**Critical reflection report – Group 6**

**Underlying software structure –**

During the development of the application we decided to make a few changes to the underlying software structure compared to our original designs. Some aren’t too large, and some might be a little larger than others.

One main changed we made to the underlying system was how the front end communicates with the backend. Originally the staff and the patient side of the system would have their own backend servers which would both communicate with each other when new data reaches the patient server. Instead we opted to have 1 large server that both the patient side and the staff side of the application could communicate to. We did this because we felt during the development that hosting 2 separate servers that essentially do the same thing would be pointless and would decrease the speed in which data is able to reach the staff side of the application if it has to go through a ‘middle man’. This way when data is sent from the patient application it will be sent directly to the staff side of the application through the server creating a much faster system. Also, adding security would be a lot easier with one server than it would with 2 separate servers with separate permissions.

Another change we made to the software structure is how data is sent from the watch. Originally, we had the idea that it would send data from the watch to the app and then data would be sent every so often from the app to the server and onto the staff side. We decided against this for 2 reasons. First off, we would again need the patient side server so that the data from the watch is sent to the app using an API. Secondly it would mean this data would be sent to the patient server to be sent to the patient app only to be resent back to the server a few moments later to send it to the staff side of the application. This is clearly not the most efficient way of doing this so now we have connected the watch to the main server and every so often the data from the watch will be sent to the server. This data will then go to both the patient app, so the user is able to check on there latest records for these items of data, and to the staff side as well, so the medical staff can get a almost live look at the data the server is receiving from the watch.

**Changes to user interaction –**

While developing this application we changed a few ways the user can interact with the UI of the application as well. Some of these reasons are simpler than others and some of these from the original designs were removed.

The first major we made to how the user interacts with the system is on the staff side of the application. Originally in our designs when a medical professional was looking for a patients specific data they would be able to click on the whole row for that patient and it would then load a model so the staff member could looking through all the data then click away to hide said data allowing them to look for other patients. This is no longer the case. We decided to add a column to the patient row that was called details which can clearly de identified as a link. This will then instead load a different page displaying all patient information as well as all historical data from the patient application. We decided to do this because first off if we made it so the staff member had to click the row there is no clear indication that, that is what you are supposed to do due to the row not having an link like features to it so it could confuse the staff member like it did the users and the filters. Also, we decided to open it up in another tab on the browser instead of a model because this will allow the medical staff to open up more than 1 patients data at once if that day they are looking at more than 1 patients details and doesn’t want to have to keep reusing the filters.

Another change to the user interaction is the blood pressure test on the patient side of the application. Looking at our original designs you can see that at first the blood pressure test was supposed to be a button on the patients app that, when pressed, would send a message to the watch to take the users blood pressure. This is no longer the case and now the only user interaction with the blood pressure is the ability to see the latest results on the patient app. The blood pressure test will now occur twice a day at the same given time. This is simulated on the watch application by adding a button below the UI that activates the blood pressure test. We thought this was the best cause of action because all the other items on the watch are automated so leaving one to be manual seemed unnecessary. Also, the original idea was to set a reminder for the patient to take the blood pressure test every so often but there is always room for error and even after the reminder the still forget to do it the results won’t be as clear and accurate as they would be if the patient didn’t have to do anything.

**Reflections on software project management –**

The software management approach we used was agile methodology , but it wasn’t as successful using this method as we would have liked as our team decreased in size by 1 so we had 2 people trying to cover a lot of the work.

We started out using the agile methodology approach by Alek working on the backend of the application getting the API ready and setting up connections so the server would run and I (Steven) work on the frontend creating the pages to make sure they all looked like the designs from the wireframes. We were able to stick to this method for a while working on a little bit at a time until we eventually finish the project a few days before the deadline so we could work on the paperwork and additional feature. This was not the case when it came around to it.

We had an issue with some of the security permissions on the backend which stunted our production of the final version of the application. To make sure we finished this on time while one of us was working on fixing the application and finishing the development the other was getting ready for the user evaluation with another group. This way once it was fixed the evaluation could get under way almost immediately to make sure everything was finished on time. This would give us just enough time to complete the final report as well as create our individual components for task 3.