Module	Engineering 1(ENG1) - COM00019
Assessment Title	Assessment 2 - Cohort 2
Team	Dragonite (Team 21)
Members	Okan Deniz, Rhianna Edwards, Omar Omar, Omar Galvao Da Silva, Craig Smith, Joel Wallis
Deliverables	Requirements

Introduction

Prior to developing and implementing code, the team decided it best to set out project requirements. This would:

- aid in the development process.
- help to keep the project on track with stakeholders' goals.
- prevent the team from working on any unnecessary features.

This was done at the start of the project to avoid common pitfalls that had been outlined through research, such as the tendency to make assumptions, and eliciting unbounded and vague requirements. [1]

The requirements were broken down into the following categories based on our research: User, Functional, and Non-Functional. [2]

Throughout the process of identifying and formatting requirements, the definition of a well-defined requirement was focused on: A well-defined requirement is a statement of system functionality that must have proof of its validation and must be met to achieve a customer's objective for a system. [3]

How requirements were elicited and negotiated

After reading the project brief document, a brainstorming session took place in which the team outlined a list of clarifications that would be needed before eliciting requirements. These were taken to a Customer meeting, in which the stakeholders were asked to make these clarifications, and their responses were noted so they could be used to outline the capabilities of the system [4]. This proved useful when finalising requirements.

With this information, a Single Statement of Need (SSON) was created alongside a final list of User Requirements for the project.

Our SSON is as follows: "The system should allow users to compete in a single player game, racing as a boat in a Dragon Boat race against AI opponents across three legs, where the 3 fastest boats across these legs will proceed to a final, that can be displayed to end users in Open Day demonstrations".

The SSON acted as a guide for maintaining focus on the system requirements that would be used in the Architecture plan.

Why requirements are presented as they are

To format the elicited set of requirements, the 'IEEE Guide for Developing System Requirements Specification' provided substantial insight into a sensible format and the necessary level of detail for the requirements.

To distinguish types of requirements, colours were used to highlight differences , making the reflection process ergonomic. A tabular approach was decided upon as it neatly stored the information and meant it would be easily adaptable in case of further change.

Alongside requirements, constraints and use cases (see Appendix) were defined to help meet the stakeholders' goals.

Risks associated with these requirements are detailed in the table below and were then addressed during risk assessment and mitigation.

User Requirements

Requirement ID	Description	Risks and Assumptions (if relevant)	Priority Level
UR_FINAL_ PLACE	If the user comes 1st, 2nd or 3rd in the final, they must be awarded the corresponding medals.	Too difficult to come in the top 3.	Shall
UR_FINAL_ RACE	If the user's fastest leg is fast enough, they will compete in the final.	Too difficult to place into the final.	Shall
UR_LOSS	The competition is lost either if the player does not make the final, places lower than 3rd place in the final or their boat energy level reaches 0.	There are enough obstacles and damage that can be done to the boat to reach an energy level of 0.	Shall
UR_PLAYABLE	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.	Shall
UR_BOATS	There should be a variety of unique boats for the user to choose from.	Some boats may end up unbalanced and make the game unfair.	Shall
UR_MIN_BOATS	There should be at least 4 boats in each race.	Too few boats could make the game boring as there's not enough competition.	Shall
UR_CONTROLS	Controls should follow standard conventions or be explained at the beginning and easy to use.	The standard controls may be followed blindly, without the user understanding fully what they are doing.	May
UR_DIFFICULTY	The game should get progressively difficult for each race.	The game could end up too difficult for the user, making them likely to quit.	Shall
UR_TIME	Each race shouldn't be too long.	Races could go on too long, making the entire competition drag on and become boring to play.	Should\
UR_CONVENTIONS	The game should follow standard conventions.	The conventions could be followed where it doesn't make sense for the game.	May
UR_POWER_UPS	The game should include a variety of power ups for the user to collect and make use of.	Power ups may be too similar in functionality and not lead to an ‘as-exciting gameplay’.	Shall
UR_POWER_UPS_COUNT	There should be 5 different power up packs floating down the river for the boats to pick up.	Excessive power-ups could make the game unnecessarily complex as players would have to learn about every individual power up.	Shall
UR_GAME_DIFFICULTY	The game should include difficulty levels for the user to choose from (Easy, Medium, Hard).	It may become impossible to win on harder difficulties or too easy to an extent where it’s not possible to lose on easier difficulties.	Shall
UR_SAVE	The game should allow users to save the state of the game at any given point and allow for it to be resumed later on.	Files saved during older versions of the game may have compatibility issues if the game is updated.	Shall

