Revised Methods and Planning2

Cohort 1, Team 1:

Connor Blenkinsop
David Luncan
Emily Webb
Muhidin Muhidin(MO)
Sam Laljee
Sooyeon Lee

Engineering Approach:

We used the agile approach for our software engineering project, since this allows us to develop features rapidly, see if they work, test them, and tweak them if they are deemed to be insufficient. The agile method allows multiple members of the team to work independently on their own tasks, then have their work reviewed and changed accordingly. This makes it significantly quicker and more efficient to create our codebase and have all features that need to be developed, developed to their highest quality. Once all the prototypes had been developed individually, we then compiled them into the main code base where everything was combined to create the finished project.

Tools Used:

Documentation:

We used Google Drive and Docs to track what needed to be done, keep a meeting log and store our files. The Google drive also contained shared files such as the deliverable outlines and the project brief, which were used to complete the project accurately and match the project guidelines. Google Docs were suitable because they allowed us to work synchronously and collaboratively. Being cloud based we didn’t have to be concerned about losing work and it was accessible from any device at any time. For these reasons it was more appropriate than applications such as Microsoft Word. It is also highly compatible with many of the other tools we elected to use.

Version control:

GitHub was one of the main resources we used to aid our team in keeping up to date with what had been accomplished and to share code written by multiple people. This resource allowed us to create the project as a team, using branches and merges to simplify and greatly improve the efficiency of the development cycle. GitHub lets each developer commit and push changes to the project to be reviewed and added to the final product. This allows us to keep track of development, seeing who did what and at what time, revert any changes if we need to or discard them completely. Thus, preventing conflicts and making collaborative programming possible.

GitHub was essential in the development cycle, giving the capability of sharing and modifying code written by multiple people synchronously. It also allowed us to view the history of the code base, giving us the opportunity to modify any of our work whenever we wanted to. However, we could have used programs such as GitLab or tortoise git. We didn’t use GitLab because most of our team were familiar with GitHub and already had an account, so it was seamless in our integration. We decided against tortoise git because most of the developers liked the convenience of having a GUI to guide the process of the version control system, we weren’t too keen on the idea of having to use the command line to do everything, even though this might have been the preference of some of the developers.

Communication:

We chose to use Facebook Messenger as our main method of communication. This allowed us to send links, discuss our availability and update each other on our progress between meetings. A group chat was preferential over private messaging as everyone was notified and kept in the loop. When we were unable to meet in person, for example over the winter break or when in isolation, we used Zoom to ensure we could still interact synchronously even when we weren't physically together. This meant we could communicate more effectively than over Messenger and keep each other accountable. We choose these tools because we all had access to and familiarity with them.

Implementation:

Following the product brief, the implementation was required to be written in Java. We considered various development frameworks such as Unity but ultimately, we chose to use LibGDX to implement this, as it well-documented the platform we had the most experience with. It is suitable for this project as it offers a vast range of features and is cross-platform, supporting MacOS, Linux and Windows – the operating systems we were required to cater for.

UML diagram:

We thought Draw.io would be the most suitable tool for creating our architecture diagrams as it is compatible with Google Drive and has a free trial. Its built-in shape library keeps creating diagrams quick and simple.

Gantt chart:

We chose to use Microsoft Excel to create our Gantt chart. We found an appropriate pre-set we could edit to fit our purposes simplifying the process of setting out our plan. Excel is also compatible with Google Drive.

