Requirements

Team 14 & Team 12

Katie Maison
Saud Kidwai
Jacob Poulton
Cody Spinks
Felix Rizzo French
Joachim Jones

Keira Lauder
Amy Mason
Dan Ho
Sina Khosravi
Yibing Wang

2.(a)

To start the **requirements elicitation process**, we researched options and decided on the following: - First we carried out a document analysis of the product brief. [1]

- Then had a meeting with the customer who is interested in trying to market and sell the game. - This was a semi-structured interview in which a combination of predefined and unplanned questions were asked [1].
- It was useful to clarify some questions the team had about the product brief and ask if any additional requirements would be necessary.
- The next stage was to do a use case [WB.1] which is the scenario in which an actor interacts successfully with the system, in this case, successfully plays through the game.[2]
- From this, we have picked out the requirements necessary to make that scenario successful.

We have ensured that each requirement possesses the characteristics laid out in the guidelines laid out in the additional guidance for requirement-related activities in ISO/IEC/IEEE 12207 and ISO/IEC/IEEE 15288 [2]. These characteristics are that each requirement should be: **Necessary, Appropriate, Unambiguous, Complete, Singular, Feasible, Verifiable, Correct, Conforming**. [2]
To make sure the requirements themselves are well worded, we will use the language criteria in the guidance. This involves not using unbounded or ambiguous terms such as superlatives, vague pronouns, and subjective language.
To present the requirements we have used a table consisting of:

- ID
- Requirement description,
- Priority for the requirement,

And added additional information afterwards for those with:

- Environmental assumptions made
- Risks
- Alternatives

In the priority column, we decided [3]:

- Shall = a critical requirement
- Should = supports necessary system operations and is required eventually but is not as essential as the 'Shall' requirements.
- May = Design Decisions/Implied but is not essential

This is so that when implementing the requirements, we can decide which features are integral to meeting the brief and focus on the ones with higher priority.

SSON

The brief requires us to create a game that allows users to roam around a body of water and engage in combat with enemy colleges and ships in order to attempt to take them over all while avoiding a number of obstacles (e.g. Lake Monsters, Bad weather etc.). The user can also decide the level of difficulty that they want to play, depending on which they pick, there would be more or less obstacles and stronger/weaker colleges to combat. As the game progresses, the user will be able to collect 5 power-ups that they can use throughout the game. The brief also requires us to be able to save the game in its current state, and resume it at a later stage.

2.(b)

Risks and assumptions write up

Many of our technical and user requirements have associated risks in their implementation. For example some of the graphical requirements have both technical and political risks since there is an inherent risk of graphical glitches with implementation but there also may be disagreements about how certain graphics should look. Furthermore, the implementation as a whole has a number of risks related to feasibility, inner team politics, schedule and software.[2]

User Requirements

ID	Requirement	Priority
UR.START_SCRN	User can start the game via a start screen	May
UR.SEE_POS	The user must be able to see their sprite on the lake	Shall
UR.TUTORIAL	The user must be able to see an in-game tutorial alongside the actual game, so this is not separate.	Shall
UR.CLG_POS	User must be able to see where the colleges are relative to them	Should
UR.UPDATE_POS	The user must be able to move the ship on the lake	Shall
UR.COLLECT_PNTS	The user must be able to collect points	Shall
UR.ATK_CLG	The user must be able to attack colleges	Shall
UR.COLLECT_LOOT	The user must be able to collect loot	Shall
UR.VIEW_PNTS	User must be able to view their points/score	Shall

UR.VIEW_LOOT	User must be able to view their loot	Should
UR.CPTR_CLG	The user must be able to capture colleges	Shall
UR.SEE_TASKS	The user must be able to see tasks to complete	Should
UR.RESTART_GAME	The user must be able to restart the game at any time	Should
UR.FINISH_GAME	The user must be able to finish the game by returning ‘home’	May
UR.LOSE_GAME	The user must be able to lose/die	Shall
UR.POWER_UP	The user will be able to use 5 special power ups	Shall
UR.DIFFICULTY	The user will be able to choose the level of difficulty	Shall
UR.SAVE	The user can save the state of the game at any point and resume later	Shall
UR.COLLISION	The user (or other boats) will not be able to sail into obstacles or land. It cannot collide	Should
UR.SPEND	The user will be able to spend their loot throughout the game	Shall

