**Requirements**

Group 28

Piazza Panic

By OuseWorks

Ben Harris

Joshua Gill

Niamh Hanratty

Amy Raymond

Matthew Czyzewski

Matt Rohatynskyj

**Part A**

**Requirements Elicitation**

In order to elicit requirements from our client, we arranged an in-person interview to discuss the product brief. Before, we went through the product brief to brainstorm queries we had about the client’s expectations. This was early in the project to ensure our ideas for the project aligned with the client’s. We decided a face-to-face interview would help us best elicit requirements[1] as non-verbal cues, such as body language, are useful to understand the client’s expectations[2].

During the meeting, the client summarised the context of the product. This was used to create the single statement of need (SSON), described as a broad understanding of the overall goal of the system and its capabilities [3]. Written notes were made throughout the meeting by multiple team members to ensure any important details were not missed as they are “unobtrusive and flexible”[4]. [4] suggests recording the client with a camera may distract them from the meeting and make them act unnaturally, which we wanted to avoid. The ethical implications (such as obtaining informed consent and securely storing collected data) of videoing our client also made recording notes the more suitable option. These notes were analysed to specify user requirements, which are statements regarding the tasks that users should be able to carry out using the system. Once we established the user requirements, we created detailed descriptions of the system’s functionality, services and constraints.

**Requirements Presentation**

These user and software requirements need to be documented in a way which helps us to ensure that the final product is of high quality compared to the product brief. In order to format our requirements in a professional and beneficial way, research into software requirements specification was employed. [5] describes methods for presenting requirements through text, use cases, prototypes, models/diagrams, user stories and work-breakdown structure. Furthermore, [6] describes good practices to follow when outlining requirements. For example the need to “write succinctly” omitting technical jargon to avoid misinterpretation, and to create functional requirements which are easy to test and verify that they have been met. [5] states that the correct documentation of requirements helps to avoid project failure, ensure the system is functionally what the stakeholders anticipate the product to be, and can even lead to further requirements which may not have been thought about before. These qualities aligned with what we need to achieve success in our project, so we decided the best layout was to split the requirements into 3 tables of textual descriptions - this made it possible to describe requirements in a clear and concise way which omits any useless information which may occur when using methods such as user stories or use cases. In order to be referenced quickly and effectively we have provided each requirement with a unique ID so in different contexts, when requirements are referred to, we are able to understand the premise without checking the documentation.

The User Requirements table contains statements about what the user should be able to do using the system. Alongside these descriptions, a priority rating is given to show the need for each specific requirement to be implemented in the product. Therefore some user requirements are more essential to the user than others.

The Functional Requirements table contains descriptions of how the system should behave in order to provide the user requirements given in the table above it. Each functional requirement has a reference to a specific user requirement, this allows us to be able to verify whether a user requirement has been met by considering the functional requirements which are linked to it.

The Non-Functional Requirements table contains descriptions of the qualities the system should have which act as restraints on the performance of functional requirements. Each NFR has a fit criteria which is a statistical measure of how well the system should perform which should be easy to test and verify whether they are met.

**Part B**

**SSON - The system shall provide a fun-to-play game which requires little to no previous gaming experience to encourage visitors to engage with the department on a University open day.**

| User Requirements | | |
| --- | --- | --- |
| ID | Description | Priority |
| UR\_CONTROL\_COOK | The system shall allow the user to control 2 cooks to complete the objective of the game and move the cooks by interacting with the game environment. | Shall |
| UR\_SCENARIO\_GAMEMODE | The system shall provide the user a single player scenario based game mode, serving 5 customers sequentially at the counter. | Shall |
| UR\_UX | The system should provide an enjoyable user experience | Should |
| UR\_PLATFORMS | The system shall be playable on different platforms(Windows, Linux, Mac). | Shall |
| UR\_INTERFACE | The system should provide a graphical interface to allow the user to navigate between gameplay, settings or the help page. | Should |

