**COVID Tracker System**

**Software Architecture Document - SAD**



***Revision #4: 03 - Apr - 2022***

**Contents:**

* Introduction
* The Domain Model
* The Class Diagram
* The Component Diagram
* The User Use-Cases Diagrams
  + Sign-in Use-Case
  + Sign-up Use-Case
  + Sign-out Use-Case
* The Patient Use-Cases Diagrams
  + Update Health Status Use-Case
  + Contact Doctor Use-Case
  + Schedule Medical Appointment Use-Case
  + Show QR Code for Covid Information Use-Case
  + Check Contact Tracing Use-Case
* The Administrator Use-Cases Diagrams
  + Manage user accounts Use-Case
  + Assign Doctors to Patients Use-Case
* The Doctor Use-Cases Diagrams
  + Monitor Patient Health Status Use-Case
  + Arrange Patient Appointment Use-Case
  + Contact Patient Use-Case
* External Libraries
* Database

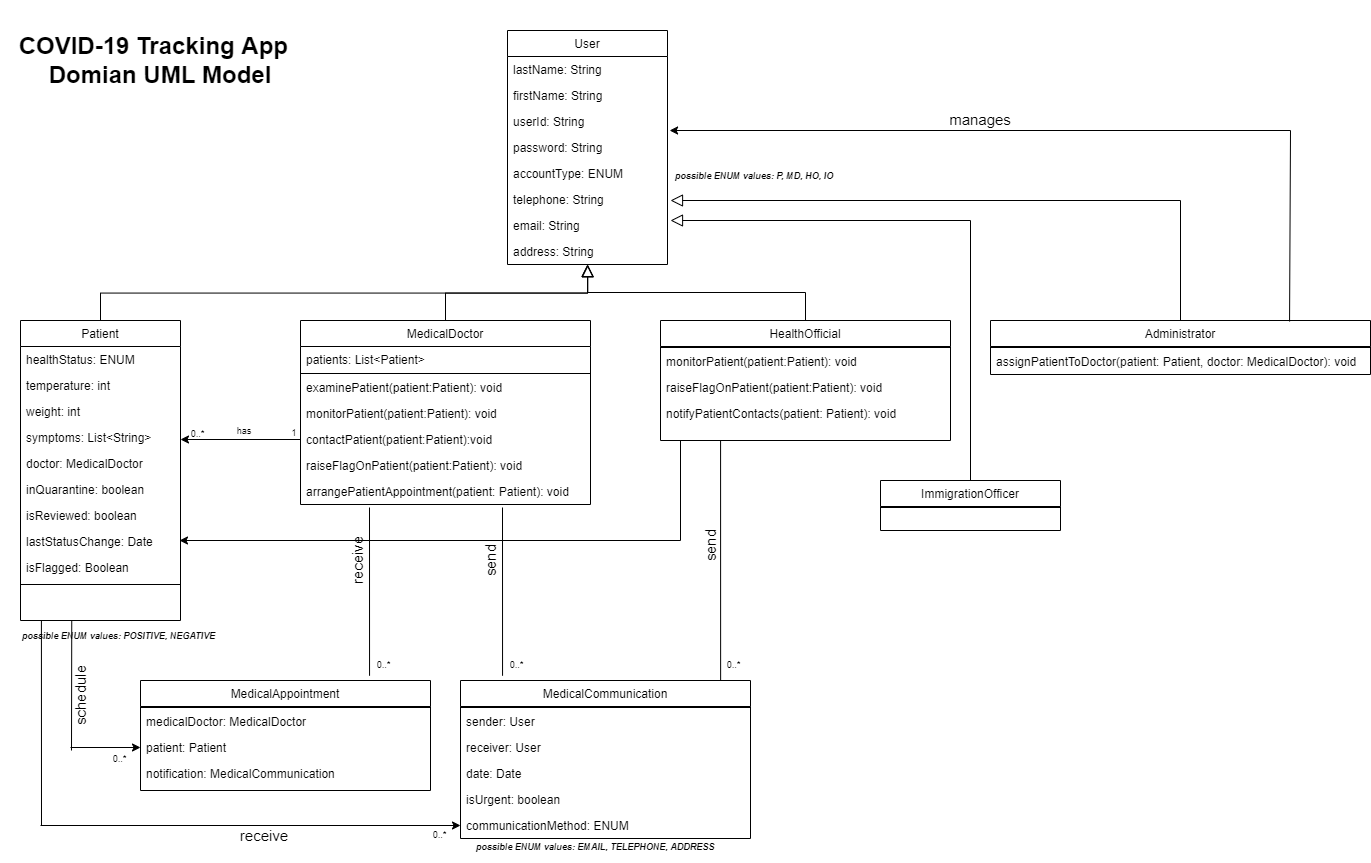
**Introduction:**

In this document we provide the UML designs of the system according to the requirements.

