**Method Selection & Planning**

Group Number: 10

Team Name: Decassociation

Group Member Names:
Tom Broadbent
Poppy Fynes
Owen Lister
Michael Marples
Lucy Walsh

***Assessment 1:***

**Methodology & Tools**

For this project, we shall be using the SCRUM methodology. The reason we chose this methodology was that it takes a short amount of time to complete the project, with a small number of people in the development team. The key is the customer's needs and this SCRUM enables us to adapt quickly to changes in requirements, and updating the customer on our current progress to provide a more satisfactory end product. Additionally, since the project development would happen over Term 1 and Term 2, resulting in a change of timetable, SCRUM allowed for the extra flexibility. Scrum is an agile methodology based on iterative sprints (short periods of time where we'd work on a set number of features). Each week, we would have a sprint and during the weekly meetings, we discuss our implementation and new goals for the next sprint. This made sure we kept on track of the core features and understood what features each member (those working on the implementation) would implement along with whether there were any problems during the previous sprint.

We used GitHub to host the project and Git for version control. These applications work hand in hand with Git allowing us to push project files to the online repository. This allowed those coding the project to remotely update the project and to revert changes / look back on previous versions if needed. Furthermore, using our chosen methodology was made very simple since we could all push and merge our respective implementations and organise required tasks using the GitHub Projects page.

Our team used a range of IDEs which included:

- ➔ Eclipse
- ➔ IntelliJ
- ➔ VSCode

Overall, we picked LibGDX as the development framework for the game. This is due to it being reliable with many resources available, use of Java which many members were familiar with and it was suitable for 2D games which we planned to make. To collaborate, we used Discord, an online instant-messaging program. This helped us communicate, organise and have meetings remotely. We initially chose WhatsApp but made the switch as we realised that Discord would provide a better experience for the team due to its file sharing features and how it's primarily used on PCs, Laptops and Mobile Phones while the majority of users only use the mobile version of WhatsApp.

For our website, we used GitHub Pages to host the site as well as using its domain name. This seemed like the most appropriate option since we could easily link our implementation and other documents using the GitHub repository. To store and collaborate on deliverables and any additional documents we used Google Drive and Doc. This allowed all of us to

work on the documents simultaneously and at any time.

**Team Organisation**

We have assigned different roles amongst the team members. We would arrange meetings twice a week on Mondays and Wednesdays. During these meetings, we would check the progress of each member's assigned task to make sure they were on track, and then when the task is completed, assign new tasks for the next prototype. We decided twice a week as we already had a set time to meet on Wednesday, however if any problems arose after, we could meet again on Monday and understand what will need to be done before the next Wednesday meeting.

If members weren't available for the meeting, we would relay information through our Discord group chat. We would also upload a brief summary of the meeting with tasks assigned. Additionally, during the implementation, we would often upload screenshots and discuss aspects of the features we were implementing. This is so all members of the team are up-to-date, whether they are present or not and can provide instant feedback if needed, as well as, allowing people to have something to look back to.

Our group chat was organised into chats for meetings, documents, images, websites, and general. We separated roles into implementation and documentation. For the most part, all of us worked on the implementation. This helped the project as all members of the team could give insight on what works, what doesn't and what needs changing, leading to a better understanding of the task as a team and a better product in the end.

**Planning**

The plan evolved as people grasped the requirements of the project and as the project went on. Firstly, we had to plan out the roles people were given and what was needed earlier in the process. We outlined what everyone was doing from their strengths and weaknesses. Some team members had never created a game before and were unfamiliar with LibGDX, others were good at documentation and web development. Inevitably, everyone had an assigned strength that they were contributing in the making of this project.

An early task that needed to be completed was the meeting with our customer. This would help in narrowing down what was needed for the project. This took place on the 22nd of November and we decided to send 3 members of the team to this meeting as the task did not require a full audience present and only a few people to listen and note down their desires.

Next, we all contributed to the Risk Assessment in the two meetings on the 23rd & 28th of November. Due to the upcoming winter break, we set specific tasks that needed to be done

and that we would begin the implementation stage and work on the Architecture and Method Selection & Planning documents. Each member of the team had an assigned task and we would meet once a week regularly, throughout December and early January, to provide our progress with one another.

