Requirements

At the beginning of our SEPR project, we brainstormed ideas for what we wanted our game to be like. After deciding upon a basis of our game, we knew that in order to ensure that our game met our customer's requirements that we would have to arrange a meeting and ask questions in order to clarify what their requirements were.

Software requirement specifications must have several qualities to ensure that they are useful to both the developers and the customer. They must be unambiguous, correct and verifiable [1]. This means that the requirements will have to tell us exactly what has to be done which prevents misunderstandings between us as developers and the customer, they must also be able to be proven in some sense that they have been fulfilled. Some requirement specifications are also ranked depending on their importance for the final product so that it is easier for developers to know which requirements they have to prioritise.

For our specifications, we have decided to write them in a table format so that they are easy to read and understand what is needed for each of them. We have separated them into separate tables depending on what part of the system they are involved with in order to make it easier for us, as developers, to know exactly what is required where. As well as this, we are also ranking them by priority so that we know what we must tackle first.

In keeping with the theme of legibility, for each requirement we will describe any environmental assumptions or risks that are related. This allows both us and the customer to have a clear idea of the potential problems that each requirement could cause during development along with what we are assuming about the user. By using this method, we eliminate any chance for us to fail to take into consideration the user's interaction with the system as we can directly see what we are assuming for our total requirements.

General:

ID	Requirements	Notes
1.1	The game must use a 2D style	Ensures it is easy to navigate and develop
1.2	The game must use keyboard and mouse for input methods	The player needs to input their name for the leaderboard as well as using the arrow keys/mouse to move around the map and navigate any menus. We assume that the user is going to use a standard keyboard and mouse that have arrow keys as that is the most common. Alternative input schemes were thought of but would be harder to implement and much less likely to be used, plus keyboard and mouse was specified by the customer as a must [2] (Q1.3)
1.3	The game must have a leaderboard which tracks the highscores of players locally	An alternative would be to use a database and track the highscores globally, but tracking locally has a lesser chance of errors and therefore a lower risk. The idea of a leaderboard was taken from our interview with our customer to refine our requirements [2] (Q4.1)
1.4	The game must have an objective list so the user knows what to do to succeed	Prevents the user from not knowing what to do to progress through the game thus reducing the risk that users get bored due to lack of understanding.
1.5	The game could have a tutorial system to instruct the user on how to play the game	This would be chosen by the player when they start their game. This reduces the risk that the player would get frustrated due to not knowing the controls as they will be taught them when they start playing
1.6	The game should be able to run smoothly on most laptop or desktop computers	This game is being developed for use in the CS department and so must be playable on most if not all of the computers available there. As such it should be playable in the operating systems installed on CS department computers.
1.7	There should be a story to give the player background to the game	This could be told before the main menu through on-screen text//cutscene or through dialogue. This helps the user to understand why they are completing the game's objectives, hopefully making it more enjoyable

Sailing Mode:

ID	Requirements	Notes
2.1	The game must use a grid-based movement system for sailing	This is the easiest method to both develop and play on as it allows the users to know exactly where they're going to go on their turn. An alternative would be to allow the user to move wherever they want use the arrow keys, but would reduce the effectiveness of the weather and make it more difficult to ensure the user gets into combat when near another ship
2.2	The map must be based off the University of York campuses	Assume it is a 2D grid map. We chose this as it was what was specified for us in our initial brief by our customer.
2.3	The playable map should be split in two, with one area harder than the other	The map isn't going to be physically split, just visually so that it is easier for us to develop and easier for the user to understand. An alternative was to make it one difficulty throughout but we felt that having one half be harder than the other gave a better sense of progression to the player. This was also suggested by our customer in our interview [2] (Q2.3) as the bosses should get progressively harder.
2.4	The user must be able to sail around the map in order to	This was given to us in our initial brief.

