Requirements

To define requirements we decided upon using IEEE's 12207.1 [2] standards as a base and also followed their terminology. As a supplement, we also used guru99's article [1] to help us clarify the system requirements. to assist us in defining the system requirements in particular.

The requirements defined here were based on a few factors. We first analysed the assessment product brief and used specific features as a base. Then we expanded on them, discussing as a group how to implement this, ensuring that each requirement was feasible and achievable.

We also organised two meetings with the customer so they could review our proposed requirements and understand how we were going to develop the system. This allowed us to reduce the chance of misinterpretations, clarify what the end product was, and resolve any early issues we had.

We have sorted our requirements, into three separate tables:

1. User requirements are high-level abstractions, designed to be presentable to our customers and non-technical people in order to help us develop our functional requirements.
2. Functional requirements, a subset of system requirements we split up the user requirements into a more concrete implementation. This is the "business logic" of the game. Defining the behaviour (inputs and outputs) and how everything interacts together.
3. Non-functional requirements, also a part of system requirements, focusing more on the quality of the game. Not completing these requirements would result in a game that would be unsuitable for the customer.

To represent this data we have chosen to present through tables each focusing on different groups of requirements. The tables all follow a similar layout, the ID column uniquely identifies every requirement, with a person-friendly identifier and a number as a short representation (1st number representing the type i.e. functional / non-functional, etc. The 2nd number represents the object to which a group of requirements relates to i.e. boat. Finally, the 3rd represents the position in the sub-table to help us identify each requirement in the list.

To show the priority of User Requirements we use colour coding. This allows us to prioritise different requirements and the order in which they are worked on.

- "Low", this requirement would be implemented for the final release build, but not in the earlier beta builds.
- "Medium", this requirement would be implemented on our second build (v0.2) as it isn't a fundamental requirement which others build off, but it should be done before the deadline as it is still a core requirement, ensuring the implementation is to a good standard and any issues are ironed out.
- "High", needs to be implemented in our first stable version (v0.1). A lot of other requirements depend on this being present.

Requirements in this document are used for designing of tests, which can be found here TRACEABILITY MATRIX and BLACKBOX TESTS

User Requirements

ID	Description	Risks and assumptions (if relevant)
UR_UNIQUE_BOATS 1.0.0	In all the different boats, they must all be unique in their own ways. Acceleration, Speed, Maneuverability, and Robustness. Create boats based on these.	There is a base boat Boats have attributes to control their characteristics Boats can be represented by an image
UR_DAMAGE 1.0.1	Colliding with obstacles damages the boat	Boat, movement, and obstacles have been implemented
UR_CONTROLS 1.0.2	The boat responds to the controls inputted by the user in a comfortable way	A base boat has been implemented and is currently stationary
UR_MIN_BOATS 1.0.3	There should be at least 4 boats in each race.	There are enough unique boats in the game.
UR_MAP 1.1.0	The map through which players race includes obstacles	Need to make a background graphics to represent the map
UR_LANE 1.1.1	Stay in your lane, or you get a penalty of X seconds for the amount of time you spend out of it	Map has already been implemented
UR_HUD 1.2.0	Have a clear UI that is easy to understand and see what corresponds to what	Boats with variables to store their statistics have been implemented
UR_MENU 1.2.1	The game has a main menu screen	Screens have been implemented
UR_DISPLAY INSTRUCTIONS 1.2.3	The game should display instructions about the input it is expecting from the user	Game works, and the input used for the game has been determined
UR_PAUSE_GAME 1.2.4	The user should be able to pause the game at any point and see a pause menu with options.	The game state can be paused and there are features that the menu buttons can access.
UR_SAVE_GAME 1.2.5	The user should be able to save the current game state to resume from later.	There is a way to create, delete and open saves.
UR_QUIT_GAME 1.2.6	The user should be able to quit the game from the main menu or quit to the main menu from the pause menu.	There are buttons for quitting.
UR_SELECT_DIFFCULTY 1.2.7	The user should be able to select the level of difficulty to play at.	Different difficulty levels are implemented.
UR_DIFFICULTY 1.3.1	Each leg of the race becomes harder	Map, AI, unique boats, legs
UR_DIFFICULTY_LEVELS 1.3.2	There are different levels of difficulty that the player can select at the start of the game.	Objects and UR 1.3.1 are implemented.
UR_POWER-UPS 1.3.3	There are power-ups along the course that boats can use to gain an advantage.	Power up obstacles can be spawned randomly on the course.
UR_FINAL_PLACE 1.3.4	If the user comes 1st, 2nd or 3rd in the final, they must be awarded the corresponding medals.	The user is able to place in the top 3.
UR_FINAL_RACE 1.3.5	If the user is fast enough, they will compete in the final.	User is fast enough.
UR_LOSS 1.3.6	The game is lost if the player loses all robustness, doesn't make it to the final or doesn't place in the	Robustness is implemented.

	top 3 in the final.	
UR_PLAYABLE 1.3.7	The game should be playable and enjoyable to the target audience.	May result in the game being considered boring in an effort to please everyone.