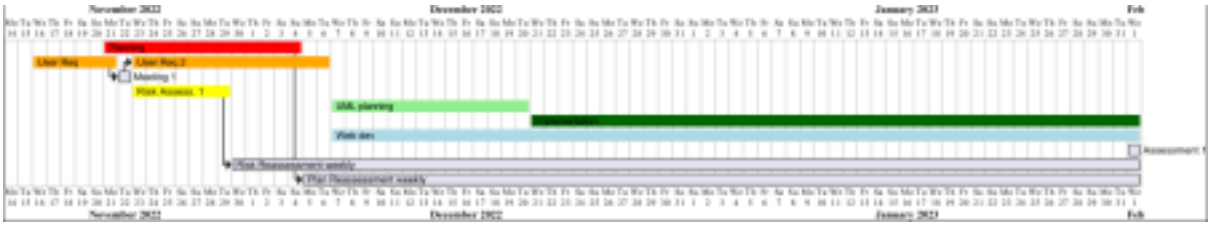
After the break, we had meetings on the 16th & 18th of January to recap what we needed to do. Our plan had to evolve as certain features, such as the leaderboard and food stations, needed extra work before completion and documents needed finalising. This led to members working on specific parts of the project and allowed us to stay focused on what needed to be done.

In the weeks of the 23rd & 30th of January, we completed these tasks and focused on last minute adjustments such as replacing any temporary assets and finalising the commenting of our code. To be safe, we increased the number of meetings we had which ensured we included everything from our brief and could bring up any issues easily with each other present.

Below find the following gantt charts:

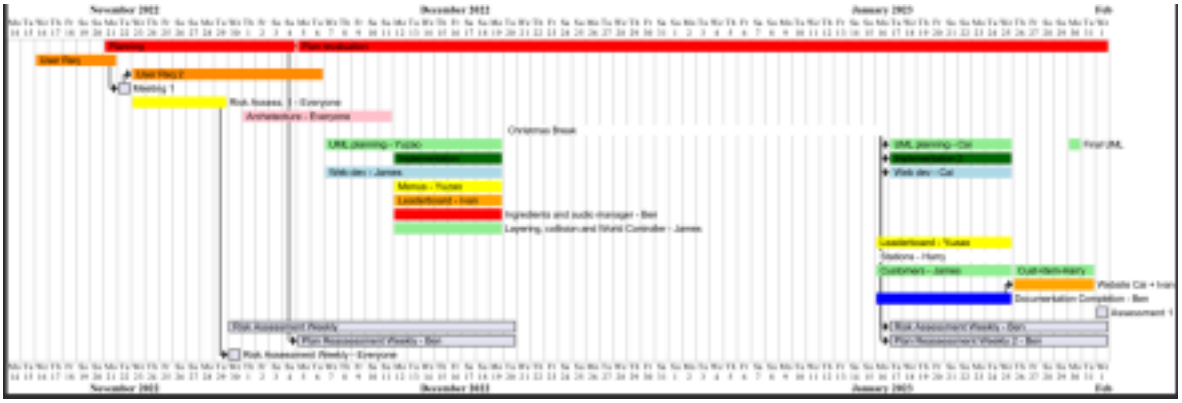
Initial gantt chart - 21/11/2022

Second gantt chart - 3/12/2022



Third gantt chart 19/01/2023


Final gantt chart - What happened 31/02/2023



***Assessment 2 Team 10 Takeover:***

**Methodology & Tools**

Our takeover of Team 3's project in terms of planning and methodology has been quite smooth because we worked in a very similar way to Team 3, using SCRUM for a fast-paced project, using Discord to communicate, using the same tools and in assigning roles.

We continued to use Discord to communicate, as both we and Team 3 did in Assessment 1, but we had meetings once a week because we didn't feel it was necessary to have more than one meeting per week due to the rate at which we were working and how infrequent issues arose. These meetings acted as our sprint review and sprint planning ceremonies; usually we would go through all the work that was completed since the last meeting first and use that information to plan the next sprint (the week until the next meeting). Owen was appointed as SCRUM master so led the meetings in a way that allowed full collaboration instead of top-down management style (as a project manager would). The sprint reviews and sprint plans were documented as sort of meeting notes / a to-do list which was updated with new jobs, jobs being crossed off when completed etc.

