Reflective Report

Roles and Responsibilities during the SE Phases

Everybody in my group had a different set of qualities coming into this project. Our initial introductions consisted of discussing these skills and assigning each other roles that aligned with their agreed Belbin Team Roles. As I already knew some of my team members before we started, it was easy for them to describe which ones I fit the most. I naturally suited the Cerebral roles, specifically the Plant and Specialist ones. I consider myself able to solve difficult problems, though may dwell on technicalities for example.

Requirements

We used the Waterfall methodology for our project. After reading the mandatory requirements for all projects, it was here that we discussed what parts of the development we each wanted to work on. Quite a few of my team members preferred working on front-end, whereas I was the only one who openly enjoyed back-end. As a result, it was very simple to delegate tasks to me as I agreed to do most of the parts they did not want to. However, since there was a large amount of back-end to do, I made sure I was only assigned a fair share of tasks and not too much. The remainder of the back-end tasks were shared out equally so that no one was forced to do too much of something they enjoyed less. As for the database, there was someone on the team who was keen to work on that part themselves.

Design

When it comes to the design of the application, I avoided the parts that involved UI/UX such as GUI design as I do not enjoy it, and there were multiple other group members who wanted to work on that aspect. I contributed ideas for the design that I felt were mandatory, such as which buttons are more important and should thus be larger than others, but nothing much more than that. As for my participation in the technical document, I worked on the assumptions section and also picked up some tasks that my team members were struggling with, for example I did a few diagrams for section 9 and provided explanations for my team members' GUI design choices. I also helped with filling out the Contribution Matrix for the document.

Most of the time during this phase, however, I spent doing a lot of research on Unity and Android Studio. I had learned that it was possible to export Unity projects into Android Studio projects and was testing if it were possible to create an Android app and then execute the game when a button is pressed, so that the game was only part of the application as a whole. After some struggles, I did manage to achieve this and so we went ahead with the minigame implementation ideas we previously discussed. It was also at this point that we decided to split our group into two smaller groups, with half of us working on the main application and the other half working on the game.

Implementation

This was my favourite part of the project so far. I had a lot of fun with programming for the game, as I was asked to work on that half of the project. My main responsibilities involved working on the core mechanics and writing most of the scripts. I also worked on implementing my teammates' sprite designs and animating their sprite sheets to create fun and unique enemies that the main character has to evade. I have been using Unity and C# for most of this phase. This played into my strengths as I enjoy scripting and writing code in

general. My favourite part was procedurally generating platforms that appear at reachable heights relative to the previously spawned platforms.

I had the most confidence with GitHub out of everyone else in my group, so I was also in charge of making the repositories and also explaining how branching and pull requests worked. If anyone had queries or uncertainties with git and how to perform an action they required, they would come to me and I would help them with it.

Testing

Besides the previously mentioned testing with Unity and Android Studio compatibility, we have not actually reached this phase of the cycle. However, when we do reach this phase, we will make a testing plan similar to the ones seen on Canvas (a table of things to test, along with expected outcomes and actual observations). Furthermore, we will write unit tests for the main app and potentially for the game, though I will need to do a bit more research into how unit tests work with unity projects. There will also be a playtest session where I will invite my friends to try the game out and see if they can find anything wrong with it or let us know if there are improvements they would like to see, almost like a closed alpha test.

Although not directly testing, I added a simple pipeline to the main application's GitHub repository. This was my first time writing a YAML file and so I kept it very basic. The only requirement for the action to pass was that the project would successfully build. This was important as it allowed those working on the app to check if their work did not interfere with the existing project, which saved us a lot of time when pushing to the main branch - it would always run first try.

Skills developed and improved during the project

Unity and C#

I have never used a game engine prior to this project, and had no experience with C# either. Albeit basic, I now have developed a lot of skills involving game development and now know how to use most of the features that Unity has to offer. I've learned about renderers, physics bodies and colliders/triggers and know how to combine them together to incorporate simple gameplay mechanics, such as jumping on enemies' heads to kill them, and two-dimensional projectiles.

Learning C# was very easy for me as the Unity library felt very similar to Python libraries. Whenever I wanted to implement something into the game, most of the time there was a pre-existing method for it, so I had very little trouble. Additionally, I have previous experience with both Java and the other members of the C family, so the syntax was very easy to get used to.

CI/CD, YAML and GitHub Actions

Discovering Continuous Integration during lectures was very fascinating to me. Although almost trivial, the concept of needing certain tests to pass before being able to push a commit to the main branch had never crossed my mind before. As such, this was also a new skill I have developed across the course of this project. I wrote a basic YAML file that checks whether the app code would successfully build and compile whenever someone attempted to push a commit. I learned a lot about the basics of how GitHub Actions works and produced a simple pipeline for our project.

Java and Android Studio

All of my team members have done projects in the past in Java, so I did not need to learn it coming into this project. We decided to use Java over Kotlin with Android Studio for this reason. Since I worked on the game aspect for most of the project, I have yet to do much with this part yet. However, when it comes to exporting the game into an Android Studio project, my skills will be put to the test as I learn more about how the software works. Additionally, I was the one that set up the initial project in an empty repository and will assist my team members towards the end of the project with parts that have not been complete yet, such as the backend for some of the pages for the main app.

Git and GitHub

I have used GitHub in the past before, but never really for team work. As a result, I have learned a lot more about branching and certain git commands, such as git fetch and git pull. In the past, I used to redownload the repository every time I wanted to continue working on it, which I now no longer need to do. Furthermore, I have become confident enough in these commands that I have been the person to turn to in my team for all git related enquiries.

