**Refactoring Report**

While creating our Neighbourhood Doctor’s mobile application, we were required to refactor our code on many cases to match the needs of the client on a few occasions. Each instance that a major code refactoring was necessary will be discussed below.

Our first experience with refactoring was the initial request to switch from a JavaScript/HTML website to a Flutter/Dart web app with a spring boot framework. As most of our team was experienced with JavaScript/HTML and none of us had experience with Flutter/Dart before, it was quite a challenge to make the switch. This meant that our team had a much slower sprint 1 than we had hoped for, largely due to needing to learn how to use Flutter/Dart. While we initially lost time in sprint 1 by having to convert to Flutter/Dart, by sprint 2 we had figured out how to use Flutter/Dart efficiently and it made our project much easier to complete and allowed us to make up for the lost time in sprint 1.

Our second major challenge with refactoring our code was the switch from web app to mobile app. Towards the end of sprint 1, our client decided that they would prefer a mobile app over a web app. This refactoring was easy to handle since we hadn’t begun much development on the front end yet. A small amount of code needed to be changed and a new environment to test our app on a mobile was set up. Flutter was extremely good at handling the switch from web to mobile which further decreased the amount of time we had to spend changing our code to fit the new format.

Another major refactoring was our switch from hosting our backend through Docker. The switch to Dockerising our backend wasn’t as hard as it was time consuming. There were many resources that we had to work with in the switch however different error messages would pop up that needed to be resolved each time. The switch to Docker caused lots of time in sprint 2 to be lost. Since each step of Dockerising had to be done in sequence, it was hard for each member to help with it. As such only one of our members (Chrisio) could work on Dockerising our web app and only once he was finished setting it up were we able to convert on our GitHub repository and then our local machines. Eventually this backend would eventually be pushed to Heroku Cloud to be deployed. This switch took much less time than Dockerising did and required minimal effort and time to be spent.

We also needed to refactor our code to include JWTs (JSON Web Tokens) to keep our code secure. This change was also more time consuming than it was difficult because it required that we change the parameters for almost all our functions in our front end to include a token instead of an id and then needed to retrieve that id each time using the token we were provided. This meant that most of our frontend code was refactored to include the tokens and new functions in our backend were needed to retrieve the id for a given user based on this token. This caused us to lose a bit of time in sprint 2 however once Alex had set up the JWTs on the backend, Marco, Rylan and Ethan were able to convert their functions quite easily.

Another major refactoring was converting from a monolith structure to a microservice structure. Initially we had decided to use a monolith structure however as we continued working on the project, we decided that a microservice structure better suited our needs. This refactoring meant splitting our controller, repository and services into multiple controllers, repositories, and services, one for each class. This move, while time consuming made it much easier to finish our product since it meant that all of our code associated with an object type was contained in the corresponding controller, service or repo. While some time was spent on this change, we believe the time saved by the change has made up for it.

A minor code refactoring that needed to be done was some changes of back end logic. As we were developing the front end, we realised that some of the backend code gave a response that was in a format that couldn’t be used efficiently. As such, we decided to change our backend code so that it was able to be used by the front end. This refactoring usually was quite simple so only minor amounts of time were lost.