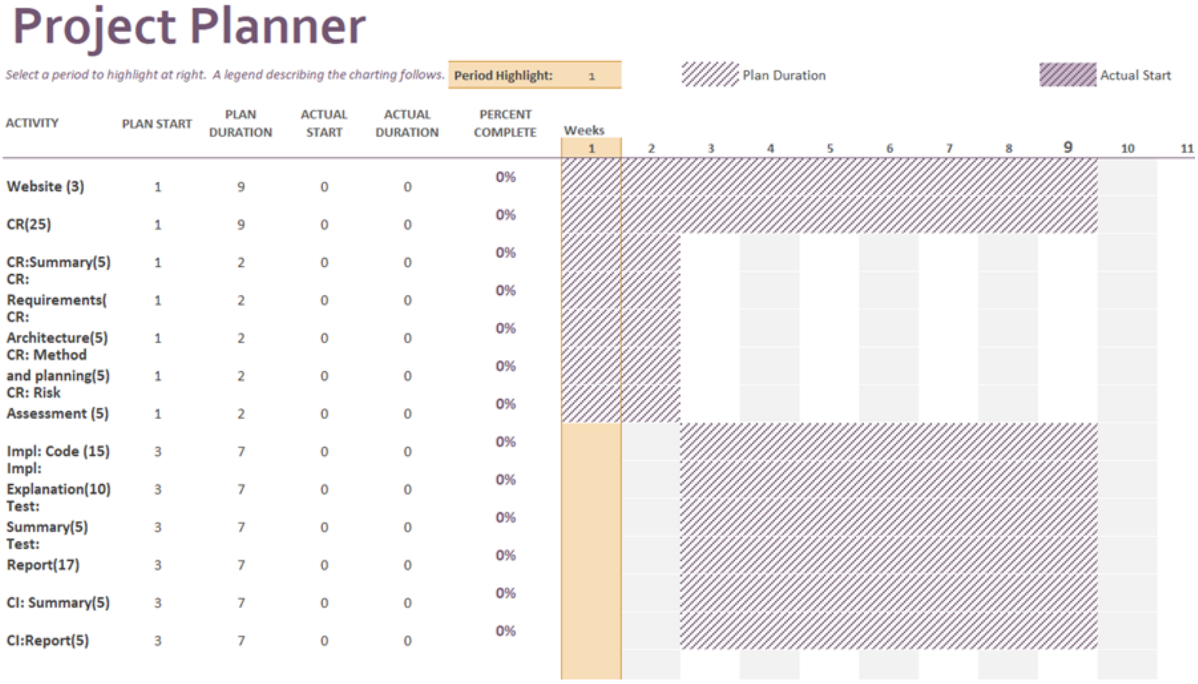
Team Organisation:

Our approach to team organisation avoided having an overall group leader, but instead section leaders, who had the most knowledge of that aspect of the software development process going into this part of the project. They would be responsible for ensuring their section was complete on time. We were all kept accountable by each other making sure each person was on track and had all the information they needed.

We shared the workload among us in the first session to ensure the work was evenly split. We divided the change report amongst the people who had been working on these deliverables previously as they were the most knowledgeable in each respective section. We then appointed teams of two to both the testing and implementation deliverables. We decided it would be best to have two people working on each section as they could work together to overcome problems and provide support to one another. With another person working on continuous integration, this left one person not fixed to any particular section but available to anyone who required help or was falling behind to make sure no one was struggling. We found this was helpful particularly in implementation where we had a lot of new features to implement. We then split the testing into black and white box testing and the implementation into the features that needed to be added so that they could be assigned between us as evenly as possible. This division of labour remained flexible while also making sure everyone knew what they should be working on between each meeting.

Plan and Description:

We made an initial Gantt chart to plan out what we hoped to achieve and updated this each week to reflect our progress and make sure the plan was realistic. The chart breaks down the deliverables we were required to produce and their component parts. Each component was then assigned to one or two people responsible for its completion, ensuring workload was equally distributed based on the number of marks it was worth.



Project Planner

Select a period to highlight at right. A legend describing the charting follows. Period Highlight: 1