Please note that these diagrams are still in draft-mode and may be changed and modified during the design phase.

**The Domain Model:**

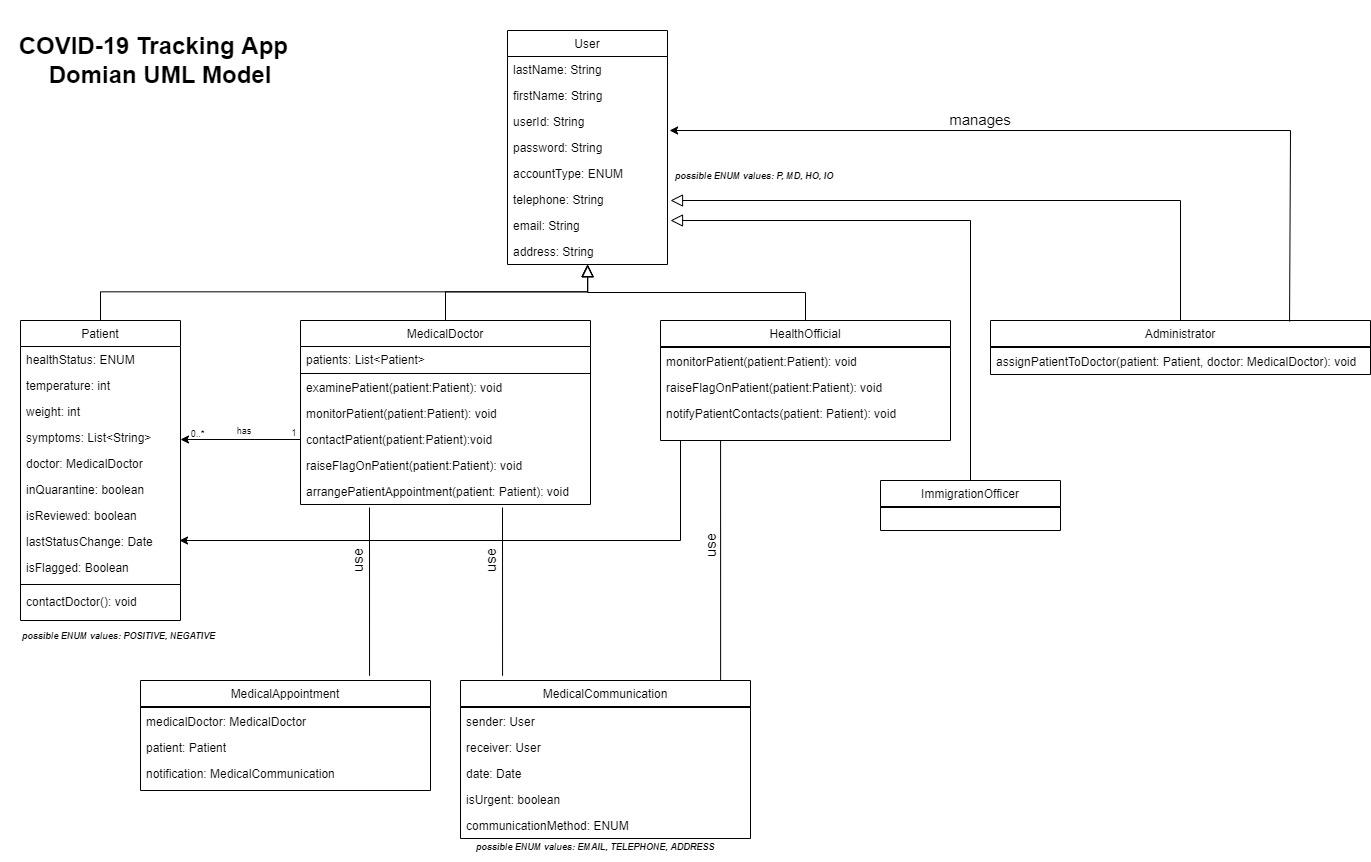
The following represent the Domain Model of the Covid Tracking System under design.



