Change Report

With the introduction of new requirements, and taking over another team's project we needed to look at change management when referring to deliverables, documentation and code. As mentioned later on in this document (methods and planning), our team was divided into smaller teams to deal with the development of the product, the software testing and the documentation.

The members responsible for developing the code and the members responsible for testing the software approached the changes in the code. Mentioned later on in the documentation it is also stated that there was not much change to the code more just updating it with new features

The pair that were responsible for the documentation, of course, managed the changes in the documentation. The first main change that was made was by converting all the documents from GitHub Markdown to .docx files so that they could be easily worked on by us as we were using google docs to work on the project collaboratively and simultaneously.

i. Requirements: https://lyrenhex.github.io/Team21Direction/Req1.pdf

Updated the requirements with the new requirements requested for Assessment 2.

User requirements:

UR_PWRUP	The player's ship should be able to obtain 5 special power ups	Shall	
UR_LEVELDIF	The player should be able to choose different levels of difficulty in the game	Shall	Some players may be younger, or have less experience with playing games so having different levels of difficulty can be inclusive for all players.
UR_SAVEGAME	The player should be able to save the state of the game and be able to resume the saved game later	Shall	

Functional Requirements:

FR_DIF_PAGE	There should be a difficulty choosing page where players can choose the difficulty of the game	UR_LEVELDIF
FR_POWERUP	The player should be able to choose from a selection of 5 power ups which will improve their chances of winning at the game	UR_PWRUP
FR_SAVEGAME	The player should be able to pause and save mid-game and come back to the current progress later on.	UR_SAVEGAME

Non-Functional Requirements

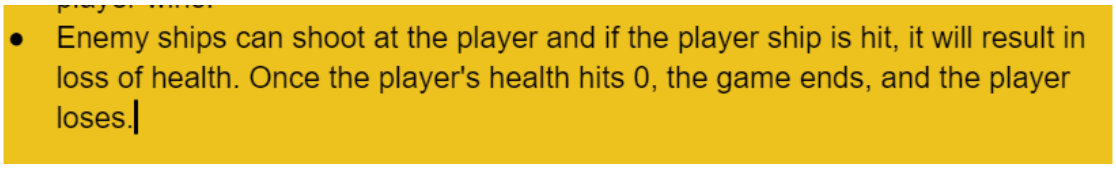
NFR_DIFFICULTY	The player should be able to have different difficulties of the game	UR_LEVELDIF	There should be 3 different difficulties
NFR_SAVE	The game should be able to be paused and saved	UR_SAVEGAME	There should be at least one save slot for the game

ii. Abstract and concrete architecture:
https://lyrenhex.github.io/Team21Direction/Arch1.pdf

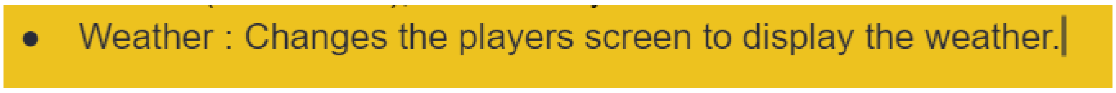
We carried on using the previous team's architecture as it made sense to keep the whole project under one format. Though this was a little bit troubling for our programming team as the previous team's work was confusing for them. An example being that the player class did not inherit from the ship class. They would have also preferred if their gameactor class was more generic.

When it came to updating the document, we took a different approach compared to the other deliverables. Whereas in the other documents we had to change, where we added our methods on a separate page at the end of the document, we decided to change the original document rather than adding bits to the end. This is because in the other documents, we were using a different approach to the previous team but when it came to the architecture we were using near about the same approach and we were building the software on top of what they had developed.

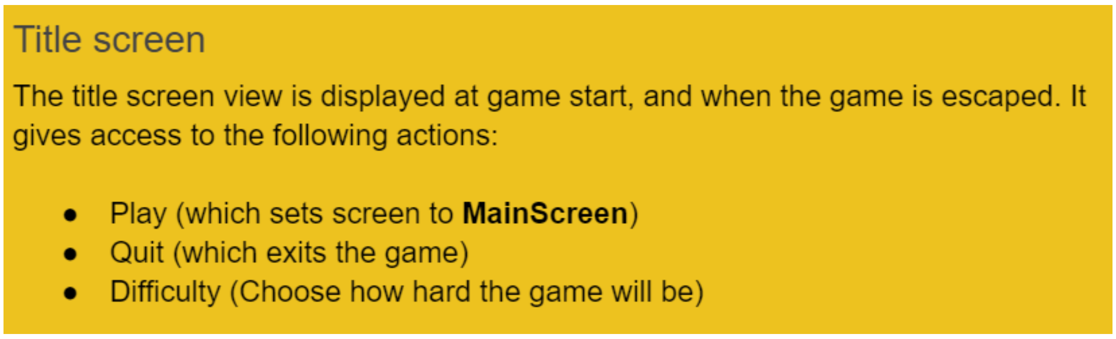
There is not too much that we added to this document as we were building upon the previous team's foundations, as mentioned earlier, but here are some examples of things that were added to the document:



- Enemy ships can shoot at the player and if the player ship is hit, it will result in loss of health. Once the player's health hits 0, the game ends, and the player loses.



