Requirements

ENG1 Team 7
Broken Designers

Adam Brown
Morgan Francis
Oliver Johnstone
Shabari Jagadeeswaran
Laura Mata Le Bot
Rebecca Stone

Introduction

The purpose of the requirements statement is to unambiguously define the user and system requirements for the game. The system requirements divide into functional and non-functional requirements. The document will be used by the client and software developers, and is written considering the different knowledge of these stakeholders.

We were provided with a product brief from which many of the functional requirements could be drawn without the need for more information. However, there were some instances where the requirements were ambiguous or not included in the brief and needed to be established. We chose to use a hybrid open and closed interview format. This allowed us to have an agenda and cover all bases, while also encouraging discussion of requirements we may not have considered.

The interview began with some questions about non-functional requirements: the controls, visual display, and audio. Next, we used a scenario to help the client visualise our ideas and give more concrete and detailed requirements. **[1]** We then asked questions about the non-functional requirements; including documentation, user profiles, the availability of the game, and accessibility. The interview ended with a discussion around prioritisation and negotiation of the requirements we had established. We discussed how often the client wanted updates, and what their main priorities were. For example, the client asked that our first priority should be to create a 'vertical slice' of the game, meaning that there would first exist a very basic game with a very simple recipe that can be downloaded and playable. The client suggested that additional features should be added from there.

Having elicited the client's requirements for the game, we were ready to create a formal requirements statement. We presented the requirements in a table as demonstrated in the lecture, which follows the IEEE Standard Requirements Engineering document. **[2]** The idea is to create a brief overview table of the user's requirements. Each of the user requirements is then divided into detailed functional and non-functional requirements in separate tables, using IDs to refer to the user requirements.

Requirements should be written in clear, concise and plain language. The focus of requirements should not be to provide any insight into the implementation of the system, but to provide comprehensive information about what the system should do. Using consistent vocabulary removes ambiguity; for example, 'shall', 'should', and 'may' are used to describe the priority of the requirements.

Having transformed the interview material into a formal requirements specification, we sent the document back to the client for review and further negotiation. The client expressed that they were happy with the majority of the requirements, and gave some small suggestions for improvements. The client suggested moving the previously named NFR_TUTORIAL and NFR_LEADERBOARD away from the non-functional requirements table to the functional one, where they became FR_TUTORIAL and FR_LEADERBOARD. The client also clarified that the NFR_PLATFORMS requirement should state that the game shall have a clean installation process for downloading and running the game on both Windows and MacOS.

Following this review, we used the requirements statement to design the initial architecture of the system. The requirements did not evolve very much throughout the project, as they are designed as a fixed guide. Instead, the architecture and implementation evolve to better meet the established requirements.

Bibliography

https://ajbrown-york.github.io/html/bibliography.html

Requirements statement

Single Statement of Need

The objective of this desktop game will be for the user to prepare and serve the orders of five customers arriving at the restaurant.

User requirements

ID	Description	Priority
UR_UX	The user should have a relaxed and enjoyable experience.	Should
UR_DEVICE	The game shall run and display on Desktop.	Shall
UR_MENU	The user should access a main menu with choices to start a new game, view a 'How to Play' tutorial, or view the leaderboard.	Should
UR_CHEF	The user shall manage and control the movement and interactions of two chefs by switching control between them.	Shall
UR_ITEMS	The user shall interact with items. Each chef shall carry a stack of up to three items.	Shall
UR_STATIONS	The user shall direct the chefs to the ingredient stations, cooking stations and the serving station.	Shall
UR_GAMEPLAY	The user shall use the chefs, items, and stations to follow the recipes provided and fulfil customer orders.	Shall

System requirements

Functional requirements

ID	Description	User requirement
FR_NEW_GAME	The 'New Game' button on the main menu shall create a new instance of the game.	UR_MENU
FR_TUTORIAL	The 'How to Play' button shall take the user to a tutorial.	UR_MENU
FR _LEADERBOARD	The 'Leaderboard' button shall display the top ten times for the day alongside each player's initials.	UR_MENU
FR_SETTING	The system shall display an overhead view of the kitchen. The cooking stations shall be inside the kitchen, with the serving station along one edge. The pantry shall be a room which the user can visit by moving the chefs into the room.	UR_DEVICE
FR_CHEF_INFO	The system shall store information about each chef including their ID, whether they are currently selected, and their hand class (see FR_ITEM_STACK).	UR_CHEF
FR_CHEF _SWITCHING	The system shall switch control between chefs when the user presses the 'TAB' or corresponding '1', '2', '3' keys for each chef by updating the information stored regarding which chef is currently being controlled.	UR_CHEF

