Method selection and planning

Group 30   Triple 10

Team Members:
Kelvin Chen, Amy Cross, Amber Gange, Robin Graham, Riko Puusepp, Labib Zabaneh

Assessment 1

Part a)

Within our team we discussed the tools that we thought were the best fit to optimise the collaborative work. We decided that Github, intellij, discord and google drive would be the best tools to aid our game development. For the following reasons:

GitHub is the platform we chose because it provides a convenient way for developers to host and share our code, track changes, and collaborate with each other on our project. It also offers a variety of tools for managing and reviewing code, such as pull requests and issue tracking. Additionally, GitHub is widely used in the industry, making it a popular choice for developers to share and discover open-source projects.

IntelliJ is the IDE that we chose as it provides advanced debugging features, such as the ability to set breakpoints, step through code, and inspect specific variables. This makes it easier to fix bugs in code. IntelliJ also provides intelligent code assistance, including code completion, error highlighting, and refactoring. This helped write code more efficiently and with fewer errors.

Another reason why we chose IntelliJ is that it is available for Windows, Mac, and Linux, making it accessible for the whole team to collaborate on the different operating systems. IntelliJ IDEA has a large and active community of users and developers, which means support is readily available when a problem arises.

We chose IntelliJ over VS Code as it is a more powerful and feature-rich IDE that is best suited for larger scale Java projects, while VS Code is a lightweight IDE and is more suited to other projects. Another option for an IDE would be eclipse. However, A few of the team members were unfamiliar with eclipse and so we decided that IntelliJ was the best option for our team at this time.

Discord was our method of communication outside of group meetings. Discord servers are divided into channels that are organised by topic or purpose which makes it easier to find information. Whereas our team did not select Slack as Slack channels are organised by team or project. This would not have been utilised as we did not have multiple projects but we had multiple topics ongoing. Also, Discord has better-calling facilities compared to Slack.

Discord is designed to help teams collaborate and stay organised by allowing us to create channels for specific projects or topics. For example, we created a channel about the implementation. Team members can also send direct messages to one another and make calls or start video chats. A useful feature was screen sharing as this allowed us to easily work on code and mention specific parts of the project without confusion. Discord also allows users to receive notifications for specific channels or mentions, which means that team members will be alerted when there is a new activity or important information that they need to know. Discord allows integration with third-party tools such as GitHub, which can be useful for version control.

Google Drive is a useful tool for a Java coding project as it allows for cloud-based file storage, real-time collaboration, version history, access control, and integration with other tools. It is also a good complement to other tools like GitHub to improve the collaboration of our Java game.

Google Drive provides multiple users to edit and collaborate on a document in real time. This makes it easy for all team members to work together on our game, even when we were not in the same physical location.
We used Google Drive over DropBox as Google Drive can be integrated with a variety of third-party tools and apps, such as Google Docs, Sheets, and Slides, which can be useful for project management, documentation, and collaboration. Dropbox also offers integration with third-party apps, but the list of apps is more limited than Google Drive. Also, every member of the team has a Drive set up that was readily available to use due to our York university accounts.

All of the tools above aided our collaboration on our game development as we looked for the following attributes within these tools:

1) Version control: Using a version control system like Git allows team members to collaborate on code and track changes, with the ability to revert to previous versions if needed.

2) Communication: A real-time communication platform like Discord can be used for team members to discuss and collaborate on code, as well as share files and screenshots.

3) Cloud-based file storage: A cloud-based file storage platform like Google Drive or Dropbox allows team members to access project-related files and documents from anywhere with an internet connection.

4) Project management: Can help keep track of tasks, deadlines, and progress, and keep team members informed of the overall project status.

5) Code review: Establishing a code review process allows team members to review and provide feedback on each other's code, which can help improve the overall quality of the codebase.

6) Access control: Control of access levels for different team members, allowing for managing who can view, edit, or share files and documents.
7) Security: Ensuring that all tools used are secure and that sensitive data is protected by encryption, two-factor authentication, and other security measures.

Taking the project for assessment 2 we decided that due to our familiarity with using a waterfall model before we wanted to follow a similar system and so chose an Iterative and Incremental Model made of mini waterfall models. The Team before us used an iterative plan but we felt that this did not reflect the relationship with the stakeholders.

