Momodou Bah C0559 Report

For the first assignment, I came up with 2 user stories. One of them was to enable two-factor-authentication through email and the other was to allow the option for the user to choose a doctor they were familiar with, possibly through a filtering or search component. After our first meeting, we shared our ideas and looked to improve them where possible. My contribution with others was to help develop ideas and help to keep them consistent with the requirements of the project. There were examples of some user stories that did not fully meet the brief. For example, they would state a particular feature that needed to be implemented but would not describe the purpose of these features. I also found this task to be very useful as it helped us to ask questions that we would not have asked without the task. For example, one question we asked was 'how many times can a user change their doctor?'. This question, yet simple, is very effective because it means that once we got to the stage when we were allowing the user to select their doctor, we were prepared to come up with the right solution. We were also very firm on the idea of two-factor authentication as it allowed us to link the security areas of authentication and authorisation. This is important as the data stored on health systems can be sensitive and two-factor authentication can be very helpful to keep it secure. We had a sight bottleneck in the early stages of this task as one of our team members was 'inactive'. This was frustrating at first but after letting the supervisors know about the problem, we were able to progress swiftly.

For the first Sprint, I wrote the documentation for how we would use Git, the structure for our project as well as adding all relevant parties to the project. I also designed the Login and Registration components and built the first template for the 'home panel' of our application. Many lessons were learnt including working with Git and designing and implementing interfaces. All the way through, I used the Git Bash command line interface to make commits; push, merge and pull files; I was also switching between the master and my personal branch. Although my IDE provided a well-built interface, dealing with collisions when merging was difficult at times. I also had to learn to pull the latest version of our program every time I wanted to contribute as pushing to the master branch for example after making changes was not allowed due to conflicts. Working with Java Swing was challenging as this was something I had to re-learn, but Java documentations online helped me to learn the basic concepts. During our sprints, we discussed how we wanted to layout our project folder. We agreed that everybody in the team would work on a specific section of the project, each sprint and there would be times when we would need to work in the master branch – for example when developing the 'home-panel' which was integral to how the system worked. When building the Login and Registration components, I designed a user interface first before producing the component on Java Swing. This was helpful as it allowed me to have a clear vision of what I was making before it would be made. One problem that came up was the deprecation of the getText() method in the JPasswordField due to security concerns. The problem with this was that my IDE was complaining that this method was deprecated. My team mates on the other hand did not have this issue so whilst I was getting errors when loading the program, the rest of the team were not. We solved this issue by using the getPassword() method. Instead of returning a string, this return a 'char' array instead and so the method '' was introduced to concatenate the chars into one single string. To develop this component, we could have made use of encryption methods such as a Caesar cipher which would have made our system more secure.

For the second assigned I also designed and implemented the ability for a user to set their doctor whether that be to create a new one or to change their doctor; this includes the SQL queries and adding doctors to our databases. In this sprint, I was introduced to the SQLite software and started working with that. I found the set-up process on my machine was difficult, but a member of my team helped me to get through this. My weak understanding with java swift prevailed at this stage as I struggled with managing the layout of the doctor frame. The idea was that a scroll view would be available. Although this was not necessarily the case, I found videos on You-tube and other resources to be useful in finding the optimal solution at the time. However, what I found most difficult in this part of the assignment was handling concurrent components. For example, when on the 'home-screen' and clicking the 'select a doctor' button, there were 3 major challenges. The first one was making sure that the doctor class could return the data back to the user, the second was making sure the results would update the home panel to reflect the changes made, and the third was ensuring the user would not click on the 'select a doctor' button more than once at a time. The first one was dealt by having a method which could be accessed from the parent class which would track the change and return the value when called. The second was to repaint the 'home-screen' and fetch the changes made in the database. The third was dealt by disabling the 'home-screen' whilst another component was in use. I struggled with the implementation of the SQL queries. Whilst working with it, we used the SQLite Software to view and manipulate the database. One challenge, that I later found out about, was that the database would lock up to prevent concurrent writes from out application and the SQLite application. This was deeply frustrating as errors would pop up at irregular intervals and a lot of time was wasted trying to identify why.

For the third Sprint, I implemented the behind-the-scenes functionality for Registering a user. The SQL queries were already set up for registration, I just had to link it to the component itself. I then refreshed the styling to ensure the interface was presentable. I built the Booking component which included creating, viewing, and modifying bookings alongside the functionality aspects; this also includes the SQL setup. After this, I connected those components to the home screen as well where they would be made available for use. This was a very challenging sprint as we had another team member who was 'inactive'. This meant that it was difficult to focus on other components as these components had to be prioritised. What I learnt in this stage was how valuable communication is. Had there been better communication between myself and some members of my team, I would have been able to track the progress of other components to ensure that everything was on-track for the given time frame and that there would not have be an overload of components to complete. The hardest part in this sprint was arguably the development of the calendar. This required extensive research on how to develop and get correct. There were many minor parts that had to be dealt with, such as what happens when the user chooses a date that is before the current date, dealing with the user clicking an empty box, interaction between the time and the time components. Most of these problems were solved by making the submit button active only when the correct input had been given. Alongside this were error messages which would inform the user of why the submit button was inactive as they were using the system. This was an effective solution as it allowed for a good user experience that was not being restricted by pop up boxes every-time something went wrong. When dealing with the format of the time and date, this was dealt by splitting them into two separate components. Once more, had the user entered in an incorrect time or a time that was out of bounds, error messages would pop up. I better way to implement this however would have been to set the increments to every 30 minutes rather than allowing the user to