| Functional Requirements | | |
| --- | --- | --- |
| ID | Description | User Requirements |
| FR\_ARRIVE\_ALONE | Customers will arrive at the counter in groups of 1. | UR\_SCENARIO\_GAMEMODE |
| FR\_INFINITE\_WAIT | Customers will wait at the counter indefinitely. | UR\_SCENARIO\_GAMEMODE |
| FR\_SERVE\_5 | There will be default 5 customers to serve. | UR\_SCENARIO\_GAMEMODE |
| FR\_COMPLETE\_GAMEMODE | The system should inform the user how long it took them to complete the game mode. | UR\_SCENARIO\_GAMEMODE |
| FR\_RECIPE | Recipes (Burger + Salad) will be displayed on screen. | UR\_CONTROL\_COOK |
| FR\_MULTIPLE\_COOKS | System will provide multiple cooks which the user controls | UR\_CONTROL\_COOK |
| FR\_COOKING\_STATIONS | There should be cooking stations for cutting, baking, frying, and serving. | UR\_CONTROL\_COOK |
| FR\_INGREDIENT\_STATION | Ingredients station that users can interact with to do food preparation. | UR\_CONTROL\_COOK |
| FR\_SERVE\_COUNTER | There should be a counter where customers wait to be served. | UR\_CONTROL \_COOK |
| FR\_STANDBY | The main page will appear after the system is left idle. | UR\_INTERFACE |
| FR\_TIMER | The system should tell the user how long it took them | UR\_UX |
| FR\_HELP | There will be a help page the user can access, which includes a game tutorial. | UR\_UX |
| FR\_MUTE | The user will be able to mute the game. | UR\_UX |
| FR\_MAIN\_PAGE | There will be a home page at the beginning of the game. | UR\_INTERFACE |
| FR\_CONTROLS | Keyboard and mouse will be used to play the game. | UR\_UX |
| FR\_COLOUR\_SETTINGS | The game will be accessible to users’ with impaired vision. | UR\_UX |
| FR\_HD | The game will be seen from further away by spectators of the gameplay. | UR\_UX |
| FR\_VOL | The user can change the volume to their preferred level. | UR\_UX |
| FR\_ REPUTATION | The system shall implement a reputation points counter | UR\_SCENARIO\_GAMEMODE |

| Non-Functional Requirements | | | |
| --- | --- | --- | --- |
| ID | Description | User requirements | Fit Criteria |
| NFR\_AVAILABILITY | The game shall be highly available. | UR\_UX | The game shall be available to play 99% of the time. |
| NFR\_RESPONSE\_TIME | The game should not take too long to update the state of the environment onto the screen. | UR\_UX | The game will take 15-45ms to respond to user input. |
| NFR\_IDLE | The game will return to the main page after no response. | UR\_INTERFACE | Game will return to the main page after the system is left idle for 1 minute. |
| NFR\_TUTORIAL | The game will have a detailed tutorial | UR\_UX | 100% of the game’s functions will be described on this page. |
| NFR\_UNEXPECTED\_INPUTS | Unexpected inputs will not have an impact on the rest of the system | UR\_UX | 0% of unexpected inputs will have an effect on the gameplay. |
| NFR\_EASE\_OF\_PLAY | The system should be operable by people with little to no gaming experience. | UR\_UX | 90% of people should be able to play this game. |
| NFR\_VISION\_ACCESSIBLITY | The system should be accessible to people with impaired vision. | UR\_UX | 100% of people with colorblindness should be able to play the game. |
| NFR\_USABILITY | All messages for the customers will be in plain English, with no technical jargon. | UR\_UX | 90% of English speakers will understand user messages |

References:

[1] - I. Sommerville, Software Engineering. Pearson Education; 2015, page 115

[2] - K.Peleckis, V.Peleckiene, K.Peleckis, “Nonverbal Communication in Business Negotiations and Business Meetings” <https://www.learntechlib.org/p/176672/>, <https://www.scipress.com/ILSHS.62.62.pdf> (accessed 16/12/22)

[3] - Requirements Engineering lecture by Dimitris Kolovos

[4] - Y Rogers, J Preece, H Sharp, Interaction Design : Beyond Human-Computer Interaction . 5th edition. Wiley; 2019, page 267-268

[5] - P. Gorbachenko, Enkonix, enkonix.com “What are Functional and Non-Functional Requirements and How to Document These” - <https://enkonix.com/blog/functional-requirements-vs-non-functional/> (accessed 16/12/22)

[6] - A. Mateja, Studio Software, “Functional Requirements in Software Engineering: Top Tips for Writing a Clear Document” - <https://studiosoftware.com/blog/functional-requirements-in-software-engineering-how-to-improve-project-performance/#How_to_write_functional_requirements>