Method Selection (agile scrum/spiral)

The method we have selected to use during this project is a Spiral method called a scrum. This is a repeated cycle of identifying what needs to be done and its risk, followed by some development and then an evaluation of the product so far. This isn't too big of a project, so a plan-driven model is definitely viable. We thought since it is some of our first times doing a project like this, we should go with an agile method to allow more space for any issues that arise:

1. Identify Objectives

   New requirements are identified in as much detail as possible. This could involve interviewing the "client" or other aspects of the existing system to identify what needs to be done in the next couple of days.

2. Perform Risk analysis

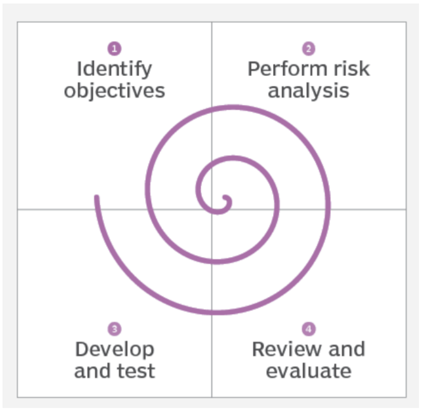
   At this stage, we take a look at the new objectives that we have made and identify which people will be taking on which objective while also looking at each objective and identifying any potential risks that may occur.

3. Develop and test

   This is when we all go and work on our designated parts of the project, we have a group discord allowing for communication and sharing of resources if more than one person is working on one part of the project.

4. Review and evaluate

   We then come together again for a meeting and discuss how our individual tasks have gone, this time allows us to work together and see if anyone else can solve the issues encountered by another group member.



There are a few main reasons why we chose this method:

- This method is a good way to break down the projects into smaller, more digestible chunks.
- It is also a very flexible method, as it is a cycle that repeats often; we are able to identify any changes to the requirements quickly and make changes appropriately.
- Allows for frequent risk assessment, as each cycle contains a risk assessment segment; we improve the security of our system while also trying to eliminate any risks that may come up.
- If at any point we feel like we are confused about any requirements or want to get more customer input, we can set up customer feedback as the flexible method allows for it.
- We thought it would be a good fit as we were thinking of splitting up the projects into smaller one-week tasks.

Development and Collaborative Tools used

The Java framework we chose to use was libGDX

- The Java library LWJGL, which gives users access to native audio and graphics APIs, was something we had thought about adopting. But as code reuse is a crucial component of software engineering, we opted to employ it since libGDX uses it in its framework.

- As LWJGL offers low-level access, this also enables us to spend more of each Scrum sprint working to resolve problems that are connected to the task that was assigned to us by the customer.

For version control, we used Git and stored our repository on GitHub

- Avoids collisions

- Hosting the repository on GitHub gives us access to GitHub Pages, which makes it simple to view documentation, as well as version control history maintenance.

Our IDE of choice is IntelliJ.

- Has the best framework to work with libGDX

- Developing a project is easy using IntelliJ. It has attributes that are absent from VS Code, making it easier to work on large projects with numerous classes. Examples include automatic class refactoring.

Weekly meetings

- Allows us to properly utilise the spiral method that we have planned to use throughout the project to keep us all updated and aware as to what's going on with the project.

Discord

- this is just a messaging platform that allows us all to communicate and share work with each other. We chose this because we all already had it, and it is easy to set up and use.

- Allows those who are unable to attend an in-person meeting, can attend on discord

Google docs

- An easy way to share and collaborate on regular word documents.

Team Organisation

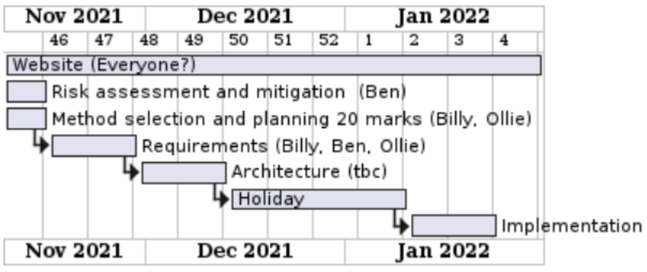
We decided to split the team into two groups as we had such a large group, one for Coding (**names**) and one for documentation (**names**). We thought this approach would be beneficial as it would allow those on the coding team to focus more on the technical aspects of the development and programming, and while the coding team is working on that, we can simultaneously have the documentation team work on planning, while getting started with the software architecture and risk assessment.

The reason we chose to split the teams was so that people could work on parts they think they're especially strong at, thus improving efficiency, and with a team larger than normal, we thought we could use this to our advantage. And because the documentation is worth a lot more than the code, that's why we have a larger documentation team. Another reason for choosing to split too two teams is you can have weekly scrums for those teams, allowing for more collaboration in the scrums.

This allows for an effective weekly scrum method, and with the two teams, we would have weekly scrums in which we lay out tasks for both teams, but also scrums within each team.

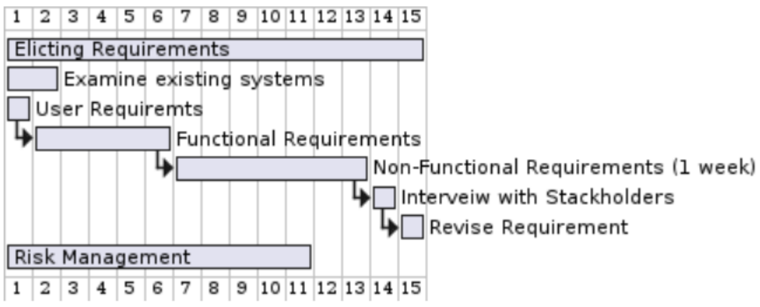
Project Organisation

Project Gantt chart



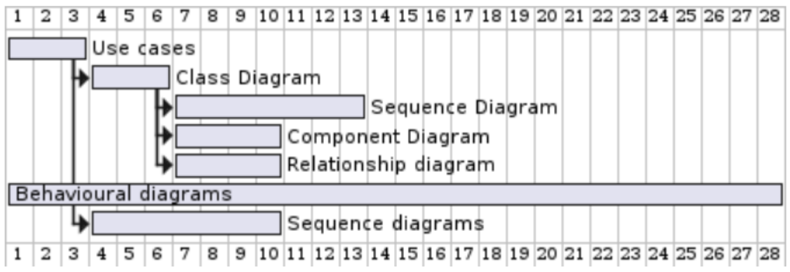
This was our original project Gantt chart, giving a brief outline of the project. This wasn't detailed enough so that we could express task priority and who should be on each task. We kept it as is for the start as we knew there were going to be changes made to this later down the line.

Eliciting Requirement Gantt Chart



This was our timeline for our requirements, which are first extracted from the product brief, then customer interviews and feedback.

Architecture Gantt chart



The Architecture Gantt chart is created to allow the tea and the client to understand how the implementation is laid out, with a proper representation to be made.

Systematic plan