See the page for more information on this model.

https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/

Part b)

To effectively work as a team, it is important that we play to the individual strengths of each team member. Such that everyone feels comfortable in the role that they have been assigned to, thus performing to the best of their abilities. We held a group discussion about prior software experience and group experience to determine where each member would fall in which role. Furthermore, we had to have a discussion about the roles we would need in the lifecycle of the project. This session was called "Forming".

During the discussion, the notable roles for members to play were: Meeting Chair, Secretary Librarian, and Report Editor. The purpose of the Meeting Chair is to ensure that the group remains on task during our meetings and to assign tasks at the end of the meetings. The purpose of the secretary is to record decisions made in meetings for future use through meeting minutes. The librarian's role is to keep documents and other resources and to oversee version control. The purpose of the report editor is to oversee and organise the production of reports. Some roles were divided between two teammates.

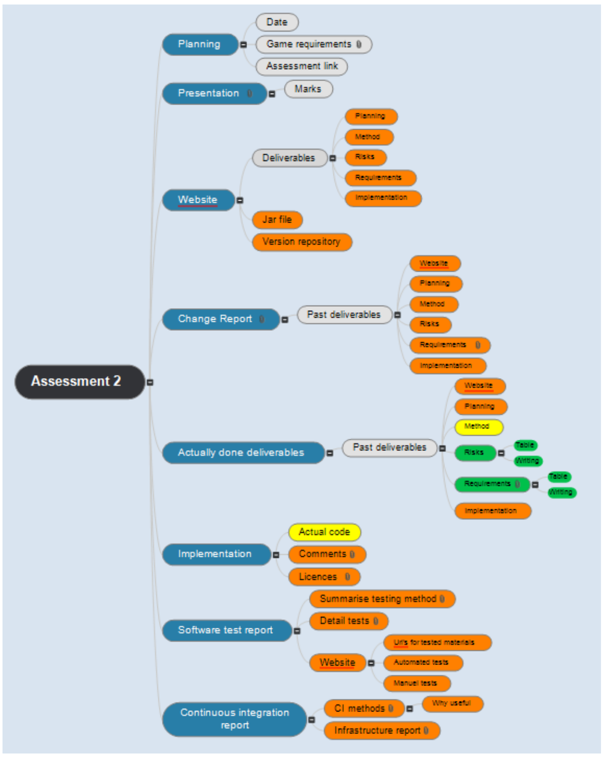
Commencing the project, it was decided that all members would contribute to the code. However, in practice, we soon realised that it would benefit the team for only a few select members to work on the code, while the rest dedicated their time to the deliverables. This was because, in order for all the members to be involved in the coding, tasks were split up incredibly small, which resulted in some tasks relying on the completion of tasks by other team members. This violated one of the risks in the risk register, R_PROJECT_06. Therefore, the owner, James Wild, decided it would be best to only allow three members of the team to code. The split was decided by preference and experience. R_PROJECT_06 is reflected in the new risks by R6 and R11.

Megan: Secretary, Kajol: Librarian, Isselmou: Report Secretary, Matt: Secretary, James: Report Editor, Alistair: Meeting Chair.

Assessment 2

Our teams' roles

Amber	**Report Secretary**
Amy	**Librarian**
Kelvin	**Secretary**
Labib	**Report Editor**
Riko	**Secretary**
Robin	**Meeting Chair**



To organise our team we followed a similar plan to our assessment 1 where we split up the tasks into smaller parts so that we could allocate them out to different team members. The mindmap shown to the left shows how we split up the project into smaller parts and the colour coding helped us keep track of what we needed to work on.

- Additional cook
- Extra recipes and ingredients
  - Pizza
  - Jacket potatoes
- Failures
  - Preparation step failures
  - Losing reputation points
    - Initial points at 3
    - Timer for serving customers
- Game modes
  - Endless mode
  - Scenario mode
    - Changing number of customers
- Customers can arrive together
  - At first, they arrive themselves
  - After some time, arrive in groups of 2 or 3

- Buying more staff and cooking stations
  - Some form of currency
- Power ups
  - Speed up chef
  - Shorter cooking times
  - Reset customer wait time
  - Gain reputation point
  - Coin boost when serving customers
  - Completed dishes rain down from the sky
- Saving progress
- Difficulties
  - Shorter or longer wait times
  - Number of customers arriving at one time

We also created a logical criteria that we needed to add for assessment 2 and zoomed into particular requirements that needed to be expanded.