***Figure: Domain Model***

The main domain classes are:

* **User:** This represents a user's personal information. A user can be: Administrator, Patient, Medical Doctor, Health Official or an Immigration Officer.
* **Administrator:** This represents the state and functionalities needed to manage the system and perform some system functionalities.
* **Patient:** This represents the state and functionalities for the registered patients who require medical and health tracking services.
* **Medical Doctor:** This represents the state and functionalities of doctors who perform the health services and checkups and medical appointments with requesting patients.
* **Health Official:** This represents the state and functionalities performed by a health official to monitor the health status and communicate with patients.
* **Immigration Officer:** This represents the state and functionalities performed by an immigration officer to monitor the health status and track patients.
* **Medical Communication:** This represents the state and functionalities needed to satisfy any communication between system users; for appointments, status update requests, tracking etc.
* **Medical Appointment:** This represents the state and functionalities needed to book medical appointments between medical doctors and their patients.

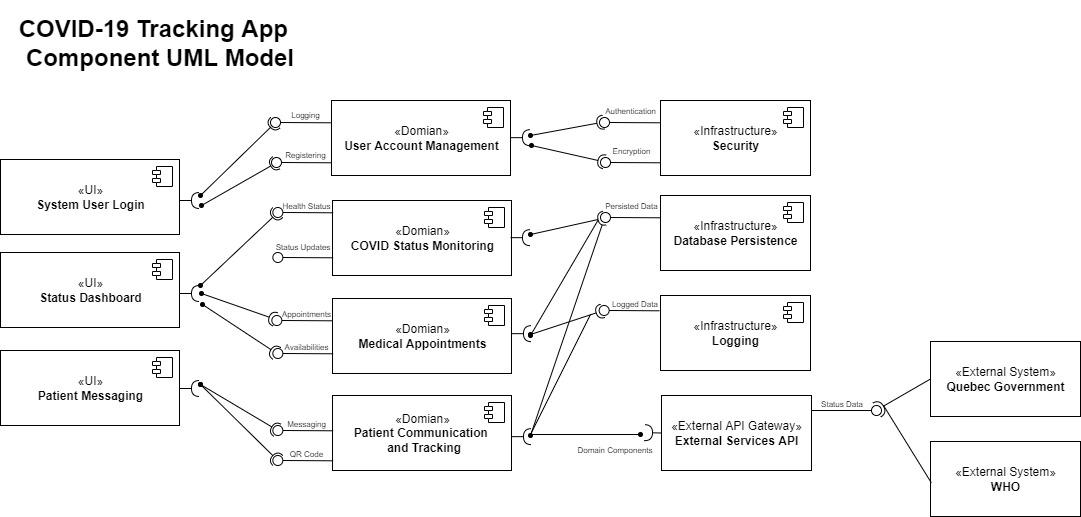


**The Class Diagram:**

***Figure: Class UML Diagram***

**The Component Diagram:**

The following represent the Component Diagram of the Covid Tracking System under design.



***Figure: Component UML Diagram***

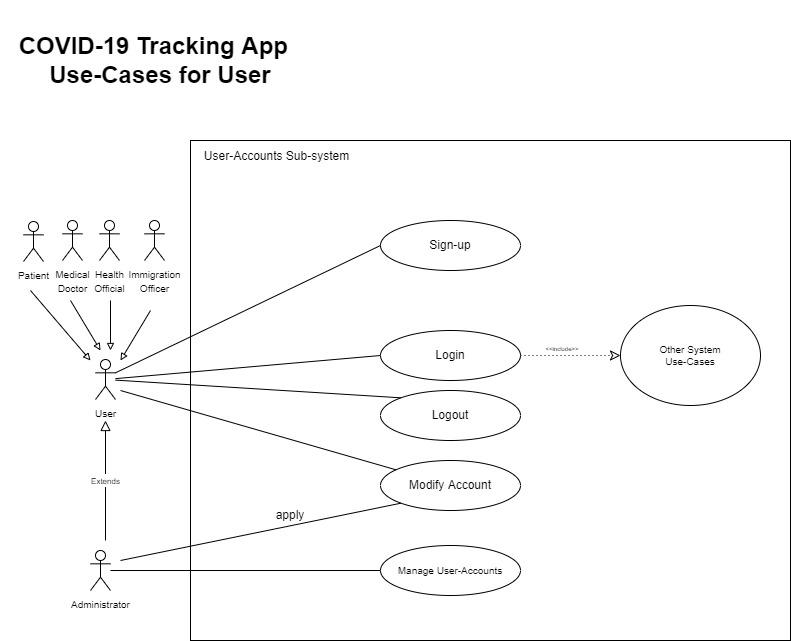
The main system components are:

* **System User Login:** This is a UI component for handling the front-end components needed for logging the system users into the system.
* **Status Dashboard:** This is a UI component for handling the front-end components needed to show information and graphs of the health status and appointment requests in the system.
* **Patient Messaging:** This is a UI component for handling the front-end components needed to handle messaging between the users; patients, doctors etc.
* **User Account Management:** This is a domain component for handling the various processes needed to manage the user-accounts; login, logout, signup, change password etc.
* **COVID Status Monitoring:** This is a domain component for handling the health status parameters collected from patients and needed to track and monitor health status.
* **Medical Appointments:** This is a domain component for handling appointments; schedule, cancel, modify, between patients and available doctors in the system.
* **Patient Communication and Tracking:** This is a domain component for handling the communications and messages between users, and for tracking health updates.
* **Security:** This is an infrastructure component needed to provide user authentication and encryption for system messaging and external queries.
* **Database Persistence:** This is an infrastructure component needed to persist all data collected and required by several functionalities in the system.
* **Logging:** This is an infrastructure component needed to perform low-level logging services; useful for debugging or system upgrade and performance analysis.
* **External Services API:** This is an infrastructure component needed to provide a gateway of services for external systems.
* **External Systems:** The Quebec Government and WHO are an example of external systems that might communicate with the system to query data from the system.

**The Use-Case Diagrams:**

The following diagrams represent the several Use-Case Diagrams of the Covid Tracking System under design.

**Use-Cases for User:**



***Figure: User Use-Cases UML Diagram***

***Use-Case:*** Users can sign-up for an account in the system,

**Actors:** A User in the system; this could be administrator, patient, medical doctor, health official or immigration officer.

**Pre-conditions:**

* User should not have an account previously in the system.

**Post-Conditions:**

* User is granted an account with access to sign-in and use other use-cases in the system depending on his Account-Type.

**Triggers:**

* This Sign-Up Use Case is triggered when a User clicks on the Sign-up button.

**Main Success Flow Scenario:** User clicks button, enters valid information. Then, information gets validated by Database and then granted access. Message: Success Signup is displayed.

**Alternative Flow Scenario:** User clicks button, enters invalid information. Then Database can not validate the data and proceeds to send error and Message: Unsuccessful Sign-up.

***Use-Case:*** System Users can login to the system into their accounts.

**Actors:** User in the system; this could be administrator, patient, medical doctor, health official or immigration officer.

**Pre-conditions:**

* User should have an account for Sign-in.

**Post-Conditions:**

* User is granted access to Home-Page Dashboard and able to access other use-cases in the system.

**Triggers:**

* The Sign-In Use Case is triggered when a User clicks on Sign-in button.

**Main Success Flow Scenario:** User clicks button, enters valid username and password information. Then, information gets validated by Database and then granted access. Message: Success Logging.

**Alternative Flow Scenario:** User clicks button, enters invalid information. Then Database can not validate the data and proceeds to send error and Message: Unsuccessful Logging.

***Use-Case:*** System Users can sign-out of their accounts.

**Actors:** User in the system; this could be administrator, patient, medical doctor, health official or immigration officer.

**Pre-conditions:**

* User should have an account for Sign-in.
* User should have already signed-in for Sign-out (Log-out).