Functional Requirements

ID	Description	Alternatives	UR_ID
FR_STAMINA 2.0.0	Each team gets tired and slower as the race progresses.	Boat speed is reduced.	1.0.0
FR_STATS 2.0.1	Define each stat and how they impact the boat and each other	Boats all act in the same way and races must be won through skill	1.0.0
FR_ASPECT 2.0.2	Each boat appears to be different	Use boats of different sizes with the same texture to differentiate between them	1.0.0
FR_OBSTACLES 2.0.3	Different obstacles appear on the course randomly.	Instead of every obstacle having unique damage amounts. Make it randomly generated	1.0.1 1.1.0
FR_PENALTY 2.0.4	Get a penalty for time spend outside the lane	Boat is frozen / on a cooldown before they can move again.	1.1.1
FR_LANE 2.0.5	Determine if a boat crosses out of its lane	A pop-up could occur to notify them to move back	1.1.1
FR_DAMAGE 2.0.6	Detect a collision between a boat and an obstacle	Sound effect when a collision occurs	1.0.1
FR_HEALTHBAR 2.0.7	Each boat has a health / robustness bar that decreases when obstacles are hit	Update the robustness bar on a timer, rather than immediately after a collision	1.0.1 1.2.0
FR_MINIMAP 2.0.8	The player can see a minimap in one of the corners of the screen	Have the minimap just show upcoming terrain	1.1.0 1.2.0
FR_CONTROL S 2.0.9	The player's movement should be based on the player's input	Use simple movement that doesn't involve the rotation of the boat	1.0.2
FR_VARIABLE_ CONTROLS 2.0.10	The player's ability to turn, as well as its speed, is based on stamina	Have a set time of movement, then a break needs to be taken	1.0.2
FR_SAVE_SLOTS 2.0.11	There must be save slots for the game state.	A single save slot that can be resumed or overwritten by starting a new game.	1.2.5
FR_PAUSE 2.0.12	The game can be paused and options appear on the screen.	Options are brought up, but the game continues in the background.	1.2.4
FR_QUIT_BUTTON 2.0.13	The game can be quit from the main menu or pause menu.	Shortcut to exit the game (Alt + F4)	1.2.6
FR_DIFFICULTY_LEVELS 2.0.14	The game has different levels of difficulty, selected by the user.	Just the number of obstacle spawns changes.	1.3.1 1.3.2
FR_POWER_UPS 2.0.15	Power-ups that affect the boat upon collision.	Could all have the same effect, but by differing amounts.	1.3.3
FR_TIMER 2.0.16	System must track the time during the race.	Use race finish positions only for the leaderboard.	1.3.4 1.3.5 1.3.6
FR_LANE_WARNING 2.0.17	A warning appears on screen for straying out of the lane.	The user is not warned when a penalty is applied.	1.1.1

FR_AI 2.0.18	Other boats in each race should be controlled by their own AI.	Other boats follow simple path-finding.	1.0.2 1.0.2
FR_MAIN_MENU 2.0.19	The game should have a menu screen for the player to select their boat, difficulty and save slot.	Game launches straight into a race with a pre-selected boat.	1.2.1 1.2.5 1.2.6 1.2.7
FR_MUSIC 2.0.20	The game should have background audio.	Audio that only plays on the menu screen.	1.3.7
FR_UI 2.0.21	There should be a UI showing the boats robustness and the stamina of the crew.	A text warning when stamina or robustness run out.	1.2.0
FR_MAP_BOUNDARY 2.0.22	Boats should not be able to leave the boundary of the river.	Boats are just stopped.	1.1.1
FR_CANNOT_GO_BACKWARDS 2.0.23	None of the boats can go backwards down the course.	No support for boat y-coordinate to decrease.	1.3.7

Non-Functional Requirements

ID	Description	Fit Criteria	UR_ID
NFR_FAST_CONTROLS 3.0.0	When inputting the direction, the response from the game should be instant	In <0.5 seconds response to an input, if not lower	UR_CONTROLS 1.0.2 UR_PLAYABLE 1.3.7
NFR_FAST_TRANSITION 3.0.1	When switching between game states, response should be instant	In <0.5 seconds response to an input, if not lower	UR_MENU 1.2.1 UR_HUD 1.2.0
NFR_INFORMATION_TIME 3.0.2	The user should be informed of any changes quickly.	UI changes/ alerts happen within <0.5 seconds of the trigger event.	UR_PLAYABLE 1.3.7 UR_HUD 1.2.0
NFR_ANIMATION 3.0.3	Animations should run smoothly.	Game should run at a minimum of 30 FPS.	UR_PLAYABLE 1.3.7
NFR_GRAPHICS 3.0.4	Graphics look like the objects they represent.	Players should be able to identify all graphical assets.	UR_PLAYABLE 1.3.7
NFR_GAME_LENGTH 3.0.5	The game should be completable in a reasonable length of time.	Each race takes no longer than 2 minutes.	UR_PLAYABLE 1.3.7

Constraints

Requirement ID	Description
CON_APPOPRIATE	The game shouldn't contain explicit or graphic content, such as blood or gore, and should be suitable for any user in the target audience.
CON_LANGUAGE	The game must be programmed in Java.
CON_RUN	The game should be able to run on any University of York Computer Science Department computer, running Windows or Linux.
CON_ACCESSIBLE	The game must use colours with high contrast to ensure all users can play it.
CON_COURSE	Must be based on the river course in York

Bibliography

1 https://www.guru99.com/functional-vs-non-functional-requirements.html

2 https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=669648