Software requirements

Functional requirements

ID	Requirement	User Requirement	Priority
FR.START.SCRN	Software must display a start screen	UR.START_SCRN	May
FR.START.START	Start screen shall have a ‘start’ button	UR.START_SCRN	May
FR.START.EXIT	Start screen shall have an ‘exit’ button	UR.START_SCRN	May
FR.DISPLAY.GUI	The software must render environment with a Graphical user interface	UR.START_SCRN	Shall
FR.DISPLAY.EDGE	The software must display a lake with boundaries	UR.COLLISION	Shall
FR.DISPLAY.SHIP	The software must display the user in the form of a privateer sprite	UR.SEE_POS	Shall
FR.DISPLAY.CLG	The software must display colleges	UR.CLG_POS	Shall
FR.DISPLAY.DOCK	The software must display other ships docked at colleges	UR.CLG_POS	Should
FR.DISPLAY.HUD	The software must include loot, points, health and a mini map in HUD	UR.COLLECT_LOOT UR.COLLECT_LOOT UR.VIEW_PNTS UR.VIEW_LOOT	Shall
FR.DISPLAY.CAM	The software’s camera must follow the users sprite	UR.SEE_POS	Should
FR.TUTORIAL	The software must display a tutorial embedded within the gameplay to have during the actual game.	UR.TUTORIAL	Shall
FR.FREEMOVE	The software must allow the users sprite to freely move around the lake via input	UR.UPDATE_POS	Shall
FR.COLLISION	The software must implement a object collision system	UR.COLLISION	Shall
FR.BOUNDARY	The software must implement a boundary to the gameplay area	UR.COLLISION	Shall
FR.ATTACKCURSOR	The users sprite must attack the colleges when they are clicked on	UR.ATK_CLG	Shall
FR.CLG_HEALTH	The college must lose health when attacked	UR.ATK_CLG	Shall

FR.CLG_CONVERT	When college loses all health, college becomes friendly	UR.CPTR_CLG	Should
FR.AWRD.POINTS	The software must award points passively and for completing tasks/defeating colleges.	UR.COLLECT-POINTS UR.VIEW-PNTS	Shall
FR.AWRD.GOLD	The software must award gold for completing tasks/defeating colleges.	UR.COLLECT_LOOT UR.VIEW_LOOT	Shall
FR.CLG_ATTACK	The college must attack the player (**added 25/01/22 due to missed requirement)**	UR.ATK_CLG	Shall
FR.CLG_INFO	College must be implemented with [.1] colour, [.2] name and [.3] friendly docked boats [4].Health [5]. Plunder		Shall
FR.OPTNL_TASKS	The game must generate a series of optional tasks that the user may complete	UR.SEE_TASKS	Shall
FR.KILL_SCRN	The game must display a kill screen on [.1] victory, [.2] loss, [.3] on restart button	UR.FINISH_GAME	May
FR.GAME_SOUND	The game shall have a sound: [1] Music [2] Sound effects [3] Be mute-able		May
FR.BAD_WTHR	The software must display bad weather on the screen and as the player sails over it, it will decrease the speed of the player.		Shall
FR.DISPLAY_POWER_UP	The software must display the power ups on the screen when the user has collected it, ready to be used.	UR.POWER_UP	Shall
FR.DISPLAY_DIFFICULTY	The software must give difficulty option in ‘start’ screen	UR.DIFFICULTY	Shall
FR.SPEND	The user will be able to spend loot that they have earned throughout the game. The loot will exchange for something else.	UR.SPEND	Shall
FR.SAVE	There will be a button on the screen that will be able to save the current game.	UR.SAVE	Shall
FR.ENEMY_ATK	Enemy boats will follow and shoot at the user. The user will be able to shoot at the enemy boat.	UR.ATK_CLG	Shall