**Post-Conditions:**

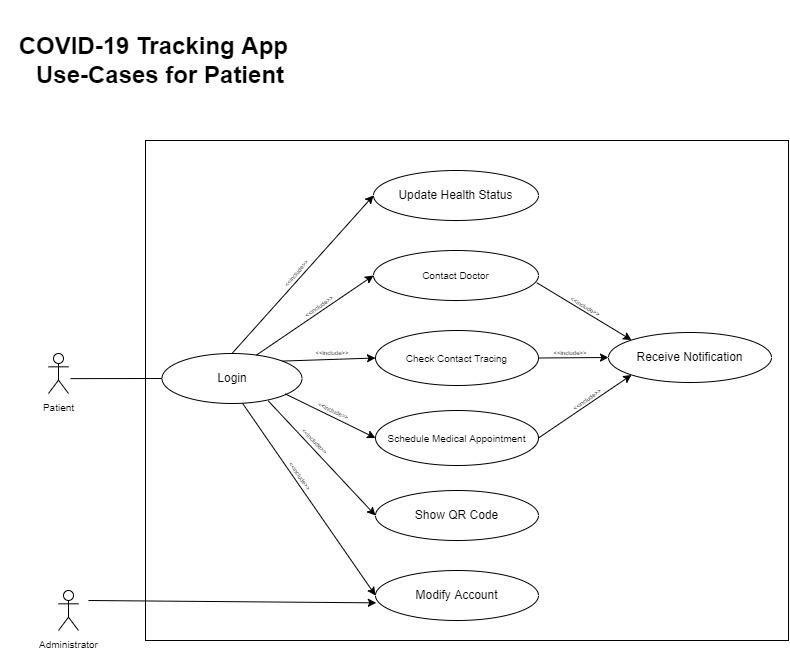
* User is disallowed from system services and routed back to Welcome-Page.

**Triggers:**

* This Sign-Out Use Case is triggered when a User clicks on Sign-out button.

**Main Success Flow Scenario:** User clicks button, Then, user gets invalidated and prevented from accessing the system services. User is routed to the welcome screen.

**Use-Cases for Patient:**



***Figure: Patient Use-Cases UML Diagram***

***Use-Case:*** Patients update their health status (temperature, weight, symptoms).

**Actors:** Patients as users of the system.

**Pre-conditions:**

* Patient should have an account and already signed in.

**Post-Conditions:**

* Patient has an entry of his/her health status, persisted in the database and can be retrieved.

**Triggers:**

* This Use Case is triggered when a Patient clicks on the Update Health Status button and fills in the form to be submitted and persisted.

**Main Success Flow Scenario:** Patient clicks button, enters valid information. Then, health information gets stored in the database and a message for Successful Update is displayed.

**Alternative Flow Scenario:** Patientclicks button, enters invalid information. Then Database can not accept the data and proceeds to send an error and a message for Unsuccessful Health Update.

***Use-Case:*** Patients can contact their assigned doctor regarding covid health updates or issues.

**Actors:** Patients as users of the system. Doctors as secondary actors.

**Pre-conditions:**

* Patient should have an account and already signed in.
* Patient has already been assigned a doctor (use-case for Administrator).

**Post-Conditions:**

* Patient has contacted the assigned doctor and a confirmation is sent with the date and details for both patient and doctor.

**Triggers:**

* This Use Case is triggered when a Patient clicks on the Contact Doctor button and fills in the request form (message body) to be sent to the doctor.

**Main Success Flow Scenario:** Patient clicks button, enters valid information. Then, the message is sent to the doctor who gets notified.

**Alternative Flow Scenario:** Patientclicks button, enters invalid information. Then a message won’t be sent to the doctor, Possible to invite the patient to resend or modify the message.

***Use-Case:*** Patients can schedule a Medical Appointment with their assigned doctor.

**Actors:** Patients as users of the system. Doctors as secondary actors.

**Pre-conditions:**

* Patient should have an account and already signed in.
* Patient has already been assigned a doctor (use-case for Administrator).

**Post-Conditions:**

* Patient has a valid confirmed appointment with his/her doctor. A possible notification or reminder of the arrival date of the appointment.

**Triggers:**

* This Use Case is triggered when a Patient clicks on the Schedule an Appointment button and fills in the form to be validated, submitted and persisted.

**Main Success Flow Scenario:** Patient clicks button, enters valid information. Then, appointment notification is sent to the assigned doctor to be confirmed.

**Alternative Flow Scenario:** Patientclicks button, enters invalid information. Then appointment is not scheduled with errors displayed explaining reasons for re-attempting possibility.

***Use-Case:*** Patients can contact their assigned doctor regarding covid health updates or issues.

**Actors:** Patients as users of the system. Doctors as secondary actors.

**Pre-conditions:**

* Patient should have an account and already signed in.
* Patient has already been assigned a doctor (use-case for Administrator).

