Requirements

Mario

GROUP 28

Joseph Frankland

Anna Singleton

Saj Hoque

Leif Kemp

Shi (Lucy) Li

Hugo Kwok

2(a).

We started eliciting requirements by having an interview with the customer. This interview was a mix of open and closed, with the team having a predefined set of questions to ask the customer and then afterwards allowing any team members to ask new questions that may have arisen from how the customer answered the initial questions. We went into the interview willing to listen to the customer to make sure that the system meets the customer's vision. During the open section of the interview, we made sure our questions would prompt the customer rather than expecting the customer to tell us everything they wanted from the get go, to make it easier for the customer to talk about their expectations. We went with this approach because we felt as though we needed to guide the customer so that we could acquire all the required information to go forward with our requirements elicitation. The priority was ensuring that the customer felt comfortable and not overwhelmed, hence the open section was intended to encourage the customer to collaborate with us in order to achieve the best possible product. In reflection, this approach worked quite well considering our limited time. We were able to collect sufficient information on the customers requirements allowing us to progress to the next stage of the requirements elicitation. This approach was deemed better than our initial approach of having a solely closed question based interview which was swiftly deemed unsuitable due to the possible need of requiring many clarifications. This was further backed up by reflection on the interview, finding most of our useful information came from the open questions, where we were able to probe the customer with more detailed answers.

After the interview we had a team meeting to go over any notes we had taken and concentrate them into a first draft of requirements. We then took time to think about the system from a user perspective to help refine certain requirements and split our draft into both user requirements (UR) and system requirements (SR). User requirements consider what a user would need to be able to do using the technology and are generally more surface-level, written from an end-user point of view. System requirements refer to functionality that the system needs in order to fulfil the user requirements, and are written from a more technical point of view. Having these two groups aids in the design of the architecture of the program - the system requirements are more critical to the final product, and the user requirements are built on top of them. With the help of the User and System requirements we have also been able to distinguish further into functional (requirements that describe what a system should do) and non-functional (define system attributes such as security, reliability, performance, maintainability, scalability, and usabilities) requirements.

Using this information we formed our User Requirements table, which was further broken down into two more developed tables which consisted of functional and non-functional requirements. The User requirements table consisted of a column for a Unique ID with a supporting description column, helping us uniquely identify these requirements as well as give us a method of linking the requirements in other contexts. We've also introduced a priority column in this table in order to indicate how significant each requirement is in terms of need of fulfilment. This allows us to focus on specific requirements that need more attention preventing us from getting into a "feature creep" situation. The System requirements that are split into functional and non-functional requirements tables will be similar in the sense of both having a column to link back to its user requirement it was derived from. However, the non functional requirements shall also have a fit criterion column as this will be used for checking whether each requirement is valid, making our implementation stage more efficient with set criterias to be met.

Our methods towards requirements elicitation and the presentation of these have been achieved through research and analysis of the IEEE 29148-2018 International Standard - e.g. 9.4.15 - User Requirements

2(b).

User Requirement ID	Description	Priority
UR_MOVEMENT	The player should be able to control a boat allowing movement around the map. Enemy boats should also be able to move around the map.	Shall
UR_COMBAT	The player should be able to encounter other boats and colleges as well as engage in combat and vice versa.	Shall
UR_ COLLEGES	The player should be able to choose a college. There must be at least three colleges in the game.	Shall
UR_POINTS	The player should be able to accumulate points which must be achieved through passage of time. This rate may be altered through weather events or power ups.	Shall
UR_PLUNDER	The player should be able to accumulate plunder which must be achieved through defeating boats/colleges.	Shall
UR_OBJECTIVE	The player should be able to complete an objective that is not immediately achievable.	Shall
UR_PLATFORM	The staff should be able to run the game on any desktop OS.	Shall
UR_AUDIO	There must be background music and sound effects.	Should
UR_TIME	The player should be able to play for 5-10 minutes.	Shall
UR_ACCESSIBILITY	The player should be able to identify the game without relying on colours.	Should
UR_SAVE_GAME	The user should be able to save the game and reload when they want	Shall
UR_DIFFICULTY	The user should be able to choose from different difficulty levels before starting the game	Shall
UR_OBSTACLES	Gameplay includes obstacles in the map that the player has to avoid	Shall
UR_WEATHER	Gameplay will have a weather system which affects the stats of the player	Shall
UR_POWER_UP	Gameplay will involve providing power ups that can be collected and gives the players special powers, i.e. different weapons, speed boost etc.	Shall

Functional Requirement ID	Description	User Requirements ID
FR_PLAYER_MOVEMENT	The system shall allow the player to be able to control the movement of the user's boat (through keyboard controls).	UR_MOVEMENT
FR_PLAYER_COLLISION	The system shall ensure collisions between players and other objects are detected and hence the appropriate effect is delivered.	UR_MOVEMENT UR_OBSTACLES
FR_ENEMY_BOAT_MOVEMENT	For enemy boats the system should have an AI, complex enough to mimic a ship's movement.	UR_MOVEMENT
FR_ENEMY_BOAT_COLLSION	For enemy boats the system should have an AI, complex enough to allow their boats to avoid collision with objects in the map.	UR_MOVEMENT
FR_PLAYER_COMBAT	The system shall allow the player to engage in combat with the enemy boats or colleges.	UR_COMBAT

