Sprint Retrospective

# What went well

Afzal – Programming went smoothly. The team was consistent day to day in terms of time used working (punctuality was good).

Topu – Teamwork was good. Merging had no issues

Henry – Bugs that were found during development were quick to fix.

Matthew – Programming went smoothly, every task was continuation of the last task. Attendance was good.

Arnav – Document writing went smoothly. Review process of the documents were good.

Tony – The group we committed and motivated, we communicated well and worked well with each other.

# What didn’t go well

Afzal – Hardware Issues, unable to use lab pcs because they couldn’t allow connection to the database so was forced to use another teammates laptop to program. This caused me to be unable to push as myself to the git and instead had to push my changes as my teammate on GitHub. Peer programming was too unorthodox, should’ve peer-programmed on one PC together rather than using a big screen.

Topu – Couldn’t push from labs pc so was forced to pass the program to someone else to push.

Henry – There were merge conflicts during development, causing my own work to be replaced by the version on GitHub.

Arnav – Estimating the priority and complexity was difficult, particularly the priority as we did not meet the client until the sprint review.

Matthew – The test written for Test-Driven Development required more detail.

Tony – GitHub could have been used better

# What would you do better next time?

Henry - Be more informed on what documentation is needed.

Topu - Have more interaction between Scrum members in a scrum meeting.

Arnav - Be more organized in the way we store our documents in GitHub.

Tony – Make sure to bring up any issues no matter how small or if they were easily resolved in the scums

Matthew - More frequent commits for each function. Write more detailed test.

Afzal – Get better understanding of GitHub and effectively push/commit/pull request more often from my account so there is evidence.

# Actions (What will we do in Sprint 2)

1. Create a test plan before implementation in code.
2. More detailed discussions in the Scrum meetings.
3. Update product backlog and sprint backlog daily.
4. Create branches in GitHub for all the user stories we are going to carry out in Sprint 2.
5. Commit each user stories code to the branches created in GitHub.
6. Merge all the user stories in the branches with the code in the master branch.
7. Do more regular commits in GitHub.
8. Perform pair programming using one screen and switch when it is appropriate.