▨ Plan Duration ▨ Actual Start

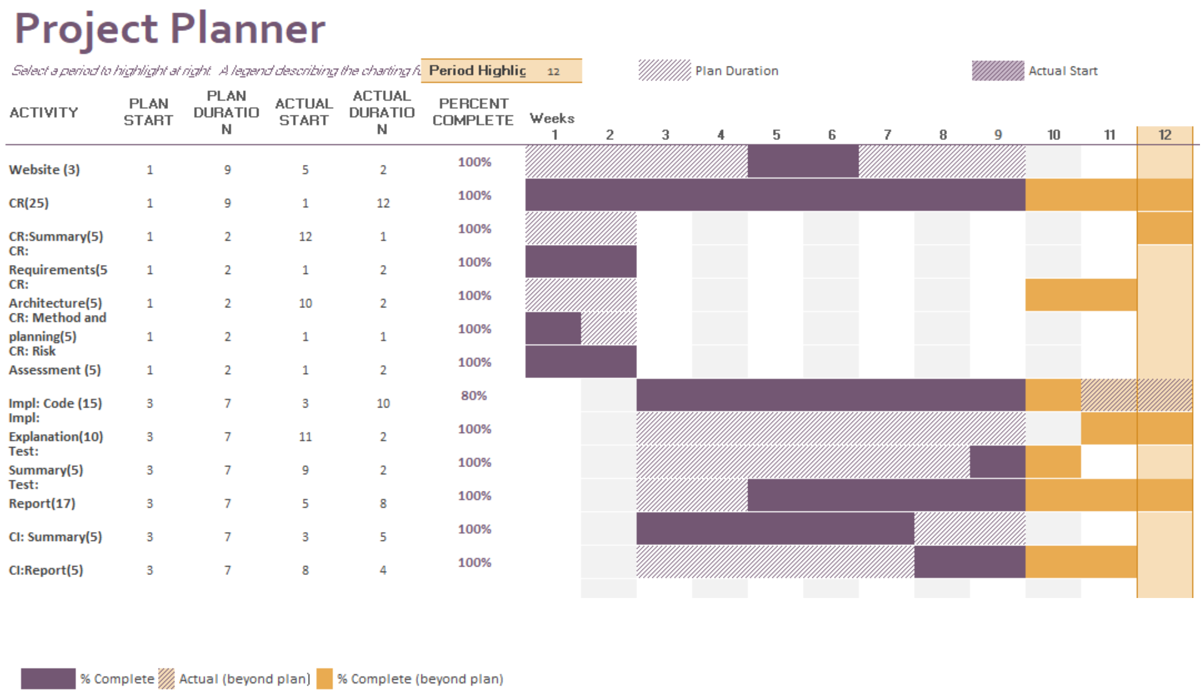
ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Website (3)	1	9	0	0	0%
CR(25)	1	9	0	0	0%
CR:Summary(5)	1	2	0	0	0%
CR: Requirements(	1	2	0	0	0%
CR: Architecture(5)	1	2	0	0	0%
CR: Method and planning(5)	1	2	0	0	0%
CR: Risk Assessment (5)	1	2	0	0	0%
Impl: Code (15)	3	7	0	0	0%
Impl: Explanation(10)	3	7	0	0	0%
Test: Summary(5)	3	7	0	0	0%
Test: Report(17)	3	7	0	0	0%
CI: Summary(5)	3	7	0	0	0%
CI:Report(5)	3	7	0	0	0%

Our initial plan was to attempt to complete the project in 9 weeks leaving us two weeks as a buffer in case we fell behind schedule. This would require working through the Easter break when the team would be away from York and only able to meet through Zoom.

We chose to conduct bi-weekly meetings to help us stay on track and keep up to date with each other. Meeting in person helped us to communicate more easily and showcase what we had been working on, however, we also messaged or met online when necessary.

Over the course of the project, we fell behind slightly especially with the difficulty of working apart over the break. The buffer weeks we had left ourselves helped us to catch up but unfortunately, not all features could be implemented. This did also have an effect on testing as the new features were not complete in sufficient time to be tested thoroughly.

We updated our Gantt chart each week and included it in the weekly log so that we could see our progress over the weeks and how much time we had left. The final chart is shown below:



Project Planner

Select a period to highlight at right. A legend describing the charting fe Period Highlig 12

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Website (3)	1	9	5	2	100%
CR(25)	1	9	1	12	100%
CR:Summary(5)	1	2	12	1	100%
CR: Requirements(5	1	2	1	2	100%
CR: Architecture(5)	1	2	10	2	100%
CR: Method and planning(5)	1	2	1	1	100%
CR: Risk Assessment (5)	1	2	1	2	100%
Impl: Code (15)	3	7	3	10	80%
Impl: Explanation(10)	3	7	11	2	100%
Test: Summary(5)	3	7	9	2	100%
Test: Report(17)	3	7	5	8	100%
CI: Summary(5)	3	7	3	5	100%
CI:Report(5)	3	7	8	4	100%