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. A 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
Encounters	UR_FRIENDLY_SHIP_ENCOUNTER	The user shall encounter friendly NPC ships	Shall
Encounters	UR_BULLET_DODGE	The user shall be able to manoeuvre their ship to dodge ~~fired munitions~~ cannonballs	Shall
Encounters	UR_FRIENDLY_BUILDING_INTERACT	The user shall interact with friendly buildings	Shall
Earnables	UR_EARN_POINTS	The user shall earn points	Shall
Earnables	UR_EARN_XP	The user shall earn XP	May
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	~~Assessment 2~~ Shall
Encounters	UR_OBSTACLE_ENCOUNTER	The user shall encounter obstacles while sailing	~~Assessment 2~~ Shall
Encounters	UR_WEATHER_ENCOUNTER	The user shall encounter bad weather while sailing	~~Assessment 2~~ Shall
Earnables	UR_SPEND_MONEY	The user shall spend the money earned	~~Assessment 2~~ Shall

New requirements from the stakeholder:

Earnables	UR_POWERUPS	The user shall be able to spend plunder on powerups	Shall
Encounters	UR_DIFFICULTIES	The user shall be able to choose more than one difficulty	Shall

Functional Requirements

ID	Description	User requirement	Priority
FR_MENU_KB_INPUT	The game shall accept keyboard input for menu navigation	UR_PLATFORM	Shall
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_FRIENDLY_AI	The game shall control the actions of friendly ships	UR_FRIENDLY_SHIP_ENCOUNTER	Shall
R_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	Shall
FR_POINTS_TRACKING	The game shall keep track of a player's points	UR_EARN_POINTS	May
FR_POINTS_UPDATE	The game shall give points with time survived and obstacles navigated	UR_EARN_POINTS	May
FR_XP_TRACKING	The game shall keep track of a player's XP	UR_EARN_XP	May
FR_XP_UPDATE	The game shall give XP on successful combat encounters completed	UR_EARN_XP	May
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 **all quests**	UR_GAME_WIN	May
FR_PLAYER_DEFEAT	The game shall display game stats upon player defeat	UR_GAME_LOSE	May
FR_SCENARIO_FAIL	The game shall display game stats upon game over scenario completion	UR_GAME_LOSE	Shall

New requirement from the stakeholder:

FR_SAVE_STATE	The game shall allow saving the state of the game to resume at a later time	UR_GAME_INIT	May

Non-Functional Requirements

ID	Description	User Requirement	Fit criteria
NFR_SHIP_COLLISIONS	The game shall detect collisions between different ships	UR_HOSTILE_SHIP_ENCOUNTER	Distance between drawn assets <5px
NFR_GAME_DURATION	The game shall finish within ~5 minutes in a win or loss for the player	UR_GAME_DURATION	Tester must reach the game stats screen within 4-6 minutes

The outcomes of all these requirements are to be explained in part b of the implementation deliverable.