FR_ENEMY_BOAT_COMBAT	For enemy boats the system should have an AI, complex enough to mimic a ship's combat.	UR_COMBAT
FR_HEALTH_DISPLAY	The system shall show a health count of the player's or enemy's (boat or college) current health	UR_COMBAT
FR_COLLEGE_LIST	The system should provide a generated list of colleges that the player can choose from. The rest of these will be randomly distributed across the map as enemy colleges.	UR_COLLEGE
FR_ENEMY_COLLEGE	The system shall allow enemy colleges to engage in combat with the player boat.	UR_COMBAT, UR_COLLEGE
FR_PLAYER_COLLEGE	The system shall allow the player college to heal the player and also provide the player a base / friendly point to return to.	UR_COLLEGE
FR_POINTS_EARN	The system shall allow for points to be earned through the passage of time or through combat with other ships or colleges	UR_POINTS
FR_PLUNDER_EARN	The system shall allow for plunder to be earned through combat with other ships or colleges	UR_PLUNDER, UR_COMBAT
FR_PLUNDER_SPEND	The system shall allow for plunder to be spent for various upgrades on the shop screen.	UR_PLUNDER
FR_AUDIO_MUTE	The system should have mute-able audio.	UR_AUDIO
FR_VICTORY	The system should be able to determine whether the objective has been completed within the time limit, if so being forwarded to a victory screen.	UR_OBJECTIVE, UR_TIME
FR_DEFEAT	The system should be able to determine that the objective has not been completed within the time limit or the players health has been diminished, if so being forwarded to a defeat screen.	UR_OBJECTIVE, UR_TIME
FR_SAVE_GAME	The system should be able to save the game at any point of playing.	UR_SAVE_GAME
FR_LOAD_GAME	The system should be able to reload the game at any save point.	UR_LOAD_GAME

NF Requirement ID	Description	User Requirement	Fit Criterion
NFR_INPUT_DELAY	The system's inputs should have little delay.	UR_MOVEMENT	Input lag should never exceed 35 ms
NFR_DISPLAY	The system should be able to display different things such as objectives, time, health, points, plunder etc in real time.	UR_POINTS, UR_PLUNDER UR_OBJECTIVE UR_TIME	All these figures should be displayed on screen at all times and should also change when appropriate (lag shouldn't exceed 50 ms)
NFR_PLATFORMS	The game should be able to run on many desktop OS.	UR_PLATFORM	The game should run on Windows, Linux and Mac OS.

NFR_TUTORIAL	The player should be able to access a tutorial to explain the controls. Keyboard controls must be available.	UR_MOVEMENT UR_COMBAT	Provide an obvious option to view this on the initial menu screen
NFR_ACCESSIBILITY_VISUALS	The game will make sure that the player is able to understand the visuals from a few metres away.	UR_ACCESSIBILITY	Ensure objects are large enough and colours distinguishable.
NFR_ACCESSIBILITY_COLOURS	The game will make sure enemy boats will be distinct from player boats without relying on colours. This also stems to other imagery in the game (e.g. a rainstorm and the water)	UR_ACCESSIBILITY	Ensure objects in the game are distinguishable - different shaped boats, different textures for water.

2(b) Constraints

There are a few constraints when working on this project, one of the main constraints is the availability of team members as there may be times where not everyone is on the same page or working at the same pace, that again can be resolved by arranging frequent group meetings so that we make sure everyone is caught up, as well as this, we have a Trello board so that everyone knows what tasks are to be completed and what stages they are at. Accompanying this constraint is the time we have to complete the project. As we have a very limited amount of time to get it done, we need to continue this good management by continuing with group meetings for communications through the team and develop a good working schedule in order to have everyone on track for certain deadlines. Another constraint that has been elicited to us is the availability of different platforms for the game. This constraint however has already been considered and should not be an issue considering the language we are using (Java) is platform independent. To further strengthen this, we will test the product on several platforms, mainly including Windows, Linux and Mac OS.

2(b) Environmental assumptions, risks and alternatives

The biggest risk that comes with this project is feature creep. If we try to make features too complicated or add too many we risk not being able to complete the game within the deadline. An example of this would be in UR_POWER_UP, if the power ups are too complicated and difficult to implement it will jeopardise the plan as it would take too long to complete one feature. To alleviate this we will make sure to discuss possible features that will be implemented and then simplify or remove them if necessary. When making the game we have assumed that it will only be run on a few OSs, such as Mac OS, Windows or Linux and that they will have access to something to control the game, such as a mouse or a keyboard, and within this assumption we will assume that the keyboard is of a QWERTY layout so that we can make a nice control scheme. An alternative style for the game was discussed in an isometric approach however this was argued against as a group with the more preferred approach being top down.