**Post-Conditions:**

* Patient has contacted the assigned doctor and a confirmation is sent with the date and details for both patient and doctor.

**Triggers:**

* This Use Case is triggered when a Patient clicks on the Contact Doctor button and fills in the request form (message body) to be sent to the doctor.

**Main Success Flow Scenario:** Patient clicks button, enters valid information. Then, the message is sent to the doctor who gets notified.

**Alternative Flow Scenario:** Patientclicks button, enters invalid information. Then a message won’t be sent to the doctor, Possible to invite the patient to resend or modify the message.

***Use-Case:*** Patients can show QR Code for Covid Information.

**Actors:** Patients as users of the system.

**Pre-conditions:**

* Patient should have an account and already signed in.
* Patient has already updated health status at least once.

**Post-Conditions:**

* Patient is able to show a QR Code generated from the information and details about the recent updates of his/her covid health status.

**Triggers:**

* This Use Case is triggered when a Patient clicks on the Show QR-Code button which generates the appropriate code from the stored health information.

**Main Success Flow Scenario:** Patient clicks button, Then, the QR code is generated and displayed and scannable by QR Scanners.

**Alternative Flow Scenario:** Patientclicks button, but code is not generated due to missing health status information. Patient can contact Administrator if problem persists.

***Use-Case:*** Patients can check for covid contact tracing.

**Actors:** Patients as users of the system.

**Pre-conditions:**

* Patient should have an account and already signed in.
* Patient has already updated location status at least once (GPS activated).

**Post-Conditions:**

* Patients are able to receive notifications regarding the possibility of being in contact with a covid infected patient. If contact-tracing is detected, then patients are invited to self-isolate and update health status.

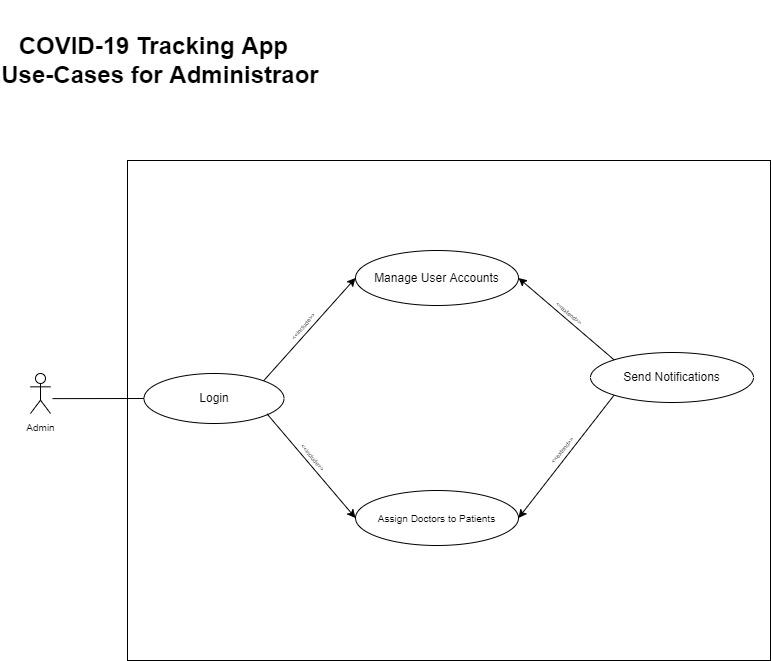
**Triggers:**

* This Use Case is triggered when a Patient clicks on the Location Check-in button which accesses the GPS information to store location data and time about the patient’s presence.

**Main Success Flow Scenario:** Patient clicks button, Then, the GPS information is saved with the current time-stamp. If the system detects any possibility of covid patient contact-tracing, then a notification is sent to patient inviting for self-isolation.

**Alternative Flow Scenario:** Patientclicks button, but GPS is not available. System can not update location. Notifications can still be sent relying on old location data only.

**Use-Cases for Administrator:**



***Figure: Administrator Use-Cases UML Diagram***

***Use-Case:*** Administrators can manage user accounts in the system.

**Actors:** Administrators as users of the system.

**Pre-conditions:**

* Administrator has received a change request to modify information of an existing account.

**Post-Conditions:**

* Administrator accepts the validated change and confirms changes with sending notification of change to requester.