Assessment 1

Part c)

Beginning the project, we looked at the key tasks for each deliverable that needs to be completed. This is demonstrated in the table below:

		Start Date	End Date
Requirements	1. Conduct interviews 2. Research for requirements 3. Write up requirements	21/11/2022 24/11/2022 26/11/2022	21/11/2022 26/11/2022 27/11/2022
Architecture	1. Research into architectural designs 2. Responsibility Driven Design 3. Structural Diagrams and Behavioural Diagrams 4. Justification of architecture	28/11/2022 28/11/2022 29/11/2022 28/11/2022	16/01/2023 28/11/2022 22/01/2023 26/01/2023
Risk Assessment	1. Research into appropriate ways to demonstrate a risk register 2. Create risk register	24/11/2022 25/11/2022	25/11/2022 01/12/2022
Implementation	1. Code 2. List of 3rd party libraries and assets	05/12/2022 23/01/2023	31/01/2023 30/01/2023
Method and Planning	1. Software engineering methods 2. Team organisation 3. Systematic plan for the project	23/01/2022 23/01/2022 23/01/2022	29/01/2023 29/01/2023 29/02/2023
Testing	1. Against the requirements	20/01/2023	29/01/2023
Website	1. Create website 2. Add the necessary details for the website	27/11/2022 27/11/2022	27/11/2022 31/01/2023

As part of our team-forming session, we created a plan for when each part of the project would need to be completed. The first thing on the agenda was to identify whether we would need to work over the autumn term break. The group decided that it would benefit the project if we continued to work during the break, however, allowing a short period of recession over Christmas and New Year's. Considering these aspects, we can develop an initial plan. Seen here under the heading first iteration.

The initial plan worked as follows: complete a draft of the requirements and risk management simultaneously by splitting up tasks amongst the group, then move on to the Architecture of the software and complete a draft before starting to code. During the period of 05/12/2022 to 06/01/2023 we would complete the development of the code in its entirety, as well as write a first draft of the Method and Planning as much as we could at that point in time. In the New Year, test and ensure that each of the requirements has been met. Finalise copies of the deliverables so that they are ready to be submitted. This plan can be seen here.

It detailed that we would first complete our client meeting as a team. From there, we can develop the requirements using the notes from the client meeting and the product brief. With a better understanding of how the life-cycle of the project is going to look, we can formulate a number of risks that could impact the development of the project while simultaneously completing the requirements. This is because requirements and risk register do not depend on each other and can be completed independently. This part of the project met the group's expectations and was completed on schedule. Therefore, there is not much differentiation between the first iteration and the second iteration of the weekly plans in terms of timings. However, we felt it necessary to add extra detail to the chart. Such as specific tasks that were completed that week with the initials of the team members who were completing them.

As a result of completing prior work, we have a clear understanding of the project as a whole and can now continue onto more technical aspects of the system.This is when we start considering the software architecture and delve deeper into the classes needed for the system. During this part of the project, we prioritised generating the Sequence and Behavioural diagrams, thus ensuring that there is a detailed description of the project that the group can follow. This is because the implementation of the project was dependent on the completion of the UML diagrams. Of course, the justification of the architecture could not be completed until the architecture was developed. As a result, writing the report was not a top priority. Thus, the Architecture report was neglected until the previous work was fully completed.

This was not in our initial plan; this is because the generation of the diagrams took longer than we expected. There were a lot of aspects of how the code would work that needed to be discussed, changed and constantly updated, such as how the chef will be controlled, where the verification of the correct recipe will take place in the code (with the cook or service station), and so on. As a result, the Architecture part of the initial plan overran, and therefore the report aspect was neglected to ensure that we did not fall further behind on the project. This is not reflected in the Gantt Charts (third iteration) however since the architecture and UML diagrams were technically complete and team member Megan Miles had written about the Responsibility Driven Design process during this time, which is reflected in the chart.

Next to be completed is the actual coding and development of the Piazza Picnic. Originally, we devised the plan such that each member of the team would contribute to the code. However, very early on, we realised that this was not feasible, and we would have to split the team into two parts: one part completing the code and the other part continuing with the deliverables. Therefore, more work can be done simultaneously. This correction was added to the Gnatt chart here where the fourth iteration displays every member contributing to the code and then the fifth iteration shows that only half of the team are continuing with the code.

