Section 4A - Justification of software engineering methods

Choosing a development method to follow is an important step in designing and building a project. We discussed which method we should choose in our first in person meeting after getting acquainted with each other. As we were a small team of only six members and we didn't have that much time, we decided that an agile method would be more suitable for us. We were debating on either choosing Extreme programming or scrum as a method to follow. We finally came to the conclusion that scrum works best for everyone in the team, as the use of sprints lined up well with our weekly meeting schedule.

We decided to make the weekly in person meeting as the start of each sprint, since that's when we are most likely to be all present. Hence, we talk about what everyone has done in the previous week and then proceed with planning and dividing the workload for the next week. These were typically led by one of our team who acted as the “scrum master” and organised the team that week as well as knowing roughly where each team member was on their work. These sprints proved to be useful because it gave everyone the freedom to do their work in their own time throughout the following week.

We decided to at least have one online meeting organized in the middle of the week to check on everyone's progress with the workload that they were given and if they needed any help with it, we would also write what we have discussed over the meeting on the discord group chat, that way even if some members were to miss the online meeting they will still be able to be up to date with everyone else. We used discord as our main way to communicate online, the reason for that is that everyone was familiar with it and the fact that you can send messages or organise video calls with ease, allowing potential design ideas and sprites to be discussed between meetings. We were also able to share important links for useful tutorials and articles via discord.

For documentation and sharing important documents with each other, we decided to use Google Drive as it’s easy to use and keeps our documents in the cloud, mitigating any potential risk of losing important documents.

We then started talking about how we will design and implement the code for the project. Starting with architecture we decided to make a UML diagram to help with the design process. We used PlantUML class diagram while using VScode as the text editor since it had an extension for PlantUML which allows the user to preview the diagram while editing it without having to produce a png each time to check see the changes after changing the code, it also had syntax highlighting which made it easier to write changes with fewer errors.

Java was the language that we were tasked with using to make this project, and since the project was a game, we had to choose what game development platform we will be using. After reviewing our requirements, we decided that libGDX was the best choice, and we chose the program Tiled, a free level editor tool to help with level design since some members were familiar with it already.

Most of the team were inexperienced with using LibGDX so we relied a lot on self-research and self-learning, Youtube was useful since there were many helpful LibGDX tutorials. Any

useful websites or videos were shared on our discord group chat to help everyone get accustomed to using LibGDX.

We decided to use Git as our way to share code and collaborate with each other, so we made a repository for our project. Some members used Sourcetree while others used GitHub desktop, in both cases the GUIs provided in these programs simplified using Git to retrieve and push changes to and from the repository. This proved useful especially for team members who were not familiar with using Git. Using Git also means that members can see the code being updated each time and get a grasp of the progress done by other members, which fits well with the agile method we chose.

The use of Git was particularly powerful due to its “branch” feature. Allowing us to produce separate feature branches at the beginning of a sprint to implement new functionality. Concluding the sprint by merging the changes from any feature branches back to the “Main” branch and resolving any merge conflicts to produce a new stable version of the game to build on in the next sprint. This process ensures that the team can work on independent features, building on a stable version of the game, without having to wait for other members to conclude their work.

Section 4B - Approach to team organisation

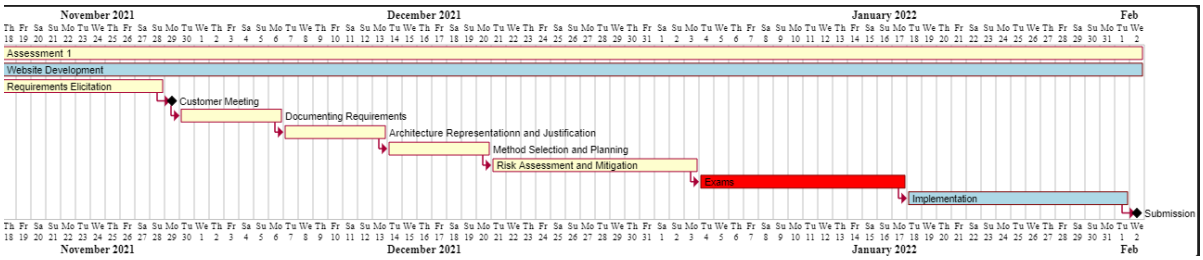
A good approach to organising the team can make the project's development process much more efficient, hence why we discussed everyone's strengths and weaknesses in our first in person meeting to get an idea of how roles might get allocated. We then decided to give different roles to members depending on what they're good at.