Discord was chosen over tools such as Slack and WhatsApp. We didn't select WhatsApp because it's not professional in any way and didn't allow for easy communication on desktop devices. Discord was favoured over Slack due to the team being familiar with Discord and not Slack, and the two seemed very similar so choosing Discord was personal preference.

For developing the game from we had 3 stages:

1) Brainstorming / Design - Coming up with ideas for what exactly we want to implement and how. We of course have the requirements for 'what' we need to implement but we needed to decide how those concepts would transfer into the actual gameplay. We all had to agree on an idea to move forward.
2) Production - We put our brainstormed ideas developed from the requirements into the actual implementation.
3) Testing - We tested, through unit tests and manual tests where appropriate, the features we implemented in the production stage.

These three stages were looped through rather than just happening once. While we had a large brainstorming session at the very start of Assessment 2, we would work on only one or a few new requirements / features at a time, so would have to go back to stage 1 once we had finished implementing one set of features.

To develop the game and therefore follow the three stages of the development process we used various tools. As Team 3 did, we used GitHub as a code repository because of course it allows for easy collaboration on our game code, and includes various supporting features in pulls and pushes for conflicts and merges, amongst others. We also chose GitHub over an alternative such as GitLab because first of all it's free, and secondly because it is a highly trusted, professional, well-supported platform that is already used by thousands of organisations; this is for a reason - it's high quality.

In terms of IDEs, the whole team used IntelliJ due to it being specifically for Java, meaning it has lots of great features just for Java that other IDEs such as Visual Studio Code don't have. IntelliJ also allows PlantUML integration which enabled us to develop architectural class diagrams in IntelliJ using PlantUML (as Team 3 did) and keep them under version control. With just one IDE across the team we developed shared knowledge that allowed us to help one another with issues in IntelliJ.

PlantUML was used to develop architectural class diagrams because its a simple language to learn so we could quickly understand how to create the diagrams and it also integrates into IntelliJ as previously mentioned. This allowed us to add the class diagrams to the repository which made it extremely easy to make changes to them (we had quite a lot of trouble with using PlantUML in Google Docs during Assessment 1; this was much easier).

Google Drive was used to store all of our project documentation and Google Docs was used for the actual documentation. These choices were made because we all regularly use Google Drive so are familiar with it (including Google Docs), and Google Docs obviously works as part of Google Drive. Most significantly though, this allowed synchronous collaboration on documents and they are all stored in one single, central location. GitHub Pages could have been used but as stated, we were all familiar and confident in using Google Drive and Google Docs.

**Organisation**

Overall, we tended not to have task-specific roles set in stone and just assigned randomly because we were using SCRUM - we all decided on what we wanted to do as a team. This meant we didn't run into any conflicts over who was doing what work although compromise was required sometimes.

In terms of task-specific roles, generally, Tom and Owen were assigned to unit testing, Michael was assigned to manual tests, Tom, Michael, Lucy and Poppy were assigned to implementation / code, Poppy was assigned to the website, Tom, Michael, Lucy and Owen were assigned to documentation. We had a good mix of roles that were very flexible and allowed members to do the work they wanted to do essentially.

Team roles (non-task-specific) were not assigned. There wasn't a 'project manager' because we all decided as a team what we wanted to do and generally if a team member worked on something, they were then responsible for checking back up on that work if anything needed updating.

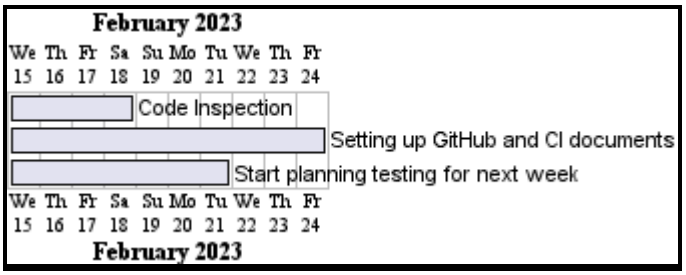
We never had any conflicts in terms of ideas; everyone was open to other members' ideas and the best ideas were agreed on by the whole team - we bounded ideas off of one another to work together rather than 'fight' for our own ideas constantly. The plan was to take a vote if we ever did run into conflicts.

