Sprint 2 Review

Pair programming

Pair programming in sprint 1 was not always done in pairs, so the aim for sprint 2 was to make sure that this was always the case. This was done to great effect, as we made sure to stick to one driver and one navigator, when integrating code and working/helping each other with features. The driver explained the code to the navigator and made sure they understood what the code was doing. We switched around the driver and the navigator often when integrating code. This helped immensely for integrating code and fixing errors.

Burndown chart

The burndown chart was not posted on GitHub daily in sprint 1 so progress was poorly recorded. For sprint 2, Melvin kept account of the burndown chart and story points for each member of the team. He kept and updated the sprint backlog, and we estimated the user stories better than in sprint 1. The status of the burndown chart was recorded and changed every day and the progress of the project was uploaded daily to GitHub so it is easy to see how it changed over time. The source of the template used for the burndown chart and log was sourced in comments on GitHub. The progress for this sprint was very well recorded on the repository.

Daily Scrums

Scrums were not recorded in Sprint 1. For Sprint 2, Arran kept account of every daily scrum. Jason stayed scrum master and each scrum was scheduled and notes taken to record what each team member said. The record includes what the member of the team has done and what they were planning to do. The scrum notes were updated daily on a Readme on GitHub. The date of every scrum was recorded but not the time. At the end of every scrum, if a team member was struggling, they would let the rest of team know so they could arrange pair programming and extra help. This was effective for highlighting issues in development.

GitHub procedure

In sprint 1, GitHub was used but the advanced features were not used and there were few pull requests. For Sprint 2, we aimed to follow standard GitHub Procedure, by making branches and using them to integrate code, and making pull requests before merges occurred. Jason made sure this procedure was followed. To avoid too much work overwriting the master branch, a develop branch was created, as a staging area for work to sit before being merged into master. For features, branches were made to add features which would then be merged back into its parent branch. This was helpful for integrating code as the integrated code would be put into a pull request, reviewed and then approved and merged back. This would then be branched off to add more code, which would be integrated. Pull requests were commented on and reviewed. On the master branch, continuous integration was implemented, to make sure the master code compiled. A good effort was made to see that standard procedures were followed on GitHub.