Re-medi EHR System

Code base link: https://github.com/FreyaXiang/ehr
For installation rules, please refer to the README file in the repository.

Team Organization:

Andrew Lee	Backend Support
Andrew Liu	Backend Database Management
Xin Xiang	Frontend Components, Backend API Design
Zane Fadul	Frontend Routing, Test Writing, UI/UX Design

Table of Contents:

Description 1

Process 2

Requirements & Specifications 2

Architecture and Design 7

Reflections and Lessons Learned 10

Description

Our product is a modern Electronic Health Record (EHR) system that improves on existing User Experience Design. We developed a user-friendly product that provides the key services that stakeholders involved in the healthcare system requires.

By improving the existing EHR infrastructure we seek to introduce an improvement and alternative which is not limited towards serving specific organizations. Our product attempts to greatly increase ease of access to personal medical information while also serving the needs of Medical Organizations. Additionally, we built the product with the intention to allow for ease of scalability in the future.

Process

At the beginning of the semester, the process that our team planned to follow centered around 2 main ideas. Firstly, since our team did not have any prior experience working professionally in software engineering, we did our best to follow the version of XP that we were taught in class, the Agile development methodology. This allowed us to develop our project in an iterative manner, as we worked in sprints and released and pushed code as often as possible.

We treated our 3 major sprints of the semester as epics, and broke down our user stories into individual tickets. To work on a ticket, we each created a branch for development use, and made sure to conduct code reviews across the team before code was merged into our master branch. The iterative development portion of our project went very smoothly, and we had no issue working on individual tickets that contributed towards the end goal.

The second key idea we planned to follow was to create our project using the latest and most modern web technologies, while following the best practices involved in industry. A large portion of the initial time we spent was deciding which tools we would use. We ultimately decided on using the tools of JIRA for workflow management, Github for source control, Illustrator for Mockups, the MERN stack for development, and Jest.js for testing.

The best practices we attempted to implement were practices that we covered in class, more specifically refactoring, testing, pair programming, and code reviews. Our team did not encounter any problem whatsoever with refactoring, as we did our best to plan out our file structure and make our code modular in advance. We refactored our code during our first sprint, when we noticed that our server file was handling too many functionalities all at once.

Testing was also not an issue with our team, as members wrote tests on the code before it was pushed. Additionally, we also had tests in jest.js to further confirm that there was no problem with our code. However, for collaborative development, of which we attempted to conduct pair programming and code reviews, our team did encounter a lot of issues.

Most of the issues we encountered were logistical in nature, due to the timezone difference between our team members. We did not manage to release code as quickly as we would have liked, as blocked members and tasks lasted longer than they should have ideally been. To overcome this problem, we had to leave detailed comments and notes regarding pushed code for other members to read, but this also had a limited effect as compared to direct conversation.

Requirements & Specifications

The user stories and use cases which we implemented in our project can be found below.

Fully Dressed Use Cases for the Hospital Staff Actor:

UC1 Manage Patients
UC2 Add/Remove Patients
UC3 Schedule Appointments
UC4 Access Patient Information
UC5 Access/Change Patient Health Records
UC6 Access/Change Patient Personal Information
UC7 Give Recommendations/Prescriptions

UC1 Flow of Events for the Manage Patients Use Case

1.1 Preconditions:

Hospital Staff is logged in.
Hospital Staff has selected the manage patients option.

1.2 Main Flow:

The hospital staff can manage the relevant logistical aspects of the patient's details. This is the non-medical details regarding the patient, usually to be changed by those working at the front desk. The hospital staff can choose to add/remove patients [UC2], or schedule appointments [UC3].

1.3 Subflows:

None.

1.4 Alternative Flows:

None.

UC2 Flow of Events for the Add/Remove Patients Use Case

2.1 Preconditions:

Hospital Staff is logged in.
Hospital staff has selected the add/remove patients option.

2.2 Main Flow:

The hospital staff can now add/remove patients on this page. In the case of adding patients, the hospital staff must fill out the basic required information before clicking the add button. In the case of removing patients, the hospital staff must fill out the basic required information before clicking the remove button. The requested process is then completed.

2.3 Subflows:

[S1] A message confirming patient details and the action of either adding or removing is shown to the hospital staff. Staff must confirm the details before the patient is added or removed.