Problems encountered and how you attempted to solve those problems

Task Assignment

As mentioned above, we had a lot of members of the team who preferred to not do as much coding, with preference for front-end work. In order to make sure everyone got to do at least most of what they wanted to, I suggested that we assigned two of the members the main app design tasks to work on together and one of them to focus on the game's interface, such as the menu and game over screens. Our other group members did not have too fond of a preference so they were happy to pick up the other parts of the project. Overall, I'd say this was quite successful and I was quite satisfied considering my preferences lie with the back-end side of things.

Miscommunication

My team had a lot of problems with communication. A very simple example would be with meeting time arrangements. There would be a lot of confusion about when a team meeting would occur, or if there would be one at all on that day. This did not last long however, as we agreed to discuss when the next meeting would occur at the end of all meetings, with our team captain confirming these times in our group chat later that day. However, this did not improve the attendance of some of them! Since everyone knew their roles and were getting on with their work anyway, we did not require full attendance to our meetings, provided they continued with their work in their own time and communicated whenever they had any issues.

This was working out quite well until just recently. One of our team members had expressed issues with their mental health and reported that they have not done as much of the work as they had hoped to have done by now. Considering how close to the project deadline this was, this has caused quite a big problem to the team. My plan to remedy this will most likely consist of abandoning a few of the extra features we had planned to add to the game and instead look to complete the fundamental parts as soon as possible, then divide the remaining work for the app out amongst available team members in an attempt to finish by the deadline.

Another problem that I personally had with communication was that there was very little intercommunication between the app half and the game half of the team. Although not necessary, it would have been nice to talk about both halves together more. I plan to amend this next week by talking to my group and asking them questions more, especially when we come to combine the game and the app together.

Specifically regarding the game portion of development now, we had slight trouble during implementation. We did not have a concrete plan about how the game should play going in, constantly switching between game ideas and never really coming to a conclusion. As a result, both myself and another team member had written a script for the main character's movement. This was a waste of time and resources for the person whose script we end up not using, which could have been avoided with a little more communication between us. In future, I will make sure to announce what part I am working on, and only begin once all parts of the app have been planned, including parts that we did not think would take up as much time as it did.

Combining the two halves of the project

Another potential problem we may have will be combining both parts of the project together. Although I did a lot of testing prior, it was only with a very simple unity game and an empty Android Studio project, so there is a chance that it may not work first try or with the same difficulty as before. I have attempted to remedy this by doing the testing prior, and will ask for some external assistance from a friend who is more knowledgeable with Gradle and Unity if I struggle further.

How has this module improved your employability skills and portfolio of work

Software Development and Teamwork

This project shows the ability to design a piece of software from scratch with very minimal requirements given, with a team of people that I have just met. This is a highly sought after skill amongst technology companies, as they produce products for their customers all the time. They need developers who can listen to a list of requirements given to them by a consumer and work together with people they may have never previously worked with to bring it all together into a successful piece of software.

In an environment that does not involve technology, teamwork is still a very important skill to have. Most fields of work involve some level of communication between staff and being able to work well with other people is a key skill to have for this reason. Additionally, software design involves solving a problem given constraints in a creative way, which are also very desirable among employees.

Fast Learning

Being able to pick up a new software, learn about how it works and then develop an entire project in it is not a light task. This is a highly sought after skill as it means that previous experience in the software that a company uses may not be as mandatory as they initially thought; employers would rather take someone who can learn their software fast over someone who only knows their software, as it allows for faster growth of the company (the employee who learns faster can pick up new software if the company every decides to change or expand).

Git and GitHub

This skill is more specific to technology companies only, but being able to use git is one of the most important aspects of development as part of a large team. It allows multiple people to work on a task at the same time, which is important for bigger projects. Being able to excel at git usage is highly sought after by bigger companies as they tend to deal with much larger developments.

Portfolio

Being able to show off a complete, working application made in a team is a fantastic addition to my portfolio. It shows of teamwork, communication, as well as git and coding skills all in one place. Additionally, it shows off a new skill in being able to create a game from scratch in Unity, which is a popular game engine used by many big companies in the gaming industry.

What would you do differently in the future

In the future, I would definitely want to talk to my teammates more. The nature of our meeting made it very easy to forget that they are also people with lives outside of the course. It would have been nice to get to know them better in order to increase both productivity and communication amongst ourselves. I plan to amend this by continuing to be friends with them once the project is over too, and by planning teambuilding days out with new groups in future projects.

In terms of development itself, I regret not taking on more tasks that are outside of my comfort zone. Since other members of my team wanted to do the parts I did not want to, I was almost too content and did not really go out of my comfort zone for this project. As a result, in future I will suggest to future teams that we all take on roles that we are not normally comfortable with at first and potentially provide an option to back out and swap back if we are not confident enough with it.

Additionally, I regret not working on the app side of the project more myself. Although I now have plans to provide some last minute assistance on it with my other team members, I feel as though I could have identified they needed aid sooner simply by being more involved with that half. Rather than remaining fixated on the tasks of the project that I am delegated, I will in future look to provide aid and perhaps assign myself some tasks in all aspects of the project so that I am able to help when needed and identify potential problems sooner.

In future I also would like to spend a lot more time on planning. Although we had really good plans for the app layout and specified all features that would be implemented, some of the features needed higher levels of preparation than what we had originally. As mentioned above, we started development on the game before we had a concrete plan of how it would play. This was a poor choice that led to some wasted resources that I will definitely plan to avoid next time.

In relation to planning, I also felt as though we could have asked more potential users of the software for their opinions. This would have allowed us to enhance the planning phases by making the designs more intricately tailored to our target audience. We had asked a couple during this project, but it would have not hurt to have asked a few more. This is something that I will attempt to do during future projects also.