pick any time of their choice. This would also be more practical and representative in the real world. There was also a case where methods were running twice which took me a long time to find a solution for. This happened in the 'Action-Event-Listener' for the calendar component and 'try-catch' statements were used to combat these failures. Another challenge that came up for this sprint was ensuring that a booking could be removed successfully. What made this stage tough was the formatting of our query. The format used in the SQL database was 'day/month/year'. The challenge came as the prepareStatement of the Driver manager in SQLite did not allow us to format the equerry in the form: 'select … where date=d/m/y' but instead the format of 'select … where date="d/m/y"'. When writing these queries it is important to note that the 'd/m/y' would need to be replaced with a question mark (?) and the prepareStatement would set the values we wanted in that position. After different methods were tried out, I came up with the solution of a 'manipulate' method. This would reconstruct the original query every time the method was called and set the date to the date that being passed in. The way it would do this was to look for the value of 'a' which was predefined in our string and replace it with the date. It would then leave the rest of query as it was, so the question marks for the other columns were left in place. After this, the reconstructed would be passed in to the prepareStatement and, in our case, the ClientID and DoctorID would be passed in as the remaining parameters to delete a booking.

As a group, I believe we did well in terms of planning and layout our ideas. This is shown through the in-depth documentations that we have produced throughout the sprints. Where we have lacked however is in our communication. In our case, we had to deal with some issues outside of our control such as deadlines and personal issues. Some members of our group struggled to communicate their thoughts clearly which meant that there were times we things were left late which compromised the quality of what we produced. We could have improved on this checking up on each other outside of the sprints as well as making more regular commits. By making more commits it shows level of engagement and when someone was not making as many commits as they should be, then it would be easy to identify who or what part of the project needed more support. I struggled with this throughout the latter stages of the project. This was because there was a lot to be done and, in my haste to get the components right, I compromised on the number of commits that were being made. The timing of our sprints could have been drastically improved. Although we got enough done, we did them at inconvenient times. Sometimes members would state that we should do a meeting 'later today' but that means different things for different people. Consequently, in future, starting with myself - I have every intention to be very specific with the timings and make sure that this mindset gets echoed to everyone on the team. The main downside to this was that we had members who simply did not turn up at times. Sometimes, in the case where members came up with excusable reasons to not join, we should have made more of an effort to communicate these ideas to them. Although we provided messages on the chat, this simply was not enough to get the ideas across. Apart from that, we did well to help each other out when it came to clarifying code that had not been understood or making sure everyone knew what to do for each sprint once they had happened.

I had a positive experience with this assignment. I believe I had a good contribution to all the sprints and made a good impact on our team. This is shown through the code written to develop most of the user interface and its functionality, ensuring SQL queries worked and dealing with merge conflicts whenever they arose. I feel I could have done better with the communication aspect of the project and taken more of a lead in making sure the team was

doing things on time instead of having to wait until the last minute to get things done. I also feel that I should have done more documentation. A lot of the code was easy to understand without comments since we were building interfaces. Despite this, I added multiple classes which defined what a Doctor or what a client was and they were being used in different parts of the project – whether it be the SQL methods or in the ChangeDoctor frame. As a result, it would be hard to tell what they were doing had you not commented what was going on. Despite all this, I spent a lot of time on this project and contributed to every sprint. I tried to reach out to members of the group in the early stages to try to build a rapport with the people on my team. This would help pave the way for a successful project despite all its challenges.

In future, I intend to do the following things differently: I would design how the components interact on a scrap piece of paper first before I had gone onto implementing them. For example, there were challenges when transitioning between authorising a user and displaying the 'home-screen'. This resulted in a slight glitch when you wanted to sign out as parts of the 'home-screen' components were still present when you were in the authorising section. This was also an expensive operation as every time this transition happened, the panel had to wipe off all the components and brought them back on when the opposite operation was called. What I would change about his would be to split the panels up into 2 or 3 separate JFrame components. This would mean that we could either close or inactivate one of the panels whilst the other one was in use. This would also be computationally less expensive and allow for a much better user experience. I would also improve the initial structure of our git project. I would keep the folders as they were but not include the files that were there. This would had meant that there would be less conflicts to deal with within files when merging and less redundancy as some of the files were not used. I would opt for a README file instead as that would have given a good enough description of how the project was laid out.