2.4 Alternative Flows:

[E1] If the required information is not filled out completely the hospital staff must fill out the required missing fields.
[E2] If the required information is not filled out correctly, the hospital staff must fill it up again in the correct format.
[E3] The patient's details must be found for them to be removed. Otherwise an error message saying no such patient found is displayed.

UC3 Flow of Events for the Schedule Appointments Use Case

3.1 Preconditions:

Hospital Staff is logged in.
Hospital staff has selected the schedule appointments option.

3.2 Main Flow:
The hospital staff can now schedule and manage appointments with specific patients. Staff must select a patient and choose a time. After it is entered, additional information such as which doctor or department to be scheduled for may be entered as well. [S1][E1] Afterwards, the appointment is created.
3.3 Subflows:
[S1] A message confirming appointment details is shown to the hospital staff. Staff must confirm the details before the appointment is made.
3.4 Alternative Flows:
[E1] If the scheduled doctor or department is full for said appointment, an error message is displayed and the hospital staff is prompted to schedule a new appointment.

UC4 Flow of Events for the Access Patient Information Use Case
6.1 Preconditions:
Hospital Staff is logged in.
Hospital Staff has selected the access patient information option.
6.2 Main Flow:
The hospital staff can manage the relevant medical aspects of the patient's details. This is the medical details regarding the patient, usually to be changed by those working with the patient, such as doctors or nurses. The hospital staff is given an option to choose the patient they want the records of. [S1] The hospital staff can then choose to access or change patient health records [UC7], access or change patient personal information [UC8], or give recommendations and prescriptions [UC9].
6.3 Subflows:
[S1] The hospital staff is displayed a window where they can select the patient they want the records of. They can manually fill in patient information or select the target patient. [E1-2]
6.4 Alternative Flows:
[E1] The selected patient must exist. Otherwise, the staff is required to select the correct target patient.
[E2] The hospital staff must have the necessary permission level to access the records of this patient.

UC5 Flow of Events for the Access/Change Patient Health Records Use Case
7.1 Preconditions:
Hospital Staff is logged in.
Hospital staff has selected the access/change patient health records
7.2 Main Flow:
The hospital staff can access or change the patient health records. The relevant medical health records are displayed to the hospital staff. This includes information such as past medical notes, checkups, diseases, prescriptions, etc. The hospital staff can make edits to these health records after asking for the patient's permission. [UC10]
7.3 Subflows:
None.
7.4 Alternative Flows:
None.

UC6 Flow of Events for the Access/Change Patient Personal Information Use Case
8.1 Preconditions:

Hospital Staff is logged in.
Hospital staff has selected the access/change patient personal information

8.2 Main Flow:

The hospital staff can access or change the patient personal information. The relevant medical health records are displayed to the hospital staff. This includes the personal information on the patient's end such as contact numbers, contact email, family members, age, date of birth, etc. The hospital staff can make edits to these health records after asking for the patient's permission. [UC10]

8.3 Subflows:

None.

8.4 Alternative Flows:

None.

UC7 Flow of Events for the Give Recommendations/Prescriptions Use Case

9.1 Preconditions:

Hospital Staff is logged in.
Hospital staff has selected the give recommendations/prescriptions option.

9.2 Main Flow:

The hospital staff can make medical notes and prescriptions to the patient on this page. The hospital staff can write a note or prescription, which will be posted to the patient's account and records.

9.3 Subflows:

None.

9.4 Alternative Flows:

None.

Fully Dressed Use Cases for the Patient Actor:

UC1 Manage Records
UC2 Manage Health Records
UC3 Access Personal Information
UC4 Contact Hospital Staff
UC5 View Prescriptions Information

UC1 Flow of Events for the *Manage Records* Use Case

1.1 Preconditions:

The Patient is logged in.

1.2 Main Flow:

Once the Patient is logged in, a dashboard will appear on the screen. From there, the Patient will be able to click on a *Manage Records* button, which will bring the Patient to a new section of the platform.

1.3 Subflows:

None.

1.4 Alternative Flows:

None.

UC2 Flow of Events for the *Manage Health Records* Use Case

1.1 Preconditions:

The Patient is currently in the dashboard's *Manage Records* [UC1] page.

