Sprint 1

**What went well:**

The team started to come together, moving past the forming and storming stages and steadily approaching the norming stage, meaning that the level of trust and respect between team mates was beginning to solidify and team standards were beginning to develop, allowing for both present and future success.

**What didn’t go well:**

We wasted time in the first week of the sprint, implemented only a couple of stories, and consequently didn't meet our expected velocity. Specifically, the estimations made for the sprint tasks fell on the underestimated end, as we didn’t have the experience or comprehensive enough understanding of the technologies to be able to accurately estimate the difficulty of using and implementing them. For example, the main language the team is using to set up the database is PHP, which none of the team members had any real experience with. As a result, we heavily underestimated the time it would take to implement certain backend features, and in addition, spent a large portion of time learning how to use PHP in order to use it in the first place.

Another issue was that without having a comprehensive understanding and plan of what our final full stack would look like, the implementation was sometimes poor and inefficient, i.e. writing implementations in the frontend when maybe they were more suitable for the backend.

**Improvements for next time:**

For next time, it would be a good idea to spend the first sprint on completing the technical user stories, specifically, gaining experience and knowledge of the technologies required for the project. This way, estimations made on actual user stories will be much more accurate as we will have more insight on what we are dealing with. In order to achieve that, we can use sprint 1 to build spikes and see if it is feasible to use the technologies listed in our analysis of alternatives to build our software.

In terms of not being able to complete the required stories and falling short of our estimated velocity, in the first sprint, a possible solution would be to ‘do nothing’ about it. This is because, due to the team being newly formed and also having no prior experience of some softwares which were used for the implementation of our user stories, it took the team some time to

Sprint 2

**What went well:**

* Having the first Sprint as a guide on how things worked, instead of starting from scratch, made the whole Sprint 2 process much easier. The team had a better understanding of the way things worked as a whole and also had a better understanding of the PHP technology required, having used it for the first sprint.
* Estimation of tasks have been improved since sprint 1

**What didn’t go well:**

* There were times where there was a lapse in team communication. For example, two team members ended up working on and implementing the same user story which was an issue, as it resulted in lost time and effort which could have been used for working on other implementations and features. In addition to this, we left things too last minute and found there were issues and errors with some backend implementations for one of our user stories. This meant that we couldn’t finish our sprint tasks on time, and resulted in a decrease in velocity.
* Usage of git: no branching / issue tracking
* Not adhering to our responsibilities in our role in the team, and rather everyone was just doing code development.
* No quality assurance
* Weak foundation on Git knowledge
* Not a strong framework for efficient cooperation
* Dividing user stories between different developers resulted in confusion and conflict
* Unclear when the sprint started and ended
* Sprint retrospective was not done before sprint review

**Improvements for next time:**

* Build in time for perhaps an additional meeting halfway through the sprint to discuss the teams progress and make sure that everyone is on track for completing their user stories.
* More frequent stand up meetings - make sure they are fast and efficient (2 mins), held every 3 days.
* Create a central GIT document for the team to refer to
* Create checklist of all deliverables at the end of the sprint
* Use trello more often
* Have velocity tracking document
* Have code review for each branch merged
* Writing quality assurance after each user story’s completion to
* Refine each person’s responsibilities
  + Roy (project manager):
    - I’ll keep track of version control of google documents
    - I’ll set up trello for each sprint
  + Isha (scrum master):
    - Keep track of quality assurance & retrospective
    - Velocity tracking
  + Tarin (developer):
    - Code reviewer for merge requests
  + Jaskaran (product owner):
    - Keep track of sprint backlogs and product backlogs
    - Stay in touch with the client

Sprint 3

**What went well:**

* Better use of git - branching, merging, issue tracking
* Better velocity tracking, estimation
* Better use of trello to keep track of all the deliverables
* Better project management - ensuring quality assurance for each user story and ensuring we are adhering to project timeline (sprint start and end dates)
* We made sure all documentation are up to date

**What didn’t go well:**

* Issues with git - emerged from a weak foundation of the concept
* Code review was too informal and unstructured

**Improvements for next time:**

* Make sprint times longer (2 weeks)
* Ensuring all members are working on project within the git system that is connected to the remote repository
* Have a more robust code review process