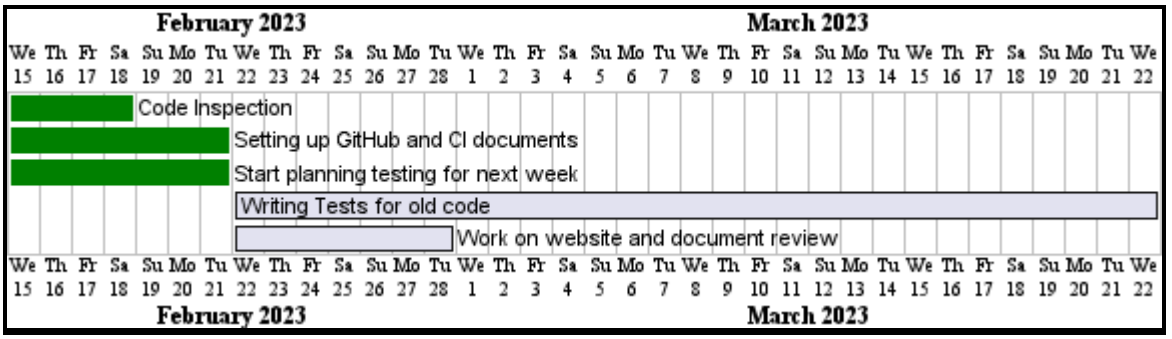
**Planning**

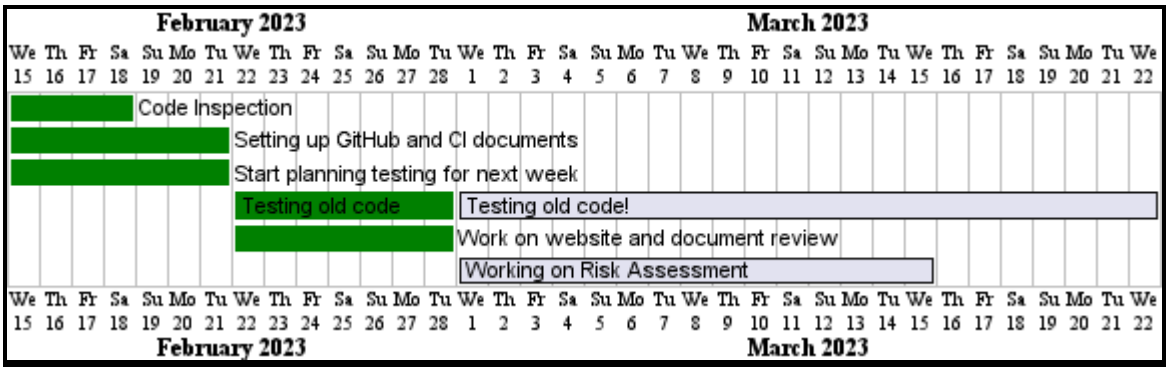
See the above 'Methodology and Tools' section for how we planned our work (sprint reviews, sprint plans etc.) as it was covered there to fit alongside the SCRUM methodology explanation.

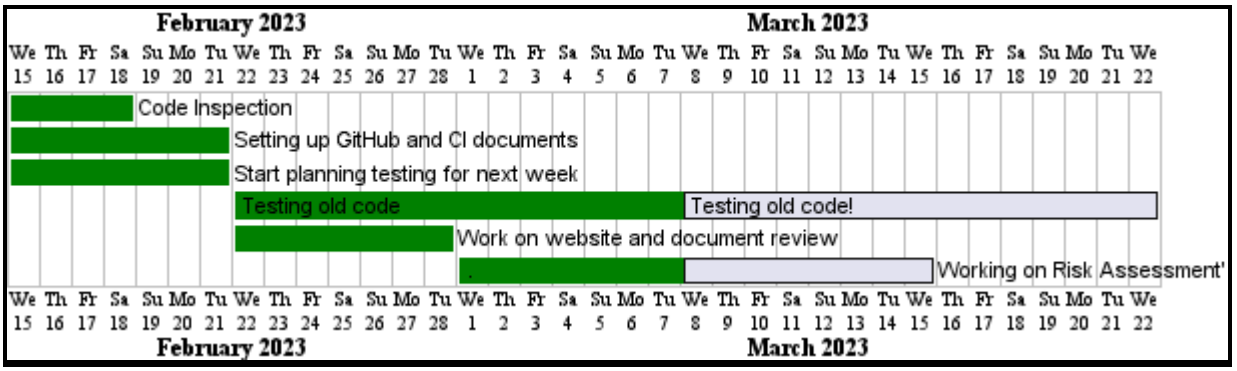
We used gantt charts to visually plan what work needed to be done next and when the deadlines were for all of that work. A gantt chart was made in almost every single meeting we had. See below