1.2 Main Flow:

The Patient will now be able to see two possible routes, *Manage Personal Health Records* and *Manage Visitation Records*. By clicking *Manage Personal Health Records*, the Patient will be brought to a preview of their medical records.

1.3 Subflows:

None.

1.4 Alternative Flows:

[E1] The Patient's health record will be read-only until the Patient chooses to edit it by clicking the edit button. Here they can edit things like personal details, allergies, past medical history (not already on the platform), etc. These records will show what patients typically fill out their first time at a new doctor's office.

[E2] The Patient will be able to share the health record via email and/or other methods of e-communication.

[E3] The Patient will be able to export the health record as a PDF.

UC3 Flow of Events for the *Access Personal Information* Use Case

1.1 Preconditions:

None.

1.2 Main Flow:

The Patient will be able to see their profile picture on every screen on the platform. At any time, they can click their icon to reveal a drop down bar. From there, they can click *Account* and will be able to access specific personal account information, such as username, password, phone number, address.

1.3 Subflows:

None.

1.4 Alternative Flows:

None.

UC4 Flow of Events for the *Contact Hospital Staff* Use Case

1.1 Preconditions:

The Patient is at the dashboard. The Patient is already registered with a Hospital on the platform.

1.2 Main Flow:

The Patient will see a tab on the dashboard *Contact Hospital Staff*. Upon clicking, the Patient will be able to see a list of registered hospital staff members corresponding to their registered hospital. They can search for the staff member or simply scroll to find them. Once they find the staff member they would like to contact, they can click on the staff member. From there, they will be able to send messages through the platform, or will receive contact information to call the hospital staff member.

1.3 Subflows:

[S1] The Patient will need to look for their preferred staff member through the list of registered accounts in the platform.

1.4 Alternative Flows:

[E1] If the Patient isn't already registered with a hospital, they will have to be added by a staff member in order to have access to things like staff contact information.

UC5 Flow of Events for the *View Prescriptions Information* Use Case

1.1 Preconditions:

The Patient is at the dashboard. The Patient has met with a doctor and has been prescribed medication.

1.2 Main Flow:

The Patient will be able to access records of previously prescribed and currently prescribed medication. Information such as the location to retrieve the prescription and details about the prescription itself will also be available for the Patient to view.

1.3 Subflows:

None.

1.4 Alternative Flows:

[E1] If the Patient does not have pre-existing prescriptions, the tab leading to *View Prescriptions Information* will not appear.

Architecture and Design

I. Project Architecture:

We are following the MVC project architecture. More details are outlined in the figure below:

M: models folder V: client folder C: routes/api folder

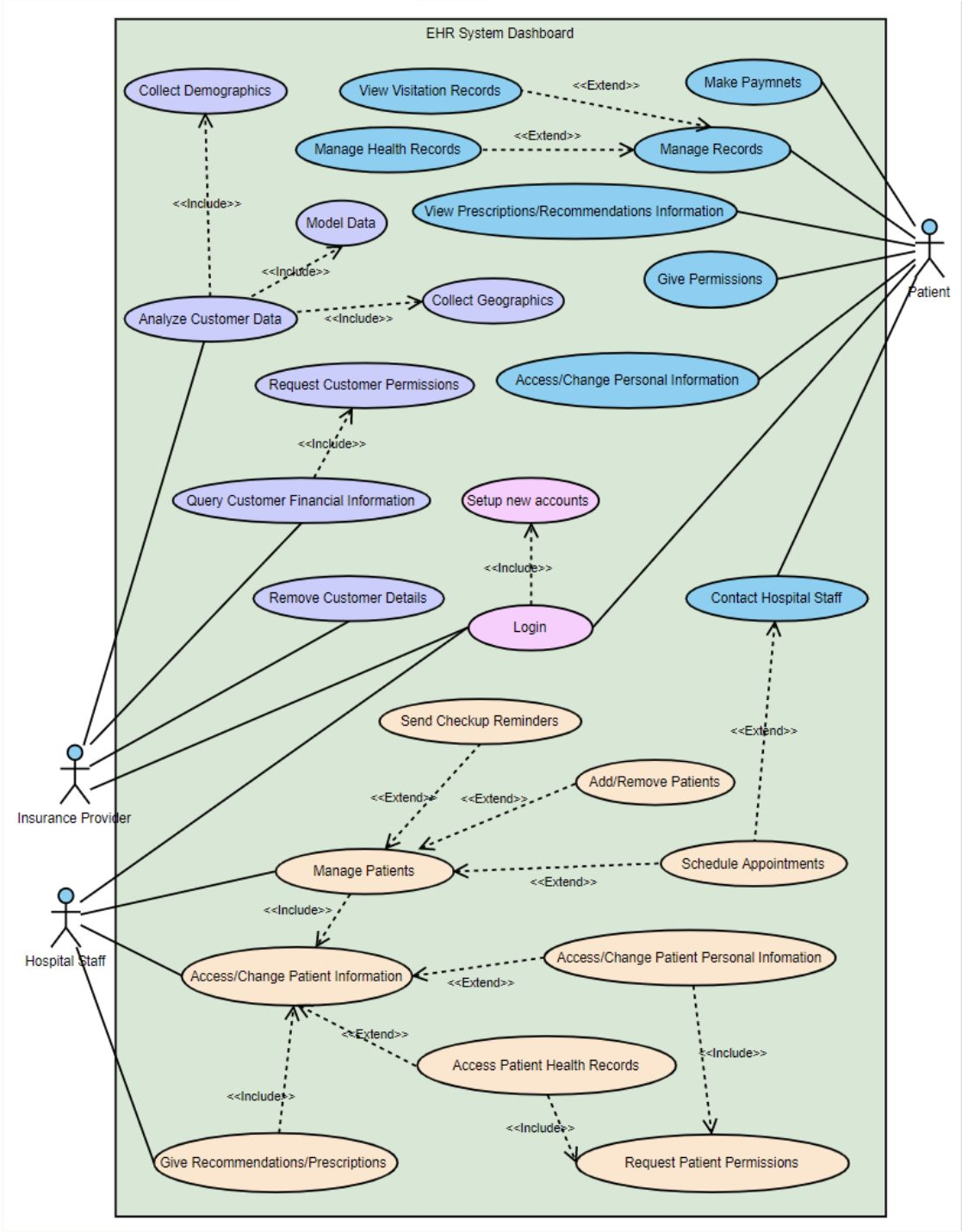
```
Re-medi
|
|-client(all the front end files)
|    |-public
|       |-favicon.ico
|       |-index.html
|    |-src
|       |-App.js
|       |-App.css
|       |-_tests_ (all the test files)
|       |-components (all the front end components)
|       |      |-Header
|       |      |-Sidebar
|       |      |-...
|       |-other folders/files on Redux reducers, utils, ...
|
|-config (database connection string)
|-models (all database schema)
|   |-User.js
|   |-Appointment.js
|   |-Request.js
|   |-...
|-routes/api
|    |-users.js (api file for our project)
|-validation (files handling user input validation)
|-package.json
|-README.md
|-server.js
```

