UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

ENG1 Group Assessment 1 Team 1

Auber

Req1.pdf - Requirements

Group Members:
Jonathan Davies
Jamie Hewison
Harry Smith
Annie Sweeney
Zee Thompson
Mark Varnaliy

Introduction to Requirements

As a group, we elicited our requirements in a few ways. Firstly, we read the project briefing that had been given to us, but we felt that some of these were slightly vague. To clarify these, we set up meetings with our customer - Prof. Dimitris Kolovos - to ask about these in more detail. These meetings took place via Zoom on the 20th October and 3rd November 2020. These requirements are presented as part of the requirements table below.

The first of these meetings elicited the majority of the gameplay requirements, referred to below as 'user requirements' and 'functional requirements'. This included questions such as "*How does Auber take damage and what happens when they die?*", "*How many rooms are expected to be on the space station?*", and "*What's supposed to happen if [the game] crashes?*". This cleared up most aspects of the gameplay requirements, but we still had a few questions about non-functional requirements - such as whether the game should be able to scale to different sized screens - which were cleared up with the customer during our second meeting.

We also have user requirements and software requirements, which include functional and non-functional requirements, which go into more detail. The user requirements are statements about the tasks that the user should be able to carry out within the game, and are written with as little technical detail as possible - they should be understood by the average user. The software requirements offer a description of how the game will deliver upon the needs of its users, and are written with technical implementations in mind.

Each requirement has a unique and meaningful ID, prefixed with either UR (User Requirement), FR (Functional Requirement) or NR (Non-functional Requirement). User requirements also state whether they are 'shall' requirements - those that *must* be implemented - or 'should' requirements - those that it would be nice to have implemented, but aren't absolutely necessary.

The table for Functional Requirements also displays which User Requirement(s) that each requirement meets. We used this so that we can easily link together the desired feature in the game to the technical implementation of it. The table for Non-functional Requirements features a column for 'Fit Criteria' - these are criteria that ensure we are meeting that specific Non-functional Requirement.

The requirements have been set out in tabular form below. Each requirement is concise so they can be easily understood when implementing the game, and they are also designed to be measurable so we can see how close to hitting the requirements we are, and it should be easily verifiable if we've covered them in the game or not. The tables below also classify the requirements as User Requirements, Functional Requirements and Non-functional Requirements, so the type of each requirement is easily identifiable [1]. We decided to show these in a tabular format as we deemed this the best way to store all the information neatly, so that a specific requirement and its information could be found quickly when needed.

Tabular Presentation of Requirements

User Requirements

ID	Description	Priority
UR_UE	The system shall offer a pleasant user experience.	Shall
UR_INPUT	The player's character shall react to any input immediately	Shall
UR-TELEPORT	The player shall be able to teleport from any room to any other room	Shall
UR-HEAL	The player shall be able to heal when in the infirmary	Shall
UR-ARREST	The player shall be able to arrest any infiltrator	Shall
UR-WIN	The player shall win and the game end when they have arrested all 8 infiltrators	Shall
UR-LOSE	The player shall lose the game when more than 15 systems have been destroyed	Shall
UR-MENU	The menu of the game should be easy to navigate	Should
UR-OFFLINE	The game should be able to play without an internet connection	Should
UR-NAVIGATE	The rooms should be easy to navigate to	Should
UR-DESTROYED_SYSTEM	The player shall be alerted when a system has been destroyed	Shall
UR-HELP	The player should be able to access information to help understand what abilities the infiltrators have	Should

Functional Requirements:

ID	Description	Meets User Requirements
FR_TELEPORTATION	The system shall provide teleport option while player is in any room	UR_TELEPORT
FR_HEAL	The system shall automatically heal player in the infirmary	UR_HEAL
FR_ARREST	The system shall provide arrest option while player and infiltrator are in the same room	UR_ARREST
FR_WIN	The game shall automatically end when all infiltrators are captured	UR_WIN
FR_LOSE	The game shall automatically end when more than 15 rooms have been destroyed	UR_LOSE
FR_MAP	The game shall provide a map with rooms locations and their current state(destroyed or not).	UR_DESTROYED_SYSTEM
FR_CONTROLS	The system shall immediately rect to user's input	UR_INPUT
FR_ENEMIES	There shall be 8 infiltrators with at least 3 unique abilities and provide description if requested.	UR_HELP

Non-Functional Requirements:

ID	Description	User Requirements	Fit Criteria
NFR_USABILITY	All customer-facing messages shall be in plain English and will not use technical jargon.	UR_UE	Most of users must speak English
NFR_ACCESSIBILITY	A colourblind mode should be offered to the player	UR_UE	
NFR_DOCUMENTATION	The game shall have in-game manual with	UR_HELP	Manual shall be easy to find and

	controls and abilities the infiltrators have		understand
NFR_OPERABILITY	The game shall be intuitive	UR_UE	Most of the users should understand how to play without reading a manual
NFR_WINDOW	The game window must be resizable to fit different screens	UR_UE	

References

[1] "Software requirements: 8 best practices to write them", *Estimancy*, 2018. [Online]. Available:
https://www.estimancy.com/en/2018/11/05/software-requirements-8-best-practices-to-write-them/. [Accessed: 12- Nov- 2020].