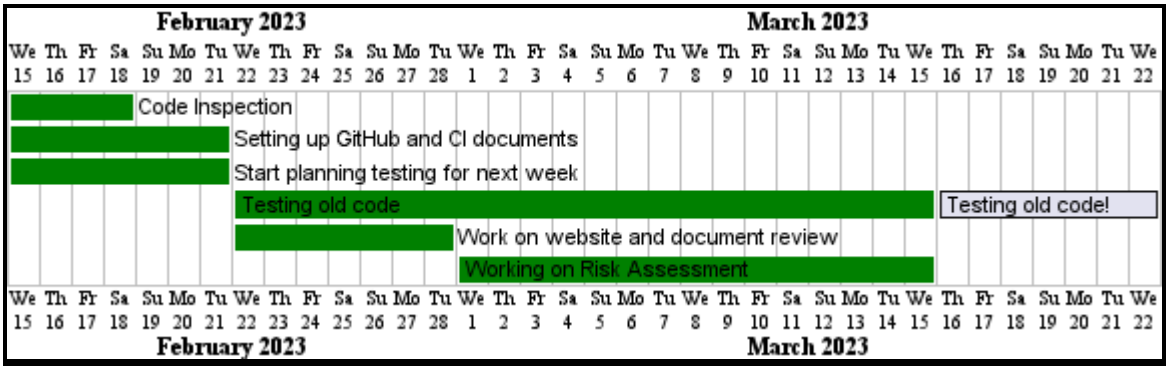
for our gantt charts.

