Engineering 1

Group Assessment 2

Planning Document

Cohort 2

Team 12 & Team 11

Part A: Method and Tool Selection

For our project we decided to use the **Scrum** method as we are delivering a small piece of software with a small team, and it is essential for us to have a flexible working environment due to team members' different schedules. Scrum is an agile method, and we chose agile because there may be changing requirements, and we believe it is crucial to have constant team reflection and development.

Scrum involves dividing development into 'sprints', each of which are a week long and preceded by a planning meeting. At the end of each 'sprint' and often much more often, there is a team review of what everyone has done and any problems they came across. "Scrum employs an iterative, incremental approach to optimize predictability and control risk."[2]

We used the following tools for completing tasks on our project:

- Trello and Github Projects for task management,
- Google Docs for writing up documents,
- Discord for communication and scheduling of tasks, meetings and weekly focuses (announcements),
- Github for source control and code development/maintenance and as result feature management and review,
- PlantUML for creating class diagrams,
- Google Drive to store documents and other files,
- Javadoc to generate HTML documentation of the source code

We used **Github** (alongside Github Pages) as the main hub for our project. It is where we uploaded the game files and relevant documentation. For our team's website, we decided to use Github Pages (although alternatives such as Google Pages were discussed). Since our project was uploaded onto Github, we thought it would make sense to use **Github Pages** alongside it, as opposed to Google Pages.

Our team started off with emails and Google Chat as initial communication, but as time went on, we switched over to **Discord** because the majority of our team was very familiar with it, compared to the alternatives. It also provides multiple useful features such as the ability to create a private server and channels for different uses within it. Zoom was used during official supervised practical sessions.

Github is very suitable for agile software development as it allows multiple people to work on features of the code simultaneously with their own local repositories and then push and pull changes to/from the main/global repository on the Github server. All pull/merge requests can be reviewed and all changes are labelled so that it is clear what has been done at each stage of the work. Developers can also create branches with new code and then merge the branch with the main to include it, which makes changes easy to do and there is no risk of breaking the working code in the main program. Github makes it very easy and efficient to do this.

Google Drive is a cloud-based storage that provides 15GB or storage for free and also comes with a suite of office tools including Docs (word processor), Sheets (alternative to MS Excel) and Slides (alternative to MS PowerPoint). It is a great tool for aiding agile development as it allows all team members to see changes that have been made and access all files at the same time. There are many alternatives to Google Docs such as Dropbox, OneDrive and Sync. However, Google Drive is the most common choice and also most familiar to all members of the team.

We chose **Trello** for keeping track of our project as it is straightforward to use and encourages cooperation on tasks, and information can be adapted easily. Initially, our team used Google Sheets instead of Trello to create and assign tasks for team members, however, after we discovered a more dedicated platform to do such things in, we switched over to Trello. It was both easier to use and easier for team members to see what needs to be done by them.

For task management we (Team 11) also used **Github Projects**, a kanban board-style todo list allowing for the creation of boards and cards to go on those boards, which all developers can then access and modify. This is ideal for our chosen development methodology as it allows for tasks to be broken down into smaller parts and put on a board where anyone can pick tasks, develop them, then move the task to the “completed” board. We favoured Github Pages over Trello as we don’t need the additional features and having everything in one place is far more convenient.

For creating class diagrams we chose to use **PlantUML**, an open-source tool that allows users to create UML diagrams from a plain text language. It is relatively easy to use and access which is the main advantage over other alternatives, such as web-based platforms Lucidchart and Creately.

Additionally, as we needed to create documentation for our entire source, we decided to use **Javadoc** to generate HTML documentation files which we could easily transfer to our project website. Javadoc reads through Java source files and generates documentation based on the method names and Javadoc-style comments that accompany them. Other solutions include Doxygen or Sphinx however these tools are primarily for C++ and Python respectively, where Javadoc is designed specifically for use with Java.

Part B: Team Organisation Approach

Both our teams had their own Discord server where members could ask questions or discuss their tasks anytime. It allowed each member of the team to have a clear understanding of the progress, decisions, project schedule and be able to share their own ideas. The Discord server that we created had multiple channels for different parts of the project (See Appendix, Figure 1).

