

**Department of Computing**

**Software Projects**

**(55-407815-AF-20201)**

**Name: A. N. Other**

**Student ID: 232208**

**Degree Route: BEng SE/BSc CS (Delete as appropriate)**

Table of Contents

[1. Reflective Assessment of Software Quality 2](#_Toc66356092)

[2. Reflective Account of Personal Development 2](#_Toc66356093)

[3. Suggestions for Future Work 2](#_Toc66356094)

# Reflective Assessment of Software Quality

# Me and my group worked diligently on our projects, and we believe that our hard work has paid off as all of the software fit user requirements and work well. Developing these projects also gave us many opportunities to improve as programmers, as we learned new skills such as database and GUI development and collaboration skills.

Software Project Two, the loyalty membership system, came out very good in the end; we managed to fulfill all of the most integral user stories and the user interface looked clean and professional, overall it met our expectations in terms of quality. Planning out the features by detailing user personas and user stories helped a lot in keeping track of which features we needed to work on and which ones were completed.

Although we think that the software does everything that it needs to do, there were some features that we did not have enough time to implement, mostly because we deemed it to be lower priority than the other features. We were originally going to have an Admin user persona, and we described user stories such as being able to add, remove or edit the offers and discounts, having the option to view the website as a regular website or premium user or the ability to see all registered users on the app on a page. We initially thought this would have been an important user to implement as it’s necessary to have administrators moderate and maintain an app, however as we developed the features for the three other users we came to the opinion that it was more worth our time to keep developing and rounding those users out. For this reason, we made the decision to cancel development on the admin features.

There were some features that we couldn’t implement more due to difficulty than time constraints, namely advertisements (which would be displayed to unregistered and registered users, and could be disabled by premium users) and text-to-speech. Text-to-speech was particularly difficult as it was very unfamiliar to us and we needed a lot more time to learn it. If we were to keep improving our software, this would be an incredibly important area to focus on.

For user satisfaction, I think that we created a very user-friendly interface that whilst having a lot of features, is still easy to navigate. This results in a pleasant user experience as anybody will be able to use our app, no matter their proficiency with technology. Despite failing to implement the administrator user, I believe that we met the requirements of the software, as every single time we showed off our software to lecturers after implementing a new feature they were more than satisfied, particularly with the aesthetics and functionality.

Our software’s standout feature, the premium membership, is similar to other systems on the market such as the Tesco Clubcard and Asda’s George Rewards, as in the customer can earn points from making purchases, and can then spend those points on things such as discounts and other products. Considering that the examples given are a lot more sophisticated and refined than what we made, it would be absurd to claim that our software is similar in quality to other loyalty schemes; however, our project could act as a foundation with which to further develop it to perhaps one day match the examples in quality. Despite not having created something which could be used comercially by a large corporation, we as a team are proud to have made something that works well for the assignment we were given.

My group and I always made sure we were organised and coordinated at every stage in the module. Our main methods of collaboration were Discord and GitHub. We used Discord for discussion, helping each other with bugs and other difficulties, and to compile all of the relevant information in one place; and we used GitHub to make programming collaboration easier, as we could easily update our projects with changes that other team members had made, and we had a simple-to-read log of all changes and additions made to the code. As soon as Stage 2 began, we detailed the user personas, user stories and acceptance tests and assigned roles to each other - based on what our perceived strengths and weaknesses were as programmers. It took some some effort to learn how to connect our GitHub repository to IntelliJ, but once we had it set up it was incredibly easy to use and convenient for us all. We had a few incidents where work was accidentally lost or overwritten because of things like people forgetting to pull before pushing code, but each of these incidents were rectified quickly, so we weren’t set back too much.

When starting Software Project Three, the attendance system, we were a lot faster with getting things like GitHub, the database and the login system set up because we were a lot more experienced and comfortable with it coming from Stage 2. We took inspirations from apps and websites such as Blackboard, Zoom and Microsoft Teams when going into development. We took the same roles as we had in Stage 2, as a lot of the required skills carried over into Stage 3.

User satisfaction for our project was very good. We kept a comprehensive list of all user requirements listed by the client and a list of user stories and checked them off as we developed the program. This resulted in the client being very impressed by what we had made and ticking all of the boxes in the feedback form, with the only bit of criticism being that whilst all the features were there, they were very basic and could have done with some refining. Whilst initially we worked hard on getting the features implemented, once we’d met all the requirements during Easter break the team became a lot more lax - and whilst the program was finished, there was still more we could have done if we’d have kept up the same determination we had at the start. All in all, despite the plateau we went through, I’m proud of what me and my team came out with, and I feel like I chose the perfect group to collaborate with; we all synergised and got along really well, we stayed very organised and always kept in touch.

# Reflective Account of Personal Development

I used a lot of the knowledge that I gained in Semester 1 in both of the projects me and my group developed. The most important was inheritance, which we covered in our programming module. In the loyalty system, we used inheritance to make sure that once the user had logged in, all their information would be able to be passed to the Controller methods when needed. We did this by using an Account class that would pull the user’s information from the Database class and store it as variables - and then we made a method in the Controller class with an Account constructor, which would then enable us to pass whatever variables we needed to other methods as parameters. This made it so much easier for us to do things that required user information, such as returning the user’s balance to display it on the interface.