Unfortunately, in this rendition of the plan (fifth iteration), we concluded that we would like the prototype to be completed by 15/01/2022 which did not come true. Due to other commitments (other modules and exams), there was a stagnation in the development of the code and the deliverables. As a result, the plan needed further editing. These can be seen here. The difference is that we had less time to ensure that the report flowed properly and that the code was well documented. However, the time added at the end was there in the event that we fell behind and needed extra time to complete certain tasks.

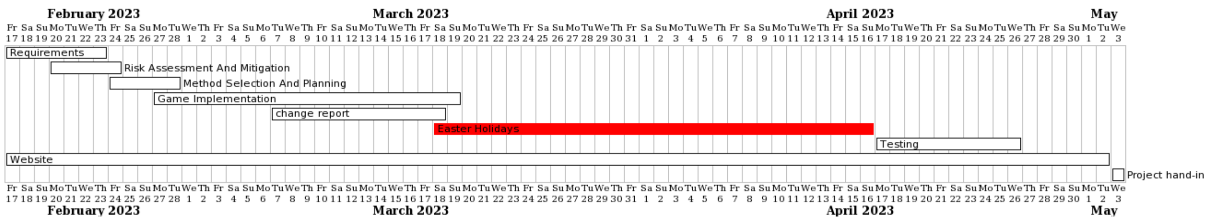
According to the project's final plan (sixth iteration), all ongoing tasks (whether coding or report writing) must be completed by January 23 before our meeting. During this meeting, we would tidy up the website, adding all the necessary information and ensuring that it was readable. At the end of the meeting, we would assign each person a deliverable that they had no involvement in writing. Then over the weekend, each person would read over the deliverable to ensure that it met the needs of the project. Thus, in the succeeding meeting on 30/02/23 we can then discuss any necessary additions or edits.

Assessment 2

Taking over this project for the second phase as the assessment, we created our meetings logbook to assist in keeping track of any progress made by each team member. We also created a new Gantt chart to give an initial outline on when we wanted different sections of the project completed by. This provided a visual plan to follow rather than listing everything in a table. The chart could also be altered throughout the duration of the project should the project be behind or further ahead than initially anticipated.

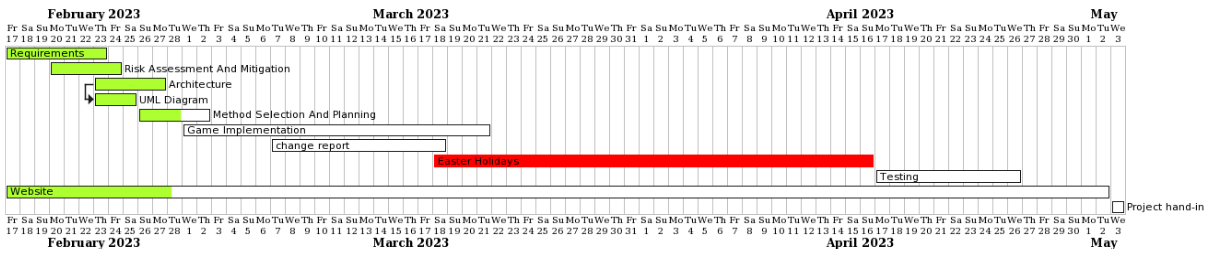
First iteration

Our first Gantt chart created, showing the basic outline we planned to follow during the project.



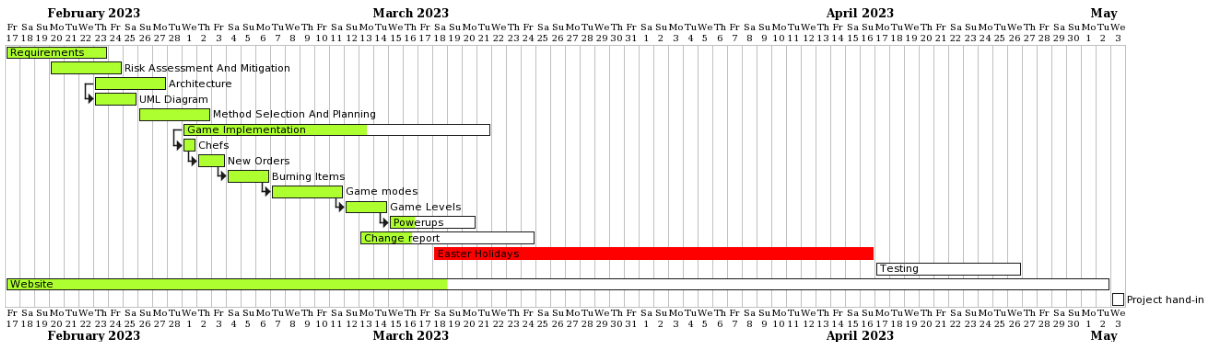
Second iteration (starting implementation)

Created when implementation began. Some progress was made, though we decided to include a section for architecture and UML diagram creation, pushing back our method and implementation deadlines.