Non Functional requirements

ID	Requirement	Priority
NFR.NETWORK	The program shall not connect to the network	Shall
NFR.STABLE	The Game shall be stable and not crash	Shall
NFR.GAME_TIME	The game shall only last 5-10 minutes - Fit criteria: 9/10 run throughs will last less than 10 minutes	Shall

NFR.UPDATE	Updated game files should be easily available	May
NFR.CVD	The game must be accessible to those with colour vision deficiency	Should
NFR.LOAD_TIME	The software must load quickly - Fit criteria: The game must load in < under 30 seconds	Should
NFR.SIMPLICITY	The game must be simple enough/have a good enough tutorial to be able to be played by those with no prior experience - Fit criteria: 9/10 players will be able to understand the game by the end of the tutorial	Should

Constraint requirements

ID	Requirement	Priority
CR.RESOLUTION	The game should be able to be displayed on range of resolutions and make good use of space - Fit criteria: Display on 13”-27” screens, test with 13”, 47”	Should
CR.LOW_SPEC	The game must run on a system with minimum specs 4gb RAM, a standard UK keyboard and mouse - Fit criteria: Game must be playable on system with these specs	Should
CR.OS	The game should work on all major operating systems. It needs to work on Windows and Linux.	Shall
CR.DEADLINE	Project must be completed by Sum/3/Wednesday	Shall

Environmental Factors:

-NFR.GAME_TIME - The game will be used at an open day where time is limited and fast throughput of people is preferable

- NFR.UPDATE - The user may have little to no technical experience, eg. with github and creating JAR files. **- NFR.SIMPLICITY, UR.TUTORIAL, FR.TUTORIAL -** The user may have little to no game playing experience. - **UR.RESTART_GAME** The user must be able to restart the game at any time

- CR.OS - The Computer Science department has Windows and Linux operating systems. The environmental assumption is that the department would still have these operating systems so the game would need to work on these.

Associated Risks/Further Comments:

- **UR.START_SCRN:** Any offensive screen names or ones that contain profanity may have to be filtered out. This could be mitigated by a potential filter in the screen box.
- **UR.CLG_POS:** Arrows could be confusing for users, especially colourblind ones and will need identifiers for friendly/enemy colleges.An **alternative** to this could be via the MiniMap
- **UR.VIEW_LOOT:** May take up too much time while being low on our priority list. Discussed in risk 004.
- **UR.SEE_TASKS:** Tasks could come across as vague and confusing for some users. ● For any requirements that are about display or graphics there is a potential for internal disagreements about how the UI should look, resulting in unnecessary lost time.
- **FR.COLLISION:** Collision systems can often result in bugs and glitches with clipping resulting in a lowered user experience.
- **FR.FREEMOVE:** Controls must be intuitive and movement must be seamless.
- **FR.AWRD.POINTS/FR.AWRD.GOLD:** Risk of differences between points and gold being unclear to users.
- **NFR.STABLE:** Although the game has low machine requirements, it is hard for us to control the number of crashes that may occur.
- **NFR.GAME_TIME:** There may be a situation in which we compromise gameplay elements in order to reduce game time.
- **CR.RESOLUTION:** Risk of graphical glitches in scaling if not implemented properly.
- **CR.LOW_SPEC:** Performance cap may limit our sprite usage as too many may cause performance issues on lower spec machines

Bibliography

[1]
M. Yousuf and M. Asger, “Comparison of Various Requirements Elicitation Techniques,” *International Journal of Computer Applications (0975 – 8887)*, Apr. 2015. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.5985&rep=rep1&type=pdf

. [WB.1] https://keiral11.github.io/Team12Website/usecase

[2]
“ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering,” *ISO/IEC/IEEE 29148:2018(E)*, vol. 1–104, 2019, doi: 10.1109/ieeestd.2018.8559686.https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8559686

[3] K.E Wiegers (Sept. 1999). “First Things First: Prioritizing Requirements”, Process Impact, https://www.processimpact.com/articles/prioritizing.pdf