While we did use a lot of the same concepts in the attendance system as we did in the loyalty system, we ended up incorporating a lot of file input and output to meet the user requirements, which we also learned in our Semester 1 programming module. This is because we had a feature where the teacher user could generate an attendance report of their class, which we accomplished by using a BufferedWriter to copy the student’s attendance data to a generated .csv file whenever they click the “Attend” button. This .csv file can be also opened in Excel for better readability.

As for personal development from developing the apps, I think the skills that I improved on the most as a result of this module were team working, leadership and, of course, programming. At the beginning of development I began the discussions on assigning roles to team members; I did not do much in the way of leading the team besides assigning roles as George and Kieren did much more in terms of organising to-do lists and giving tasks to the team, but I’ve always avoided acting as a leader in group activities throughout my life, so I feel like this is a start to my development as a leader. Examples of my team working include helping most of the team set up Version Control in IntelliJ so that they could use GitHub, collaborating with another team member on inheritance for the front-end developers. Overall, I think this module helped give me an idea of what software engineering is like, and what I need to work on to make sure that once I’m out of university I’ll work well in the industry. I feel a lot more confident in my ability to develop a project as part of a team, this module has given me a useful opportunity to develop necessary skills and I look forward to my next opportunity to code with others.

We programmed our apps in Java using IntelliJ, as we as a group had already been using it in our other modules and we felt most comfortable with it. To develop the user interface, the front end developers programmed using the JavaFX library and created the graphics using Scene Builder, because JavaFX was more modern a library than Java Swing and Scene Builder was very user friendly. To create the database and database related functions, I used DB Browser and the SQLite Library, because it made adding to the database very simple and it was very easy to attach to our codebase to run SQL queries. I feel as though we made the right choices tools-wise because we based our decisions on what felt most comfortable to use, which made development go much smoother and we ran into few roadblocks - although there were some difficulties with JavaFX as every time we pulled from the GitHub repository it would overwrite our library paths, so we would have to manually set JavaFX back up every time we updated our code.

If I could go back, I would have committed more to going beyond user requirements and put in the work to implement extra features, such as the planned text-to-speech feature in Stage 2. It was going to be my responsibility to develop this feature after completing all database related features, though because slowed down in productivity around Easter, I didn’t leave myself enough time to properly learn the text-to-speech library - and there was too much text in the program to go back and make sure was read out once I had gotten to grips with the library. Ideally, I should have practiced the text-to-speech library early on and worked with the front-end developers to make sure that all text was able to be read out as we developed the app. To summarise, I should have began implementing text-to-speech at the beginning of development, and I should have kept up my work ethic through the Easter break.

# Suggestions for Future Work

In our GitHub repository we decided to make use of the branch feature - for different reasons between stage 2 and 3. For stage 2, we made a new branch named “working”, with the idea being that we push the code from the “working” branch to the main branch once we felt the app was completed, sort of like publishing a release. However we came to realise this was unnecessary, as it was much easier to keep the code in one place, and because GitHub already has a feature for making releases, which made our method obsolete. For Stage 3, we had two branches, the desktop branch and the mobile branch. This was a much better idea than what we had for Stage 2, as these would be two different versions of the software and it made sense to keep them separate. However, this fell through during development, as we were going to develop the mobile version in Android Studio, but found it difficult to replicate the look of the desktop version. This led us to making the decision of making a “mobile” package within the desktop project, where the front-end developers simply remade the desktop version but with a small screen, which was much less sophisticated than an actual Android app, but it was easier for them to do in the time frame so it’s the solution we stayed with. Both stage 2 and 3 GitHub pages ended up having branches that went unused, which clogs up the page with unnecessary things and makes it harder to navigate, along with looking less professional. Having extra branches also meant whenever we opened our apps in IntelliJ we had to manually switch branches in the console, which was inconvenient. To improve this, we should clean up the GitHub page by removing the extra branches, and in future we should have a clearer idea of how we are going to structure our repository to make it as neat as possible - I think we’ll be much better at this now we are more comfortable with using GitHub. We should have also spent some more time to make ourselves familiar with Android Studio, as it would have looked much more professional and impressive when presenting it.

A change we could implement in future is an iOS version for the attendance system. To accomplish this, we would need new software to develop on, including a virtual machine with macOS installed and XCode as the IDE. We would also not be able to develop it in Java, as it’s not supported on Apple devices; so we would have to learn Objective-C if we wanted to start developing the app. Everybody in the team expressed interest in creating an iOS version as it would make it more accessible, and thus we would have a much wider audience for the app. Whilst the advantages of implementing this vastly outweigh the disadvantages, it was simply too much of a time investment for us to include this within the timeframe we had. If we were to go back to this in the future, we would definitely work on this as multiplatform apps are so important for accessibility and inclusivity, especially for a university.