2. Requirements

Elicitation of requirements

1. The provided product brief indicated the overall goals and intentions of the finished product and contained general descriptions of its desired functionality
2. Group brainstorming session was held to compare our interpretations of the brief and to raise a list of questions to address to the customer
3. In a group meeting with the customer, answers to the questions and any other customer comments were detailed in informal meeting notes.
4. Recorded info was formalised as a set of user requirements.
5. User requirements were distilled down into more specific functional and non-functional requirements.
    a. Functional requirements detailed concrete, specific functionality and capabilities of the product as related to its software implementation.
    b. Non-functional requirements captured the performative characteristics of the completed product as a whole which could be perceived by the user or tester.
6. Resulting functional and non-functional requirements were closely evaluated for possible risks to their implementation; these were detailed within the risk register.

Research into requirement specification and presentation

- IEEE requirements engineering document[1]:
    - Provided comprehensive information on all aspects of requirement elicitation and presentation, although sections 5.1-5.2.8 and 6.1-6.6.3 were most helpful
    - Contained robust justification for the need for requirements and their role in the overall software lifecycle process
    - Informed our choice of specific language, standardised subjects and verbs (user, shall, etc.), the choice of imperative tone, and justifications for these
    - Ultimately, aimed at larger, more critical projects than ours

- ENG1 lecture on requirements engineering:
    - Provided an excellent overview of the motivations for requirements engineering and a lucid overview of this process
    - Introduced the user/functional/non-functional requirements methodology which proved an excellent fit for our scope of project (versus lifecycle-based requirement methodology in the IEEE document)
    - Demonstrated requirements tables as a tool for writing down and detailing requirements

Informed by these resources, we chose a tabular format for implementing for the requirements register, allowing us to easily add additional metadata to individual rows as extra columns and permitting a quick, comprehensive overview of the entire register. All rows were labelled with unique identifiers, permitting cross-referencing between user and (non-)functional requirements, as well as with other sections of project documentation, such as the risk register and architecture specs.

---

[1] *Systems and software engineering -- Life cycle processes -- Requirements engineering*, ISO/IEC/IEEE 29148:2018(E), 2018.

User Requirements

Category	ID	Description	Priority
Game setting	UR_PLATFORM	The user shall use a standard laptop PC to play the game	Shall
Game setting	UR_GAME_INIT	The user shall begin a new game from an initial state	Shall
Game setting	UR_SHIP_CONTROL	The user shall control a ship sailing across the great Lake of York	Shall
Game setting	UR_COMPETING_COLLEGES	The user shall encounter at least 3 other colleges	Shall
Game setting	UR_LEARNING_CURVE	The user shall play the game without training	Shall
Game setting	UR_GAME_DURATION	The user shall be able to complete the game within a ~5 minute timespan	Shall
Game setting	UR_GAME_OBSERVABILITY	The game shall accomodate onlookers in the PCs surroundings	Shall
Encounters	UR_FRIENDLY_SHIP_ENCOUNTER	The user shall encounter friendly NPC ships	May
Encounters	UR_HOSTILE_SHIP_ENCOUNTER	The user shall encounter hostile NPC ships	Shall
Encounters	UR_FIRE_WEAPONS	The user shall be able to fire weapons from the ship	Shall
Encounters	UR_BULLET_DODGE	The user shall be able to maneuver their ship to dodge fired munitions	Shall
Encounters	UR_FRIENDLY_BUILDING_INTERACT	The user shall interact with friendly buildings	May
Encounters	UR_HOSTILE_BUILDING_COMBAT	The user shall engage in combat with hostile buildings	Shall
Encounters	UR_HOSTILE_COLLEGE_CAPTURE	The user shall capture other colleges via combat	Shall
Earnables	UR_EARN_MONEY	The user shall earn money	Shall
Earnables	UR_EARN_POINTS	The user shall earn points	Shall
Progress	UR_QUEST_PROGRESS	The user shall progress through a series of quests	Shall
Progress	UR_GAME_WIN	The user shall win the game through achieving an ultimate objective unlocked by the fulfilment of preceding requirements/quests	Shall
Progress	UR_GAME_LOSE	The user shall lose the game through being defeated in combat	Shall
Encounters	UR_SHIP_COMBAT	The user shall engage in combat with other ships	Shall
Encounters	UR_OBSTACLE_ENCOUNTER	The user shall encounter obstacles while sailing in game	Shall
Encounters	UR_WEATHER_ENCOUNTER	The user shall encounter bad weather while sailing	Shall
Earnables	UR_SPEND_MONEY	The user shall spend the money earned	Shall
Earnables	UR_POWERUPS	The user shall have 5 powerups to use	Shall
Game setting	UR_SAVELOAD	The user shall have the abiltiy to save and load the game	Shall
Game setting	UR_DIFFICULTY	The user shall have the ability to select a difficulty	Shall
Game setting	UR_QUIT	The user shall have the ability to quit the game	Shall