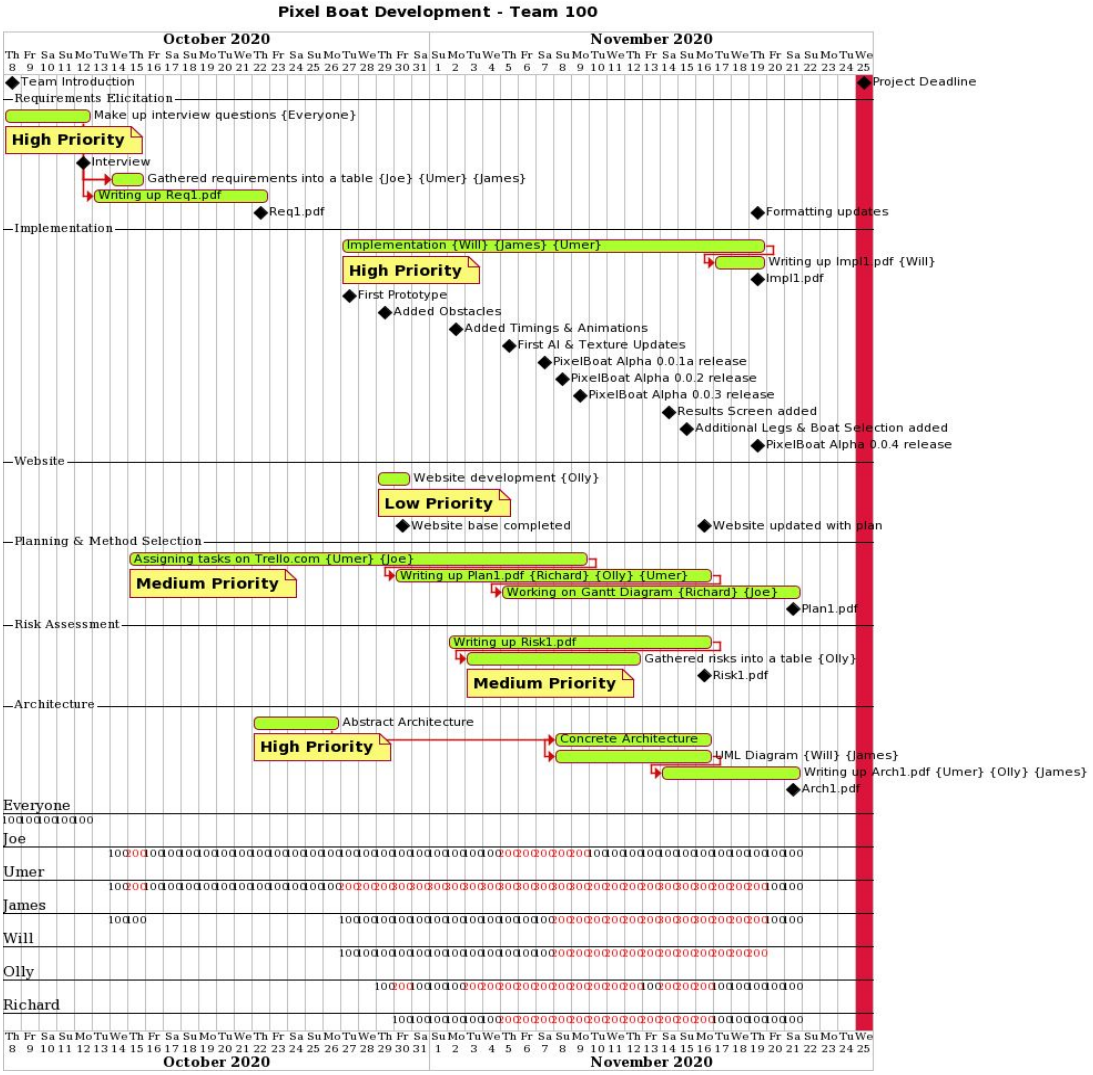
Team 12:
Our team was split into subteams for all the different parts of the assignment (coding, working on documentation, etc.) meaning that each subteam was debriefed separately by the team leader. Each task had a leader as well as one or many collaborators to ensure good implementation. These subteams were very adaptive, changing people often, given the required skill set for the task. All tasks completed were peer-reviewed by other members of the team.

