Method Selection & Planning

Group 18

Team B

Olivia Betts
Zac Bhumgara
Nursyarmila Ahmad Shukri
Cameron Duncan-Johal
Muaz Waqas
Oliver Northwood
Teddy Seddon

Software Engineering Methods

Our team uses Agile software development as our software engineering method. Short development cycles are used in the agile project management technique, which prioritises continuous improvement in the creation of products and services. Requirements and solutions are developed in collaboration amongst self-organising cross-functional teams. We chose this to ensure we were constantly checking the code we had produced against the requirements.
Other development methodologies considered but ultimately discarded were the waterfall method (outdated and not useful for changing requirements) and spiral (ideal for large, risky projects - neither of which this is).

One of the collaboration tools that we have used is Github. It is used to store our code files as it makes it easier for collaborating on our software project. It allows multiple people to work on the same codebase at once, as well as cloud storage, and the ability to manage code collisions

We also used Google Docs to write our deliverables, again to allow multiple people to edit the same document. These were stored in a shared Google Drive, so that they could be accessed easily and from multiple devices. Another bonus of using Google Docs was that the files can easily be downloaded to .pdf format, which was the required format for the deliverables.
We considered using OneDrive, however it took more effort to set up, and more annoying to ensure everyone had access; so we didn't use it in the end.

Other than that, we have created a group email so that it is easy to send an email message to everyone at once and also to ensure that everyone receives any new updates through email.

Team Organisation

Our team split into smaller sub-groups, each of which focused on one section of the deliverables. We then had one spokesperson (Zac Bhumgara) who coordinated communication with the customer, for example, emailing to check about the requirements, as well as between sub-groups. It was important to have one main point of contact, as otherwise multiple people might have been sending the same messages separately to the customer.

In these sub-groups, each organised themselves, whether that was working in a pair, or working in up to a group of 4. This was ideal, as it meant a sub-group could meet and discuss their specific problems without having to coordinate the entire group to do so. This approach worked for the project as a team of 7 people would have been too large for everyone to try and know everything happening at once. It would also have been unnecessary - for example, the method selection and planning requirement was too small for 7 people to work on at once.
The majority of sub-groups had 2 or more people. This meant that if one person was unable to complete their portion of the work - for whatever reason - there was at least one other person able to step in and do it. The exception to this was the website, which was worked on by just Olivia Betts. However, in this case the site code was also entirely on Github and documented, so another person could easily take up the work if they couldn't complete it.

We organised ourselves as follows:

Website	Olivia Betts
Requirements	Olivia Betts, Zac Bhumgara, Nursyarmila Ahmad Shukri
Architecture	Oliver Northwood, Nursyarmila Ahmad Shukri, Muaz Waqas, Teddy Seddon
Method selection and planning	Nursyarmila Ahmad Shukri, Cameron Duncan-Johal
Risk assessment and mitigation	Olivia Betts, Cameron Duncan-Johal
Implementation	Muaz Waqas, Oliver Northwood, Zac Bhumgara, Teddy Seddon

This meant we could ensure everyone got a fair share of the work, so nobody was given too little or too much (a number that worked out to ~12.8 marks of work per person)

Key Tasks

Task	Start date	End date	Dependencies
Requirements Elicitation	16/11/22	30/11/22	N/A
Initial Architecture	23/11/22	30/11/22	Requirements elicited (doesn't have to be perfectly finalised)
Risk Assessment	7/12/22	14/12/22	N/A
Implementation	18/1/23	31/1/23	Requirements, Architecture
Architecture evaluation	25/1/23	31/1/23	Implementation, Initial architecture

(We have also made a Gantt chart laying this out visually, which can be found on our website TeamBEng1.github.io)

The above chart uses very broad definitions of key tasks. We found this suited our way of working best, with individual smaller teams being able to dedicate the time they saw fit to complete the work in ways that worked for them, whilst remaining within the overall deadline.

As an example, the Risk Assessment was further split into two key tasks:

Task	Start date	End date	Dependencies
Identify risks	7/12/22	8/12/22	N/A
Risk writeup	9/12/22	14/12/22	Identify risks

We did similar for each other broad category.