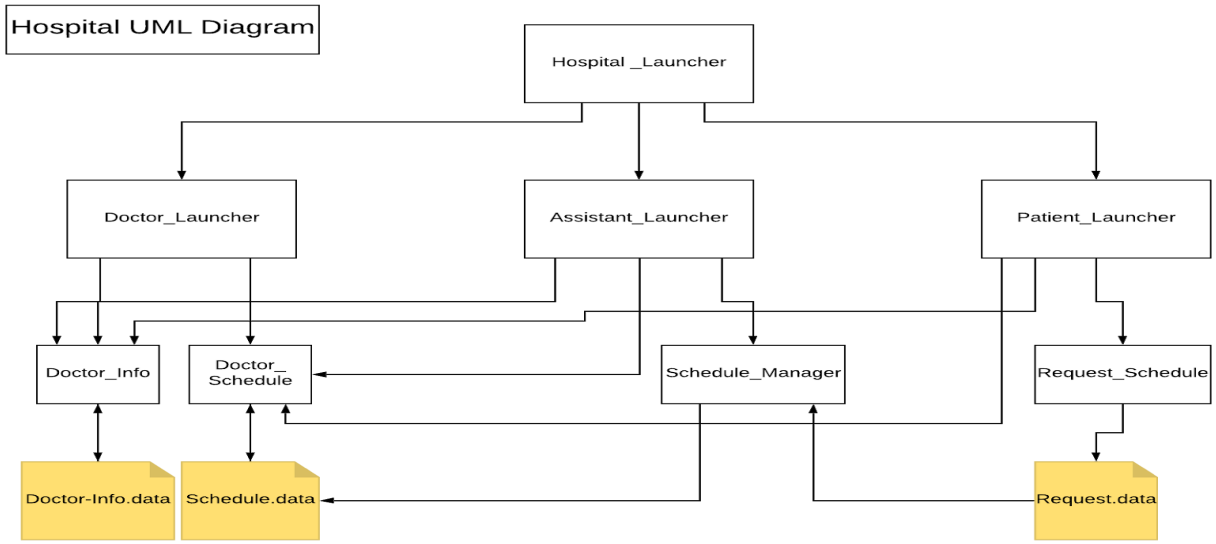
Final Project Report

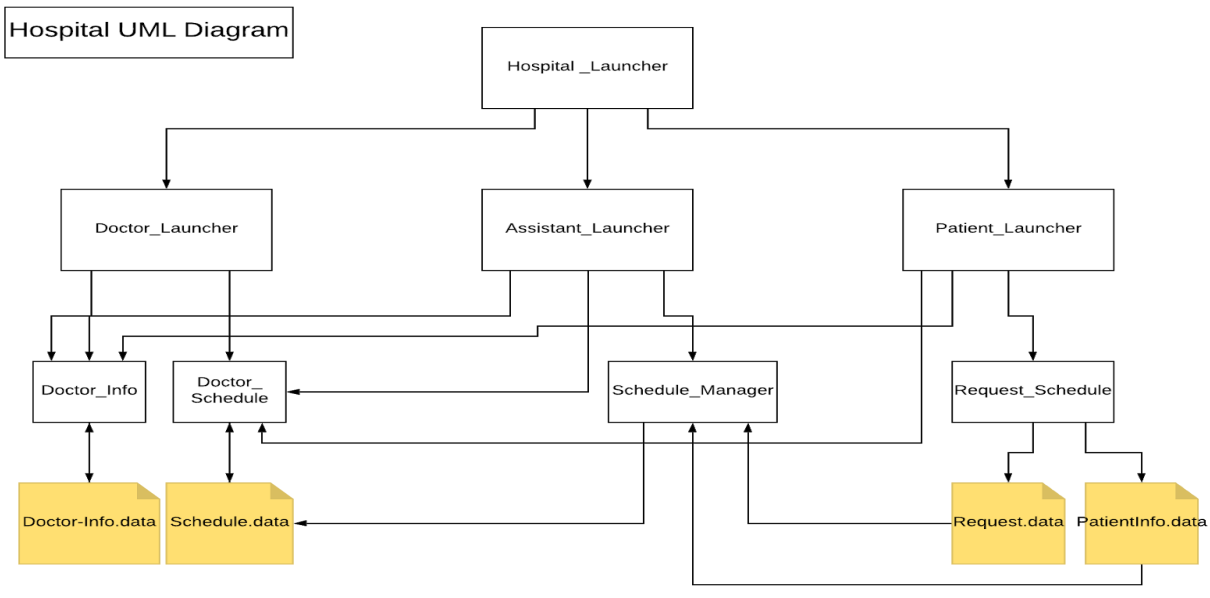
Zain Habib, Sun Jun, Jason Ryu, Taeyoon Choi