FR_CHEF _MOVEMENT	The system shall move the chef horizontally and vertically across the display when the ‘W’, ‘A’, ‘S’, and ‘D’ keys are pressed indicating up, left, down, and right respectively.	UR_CHEF
FR_CHEF _COLLISIONS	When the user attempts to move towards an object at a shorter distance than the distance moved per key press, the control will be blocked and the chef shall be displayed to be touching the object.	UR_CHEF
FR_ITEM _REGISTER	The system shall keep an item register storing a string ID for each item and linking it to the PNG used in the display.	UR_ITEMS
FR_ITEM_STACK	The system shall store information about the chef’s hand, a first-in-first-out stack with a maximum number of three items for the chef to carry.	UR_ITEMS
FR_ITEM_STACK _DISPLAY	The system shall display the chef’s stack contents to the user.	UR_ITEMS
FR_ITEM_STACK _CHANGES	The ‘UP’ and ‘DOWN’ keys shall allow a chef to add and remove items between the top of their stack and the current station or counter.	UR_ITEMS
FR_ITEM _INTERACTION	The ‘SPACE’ key shall be used for the interactions between chefs and items. The map shall include ‘interaction areas’, linking to different station classes. The interaction area that the chef is in shall determine the interaction (chop, flip, bake, assemble).	UR_ITEMS
FR_ITEM _INTERACTION _EFFECTS	The system shall use audio and visual effects (such as sizzling) to indicate interactions with items.	UR_ITEMS
FR_ITEM _CHANGES	The system shall update items according to the interaction (eg. onion to chopped onion).	UR_ITEMS
FR_DISPENSER	The system shall have a pantry containing ingredient stations from which the player can collect ingredients onto the chef’s stack.	UR_STATIONS
FR_INGREDIENT _INFO	The system shall store information for each ingredient station about which item the station stocks and whether a chef is at the station.	UR_STATIONS
FR__COOKING _STATIONS	The system shall have cooking stations with different purposes (chopping, frying, baking, assembling).	UR_STATIONS
FR_STATION _INFO	The system shall store information for each cooking station about the accepted items and interactions that can happen.	UR_STATIONS
FR_SERVING _STATIONS	The system shall have a serving station where customers queue to order items on the menu and be served.	UR_STATIONS
FR_SERVING _ARRIVALS	The system shall display five customers arriving at the serving station at fixed time intervals.	UR_STATIONS
FR_SERVING _ORDERS	The system shall randomly assign an order from the menu to each customer and display these to the customer by a speech bubble on arrival.	UR_STATIONS

FR_ASSEMBLY _STATION	The system shall determine if a recipe has been followed correctly and create the final meal at the assembly station.	UR _GAMEPLAY
FR_ERROR _DISPLAY	The system shall display an error message to the user if the order has been made wrong.	UR _GAMEPLAY
FR_SERVE _ORDER	Fulfilled orders brought to the serving station shall be taken or rejected by the corresponding customer.	UR _GAMEPLAY
FR_TIMER	The system shall display the time since the scenario was started on the screen at all times.	UR _GAMEPLAY
FR_FINISH	The user cannot lose in this version of the game. The game shall finish when the user has successfully served all five customers. Success shall be measured by the time taken to serve all customers.	UR _GAMEPLAY

Non-functional requirements

ID	**Description**	**User requirement**	**Fit criteria**
NFR _DOCUMENTATION	The code shall include technical documentation to facilitate maintenance by future developers.	UR_UX	All classes include a JavaDoc.
NFR_PLAYER _ABILITY	The user should complete the game in the first attempt without too much difficulty while still being able to improve their skills to drastically reduce the time taken to complete a scenario.	UR_UX	90% of users should complete the scenario within 10 minutes on their first attempt.
NFR_COLOUR	The game colours shall be colour-blind-friendly.	UR_UX	A colour-blind accessible palette will be used.
NFR_DEVICE	The game shall run all day on a single machine with no memory leaks.	UR_DEVICE	The game will be available for up to 8 hours at a time.
NFR_PLATFORMS	The game shall be installed and run on both Windows and MacOS.	UR_DEVICE	