System Requirements - Functional

Requirement ID	Description	Risks and Assumptions	Design Requirement

FR_UNIQUE_ BOATS	Each boat must have unique statistics. Robustness Maneuverability Max Speed Acceleration	Some boats may be 'unbalanced 'and either too powerful or weak. This would affect the competitiveness of the game poorly.	UR_BOATS
FR_TIMER	The system must track the time during the race.	Assumes that built in time tracking functions are accurate.	UR_FINAL_RACE, UR_FINAL_PLACE
FR_FINISH	The system must recognise when the user's boat crosses the finish line and calculate the time taken.	Assumes the user crosses the finish line. Also assumes that there is a working timing function.	UR_FINAL_RACE, UR_FINAL_PLACE
FR_OBSTACLES	The system must randomly generate obstacles on the course.	Obstacles may cause too much damage to the boat, meaning the player can survive very few hits - making it unplayable.	UR_DIFFICULTY
FR_OBSTACLE_INCREASE	The number of obstacles over the length of the course should increase for each race.	Too many obstacles at once could make the course impossible to pass without going outside the lane.	UR_DIFFICULTY
FR_TIREDNESS	The energy of the crew should decrease over time. This carries over through each race.	Rate of tiring may be so high that the race takes too long to complete.	UR_DIFFICULTY
FR_UI	There should be a UI showing the boats robustness and the energy of the crew.	UI may be too cluttered or have an unclear layout. Could obstruct other parts of the game.	UR_PLAYABLE, UR_CONVENTIONS
FR_ANIMATIONS	There should be animations in the game, such as movement of boats.	Assumes the hardware is capable of playing animations along with the rest of the game.	UR_PLAYABLE
FR_COLLISIONS	Upon collision with an obstacle, the robustness of the boat will decrease.	Damage done may make the game too difficult or not be enough of a challenge.	UR_DIFFICULTY, UR_LOSS
FR_PENALTY	Total time outside the user's lane will result in a time penalty.	Penalty could be too harsh or not enough of a deterrent to stop the player leaving their lane. Also assumes that there is a working timing function.	UR_DIFFICULTY
FR_LANE_WARNING	The user should receive a warning if they leave their lane.	Warning could obstruct the player's view of the game and therefore, ability to move back into their lane.	UR_PLAYABLE, UR_CONVENTIONS
FR_SPEED_CONTROL	The user should be able to control the speed of the boat	Control sensitivity could hinder the player's ability to control the speed how they want.	UR_CONTROLS, UR_PLAYABLE
FR_COURSE_ BOUNDARIES	Boats should not be able to leave the course.	Boats may leave the course and get stuck or be able to cheat.	UR_PLAYABLE, UR CONVENTIONS
FR_TITLE_SCREEN	The game should be able to launch into a title screen.	The player may not know where to navigate to from the title screen.	UR_CONVENTIONS
FR_AI	Other boats in each race should be controlled by their own AI.	AI could be too good at the game for the player to beat, or alternatively too easy to beat - making the game boring.	UR_PLAYABLE, UR_CONVENTIONS

FR_MUSIC	The game should have some background audio.	Background music could be distracting or too loud for the player.	UR_PLAYABLE, UR_CONVENTIONS
FR_UNIQUE_POWER_UPS	Each power up should alter the current game state such as boat stats and environmental features.	Some power ups may end up being too influential. This may cause unbalance.	UR_POWER_UPS
FR_GAME_MODES	The system should make changes to enemy boats/AI and obstacles depending on the difficulty level chosen.	Insufficient challenges may make the game less enjoyable at lower difficulties.	UR_GAME_DIFFICULTY
FR_PAUSE_SCREEN	The game should display a pause screen when the pause button is pressed.	The player may not know which button pauses the game.	UR_SAVE, UR_CONVENTIONS

System Requirements - Non-Functional

Requirement ID	Description	User Requirements	Fit Criteria	Risks and Assumptions
NFR_RESPONSIVE	The system should respond quickly to user input from the mouse and keyboard.	UR PLAYABLE	The response should happen <0.5 second after the input is pressed.	Assumes the user gives a valid input.
NFR_GAME_LENGTH	The game should be completable in a reasonable length of time.	UR_TIME	Each race takes no longer than 2 minutes.	Assumes the player moves the boat along the course.
NFR_INFORMATION_TIME	The user should be informed of any changes quickly.	UR PLAYABLE	UI changes/ alerts happen within <0.5 seconds of the trigger event.	Players may be unaware of changes.
NFR_ANIMATION	Animations should run smoothly.	UR PLAYABLE	Game should run at a minimum of 30 FPS.	Assumes hardware is capable of running a basic game at 30 FPS.
NFR_GRAPHICS	Graphics look like the objects they represent.	UR PLAYABLE	Players should be able to identify all graphical assets.	Players will be confused and the game may be unplayable if assets are not identifiable.

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

References

- [1] 'IEEE Guide for Developing System Requirements Specifications', *IEEE Std 1233, 1998 Edition,* pp. 14, Dec. 1998. [Accessed: 05 Nov. 2020)
- [2] D. Thakur, (2013, November.23), *What is Software Requirement? Types of Requirements.* [Online]. Available: https://ecomputernotes .com/software-engineering/softwarerequirement.[Accessed: 05 Nov. 2020).
- [3] 'IEEE Guide for Developing System Requirements Specifications', *IEEE Std 1233, 1998 Edition,* pp. 11, Dec. 1998. [Accessed: 05 Nov. 2020)
- [4] 'IEEE Guide for Developing System Requirements Specifications', *IEEE Std 1233, 1998 Edition,* pp. 16, Dec. 1998. [Accessed: 05 Nov. 2020)

Appendix

1 - **Use Case For "Qualifying for the final race"**

- Primary Actor: Player
- Precondition: Player has completed 3 legs and health has not fell below 0
- Trigger: Player has completed the third leg
- Main Success Scenario:
    1. System tracks fastest times across all 3 qualifying legs
    2. Player has achieved one of the fastest times
- Secondary scenarios:

1. Player has not achieved one of the fastest times
2. Player loses the game. Loss screen is displayed.

- Success Postcondition: Player proceeds to the final race
- Minimal Postcondition: Player does not qualify for the final race. Player loses the game.

2 - **Use Case For "Achieving a Medal"**

- Primary Actor: Player
- Precondition: Player has qualified for the final race
- Trigger: The start of the final race
- Main Success Scenario:
    1. System tracks time taken for boat to cross the finish line
    2. Player crosses the finish line
    3. Player achieved one of the fastest three times
- Secondary scenarios:
    1.1 Player does not complete the race, health has fallen below 0
        1.1.1 Player loses the game
    2.1 Player does not achieve one of the fastest three times
        2.1.1 Player loses the game
- Success Postcondition: Player is awarded 1st, 2nd or 3rd place medal.
- Minimal Postcondition: Player does not achieve top 3 position. Player loses the game.