5. Risk assessment and Mitigation

Part a)

Before blindly coding the project, it is vital to consider the risks that are involved in the project as a whole. Ensuring that each risk has a mitigation plan in place to avoid problems occurring in the future. Identifying the risks that are involved with the project was the first thing on our agenda. First and foremost, we conducted some research [1] into areas that we should consider when creating potential risks. The article highlighted key points to consider when generating risks, since the team has limited experience with software engineering risks. Initially, we had a discussion as a team to construct a list of all potential risks. As such, each team member was given the opportunity to voice their opinions, allowing a comprehensive list to be developed. Unfortunately, with this approach, there may be risks that get included that are redundant. To eliminate this danger, we reviewed the list and removed any such risks. For example, there was a risk discussed where a team member didn't complete their work, but this is quite vague since it depends on the reason, thus making it hard to create a mitigation plan. Following the list of risks, we had to analyse each one to determine its likelihood and severity. Defining an appropriate method for this part of the process was difficult since each member of the team will have different views. Additionally, some members may have stronger views than others; therefore, they could be influenced. As a team, we decided against having a discussion and instead used an Excel spreadsheet. Similar to the Collaborative Heuristic Evaluation (CHE) [2]. On this spreadsheet, each member will have their own sheet to enter the values they believe to be the best for the analysis (without looking at others for answers). At the end, averages are calculated from the data and rounded to the nearest whole number so that these will be used as the final values. The approach encourages members to think about each risk individually and avoids the possibility of peer pressure. Importantly, each risk needed a mitigation plan to limit its impact on the overall project. Completing this step required the team to meet and collaboratively determine what the best strategy for the risk would be. Consequently, it allowed members to look at the problem from different perspectives, ensuring that the members are critically evaluating the plans. Following the development of the risks, we would need to correctly and effectively format the risk register so it's easily accessible and readable. Using sources from online [3], there is a common format that the risk register should follow: risk ID, risk description, risk likelihood, risk severity, risk mitigation, risk ownership, and risk strategy. It is critical that the risk ID naming be consistent throughout the project in order for risks to be easily identified. Although the source cited states the need for a risk breakdown structure, as a group we believe this is not a necessity for our project. This is because, given the nature of the project, there are not an overwhelming number of risks, so we do not need a breakdown. In fact, having a risk breakdown structure may overcomplicate the risk register.

Part b)

project

Risk	Risk ID	Likeli-hood	Severty	Mitigation	Status	Owner
A group member misundersta-nds their task and completes the wrong thing	R_PROJECT_0 1	3	4	We will meet regularly and stay in contact through messaging to ensure we all are clear on what we are doing, and if a different task gets completed, we just mark that one as done. We will also break tasks down into small parts on a kanban board to make it easier to keep track of what needs doing. If necessary, we will finish	Not occurred	

				it together		
A member leaves the project	R_PRO JECT_0 2	1	4	Our regular meetings should make sure everyone is fine and working on the project. But if this should happen, we will be in contact with the lecturers and module leader. And quickly assign and finish the left personnel unfinished tasks	Not occurred	
The network fails before a member has committed their work and therefore they cannot keep up to date with the remote code which makes it challenging to merge their changes in	R_PRO JECT_0 3	2	3	We make sure we commit often and try to keep pull requests a reasonable size, not too big. But if this occurs, then we can review the merge conflicts together and work out the best way to resolve it. We will also try to make sure that our pieces of work are self contained and have minimal areas that may have merge conflicts.	Not occurred	
A member becomes unavailable for an extended period of time and cannot communicate this	R_PRO JECT_0 4	2	3	Regular communication should mean that we all know where we are at, and if someone becomes unresponsive, we will notice this. Attempts will be made to contact the member using alternative forms of communication.And if that does not work we will let the lecturers and module leader know, while divvying up the work between the rest of us.	Not occurred	
A member is overworked and becomes burnt out as others are doing their workload	R_PRO JECT_0 5	2	4	Our regular meetings to assign work should help ensure work is assigned evenly. We will also try to make sure that there is a second person assigned as backup, so that if people don’t do their main task, there is still someone who can do it and it does not all fall on one person.	Not occurred	
We misjudge the time it takes to complete the project, and do	R_PRO JECT_0 6	3	5	We make a schedule of when certain pieces of work should be done so we have a rough way to	Not occurred	

not finish it before the deadline				measure progress. Then throughout the project with regular meetings we can see what parts are on schedule and what we need to catch up on. There is also buffer time at the end to help mitigate this risk. We need to spend more time on our work that we can finish all the work on time.		
Tasks given in the same week are split up such that they heavily rely on the completion of tasks that need to be completed in the same week	R_PRO JECT_0 7	4	2	Tasks that clash with each other must be given to the same person to complete and therefore can be completed in order. If the tasks are too heavy for one person to complete in one week then the tasks must be split into two weeks instead.	Not occurred	

Product

Risk	Risk ID	Likelih o-od	Severit -y	Mitigation	Status	Owner
A member does not understand the use or purpose a variable (e.g. confusing variable names)	R_PRO DUCT_01	3	2	When working on code, we will be working on separate branches, and when we are done with that task, we will merge using pull requests. These pull requests can then be reviewed by someone else on the team, and that should catch confusing variable names. Though if something is still confusing, we can message the group or the person who worked on that part previously to ask for clarification.	Not occurred	
Assets that are being used are not allowed to be included in open source projects	R_PRO DUCT_02	1	3	Before using any assets we will check the licence for usage rights. If that is clear and says that we can, we will use it. If it says we cannot use it, then we will not. If it is unclear, then we can either look for something else, or we will get in contact with the asset	Not occurred	

				creator to ask for permission. Also, we can contact our lecturers		
A library being used becomes unavailable or deprecated in the middle of the project	R_PRODUCT_03	1	5	We will try to choose popular open source libraries that have a large community around them to help avoid this risk. However, should this still happen, we can check to see if the last released version is sufficient for our use case, which it should be, or we have a look for alternatives that would also work.	Not occurred	