Our approach is appropriate for the project because it involves frequent and plentiful communication and feedback between team members which is essential for building software like this. The project is small overall, but involves lots of different parts, so this approach is perfect as it permits us to organise into sub teams and get things done efficiently. This approach is also suited to our team because we have different skill sets - some team members were more suited for building the foundations of the code base whilst others were tasked with feature programming. We worked well independently as well as in subteams which were tasked with tackling goals which required subtasks to be completed. These subtasks were discussed and reviewed in large team meetings when possible and in asynchronous Discord reviews.

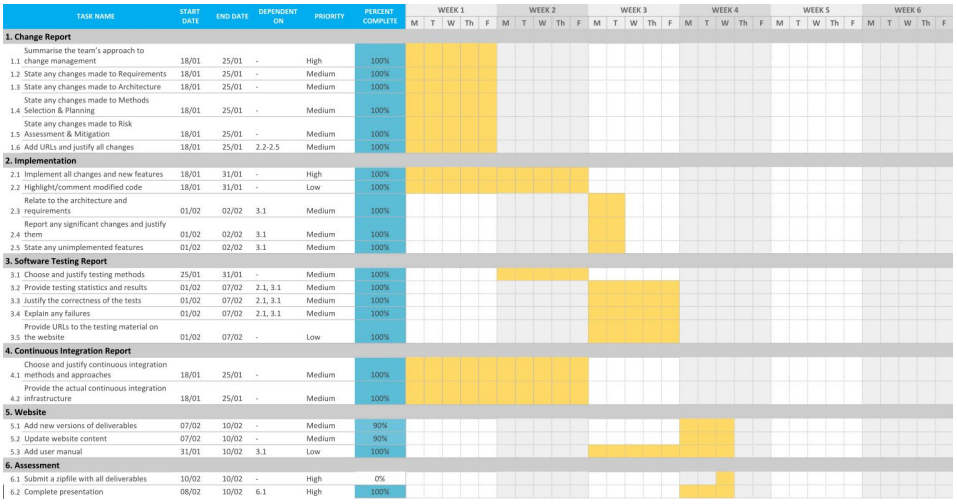
Team 11:
We tried to divide responsibilities evenly between all members of the team so that everyone has a task that suits them, given their skills and experience. Since we chose Scrum as our development framework, all tasks were done in small, week-long sprints, and during the sprint, each team member worked on their specific tasks. After each sprint we had a weekly meeting where we discussed what has been done and what is yet to be completed as well as assigned new tasks.

This method is appropriate for our team and the project as it is perfect for small teams and provides everyone with an opportunity to contribute and stay informed about all decisions made within the team. In addition, it allows a certain degree of flexibility, as the whole project is divided into small tasks that can be done by any member of the team in case if someone is sick or unwilling to participate. Additionally, our method allows us to react to changes and make new decisions without complications.

