Requirements

Introduction

This document contains a set of detailed requirements for our project, which we will refer back to throughout the software development process. It is intended to be a living document, which we will continuously update as our project's requirements change. In this introduction, we will describe the methods we used to gather and develop our requirements, as well as provide an explanation for the way in which we have categorised our requirements and presented them within this document.

In order to begin eliciting the requirements for the project we read through the project brief provided to us by the client. The brief described a number of qualities the system must have, and some of its functionality, making it a good starting point for determining our requirements.

Early in the requirements elicitation process, the team had a number of questions about the qualities and functionality of the system that we felt were left unanswered by the project brief. We decided to interview the client, in order to seek guidance on how to interpret the brief. This proved useful, as the client was able to give us a clearer idea of what they expected from the product.

We also wrote four short user stories, each aiming to describe a different "scenario of use that might be experienced by a system user". [1,p.79] Having the team consider the end user's interactions with the system helped us to write better user requirements. The user stories can be found on our website in the 'downloads' section.[2]

Before we began writing our requirements document, we conducted some research into how best to structure and organise our requirements. Sommerville [1] contains some useful insights into how to present a requirements document. Sommerville suggests that user requirements should be listed before system requirements. [1, p.128] We agreed with this assessment and decided to present our requirements in this order. This is because many of our user requirements can be broken down into sets of system requirements. Our requirements fit into three categories: constraint requirements, user requirements and system requirements. We split our system requirements into two subcategories: functional and nonfunctional system requirements.[3] We decided that constraint requirements should come first, followed by user then system requirements. We felt that constraint requirements should come first because they must be kept in mind whilst developing other types of requirements as they are "Global issues that shape requirements". [3]

We decided to present our requirements in tables, in order to make the document more readable and easier to modify. Each table contains requirements from a specific category. Each table provides the requirement definition, assigned name and any environmental assumptions that are associated with the requirement. An assumption might have been included to clarify a certain aspect of the requirement, or to provide details of the system that must be known by the reader for the requirement to make sense.

Within each table, the requirements are in priority order. The importance of a requirement is indicated both by its placement in the table, and by a number next to it's name. This will help remind the team, throughout the planning and development of the project, which of our goals for the product are the most important. The numbering system for requirements will also prove useful later in the project, by making specific requirements easier to reference.

Some of our requirements are 'riskier' than others. High risk requirements are those which we might not be able to pursue if the team runs into problems, like those described in our risk register. These requirements are generally lower priority. That means the impact on the overall system of leaving them unfulfilled, should we have to, will be small. An example of a high risk requirement is functional system requirement 8, the 'save system'. This requirement is high risk because we anticipate that creating a system for saving and loading games will be difficult, and therefore may not be possible if we fall behind schedule.

We have tried to make our requirements as testable as possible, in order to make it easier to compare the product we produce to our requirements later on in the project. A testable requirement is one which can be easily tested for fulfillment.

Constraint Requirements

Name	Requirement	Assumptions
C1.End user hardware	The game shall be playable on a laptop or desktop computer, using keyboard and mouse for input.	
C2.Content suitability	The content and presentation of the game should be suitable for use by the University of York Communications Office, who may wish to use the project during their promotional activities, e.g. at university Open Days.	
C3.Setting constraint	The game must be set in a fictionalised version of the Ron Cooke Hub, during a lock-in costume party.	

User Requirements

Name	Requirement	Assumptions
U1.Character Selection	At the start of each new game the player will be able to choose to play as one of three ‘great detective’ characters. Each detective will have three unique modes of questioning available to them. This will determine the three dialogue options the player is presented with when questioning an NPC.	There will be a character selection menu, displayed at the start of each new game, which the player can use to select their great detective for that game.
U2.Navigation and Interaction	The user will be able to navigate their chosen ‘great detective’ character through the game world, and interact with the objects and NPC characters distributed throughout it.	
U3.Interacting with NPCs	When the user chooses to engage with an NPC, they will have three modes of interaction available to them. They are: ‘ignore’ (do not interact further), ‘question’ (begin interrogating the character) and ‘accuse’ (blame the character for the murder).	
U4.Accusing an NPC of murder	If the player accuses the correct NPC of the murder, and has collected the necessary evidence, they win the game. However, if the player accuses the wrong character, or doesn’t have enough evidence to support their claim, the accusation is a failure. Any character who has been wrongly accused by the player becomes temporarily unavailable for questioning.	When the player chooses to accuse an NPC of murder, they will have to decide which of the clues they have collected support their claim.
U5.Questioning NPCs	The user will be able to ‘question’ an NPC in order to gather clues (which could help them to solve the murder mystery). When questioning a character, the user will have three different modes of questioning available to them. The modes of questioning available will depend on which ‘great detective’ character the user chose at the beginning of the game.	
U6.Using the map	The player will be able to view a map of the game world, showing the room layout and the locations of items and characters.	The player can view the map at any time.