-Figure 1: Project Architecture-

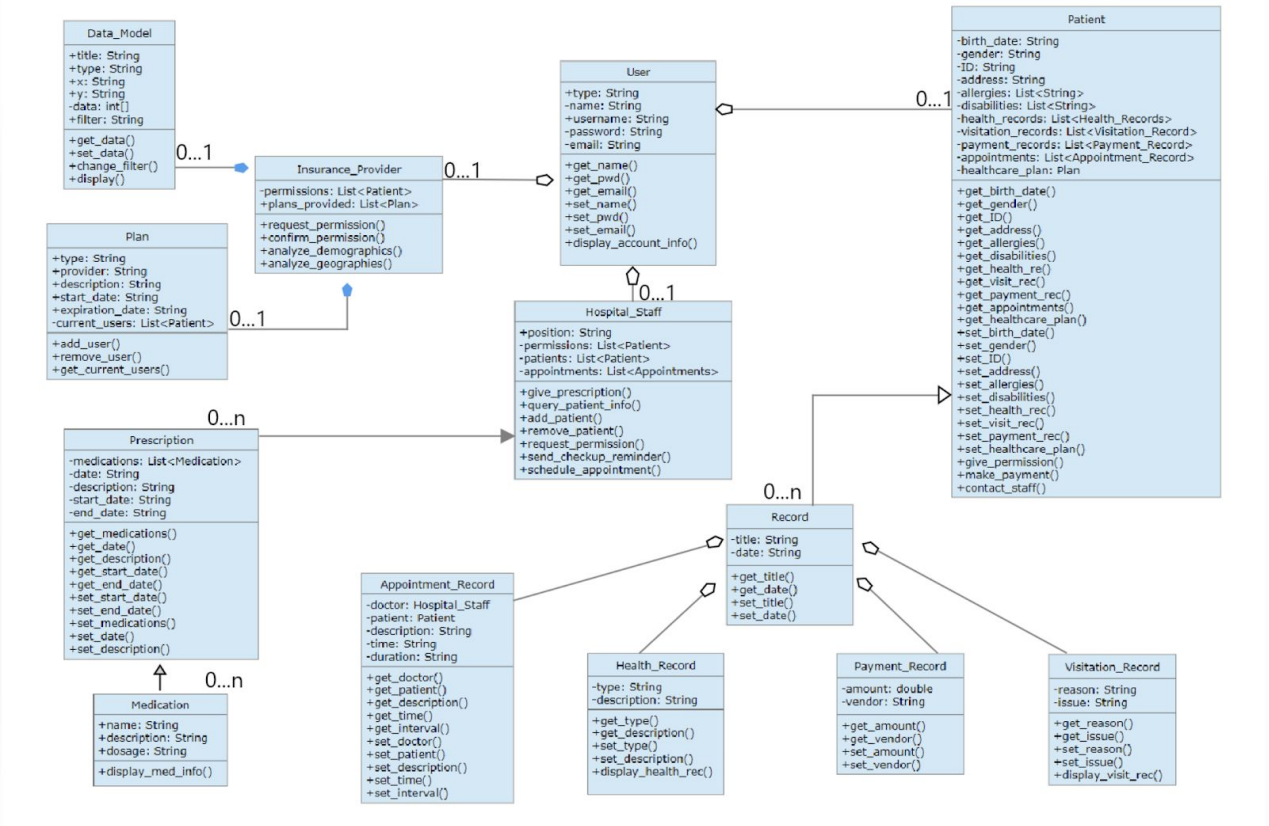
By selecting ExpressJS as our backend server framework, it provides us an easy server implementation and directs us towards developing using a combination of the client-server and MVC architecture styles.

II. UML Diagrams

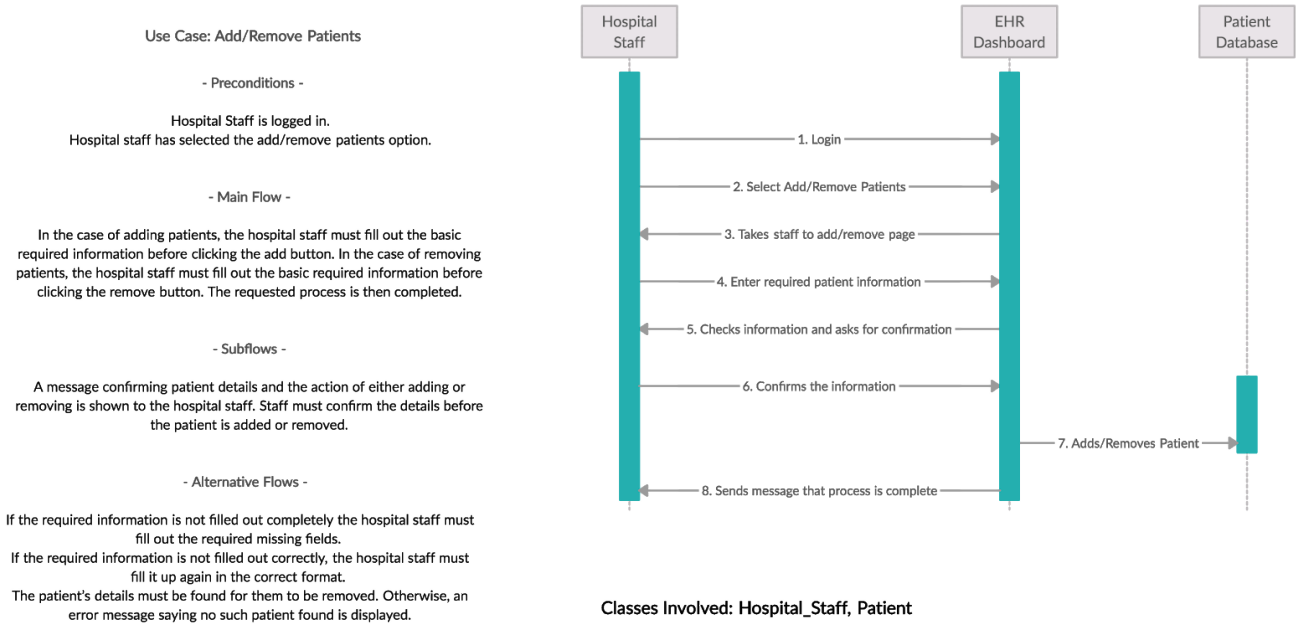
Please find below the Use-Case Diagram (fully merged all the Use-Case diagrams for the 3 Actors), Class Diagram for the Whole System and two Sequence Diagrams:



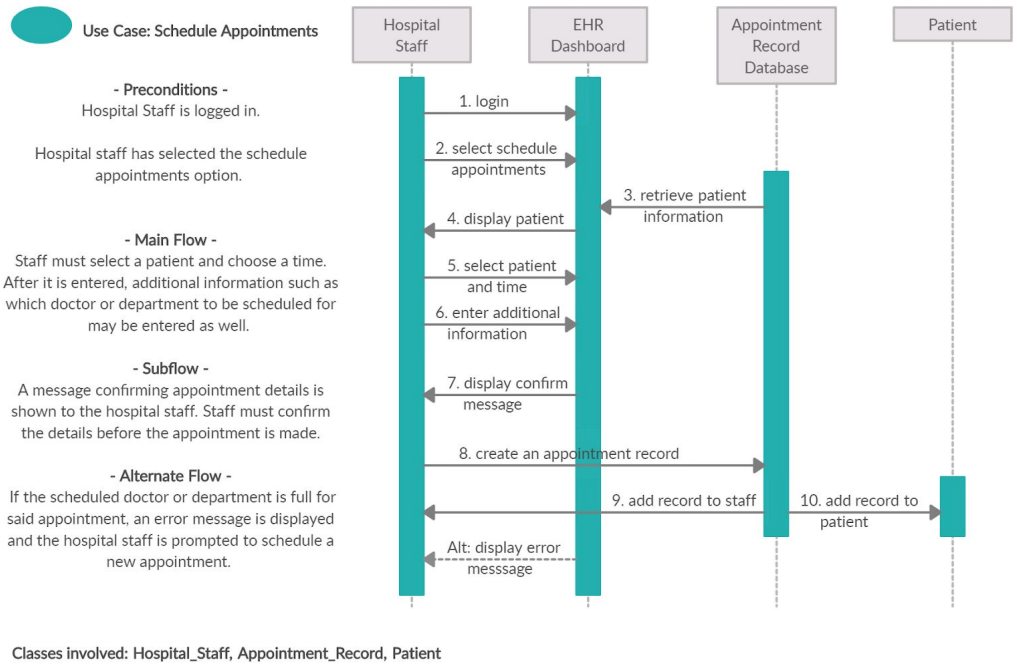
-Figure 2: Use case diagram-



-Figure 3: Class Diagram-



-Figure 4: Sequence Diagram-add/remove patients-



-Figure 5: Sequence Diagram-schedule appointments-

Note: You may have to zoom in on the image to read it. The image was too big and was difficult to compress for the use of submission. We can email the uncompressed version if required.

Reflections and Lessons Learned

Andrew Lee - Working with the latest web technologies in an XP manner was an extremely challenging, yet valuable experience. I learned so much regarding software engineering in a team setting, and will apply everything I have learned to the professional work setting.

Andrew Liu - Bootstrapping an app is an extremely useful tool in setting up foundations and implementing project features. Planning code structure and User Flow is necessary in order to avoid bottlenecking the team due to work-in-progress features.

Xin Xiang - During the development process, when some tests fail or a feature stops working, it is much easier to find the problem if we only changed one thing. If we have to refactor the code before adding a new feature, it is better to commit the refactoring first, then (in a new commit) add the new feature.

Zane Fadul - The hardest thing for me working front end is that there isn't much time to play around with designs. Given the fast paced environment, it's important to focus on the core experience and usability before making things pretty. That said, I truly valued working on a project to this scale; it's sure to help me after I graduate.