**Triggers:**

* This Use Case is triggered when a User clicks on the Modify Account button and fills in the form to be validated then accepted or refused.

**Main Success Flow Scenario:** Administrator receives an account-info request change, then validates information. The new changes are persisted in the database with a notification sent to the requester.

**Alternative Flow Scenario:** Administrator does not accept the modified information and refuses to persist the data. Possible to send a notification with information about refusal update rejection reasons to the requestor.

***Use-Case:*** Administrators can assign doctors to patients.

**Actors:** Administrators as users of the system.

**Pre-conditions:**

* Patients and Doctors exist in the system.
* Administrator detects patients without any assigned doctor.
* Administrator receives a request from doctor to re-assign a new doctor for patients.

**Post-Conditions:**

* Patients are assigned with doctors. A doctor can be assigned to many patients. Patient is informed about the assigned doctor. Doctor is informed about patients under his/her responsibility.
* The Number of Patients counter is increased for the assigned doctor.

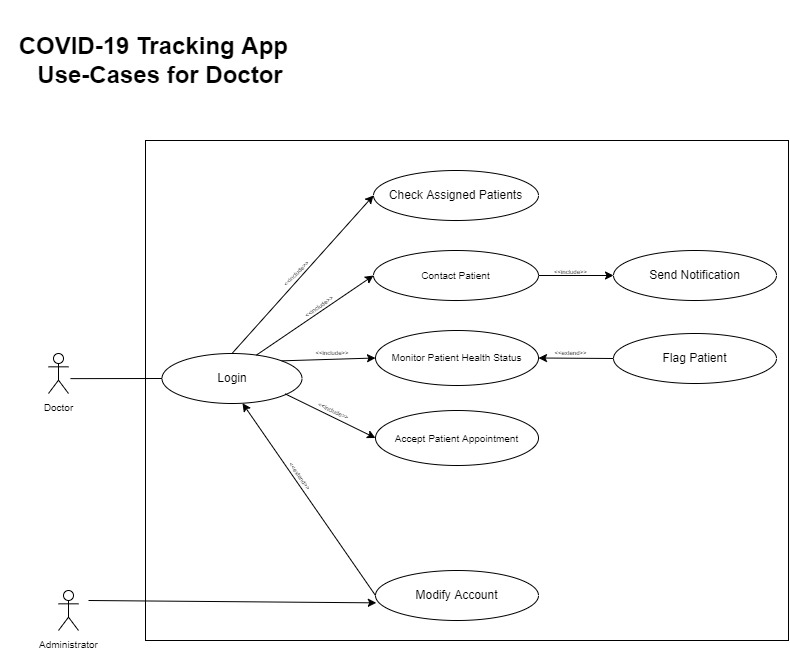
**Triggers:**

* This Use Case is triggered when an Administrator detects a patient with no doctor, or receives a request to change the assigned doctor for a specific patient.

**Main Success Flow Scenario:** Administrator receives request with details about patient and doctor. Then a doctor gets assigned to a patient if the availability allows it (Number of patients per doctor must be balanced in the system).

**Alternative Flow Scenario:** Administrator refuses to assign a doctor to a patient if the doctor is already assigned with many patients. Or if the doctor is not active.

**Use-Cases for Doctor:**



***Use-Case:*** Doctors can monitor the health status of patients.

**Actors:** Doctors as users of the system. Patient is a secondary actor being monitored.

**Pre-conditions:**

* Doctor is assigned to a specific Patient.
* A Patient is been assigned to a doctor and updated the health status.

**Post-Conditions:**

* Doctor can view status history of patient.

**Triggers:**

* This Use Case is triggered when a Doctor clicks on the Monitor Patient button. and fills in the form to be validated then accepted or refused.

**Main Success Flow Scenario:** Doctor views the status information of a chosen patient to take further possible actions (Flag a patient, Contact a Patient).

**Alternative Flow Scenario:** Doctor can not view a Patient’s status, either because of missing status information, or because of trying to view a patient not assigned to that doctor.

***Use-Case:*** Doctors can arrange an appointment with a patient.

**Actors:** Doctors as users of the system. Patient is a secondary actor for the appointment.

**Pre-conditions:**

* Doctor is assigned to a specific Patient.
* A Patient is been assigned to a doctor and requested an appointment with the doctor.

**Post-Conditions:**

* An appointment is scheduled between a Doctor and the requesting patient.

