Sprint Retrospective

Team: 2

Sprint: 1

Team Members:

Jamie Till (2505749)

Greig Smith (2599630)

Ahmid Omarzada (2527809)

Ahmed Youssef (2507690)

Giri Prasad Imayavaramban (2666739)

Anas Saad (2510059)

Hannah Kiernan (2542983)

To begin the sprint retrospective, we used FLAT retrospective technique, where we had four quadrants where:

Future directions: • Organise GitHub issues more and change how we assign tasks to each other to avoid people's work overlapping- take more advantage of GitHub's features. • Improve our definition of Risk.	**Lessons learned**: • If multiple developers are working on the same user story, assign them the tasks instead. • Risk is not just time taken to implement a feature, risk is also based on uncertainty and challenges. • The backlog needs to be prioritised based on value to the client.
Accomplishments: • All three components and the database are hosting and operatable. • The core framework and data flow is operational between ATM, switch, and network. • The UI is user friendly and functional. • Meetings and communication were frequent and effective	**Thank you**: • Anas for organising the github and fixing the connection and functionality between ATM, Switch, and Network. • To the team for completing assigned tasks for first sprint allowing the base functionality to be shown in the review.

Then, with FLAT in mind we used a DAKI retrospective technique, where we decided what to Drop, Add, Keep, and Improve.

Drop	Add	Keep	Improve
• Rushed/ unplanned sprint review	• Task labels to help clarify issues from user stories • Better Branch management and usage. • Use MoSCoW to help prioritise user stories in a better way.	• Daily and frequent communication between members • GitHub Kanban Board	• Prioritisation • Definition of Risk • Clarification of task assignments for developers

After we all made suggestions as to what to add to each category, we then wrote elaborations on each actionable item in detail below and agreed on them amongst the team.

Actionable Items:

Clarify the tasks ordered to each developer.

To help increase productivity and reduce confusion and the overlapping of work, we should assign issues on the GitHub based on tasks rather than stories. If a single developer is assigned to a whole story, it can be assumed that they are to work on each task. This can be done during discussions and meetings where developers will clarify the tasks in the issue they work on.

Keep consistent usage of GitHub Kanban Board.

GitHub's KanBan board during the first sprint was a very useful tool to let developers know which tasks were being done, completed, or needed to be done. This provided an easy high-level task

assignment overview that made the user aware of the project's current completion. Going forward, we should aim to continue with the consistency of the Kanban board.

Have a better plan and outline for Sprint Review.

For the Sprint Review, we didn't have a strong enough plan or outline for what would be shown, discussed, and asked. This led to us having a disorganised structure where we may not have left as good of an impression or given or received as much information on the product with the clients as we had run out of time. We can improve this certainly, through meeting up beforehand and planning a schedule of what will be discussed, shown, and asked, and appointing a leader and providing introductions.

Have better backlog prioritisation.

Prioritise the backlog to make sure that higher value items are addressed first. We would do this by:

- Hold regular backlog organisation sessions to address and reorder tasks based on client's needs, risk, and urgency.
- Take into account feedback from previous sprints plus client input to adjust priorities.

Improve communication and documentation.

Clear communication and documentation will help avoid misinterpretations or delays. We would do this by:

- Writing user stories more clearly with consideration of what the client or user wants to do and how it will bring the client value instead of writing it from the perspective of a developer, which would help us deliver that value for the client in the end.
- Increasing number of meetings during a sprint.
- Keeping a log of decisions, changes, and lessons learned for easy reference during retrospectives.

Accelerate task and user story completion.

To maintain momentum and deliver value faster, we have to focus on completing tasks and user stories more quickly. We would do this by:

- Breaking larger tasks into smaller manageable subtasks.

- Getting rid of user stories one at a time instead of assigning more than one user story to different people and working on them simultaneously.
- Setting clear deadlines for each task.