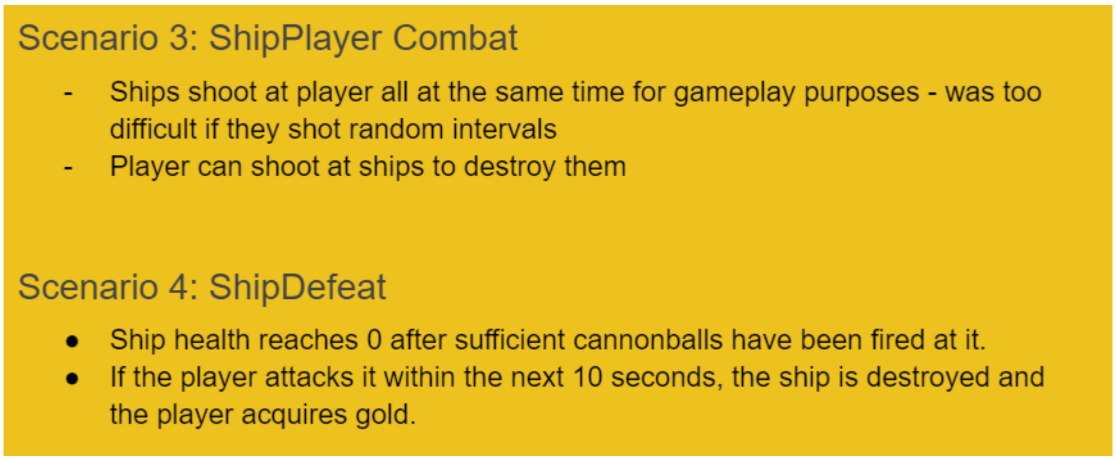
- Weather : Changes the players screen to display the weather.



Title screen

The title screen view is displayed at game start, and when the game is escaped. It gives access to the following actions:

- Play (which sets screen to **MainScreen**)
- Quit (which exits the game)
- Difficulty (Choose how hard the game will be)



Scenario 3: ShipPlayer Combat

- Ships shoot at player all at the same time for gameplay purposes - was too difficult if they shot random intervals
- Player can shoot at ships to destroy them

Scenario 4: ShipDefeat

- Ship health reaches 0 after sufficient cannonballs have been fired at it.
- If the player attacks it within the next 10 seconds, the ship is destroyed and the player acquires gold.

iii. Methods and plans: https://lyrenhex.github.io/Team21Direction/Plan1.pdf

We extended on the previous team's document but we carried on using our own method of working. This is because we were already familiar with our own methods we had used in the previous assessment and thought that changing it for this one would not be necessary. The methods included using agile software development and splitting ourselves back into the same teams we had which were: development, testing and documentation.

The following table shows our milestones that we needed to reach and when we completed them:

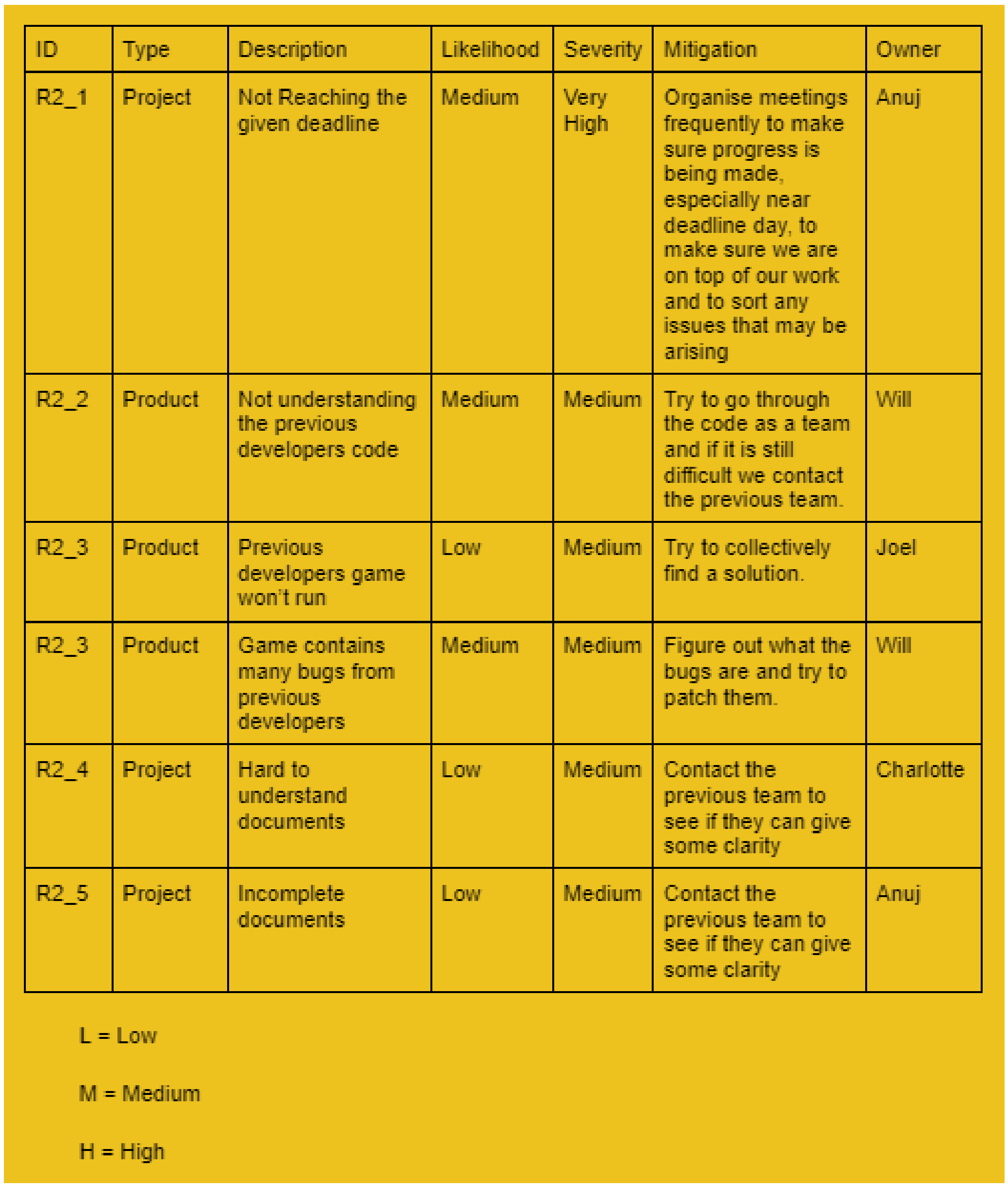
Task	**Start Date**	**End Date**	**Priority**
Method and Planning	20/2/2022	29/4/2022	2
Requirements	20/2/2022	27/4/2022	2
Risk Assessment	20/2/2022	30/4/2022	3
Architecture	20/2/2022	2/5/2022	2
Change Report	20/2/2022	2/5/2022	3
Testing Document	20/2/2022	3/5/2022	2
Implement Shop	20/2/2022	28/4/2022	1
Implement Weather	20/2/2022	1/5/2022	3
Implement Powerups	20/2/2022	2/5/2022	2
Implement Savegame	20/2/2022	2/5/2022	2
Finishing Product	20/2/2022	3/5/2022	1

iv. Risk assessment and mitigation: approach, presentation, risks, mitigations:

https://lyrenhex.github.io/Team21Direction/Risk1.pdf

Using the previous team's document, we continued on from their format and approach as we believe they had a good way of understanding analysing the risks that they may be confronted with and how to approach dealing with it. We used the four steps they had used for their risk assessment in assessment 1 but in a slightly different order. Whereas they had talked about managing the risks, then identifying them, then analysing, then tracking, we presented it in the order of identifying, analysing, managing then tracking. Using these steps we had been able to discuss a new set of risks that comes along with this assessment. The risks were all from a similar category, being the risks that would occur from taking over another teams work. Examples would include, not understanding the previous teams documentation or not understanding their code.

Below shows the risks that we discussed and added to a risk register:



ID	Type	Description	Likelihood	Severity	Mitigation	Owner
R2_1	Project	Not Reaching the given deadline	Medium	Very High	Organise meetings frequently to make sure progress is being made, especially near deadline day, to make sure we are on top of our work and to sort any issues that may be arising	Anuj
R2_2	Product	Not understanding the previous developers code	Medium	Medium	Try to go through the code as a team and if it is still difficult we contact the previous team.	Will
R2_3	Product	Previous developers game won't run	Low	Medium	Try to collectively find a solution.	Joel
R2_3	Product	Game contains many bugs from previous developers	Medium	Medium	Figure out what the bugs are and try to patch them.	Will
R2_4	Project	Hard to understand documents	Low	Medium	Contact the previous team to see if they can give some clarity	Charlotte
R2_5	Project	Incomplete documents	Low	Medium	Contact the previous team to see if they can give some clarity	Anuj

L = Low

M = Medium

H = High