Part C: Project Plan



Gantt Diagram for Assessment 1



TASK NAME	START DATE	END DATE	DEPENDENT ON	PRIORITY	PERCENT COMPLETE
1. Change Report					
1.1 Summarise the team's approach to change management	18/01	25/01	-	High	100%
1.2 State any changes made to Requirements	18/01	25/01	-	Medium	100%
1.3 State any changes made to Architecture	18/01	25/01	-	Medium	100%
1.4 State any changes made to Methods Selection & Planning	18/01	25/01	-	Medium	100%
1.5 State any changes made to Risk Assessment & Mitigation	18/01	25/01	-	Medium	100%
1.6 Add URLs and justify all changes	18/01	25/01	2.2-2.5	Medium	100%
2. Implementation					
2.1 Implement all changes and new features	18/01	31/01	-	High	100%
2.2 Highlight/comment modified code	18/01	31/01	-	Low	100%
2.3 Relate to the architecture and requirements	01/02	02/02	3.1	Medium	100%
2.4 Report any significant changes and justify them	01/02	02/02	3.1	Medium	100%
2.5 State any unimplemented features	01/02	02/02	3.1	Medium	100%
3. Software Testing Report					
3.1 Choose and justify testing methods	25/01	31/01	-	Medium	100%
3.2 Provide testing statistics and results	01/02	07/02	2.1, 3.1	Medium	100%
3.3 Justify the correctness of the tests	01/02	07/02	2.1, 3.1	Medium	100%
3.4 Explain any failures	01/02	07/02	2.1, 3.1	Medium	100%
3.5 Provide URLs to the testing material on the website	01/02	07/02	-	Low	100%
4. Continuous Integration Report					
4.1 Choose and justify continuous integration methods and approaches	18/01	25/01	-	Medium	100%
4.2 Provide the actual continuous integration infrastructure	18/01	25/01	-	Medium	100%
5. Website					
5.1 Add new versions of deliverables	07/02	10/02	-	Medium	90%
5.2 Update website content	07/02	10/02	-	Medium	90%
5.3 Add user manual	31/01	10/02	3.1	Low	100%
6. Assessment					
6.1 Submit a zipfile with all deliverables	10/02	10/02	-	High	0%
6.2 Complete presentation	08/02	10/02	6.1	High	100%

Gantt Chart for Assessment 2

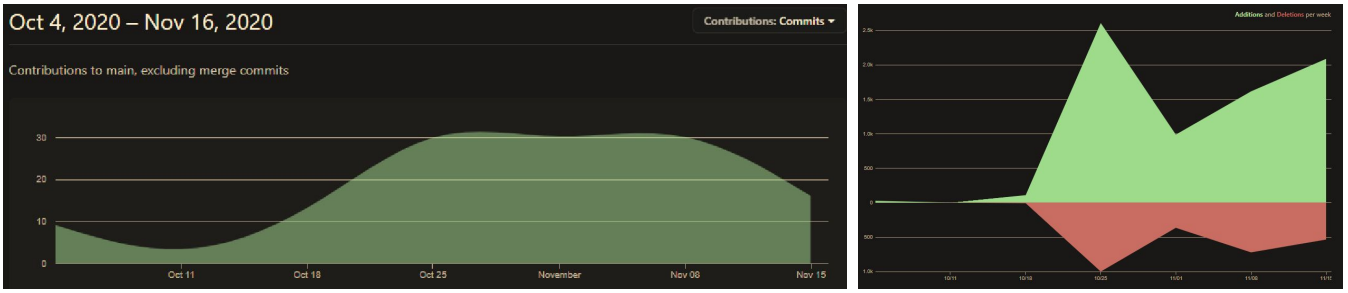
Plan Strategy

The plan evolved through ongoing meetings and “Scrums” where we discussed what tasks needed to be completed next and what each team member had been doing. Initially, we had a rough idea of what the plan was from our critical path, and after the first couple of weeks we elicited an actual plan using a Gantt chart (shown above). With our agile development methodology, we altered the plan frequently to adapt to any changing circumstances we came across such as absence of team members and technical issues, and some examples of weekly changes to the plan are shown on our project website. Furthermore, with the current pandemic, we had difficulties such as the presence and ability of team members being affected. However, we overcame these issues by keeping in contact daily with Discord and using Trello to keep track of tasks.

For Assessment 2 we created a new Gantt chart (shown above) with all the tasks that must be completed for the second part of the project. We decided to create a new chart to keep all things separate and focus on new tasks that were yet to be completed.

Start and end dates of tasks were shifted frequently due to various reasons. However, Scrum allowed for flexibility, and we were able to react to all changes and complications immediately. For example, we could appoint a different team member to finish a task, in case where the member who was originally responsible for it was unable to complete it, for example, due to the lack of knowledge/motivation or other reasons.

