Design Rationale

For our application our group has decided to implement an MVC architecture. The reason for this is because MVC by virtue is designed to follow SOLID design principles. We have decided to use Spring MVC to make our app because Spring MVC is a framework used to build MVC apps using JAVA.

MVC apps follow the single responsibility principle by separating the responsibilities of the app into 3 distinct parts, The model, the view and the controller. Our app further follows the single responsibility principle by dividing the controller into sub controllers each with a corresponding view to minimize coupling in the system.

The app adheres to the Open/close principle by the use of interfaces where multiple implementations can be done. The patientService interface in the app is one such instance of this. The service can be implemented in a different way in the future if required without having to change the current implementation of this service.

For the scope of this assignment we have not had to use Liskov's substitution principle or the interface segregation principle.

Our app adheres to the dependency inversion principle by using dependency injection for inversion of control. Spring uses an IoC container to achieve this. The spring container instantiates ,configures and manages the lifecycle of objects that are required for the app to function.

When refreshing patient data in the cholesterol monitor we have decided to poll the fhir server only once every N seconds and get the details of all patients of the practitioner. We do this instead of polling the server for each patient in the monitor because that requires multiple get requests from the server and causes unnecessary traffic to the server. This causes the app to have a slight delay when processing the server data but it is a necessary trade-off to reduce the amount of requests to the server.