	complete objectives	
2.5	There must be at least 2 different types of stationary locations: colleges and departments	This was given to us in our initial brief.
2.6	There must be at least 5 colleges and 3 departments	This was given to us in our initial brief.
2.7	Capturing a college gives ability for repairs at that location	This was given to us in our initial brief. Also, we have the option to be able to repair ships in different shops, preventing the risk that players will be low on health and be stuck without being able to repair their ship without dying.
2.8	Pirates should patrol a preset area	This was so that we could decide exactly where to have the enemies be and we could hard code their movement pattern. An alternative was to make it entirely random where they would move, but this had a lot of risks involved where there could be fringe cases where the user doesn't find any for a long time or they all end up in a small area.
2.9	Monsters should be hidden to the player and be fought when moved onto	This was designed to create a bit of variance between the types of enemies the player encounters. The risk of this could be that the user encounters lots of monsters in a very short amount of time, leading to them potentially dying repeatedly without being able to make much progress. The alternative would be to have a monster fight to have a random chance to occur whenever the player moves, but this way of development makes it easier to do, simply by making them unseeable and move every few rounds to a new location.
2.10	There must be weather effects that affect ship movement	This was taken from our customer interview [2] (Q2.5) as well as our initial brief. The weather will force the player to either take damage and be slowed, or have to go all the way around it, which provides choice for the player. There is a risk that this could hinder the player for a large amount of time, but this shouldn't happen at all/often.
2.11	The user should be able to interact with an "End Turn" button to end their turn when sailing	Having an "End Turn" button allows the player to not have to use all of their moves in a single turn, allowing the gameplay to feel less restrictive. One risk is that the user accidentally presses this button and could lose a turn so we will take steps to ensure that this is out of the way and obvious.
2.12	Players should be able to click on a tile in the map or use the arrow keys to select a point to move to	The map should highlight the square that the user is hovering over. This was chosen to reduce the risk that the user only had one type of input or, for whatever reason, they couldn't use a keyboard or mouse.

Combat Mode:

ID	Requirements	Notes
3.1	The game must have a turn-based combat system	This was chosen as it reduces the risk of having the combat system be overly difficult to develop. This requirement was gathered from our customer interview [2] (Q3.1)
3.2	There should be two different types of enemies: pirates and monsters	This was chosen to create a bit of variance in the type of enemy the player encounters as well as having them be met in different ways.
3.3	You must gain gold	This was given to us in our initial brief. The risk is that we don't have the

	when you defeat an enemy	balance of how much gold the player should be given on each enemy defeat and so the game either becomes extremely easy or extremely difficult. One alternative is to just give the player items instead but that takes away from the feeling of gathering gold and then spending it on what you want to get.
3.4	Colleges should be capturable through combat	This occurs when the player beats the boss of the college. This was given to us in our intial brief. We have to ensure that the boss of each college isn't too difficult or it will prevent progress for the players and make it hard to complete.
3.5	The game must have an end-boss in order to complete the game	The end-boss will have to be difficult enough so that it takes a while for the player to be able to not only make it to the boss but also be able to defeat him. However, it can't be too difficult as the game does need to be beatable by the average user as it is being used for display purposes within the CS department.

Items/Services/Shops

ID	Requirements	Notes
4.1	The game should have items that you can find or buy to help you in combat	You can spend the gold you get from defeating enemies at shops in order to buy items that act as abilities in combat. Other items are available that could help in map movement too. The risk is that there is too little gold or items dropping or too few shops available to make them obvious and common enough to use.
4.2	You must be able to use gold to repair/upgrade your ship	This was decided in our interview with our customer [2] (Q5.1). The risk is that we spend a lot of time developing ship upgrades that aren't entirely necessary and miss core functionality in other parts of our game.
4.3	There will be multiple shop locations, for buying services/items	These are so that the user will be able to upgrade/repair their ship in multiple places, so its not too bad for the player if they get damaged. The risk is that the shops aren't clear enough to see so that player doesn't find any for a long time which would make the game less fun as they would have to avoid combat. Another risk is that the shops are too common, which reduces the risk that a player should encounter of their ship being low after combat.

Minigame:

ID	Requirements	Notes
5.1	If you die, you have to complete a minigame to come back to life which increases in difficulty for each death	This was taken from our interview with our customer [2] (Q3.3). The risk is that the minigame is too difficult to complete so we will ensure that it is fairly easy to complete the first time and getting harder each time.
5.2	The player should only have 3 lives after which they have to restart	We wanted to make this game challenging and arcade-style by forcing the user to have to take care of how many lives they have left and complete the game in one playthrough. An alternative was to have a save game feature, but we felt that it would cause too many risks in development and would probably make the game too easy.

Bibliography

[1] Tutorialspoint. *Software Requirement Specification*, [Online] Available: https://www.tutorialspoint.com/software_testing_dictionary/software_requirement_specification.htm [Accessed: November 6th, 2018]

[2] Interview with the customer: https://davidjnorman.github.io/SEPR/assessment_1/Questions.pdf