Requirements Specification

Team: No

Team Members:

- Jack Burman
- Sam Churchill
- C Lloyd
- Sam Ralph
- Rebecca Wardle
- Jiacheng Wu

Introduction

The initial requirements were extracted from the product brief and turned into a table of user requirements as well as the single statement of need (SSON). Where the requirements raised more questions about the clients wishes we created a series of questions. At the interview we then used the questions to collect more details about existing user requirements as well as detailing new ones. From the collection of user requirements we then deducted a series of system requirements both functional and non-functional that would be required to implement the proposed product.

Project requirements, in this document, are laid out in a table. This allows easy reading as well as additional notes such as priority. Making this document accessible and up to date is vital to the document's purpose and should help ensure constant use and reference by the team.

Whilst the requirements laid out by the client place some restrictions on the product there are certain aspects that have been left for the team to decide. These uncertainties are mostly left in the balancing of the game and so fall within existing requirements but some items outlined in the interview but not contained within the design brief require novel solutions.

The requirements that were critical to acceptance of the product are given the priority “Shall” within the table. So as to properly focus the coding these requirements were given lower priority of “Medium” or “Low” as they are not vital to the success of the product and in some cases are purely superficial.

The categories used in the table were chosen based on examples:
The table outlined briefly in Product requirements documents, downsized (https://www.atlassian.com/agile/product-management/requirements) is designed to give people a quick overview of the requirements. In that aim it includes only 4 columns and uses “User Stories” to describe the requirements.
How to Write a Software Requirements Specification (SRS Document) describes a more formal and complete document, using headings to subdivide requirements and giving the readers more information about the project requirements as well as assumptions and risks.
IEEE Recommended Practice for Software Requirements Specifications. (n.d.). doi:10.1109/ieeestd.1998.88286 lays out principles and guidelines for writing good SRSs as well as common pitfalls.

SSON

A single-player game, the aim of which is to arrest the infiltrators before they manage to destroy a critical number of key systems of the station.
The game is won when all eight infiltrators have been arrested. The game is lost when infiltrators have successfully destroyed more than 15 systems of the space station.

User Requirements

ID	Description	Priority
UR_PLAYER_ CONTROL	The player must be able to control something that moves around a space station	Shall
UR_ROOM_TYPES	There must be at least 4 types of rooms in the station (e.g. cargo bays, personnel, bar, infirmary)	Shall
UR_TELEPORTATION	Rooms can have teleportation pads from which Auber (but not infiltrators) can teleport to any other teleportation pad in the station	Shall
UR_INFILTRATOR_ ABILITIES	There must be at least 3 distinct special abilities within the group of infiltrators	Shall
UR_HEAL	Auber can teleport to the infirmary to heal	Shall
UR_GAME_OVER	If Auber reaches 0 health the game ends	Shall
UR_REAL_TIME	The game must be real-time (not turn-based)	Shall
UR_ARREST	Auber should be able to arrest and move an infiltrator to the brig	Shall
UR_INFRILTRATOR_ DAMAGE	The infiltrators should be able to damage systems of the space station	Shall
UR_WIN	The game will be won when 8 infiltrators have been arrested	Shall
UR_LOSS	The game will be lost when 15 or more systems are destroyed	Shall
UR_MODES	There must be “single player mode” and a “demo mode”	Shall
UR_ART_STYLE	The game should be a 2D top-down game	Shall
UR_BYSTANDERS	There should be bystanders in the game who look like the infiltrators but who do not cause harm	Should
UR_GAME_REPLAY	Record the movements of the game being played so that it can be played back	Should
UR_DIFFICULTY_ LEVELS	The game should have different levels of difficulty for the player to choose from	Low
UR_NPC_ DESCRITION	Aubers get a photo/description of the infiltrator if they damage too many systems in front of NPCs	Low
UR_REPAIR	Aubers can repair systems	Low

Functional Requirements

ID	Description	User Requirement
FR_PLAYER_INPUT	The user must be able to use key presses to move the Auber (an on screen representation of their character) around a space station	UR_PLAYER_ CONTROL
FR_AUBER_ COLISION_ DETECTION	The system should detect when the player's character walks into wall and stop it's motion	UR_PLAYER_ CONTROL
FR_ROOM_ MOVEMENTS	Players should be able to move (walk) between the different areas	UR_ROOM_TYPES
FR_SYSTEMS	There must be at least 15 systems on the space station	UR_LOSS
FR_FIND_SYSTEM	Infiltrators must be able find their way to systems	UR_LOSS
FR_DESTROY_ SYSTEMS	Infiltrators must be able to damage and destroy systems	UR_LOSS
FR_ARREST	Auber must be able to arrest infiltrators	UR_WIN
FR_INFRILRATOR_ ABILITIES	Infiltrators should have different special abilities, or may have none at all	UR_INFILTRATOR_ ABILITIES
FR_HEAL	Auber can heal in the infirmary	UR_HEAL
FR_AUBER_ START_TELEPORT	When the Auber steps into the teleporter it Present the player with a choice of teleporter pads to move to	UR_TELEPORTATION
FR_AUBER_END_TE LEPORT	The Auber leaves the teleport at the chosen location	UR_TELEPORTATION
FR_AUBER_NOTIFY	The auber must be notified when and where a system is being damaged	UR_LOSS
FR_SCREEN_UPDAT E	The game must update the screen in real time	UR_REAL_TIME
FR_AI_DECISION	The AI (Infiltrators & NPCs) must make decisions and move in real time with the game screen update	UR_REAL_TIME
FR_DEMO	The game must be able to record the	UR_MODES

	motions and actions of all the characters in the game. This can then be played back in another mode	

Non-Functional Requirements

ID	Description	Fit Criteria	User Requirement
NFR_MODES	There should be a demo mode, and a single player mode	The player should be able to watch back a playthrough	UR_MODES
NFR_DOCUMENTATION	All complex methods should include a docstring before them to describe inputs, outputs and a general description	All methods that aren't just a get or set method must include a docstring	
NFR_SYSTEM_DIFFERENCE	There must be some difference between the systems as they are represented in the game	A user must be able to differentiate the different systems on the station	UR_ROOM_TYPES
NFR_GAME_TIME	The game must last around 1-2 minutes per round	This will have to be achieved with balancing all the aspects of the game	UR_WIN
NFR_BYSTANDER_APPEARANCE	The player must be able to distinguish the bystanders from the infiltrators in some regard whether it be: looks, sound, AI, etc	To make the game in any way challenging the infiltrators should blend in with the bystanders Only differentiate by AI behaviour	UR_BYSTANDERS