We had two members who were more experienced with coding and one of them was even familiar with LibGDX so they volunteered to do most of the coding bits, they would also share and take ideas regarding the game mechanics from the rest of the team. The code was also shared on Git so all other members would still be able to view it and see what changes had been made to the game.

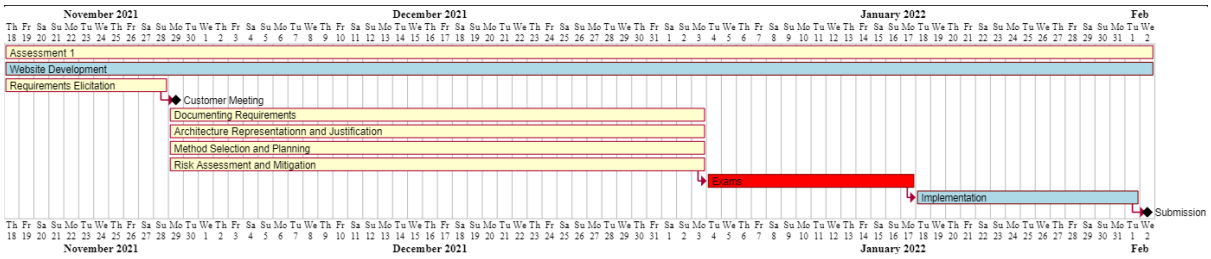
The other team members focused more on documentation, planning and architecture. If anyone has any problem with their workload or is facing an issue, they can easily message the group chat and we can figure out a solution to solve their problem.

Section 4C - Plan for the project

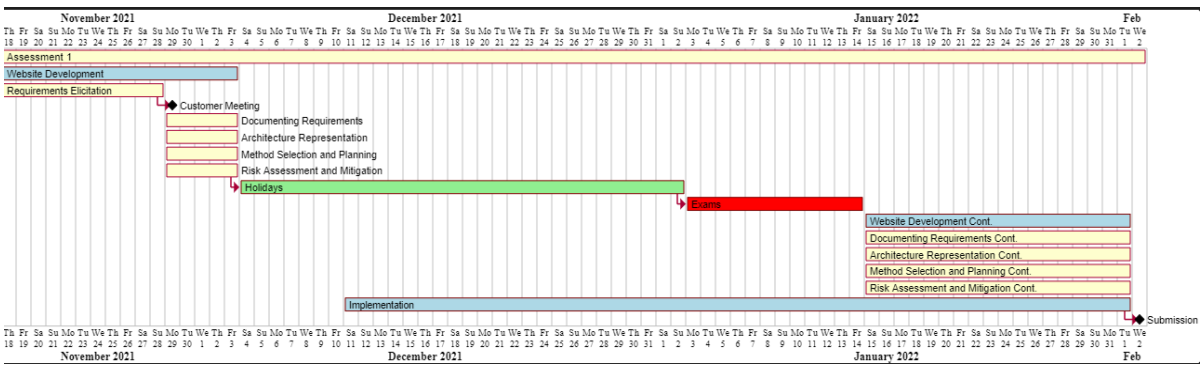
This project started on the 19th of November 2021, and we began by eliciting the requirements of the project, this was a high priority as it was essential to obtain a proper understanding of what was required of us before moving forward. We then set out an initial plan, sequentially completing each part of the documentation, spending between one and two weeks on each piece. Then taking a break to focus on exams in the new year before starting the implementation on Monday the 17th of January 2022.



This plan was created under the assumption that all the documentation for the project would be completed in the order it was laid out in the assessment brief. We soon realised this was not feasible, both in terms of dividing work between the group and each part of the documentation having varying levels of priority, leading us to decide on a new plan.



This revised plan has the team working on all parts of the documentation concurrently as this made it easier to divide work between individuals. However, this plan was made assuming the team would work over the holidays which we later decided would not be possible. We realised it would be difficult to organise meetings and complete work over the break, especially considering we had other university deadlines looming and January exams to prepare for. After having a team discussion regarding the holidays, we drew up a final plan for the project.



This final plan took into account the breaks the team would be taking over the holidays and to prepare for exams in the first week of the spring term. With this plan we decided it would be a good idea to spend some time over the holidays to learn the game library we would be using for this project, LibGDX. This was to make the implementation easier for the team as it would allow us to jump straight into coding the game. This led two team members taking responsibility for the coding of the game and one member undertaking the design of assets. All team members were responsible for parts of the documentation, with the main programmers being responsible for the architecture and the rest of the documentation being divided equally among the other members of the group.