An issue which heavily impacted me, and my team was our lack of communication. Looking back with hindsight it was the root for most problems we faced. From the very beginning we were behind since two members were completely unresponsive: they didn't reply to emails, call, and texts, nor did they attend mentor meetings and workshops. In addition to this, one team member had taken a gap year but didn't communicate this with the rest of the team.

During our first workshop, the team collectively agreed on a design alliance in which we highlighted the importance of voicing concerns, issues we are facing individually and ‘pulling our weight’, this however was dismissed. This resulted in some team members volunteering to take on roles which they did not fulfil and other team members taking on a lot more to make up for this. I believe a lot of this was due to the fact we didn’t have an effective communication method; we didn’t take on mentor advice soon enough when they recommended using formal methods of communication like Trello and Discord rather than WhatsApp.

The best way to explain how I felt is ‘tired and burnt out’. This was because I felt like I was one of the few members who committed to what they took on and, as well as this, took over on tasks other members abandoned. The team started off with so much energy and were full of ideas, but this progressively withered away as everyone felt overwhelmed with other assessments and coursework due. This was most evident in our MVP demo submission which occurred on the same day as the Software Engineering exam and Cyber Security students’ coursework submission. Our lack of dedicated focus to the MVP resulted in a poorly executed demo. Students running late led to a late setup and other students unprepared regarding the script caused the team to run out of time. A way in which this could've been prevented was when we were rehearsing the demo, we should have ensured the game and website could be set up and presented in a manner suited to what we were looking to achieve. We should have nominated one member, who wasn’t going to talk, managing what the audience would have been viewing (the website walkthrough and the game demo). Furthermore, we should have communicated what to do if someone was running late because of uncontrollable circumstances as this is what happened on the day of the demo. Another team member had brought their laptop as a backup but wasn't aware how to load the game as we assumed the member who knew how to load the game would be present- we didn’t account for all possibilities.

This experience taught me not to overlook small issues which arose in the beginning, to instead deal with them first-hand. An example which demonstrates this well is that we should have set up trello from the start to have proof of lack of contribution. This would have acted as a deterrent for members to act complacent and kept clear track of what has been completed and what is yet to be done. This would also prove useful when planning for upcoming submissions and in delegating roles. Another issue which we overlooked was not taking account for all possibilities when planning. As mentioned above, I believe this was a huge factor which led to the team running out of time during the MVP and in turn led to a poor result as we couldn’t walk through the game which was the main purpose of the demo.

Something to take away from this experience would be when one of our mentors explained that: there will be times where you’ll be able to go the extra mile and others where you can only offer the bare minimum, so be understanding of other students' circumstances. Although it did not alleviate any of the tasks we had to take on in addition to our own, it did relieve some frustration knowing that some students cannot input the same way I could, and that they are facing issues which are limiting their input.

 Perhaps a solution to our problem could be implementing a plan like the Software Development Life Cycle. By adopting this process we’d be able to conceptualise, develop and maintain our game in a controlled manner since this process involves seven fixed phases which we could then tailor to match our needs. The way the team split tasks was by creating subgroups for different areas which needed to be completed, for instance the design document, the website, and the game itself. By doing this, we completely isolated each task and limited the communication between each subgroup. This resulted in discrepancies during the MVP demo as each subgroup had a different idea of what the game entailed. This however could be resolved had we followed an existing method like the SDLC because everyone would’ve been involved in each task as we worked our way through each phase. An example of how we could’ve implemented this is by beginning with the initial phase: planning, whereby all members contribute to identifying the features we were required to implement, for example: being a turn-based, multiplayer game, and which features we wanted to implement as novelties to the game such as implementing a leader board which presents the top ten highest scores. After we’ve outlined the software requirements, we could then move onto the second phase: requirement analysis. This phase would guide our thinking by pushing us to consider the back-end development needed to accomplish the requirements outlined in the planning phase; for the leader board we’d require a database which stores the player’s name and score so it can then be compared to other players to create the leader board for the ‘top ten highest scores’. After defining each requirement outlined in the planning phase, we could then delve into the mechanics of the game in the design phase and decide which programming language would be best suited for what we were looking to achieve.

After these phases have been completed, we would be in a good position to begin the implementation phase since all the required research would have been completed. Those working on the documents would have all the necessary information to begin their tasks, and those working on the website and hack would have a plan on what they need to include when completing their tasks. The SDLC method includes a testing phase to ensure everything is executed, bugs are detected and that all requirements set out in the initial phase have been satisfied. I believe by implementing a method like the SDLC, it would resolve issues of confusion which arose due to poor communication as we would be required to keep a record of ideas discussed in meetings and it also would allow us to rethink the way we assigned roles. It would also allow for alleviation in pressure when it came to completing tasks as the ‘leg work’ would have been equally distributed within the group during the first few phases I mentioned above.

However, some of the problems we faced were realistic examples of what to expect in the real world. In the MVP, although we did check the game would load and play beforehand, on the day we still faced unforeseen issues. In addition to this, the nerves we experienced on the day weren't present during our rehearsals and so altered our delivery time on the day of the demo. This experience highlights to us that both in this project and beyond, there are some things which cannot be accounted for and the only way we can handle them is by improvising: the demo couldn't be pushed back so whilst we waited for the website and game to be displayed, we began introducing our game and its concepts. Furthermore, this project has emphasised in detail the need to research rather than wasting time diving into things which when implemented won’t satisfy what we’re looking to achieve. Although sometimes this may be inevitable, it is always best to be as prepared as you can.

After the retrospective workshop we were able to communicate our issues and understand everyone’s individual concerns and the concerns we share too. This highlighted to us what needs to be fixed and allowed us to conclude on solutions. For instance, setting up trello and emphasising the importance of posting and updating our tasks means others can see where we are at and what needs to be done. By setting a deadline for a task we can establish a rough date for when it will be completed; this is particularly useful in situations where the completion of one task is dependent on the completion of another.

The plan of action I would like to propose is as follows: firstly, I would like for the team to work together rather than trying to complete tasks individually and then combining our efforts. Previous experiences have led to discrepancies in what was drafted in the design document to what was implemented in the game; and so, working together will allow for consistency throughout the project. Secondly, following the retrospective workshop, we mutually agreed that there was a strong need to use the trello we set up previously. To regularly update it and help wherever we can. Thirdly, I would suggest we meet up more regularly and earlier than we do now rather than leaving the meeting too close to deadlines. By doing so, we limit the time we have to implement the ideas we discussed during the meetings which result in a poor submission. The final way we could resolve this issue of communication is to voice when we feel overwhelmed as soon as we can. This will allow other team members to have enough time to pick up the extra workload rather than being under a tight time constraint. Our team consists of students with various skills which so far have proven very helpful, and I have no doubt that once we iron out this issue of poor communication, we can create a successful game.