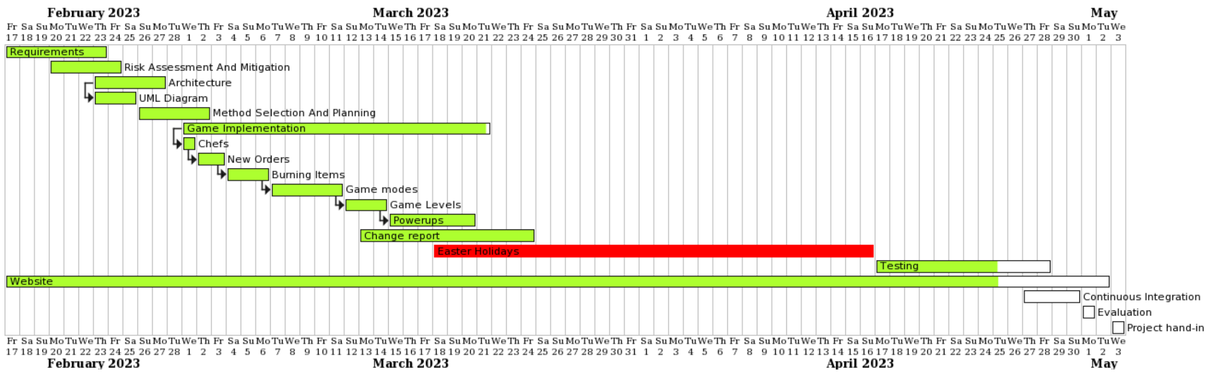
Third iteration (pre-Easter holidays)

Created before the Easter holidays. This included expanding game implementation into its different sections so that we knew exactly what had been changed. The changes report was also started at this point as shown.



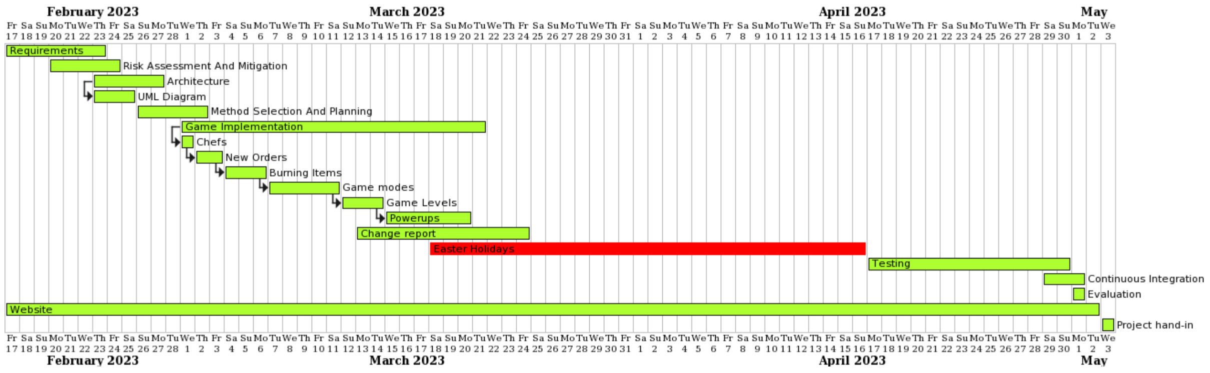
Fourth iteration

Created during testing to reflect the progress following the Easter break. Along with this, we also felt that we needed to do some additional work over Easter to stay on track of our previous chart.



Fifth iteration

Our final Gantt chart, showing that the end of the project had been reached.


February 2023
March 2023
April 2023
May
Requirements
Risk Assessment And Mitigation
Architecture
UML Diagram
Method Selection And Planning
Game Implementation
Chefs
New Orders
Burning Items
Game modes
Game Levels
Powerups
Change report
Easter Holidays
Testing
Continuous Integration
Evaluation
Website
Project hand-in