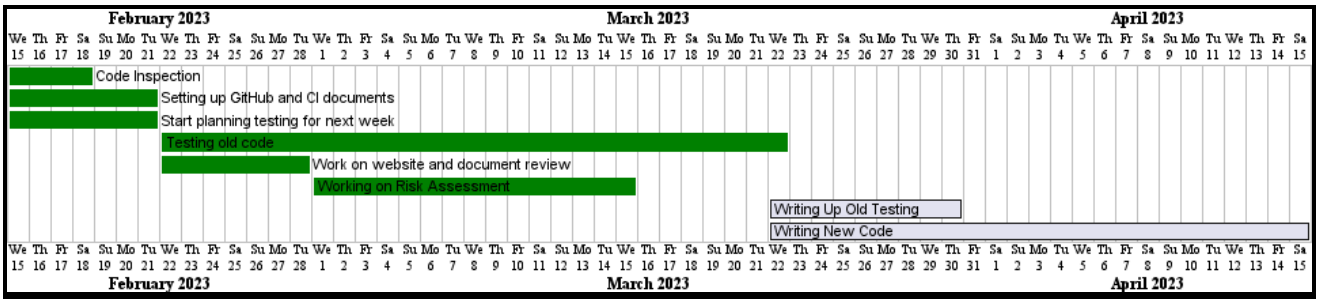


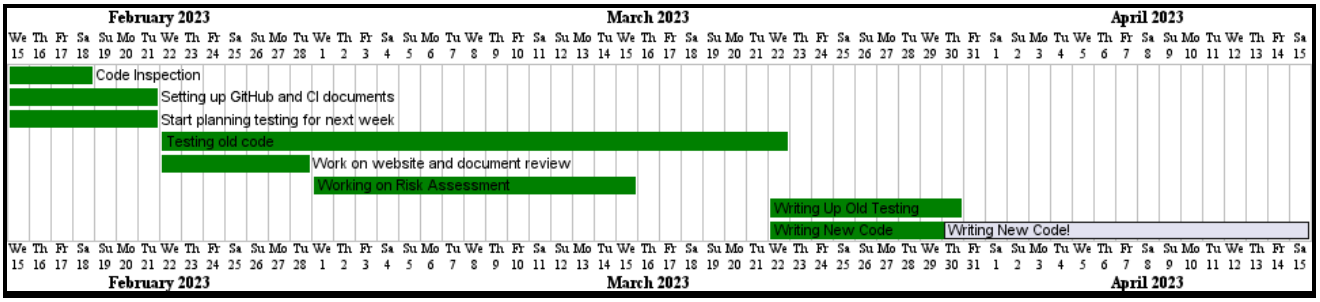



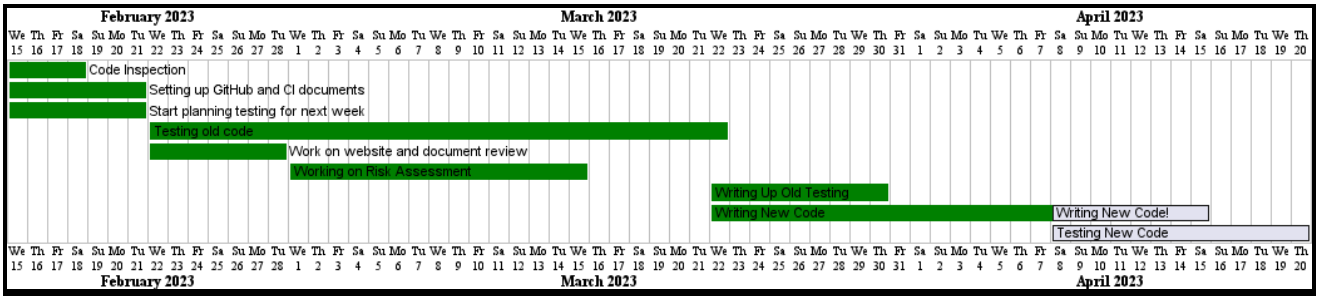



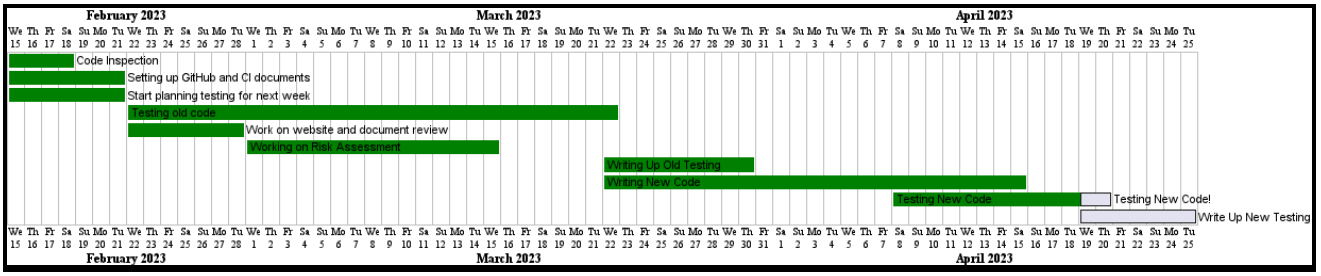




February 2023
March 2023
Code Inspection
Setting up GitHub and CI documents
Start planning testing for next week
Testing old code
Testing old code!
Work on website and document review
Working on Risk Assessment


February 2023
March 2023
April 2023
Code Inspection
Setting up GitHub and CI documents
Start planning testing for next week
Testing old code
Work on website and document review
Working on Risk Assessment
Writing Up Old Testing
Writing New Code


February 2023
March 2023
April 2023
Code Inspection
Setting up GitHub and CI documents
Start planning testing for next week
Testing old code
Work on website and document review
Working on Risk Assessment
Writing Up Old Testing
Writing New Code
Writing New Code!


February 2023
March 2023
April 2023
Code Inspection
Setting up GitHub and CI documents
Start planning testing for next week
Testing old code
Work on website and document review
Working on Risk Assessment
Writing Up Old Testing
Writing New Code
Writing New Code!
Testing New Code


February 2023
March 2023
April 2023
Code Inspection
Setting up GitHub and CI documents
Start planning testing for next week
Testing old code
Work on website and document review
Working on Risk Assessment
Writing Up Old Testing
Writing New Code
Testing New Code
Testing New Code!
Write Up New Testing