Functional System Requirements

Name	Requirement	Assumptions
F1.World generation	The system will generate a new game world each time the game is played. The system must populate eight rooms with ten non-player characters. Each room must also contain at least one 'clue', which the player might find useful whilst trying to solve the mystery.	All clues, characters and rooms are pre-set (the game constructs a new game world using them at runtime).
F2. Mystery Generation	Each playthrough will provide a new murder mystery for the player to solve. The murderer, murder weapon and motive will be different each time.	The mystery will be constructed using predetermined motives, weapons and characters.
F3.GUI Interaction	The system must display objects and game characters, and provide a user interface which the player can use to interact with them.	The GUI will be simple enough for the player to understand. Assuming the player is a 2nd year computer science student.
F4.Unlocking characters	The system shall provide a method for 'unlocking' an NPC the player has falsely accused of murder. This will involve the player speaking to another character, or speaking to the NPC again whilst carrying a specific item in their inventory.	
F5.NPC questioning	The NPC characters will respond differently, and may provide different information, depending on the mode of questioning the player chooses.	Players can questions NPCs, in order to gather information that might help them to solve the murder mystery (User Requirement 5).
F6.Item Clues	Each newly generated game world will contain a number of item clues for the user to find. Not all clues should be available to the player at the start of the game. Clues which are initially 'hidden' should become visible to the player after they have taken specific actions, such as talking to characters, or finding other clues.	Not all item clues will be relevant, but there will always be enough relevant item clues for the player to solve the murder mystery. The item clues present in each game will be drawn from a pool of pre-set item clues.
F7. Verbal Clues	Verbal clues are pieces of information provided by NPCs to the player.	Not all verbal clues will be relevant. Some of the verbal clues in each game will have been modified to include details of that playthrough's specific murder mystery.
F8.Inventory and Logbook	The system shall allow the user to collect items and store them in an inventory that can be accessed at any time. This will allow the user to keep track of what they have picked up. The system shall also record any 'verbal' clues the player has gathered by questioning NPCs in a logbook. The inventory and logbook will be displayed alongside one another in the GUI.	

F9.Score tracking	The system will award the user a score at the end of the game based on their performance. The user's score will be based on: time taken to accuse the correct character, number of incorrect accusations, and number of clues found.	
F10.Save system	The system should provide a method for the user to save their game, allowing players to leave an unfinished game and return to it at a later date.	

Non-Functional System Requirements

Name	Requirement	Assumptions
N1.Graphical Representation	There will be a graphical user interface (GUI), which will allow the user to interact with objects and characters within the game world. This GUI shall provide a representation of the fictionalised Ron Cooke Hub (RCH), which features eight distinct rooms with varying dimensions.	The GUI can display several different rooms, each with differing dimensions and a distinct visual style. (This may include different furniture, for example).
N2.Non Player Characters	There will be ten non player characters (NPCs), each with a different personality and appearance. This does not include the three 'great detective' characters.	The NPCs will be mostly the same in each playthrough of the game, but may have different verbal clues they can provide the player.
N3.Main Menu	When the game is launched, a main menu shall be displayed. The menu will present the user with two options: Start and Exit. The Start button will begin the game. The Exit button will close the game.	The player can return to the main menu at any time.
N4.Map	The system will provide the user with a map showing the layout of the game world, as well as which NPCs and clues are located in each room.	

References

[1] I.Sommerville, Software Engineering Tenth Edition, Edinburgh: Pearson, 2015

[2] Project Website, downloads page: http://www-users.york.ac.uk/~phj501/downloads.html

[3] Dr T. Kelly, "York VLE SEPR Lectures," 2015-16. [Online]. Available: https://vle.york.ac.uk/bbcswebdav/courses/Y2016-006404/2015-16/Lectures/SEPR-Lecture-RequirementsEng.pdf. [Accessed 3 October 2016].