2a) Requirements Introduction

In order to begin eliciting requirements the team went over the Piazza Panic Product Brief together to make sure everyone had an understanding of what the basic Single Statement of Need of the game was and what needed to be included/done. After that, an interview was scheduled with the client where we asked more specialised questions including “what kind of accessibility features need to be included?” and “would there be any specific map layout requirements?”, etc.

The team then extracted appropriate User Requirements from the bullet points in the Product Brief and answers provided by the client. After deciding on what the User Requirements should be, we then evaluated how vital to the game each requirement would be and gave each a priority rating ranging from “May”, “Should” and “Shall” corresponding to how important that requirement would be for the game to operate appropriately.

Following that the team brainstormed possible tailored System Requirements for each User Requirement. For which, the process for each was carefully thought out and any extra System Requirements were either removed, or deserved to be moved into their own User Requirements category.

Once all the System Requirements were negotiated, the team used a table to split every System Requirement into two categories; functional and non-functional requirements. This was done by analysing each requirement on whether it defines what the game must do ((Functional) including transformations, invariants or failures) or if it instead describes general, measurable properties of the system (Non-functional) such as security and timing. The team then decided which category each would fall under.

Now that all the requirements have been set and separated into their appropriate groups, a sustainable and quantifiable fit criteria was further brainstormed for each non-functional requirement.

One of the team members (along with a shadow) was tasked with the creation of the deliverable, now that all the information was gathered. The requirements were split into the 3 appropriate tables as required; First a table for the User Requirements with a unique and meaningful User Requirement ID, an appropriate description and a priority value; Followed by a Functional Requirements table with a Functional Requirement ID, description and the User Requirement ID it links back to; The final table is the Non-functional Requirements table which consists of the fields Non-functional Requirement ID which again has to be unique and meaningful, a description of the requirement, the User Requirement ID it links back to and and extra column for the specific fit criterion it needs to achieve.

Towards the end of the project, once the game was operational, small quality-of-life tweaks were made to some requirements, such as editing some fit criteria for some Non-functional Requirements, for example “cook speed” and “interactable stations range”. While also staying true to the product brief, these also improved how the game plays and the functionality of it. Finally, during the last meeting, requirements were checked one last time, having every member concur, to produce the final document.

2b)

User requirements:

ID	Description	Priority
UR_COOKS	The game should have two playable cooks to move and interact with that the user can switch between	Shall
UR_SCENARIO_BASED_MODE	The game should have 5 customers within a scenario-based mode that will arrive and place an order	Shall
UR_RECIPES	The game should have two recipes: salad and burger	Shall
UR_COOKING_STATIONS	The game should have interactable cooking stations	Shall
UR_REPUTATION_POINTS	The user has reputation points	Shall
UR_PANTRY	There should be pantry with the ingredients needed	Shall
UR_TIME_AT_END	The game should display the time taken at the end	Shall
UR_ACCESSIBILITY	The game should be widely accessible	Should
UR_TARGET_AUDIENCE	The game should be suitable for its target audience	Should
UR_LEARNABILITY	The user should be able to learn to play	Shall
UR_MAIN_MENU	The game should have a main menu	Shall
UR_PAUSE	The game should be pausable	Should

Functional requirements:

ID	Description	User Requirements
FR_COOK_SWITCH	The game should provide the ability to switch between cooks	UR_COOKS
FR_COOK_MOVE	The game should provide the ability to move the cook	UR_COOKS
FR_COOK_INTERACT	The cook should be able to	UR_COOKS

	interact with what's in front of it	
FR_INGREDIENT_STACK	The cook should carry ingredients in a stack	UR_COOKS
FR_ORDER_OPTIONS	Customers have an option of ordering a salad or a burger. They are generated at random	UR_SCENARIO_BASED_MODE
FR_TIMER	Customers' orders will have a timer	UR_SCENARIO_BASED_MODE
FR_LEAVE	Customers will leave when served	UR_SCENARIO_BASED_MODE
FR_RECIPES	Recipes must exist for the burger and salad	UR_RECIPES
FR_COOKING_STATIONS	The cooks can approach and interact with a cooking station	UR_COOKING_STATIONS
FR_CHOICES_OF_COOKING_STATION	Chopping board, frying pan, plates	UR_COOKING_STATIONS
FR_START_REPUTATION_POINTS	The user starts with three reputation points	UR_REPUTATION_POINTS
FR_LOSING_REPUTATION_POINTS	The user loses a reputation point when not served in time	UR_REPUTATION_POINTS
FR_ZERO_REPUATION POINTS	When zero reputation points is reached, the user has lost and the game ends	UR_REPUTATION_POINTS

Non-Functional Requirements:

ID	**Description**	**User requirements**	**Fit criteria**
NFR_MAX_INGREDIENTS_IN_STACK	There is a reasonable number of ingredients in the stack	UR_COOKS	3 ingredients max
NFR_MOVEMENT_REQUIREMENTS	The user will be able to move the current cook at a reasonable constant speed in a controllable number of directions	UR_COOKS	Moves at a maximum speed of 3 units per second. The user will be able to move in 8 directions.
NFR_MOVING_COOKS	The user can only move one cook at once	UR_COOKS	The user is limited to control one cook at a time
NFR_CUSTOMERS_ARRVING	Customers arrive one a time, at different times	UR_SCENARIO_BASED_MODE	A customer should arrive every minute.

NFR_CUSTOMERS_WAITING	Customers wait an appropriate amount of time	UR_SCENARIO_BASED_MODE	Customers will wait indefinitely
NFR_MAX_CUSTOMERS_WAITING	A reasonable amount of customers wait at a time	UR_SCENARIO_BASED_MODE	This will be a maximum of three
NFR_SALAD	The recipe for the salad will be relatively easy to follow	UR_RECIPES	Salad: cut lettuce, cut tomatoes, cut onions, serve together.
NFR_BURGER	The recipe for the burger will be relatively easy to follow	UR_RECIPES	Burger: form patty, fry patty, toast buns, serve together.
NFR_RECIPE_VARIATIONS	The customers may have a variation of the recipe	UR_RECIPES	This could be anywhere between a single ingredient and the recipe missing one ingredient
NFR_RANGE_OF_COOKING_STATION	The cook needs to be within a suitable range of the station	UR_COOKING_STATIONS	The cook should only be able to interact with stations within a 1 tile range.
NFR_REACTIONS_TO_INTERACTION	A reaction from the interaction is complete in a reasonable time	UR_COOKING_STATIONS	Interaction input from the player leads to interaction starting within 0.05 seconds.
NFR_RUNNING_OUT_OF_INGREDIENTS	Ingredients never run out	UR_PANTRY	The user will have access to an unlimited amount of ingredients
NFR_DISTANCE_TO_PANTRY	The cook has to be within a reasonable distance to interact with the stations in the pantry	UR_PANTRY	The cook should only be able to interact with pantry boxes within a 1 tile range.
NFR_REACTION_FROM_PANTRY	The reaction from the pantry should be within sufficient time	UR_PANTRY	Interaction input from the player leads to interaction starting within 0.05 seconds.
NFR_TIME_TAKEN	The measurement of time taken will be accurate to how long they took	UR_TIME_AT_END	This will be taken from the time difference between them pressing play and when they served the last customer

(Due to page limit, the rest of the Functional and Non-Functional requirements are included on the website: https://eng-25.github.io/#requirements_table)