**Triggers:**

* This Use Case is triggered when a Doctor clicks on the Appointment notification button. Then, see the appointment details sent by the patient.

**Main Success Flow Scenario:** Doctor views the appointment information of a requesting patient to take further possible actions (accept the appointment, refuse the appointment).

**Alternative Flow Scenario:** Doctor does not receive any appointments, and the notifications section is empty and clear; pending for an appointment request by a patient.

***Use-Case:*** Doctors can contact a patient.

**Actors:** Doctors as users of the system. Patient is a secondary actor being contacted by his/her doctor.

**Pre-conditions:**

* Doctor is assigned to a specific Patient.
* Patient has filled in the contact information section of profile.

**Post-Conditions:**

* A Patient receives a notification from the assigned doctor with information about a medical request (example: asking a patient to self-isolate).

**Triggers:**

* This Use Case is triggered when a Doctor clicks on the Contact Patient button.

**Main Success Flow Scenario:** Doctor selects a patient, then requests to contact the patient by filling a message box with details. Patient receives a notification message showing the message details.

**External Libraries:**

* **Ionic:**

Ionic is a framework for building mobile apps with web technologies.

Ionic allows building cross-platform applications from the same codebase.

*FROM Code (HTML, JS, CSS) TO Web-Application OR Mobile-Application*

Ionic can be used with front-end web frameworks like: Angular or React.

It can help our team to generate any of the following options:

1. Progressive Web App (PWA)

2. IOS Mobile App

3. Android Mobile App

4. Electron (Desktop) App

* **Angular or React:**

They are examples for client-side JavaScript Frameworks for developing (reactive Java-script driven) web applications.

They use TypeScript (a superset of JavaScript) which is used to develop web components that run under Ionic to generate reactive applications that can be hosted as Web Application or Mobile Application, which is the case in our requirements.

In this project, we will be using the Angular framework.

* **Node:**

Node.JS is a back-end JavaScript runtime environment. It is a required tool during our development environment setup. Furthermore, it is also a dependency package manager used by Ionic and Angular to install packages needed for libraries and third-party web components.

**Libraries Advantages Discussion:**

The reason behind our choice of libraries is that we can develop one code-base and use it via some Ionic tools “Capacitor” to build the project in the form of Web Application, or a native Mobile App that runs on both Android and IOS platforms.

Another reason behind our choice is that Ionic is supported with lots of smart web components that have the ability to communicate with native hardware kernels of mobile systems. This will be helpful when we start developing functionalities that need GPS or any other hardware support.

**Database:**

For the requested Covid-Tracker system, we need a database to persist all needed data.

We started implementing our datastore in ***MySQL as a DBMS*** Database Management System. We decided to deploy our data-store in a cloud infrastructure to support some non-functional requirements including:

**1. Availability:** The database will be a central data-store that is accessible by our dev-team.

**2. Scalability:** The database can be replicated on several servers to accommodate a high number of system users.

**3. Developing and Testing:** It can be our centralised and consistent datastore to work on during development.

We configured the firewall rules to allow our IPs to access it during the development.

Here are the details of the database:

**Server (Host) Name:** [concordia-database-server.mysql.database.azure.com](http://concordia-database-server.mysql.database.azure.com/)

**Port:**  3306

**Database Access Credentials:**

**Admin username:** databaseadmin@concordia-database-server

**Password:** Only-Shared-Among-Developers

**Database (Schema) Name:** CovidTrackingDatabase

**TypeORM:**

TypeORM is a TypeScript ORM (Object-Relational Mapper) library that makes it easy to link our TypeScript application up to a relational database database.

TypeORM supports MySQL, SQlite, Postgres, MS SQL Server, and a host of other traditional options.

TypeORM is can run in NodeJS, Browser, Cordova, PhoneGap, Ionic, React Native, NativeScript, Expo, and Electron platforms and can be used with TypeScript and JavaScript (ES5, ES6, ES7, ES8).

TypeORM Installation using Node: **npm install typeorm**

To connect to the MySQL Database hosted in Azure:

MySQL driver installation using Node: **npm install mysql2**

To connect to an Embedded SQLite Database:

SQLite driver installation using Node: **npm install sqlite3**

We created an Angular Service class that will be responsible for CRUD operations with the Database. @SourceCode: folder: *database-services*