Requirements

Team 6
Sanjel Sadikaj
Adam Hagan
Tudor Ciobanu
Na Tang
Armintas Zadeika
Liu Zhang

a)

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.

The priority column helps us decide what will come first during the implementation, each requirement is ranked on the following scale:

- "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.

ID	Description	Test	Assumptions	Priority
Boat				
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.	Boats can be assigned different attributes Measurable by comparing how much the boat movement after: player interaction when hitting obstacles The boats look visually distinct	There is a base boat Boats have attributes to control their characteristics Boats can be represented by an image	Medium (v0.2)
UR_DAMAGE 1.0.1	Colliding with obstacles damages the boat	When a boat collides, the stamina should be reduced	Boat, movement, and obstacles have been implemented	Medium (v0.2)
UR_CONTROLS 1.0.2	The boat responds to the controls inputted by the user in a comfortable way	Boat's course will be adjusted when a user uses the specific key bindings	A base boat has been implemented and is currently stationary	High (v0.1)
UR_POWER-UPS 1.0.3	The boat should receive an advantage when collecting power up objects	When the boat picks up a power up, an event should occur to give the boat an advantage.	There can be various power ups such as; increases in health, stamina, speed and maneuverability, and time reduction.	High
Map				
UR_MAP 1.1.0	The map through which players race includes obstacles	Must be long enough in length to cover an entire leg. Features obstacles which will reduce the player's stamina	Need to make a Background graphics to represent the map	High (v0.1)
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	Deliberately move the boat to the other lane, and check if a warning is given. Check to see if the penalty is different for when the time spent in it changes.	Map has already been implemented	Medium (v0.2)
UR_SAVE_GAME 1.1.2	The user should be able to save the game in its current state and load the save.	Test saving a game (should save a file). Then load the file and see by comparison if everything is saved.	Objects which are off-screen will not be saved as they have not generated yet.	High
UI / HUD				
UR_HUD 1.2.0	Have a clear UI that is easy to understand and see what corresponds to what	Displays the timer, leg, position, stamina, and robustness	Boats with variables to store their statistics have been implemented	Medium (v0.2)

UR_MENU 1.2.1	The game has a main menu screen	Displays game logo, buttons to navigate the game	Screens have been implemented	Medium (v0.2)
UR_DISPLAY_INSTRUCTIONS 1.2.3	The game should display instructions about the input it is expecting from the user	Display text that explains what the player has to do next	Game works, and the input used for the game has been determined	Low (v1.0)
UR_CHANGE_SETTINGS 1.2.4	The user should be able to change multiple settings on the settings screen	The available options should include audio, control and graphic settings	Game works with default settings	Low (v1.0)

Game				
UR_DIFFICULTY 1.3.1	Each leg of the race becomes harder	AI stats, obstacle amount, global speed movement are increased	Map, AI, unique boats, legs	Low (v1.0)
UR_CHANGE_DIFFICULTY1.3.2	The user should have the ability to select a difficulty of Easy, Medium & hard	Increases the number of obstacles for difficulty increasing and decreases the number of powerups.	The higher difficulty the more obstacles. The lower difficulty the more power ups and less obstacles.	High

Functional Requirements

ID	Description	Test	Alternatives	UR_ID
FR_STAMINA 2.0.0	Each team gets tired and slower as the race progresses.	Reaction speed to user controls progressively gets worse	Boat speed is reduced.	1.0.0
FR_STATS 2.0.1	Define each stat and how they impact the boat and each other	Each stat that a boat has has an effect on its performance	Boats all act in the same way and races must be won through skill	1.0.0
FR_ASPECT 2.0.2	Each boat has a different aspect	Check each boat texture to ensure they are significantly different from one another	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.	Boat's robustness is decreased by the specific damage of the obstacle	Instead of every obstacle having unique damage amounts. Make it randomly generated	1.0.1 1.1.0
FR_POWER_UP_ITEMS 2.0.12	Objects randomly floating along the river which can be picked up to gain an advantage in the race.	Objects should be seen on the river. After the boat passes over the object depending on the powerup, an event should take	Powerups should be randomly placed and vary. There should not be "blocks" of power ups in one place.	1.0.3

		place.		
FR_PENALTY 2.0.4	Get a penalty for time spend outside the lane	One of the boat's stats are modified as a penalty temporarily	Boat is frozen / on a cooldown before they can move again. Other players stats are improved temporarily	1.1.1
FR_LANE 2.0.5	Determine if a boat crosses out of its lane	The penalty system will engage on the person	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	Boat's robustness is decreased	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	Health bar is updated when a collision occurs	Update the robustness bar on a timer, rather than immediately after a collision	1.0.1 1.2.0
FR_CONTROL S 2.0.9	The player's movement should be based on the player's input	Boat's target position is updated to reflect the changed rotation	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	Boat movement is gradual towards the desired location, factoring in current stats	Have a set time of movement, then a break needs to be taken	1.0.2
FR_SAVE_&_LOAD 2.0.11	The player should be able to press a button which should save and a button which will load a save.	After pressing the save button a file should be created. Loading a file should show the state of the game for which it was saved.	The file should be created and loaded from the users directories. The game state should match at the moment it was saved.	1.1.2
FR_CHANGE_DIFFICULT 2.0.13	After selecting whichever difficulty, changes should be made to the game to ensure that difficulty is selected.	The number of obstacles and power ups should change accordingly to the 3 sets of difficulties.	Each set of difficulties will have a set value of power-ups and obstacles which respectively increase or reduce difficulty.	1.3.2

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_BOAT_CONTROLS
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, UR_HUD

Bibliography

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

2

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