Functional Requirements

ID	Description	User requirement	Risks	Priority
FR_MENU_KB_INPUT	The game shall accept keyboard input for menu navigation	UR_PLATFORM		Shall
FR_VIEWPORT_SCALING	The game shall render on a 13"-27" monitor	UR_PLATFORM	R6	Shall
FR_MIN_FPS	The game shall render at a minimum of 30 FPS	UR_PLATFORM	R7, R3	Shall
FR_CROSS_PLATFORM_WIN	The game shall be playable on Windows	UR_PLATFORM		Shall
FR_CROSS_PLATFORM_MAC	The game shall be playable on Mac OS	UR_PLATFORM	R9, R10	May
FR_CROSS_PLATFORM_GNU_LINUX	The game shall be playable on GNU/Linux	UR_PLATFORM		Shall
FR_GAME_RESET	The game shall allow restarting play from an initial configuration	UR_GAME_INIT		Shall
FR_SHIP_KB_INPUT	The game shall accept keyboard input for ship control	UR_SHIP_CONTROL		Shall
FR_COLLEGE_ENTITY_TRACKING	The game shall keep track of ships and buildings for a minimum of 3 distinct factions	UR_COMPETING_COLLEGES		Shall
FR_FRIENDLY_AI	The game shall control the actions of friendly ships	UR_FRIENDLY_SHIP_ENCOUNTER		Shall
FR_FRIENDLY_INTERACT	The game shall allow user interaction with friendly ships	UR_FRIENDLY_SHIP_ENCOUNTER		Shall
FR_HOSTILE_AI	The game shall control the actions of enemy ships	UR_HOSTILE_SHIP_ENCOUNTER	R1	Shall
FR_PLAYER_FIRE	The game shall enable the user to fire ship weapons	UR_FIRE_WEAPONS		Shall
FR_PLAYER_AMMO	The game shall maintain the state of the user's ship armament and ammunition	UR_FIRE_WEAPONS		Shall
FR_BULLET_TRAVEL	The game shall render the travel of a ship's fired munition	UR_BULLET_DODGE	R2	Shall
FR_MONEY_TRACKING	The game shall keep track of a player's money	UR_EARN_MONEY		Shall
FR_MONEY_UPDATE	The game shall give money on success in quests and encounters	UR_EARN_MONEY		Shall
FR_POINTS_TRACKING	The game shall keep track of a player's points	UR_EARN_POINTS		Shall
FR_POINTS_UPDATE	The game shall give points with time survived and obstacles navigated	UR_EARN_POINTS		Shall
FR_XP_TRACKING	The game shall keep track of a player's XP	UR_EARN_XP		Shall
FR_XP_UPDATE	The game shall give XP on successful combat encounters completed	UR_EARN_XP		Shall
FR_QUEST_TRACKING	The game shall maintain the state of the user's progress through multiple objectives	UR_QUEST_PROGRESS		Shall
FR_QUEST_RANDOMISE	The game shall randomise user's objectives between different playthroughs	UR_QUEST_PROGRESS		May
FR_QUEST_OBJECTIVE	The game shall associate quest objectives with game entities	UR_QUEST_PROGRESS		Shall
FR_BOSS_UNLOCK_TRACKING	The game shall monitor quest progression status prior to unlocking final objective	UR_GAME_WIN		Shall
FR_BOSS_SPAWN	The game shall spawn boss upon final objective ready status	UR_GAME_WIN		May
FR_GAME_WIN	The game shall display game stats upon successful completion of boss encounter	UR_GAME_WIN		May
FR_PLAYER_DEFEAT	The game shall display game stats upon player defeat	UR_GAME_LOSE		Shall
FR_SCENARIO_FAIL	The game shall display game stats upon game over scenario completion	UR_GAME_LOSE		Shall

Non-Functional Requirements

ID	Description	User requirement	Fit criteria	Risks
NFR_SHIP_COLLISIONS	The game shall detect collisions between different ships	UR_HOSTILE_SHIP_ENCOUNTER	Distance between drawn assets <5px	R4
NFR_WORLD_COLLISIONS	The game shall detect collisions between ships and world objects	UR_COMPETING_COLLEGES	Distance between drawn assets <5px	
NFR_BULLET_COLLISIONS	The game shall detect collisions between game entities and fired munitions	UR_BULLET_DODGE	Distance between drawn assets <5px	
NFR_USER_INPUT_LAG	The game shall be responsive to user input	UR_SHIP_CONTROL	Input lag <200ms	
NFR_AI_LAG	NPC actions' responsiveness shall approximate that of player actions	UR_HOSTILE_SHIP_ENCOUNTER	AI response time <200ms	R3
NFR_RENDER_SMOOTHNESS	The game world shall render smoothly during player movement	UR_SHIP_CONTROL	Visual render lag <200ms	R8
NFR_COLOURBLINDNESS	Game map and assets should be distinguishable by a colourblind person	UR_PLATFORM	Subjective screenshot test via colourblind accessibility evaluation app	
NFR_EASE_OF_USE	The game shall be self-explainable and feature obvious controls	UR_LEARNING_CURVE	Tester must be able to pick up and play with no prior instruction	
NFR_GAME_DURATION	The game shall finish within ~5 mins in a win or loss for the player	UR_GAME_DURATION	Tester must reach the game stats screen within 4-6 mins	
NFR_LARGE_ASSETS	The game assets shall be large enough to observe from several metre's distance away on a standard laptop PC screen	UR_GAME_OBSERVABILITY	Observer standing 2m away should be able to answer questions about gameplay state	