Appendix



Github Charts. Evidence of implementation progress (Team 12)

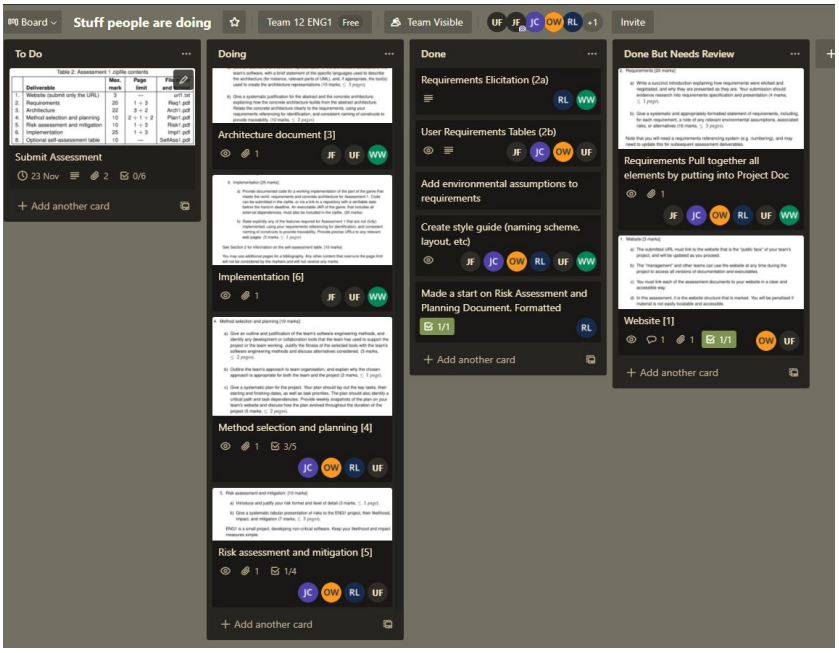


Figure 2: Trello board for documentation as of 09/11/2020 (Team 12)

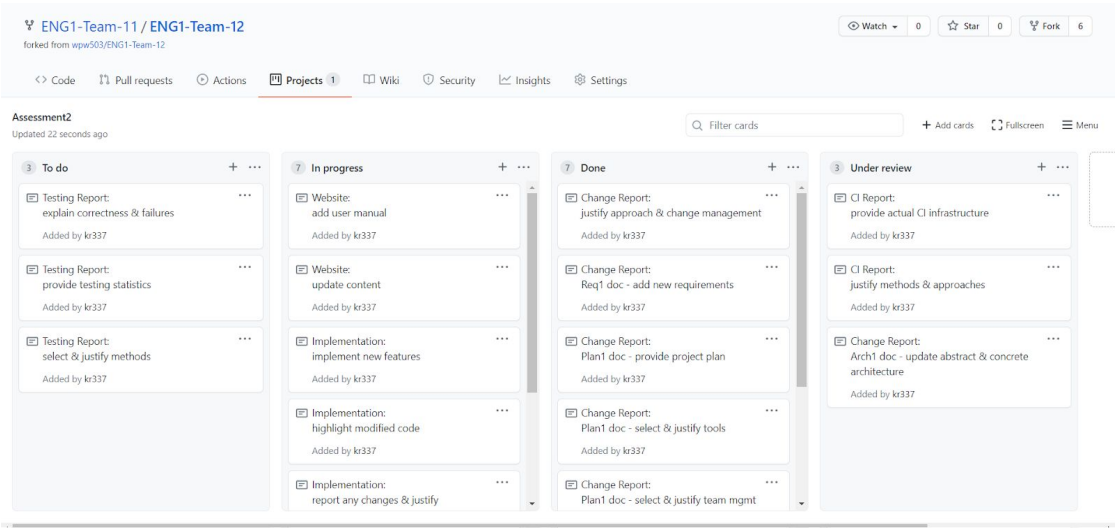


Figure 3: GitHub Projects board for documentation as of 04/02/2021 (Team 11)

Bibliography

[1] I. Sommerville, Software Engineering, Pearson Education, 2008, pp. 74-98.
[2] K. Schwaber and J. Sutherland, The Scrum Guide™, 2017 pp. 1-19