Original UML Diagram:



Our original diagram was pretty much in line with what we ended up doing. The relationships between all the classes were right and the interactions are just how we depicted them above. Our original diagram was missing a key part of the system though. While patients were able to submit requests, they couldn't submit any ancillary information like their names or any symptoms they were feeling. So we created a new data file that could hold information like that and be passed onto the doctor with the appointment request. The new diagram is below:

New UML Diagram:



How it Works:

We have designed a hospital system in which there are three views: *patient, assistant, and doctor.* Each user will be prompted to log into the system into one of those views. Below we will be explaining the functionality of each view of the system and the different classes we used to build each part of it.

Doctor

Doctors have the most functionality in terms of what they can do in the system. As with each other actor in the system, the first prompt anyone will get will be to log into the system. Once a doctor successfully enters their User ID and password, they will be let into the system to make edits. Once logged in, a doctor can edit his/her information, update his/her schedule, and look at appointment requests. Inside each appointment request, doctors can even look at the patient information such as names and the symptoms that each patient is feeling. Additionally, they can edit their own personal schedule and add or remove available hours from the hospital system. That way, the assistant won't accidentally schedule appointments for times that the doctor isn't available to see a patient. We took three classes to create the doctor experience: *doctorLauncher, doctorInfo, doctorSchedule*. The doctorLauncher is the dashboard that allows the doctors to edit anything. The doctorInfo class is there to save their important information: License number, phone number, email, name, specialty, etc. They can also make edits to their schedule in the *doctorSchedule* class. These two classes spawn data files that save all this information and the scheduleManager then compares it to the patient requests.

Assistant

The Assistant has two classes dedicated to it: *assistantLauncher* and *scheduleManager.* The assistant acts as the liaison between the patient requests and the doctors. Using scanners, the schedule manager class pulls data from the requestSchedule file and patientInfo and compares them to available times in the Doctor's schedule. If the doctor's [available] schedule contains the request, then the schedule manager approves the request and adds it to the doctor's schedule. This automatically updates the doctor's schedule so that if someone requests the same time afterwards, they will get denied. The *assistantLauncher* class serves as a dashboard for the assistant. It allows them to view the doctor's schedule and then pushes information down to the *scheduleManager* class.

Patient

The patient requires two classes as well. The *patientLauncher* class allows the patient to log in and gives them a dashboard to choose from. The patients have the most limited abilities so they can only view the doctor's list and their specialties, view their schedules that are available to the public, and request to make an appointment with a doctor. The *requestSchedule* takes their appointment requests at the top of every hours from 8:00 am to 5:00 pm, normal business hours. Additionally, it will ask them for their name and the symptoms they feel so that the doctor has a better idea of what is wrong with the patient. This information is then saved into a data file that is sent to the *scheduleManage*r class.

The hospitalLauncher acts as a main hub that directs the right people to their respective dashboards. It will ask the user if they are a patient, assistant, or doctor. Once the user chooses which one they are, it will ask them for their login information.

Test Cases

Inside the README file, you will find test credentials you can use to log into our system as a doctor, assistant, and patient. These credentials will give you access to different parts of the system and will let you see how each part of the system relates to the next.