4. Method selection and planning

Part a)

Within our team we discussed the tools that we thought were the best fit to optimise 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. 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 have not 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 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 within 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

Part b)

To effectively work as a, we decided that it was important that everyone was assigned tasks that they were comfortable doing, and were tailed towards their individual strengths.Since we had already worked on assessment 1, we already knew what people preferred to do and what their strengths were, which made it easy to have a short discussion about what to do. We decided to work the same as last time, however, whereas last time each person worked mainly individually, we decided that everyone should work in groups of 2 and try to communicate when anything needed to be changed.

During the discussion, we also decided to follow the same structure as last time, with no defined roles for anyone apart from what their tasks are, and that if they needed help they would just ask their partner or the whole group. The jobs assigned were based on the individual's strengths and weaknesses. The individuals do not have to solely work on their part of the project and do not have to complete it all by themselves, however it was used to give a rough outline on what to work on.

Name	Section
Dan	Change Report
Niko	Change Report
Matty	Implementation
Adam	Implementation/Website
Ani	Testing
Dominik	Testing/CI

Part c)

When we started the project, we highlighted the different tasks that had to be completed for each section. This is demonstrated in the table below:

		Start Date	End Date
Change Report	● Update Requirements ● Update Architecture ● Update Risk Assessment ● Update Method and Planning	01/03/2023 08/03/2023 15/03/2023 19/04/2023	07/03/2023 14/03/2023 22/03/2023 21/04/2023
Implementation	● Add new Ingredients ● Add more Chefs	01/03/2023 08/03/2023	07/03/2023 08/03/2023

	● Add Power Ups ● Add Shop ● Add Save	08/03/2023 08/03/2023 17/04/2023	14/03/2023 20/03/2023 23/04/2023
Testing	● Summarise testing methods ● Carry out test, provide precise URLS ● Report on tests	08/03/2023 15/03/2023 20/04/2023	14//03/2023 20/04/2023 25/04/2023
CI	● Summarise CI methods ● Set UP CI infrastructure ● Give brief report	15/03/2023 19/03/2023 20/03/2023	18/03/2023 20/04/2023 25/04/2023
Website	● Add all new links to work	01/05/2023	01/05/2023

As part of our team-forming session, we outlined the work that needed to be done and created a rough timeline of when we hoped to start and complete all of these. The group unanimously decided that we would keep the work during the Easter break to a minimum to give us time off as well as time to revise for other modules, and with this in mind, managed to come up with the plan as seen above.

The beginning focus of our tasks was decided to be the Change report as well as get started on the Implementation. We decided to split into groups to complete these as per Part B, and at the points where Testing and CI was not scheduled to start yet, the group members assigned to these could revise for their own modules and help out when needed

We decided we would just meet weekly at the ENG1 sessions as that was all that would be needed. As a result of the work we had done for Assessment 1, we were more comfortable with the work were had to carry out this time as this is why we decided that we would not need an extra meeting.

By starting with the change report this will make sure we understand and know what information we are going to update within the implementation. Following this, if we discover any other methods or architecture points that we want/need to add or change, it would be easy to look at what we were planning to do and compare it to what we are trying to do now.

Of course the Implementation and Testing would come next. Putting these 2 together means we can create and implement functions and tests for these functions at the same time. Naturally this means any problems that we encounter can be fixed as we go along

The CI must be done as after/ during the implementation of the game, which is why it is done later

Finally with all sections done, these new documents can be linked to the updated website