2. Requirements.

Group Number: **Team 20**

Group Name: **Gourdo Ramsay**

Group Members:

Lauren Waine
Megan Bishop
Tommy George
Bartek Grudzinski
Davron Imamov
Nathan Sweeney

Introduction:

In order to ascertain our requirements, we initially looked at the project brief and wrote a first draft of the user requirements and constraints based on its specifications. To help with this, we also referenced similar games that we had already played [1] to get an idea of how the game would function.

With this, we arranged a customer meeting with our client and discussed what their requirements were and whether they matched up to our original assumptions. In this meeting, our single statement of need (SSON) was for students, the intended users, to play with the game and see what they could achieve on open day. The game should be playable with little experience, be as simple as possible and like an arcade game to play. To achieve this, UR_STANDBY_MODE was added as a demonstration to show the student how to play the game and UR_ARCADE_GAME is meant to guide the developers so the game has an arcade machine aesthetic.

Some of our initial requirements also had to be changed, to cater to the client's preferences. Our original idea of the game had text to explain the recipes but the client preferred for us to use images over text to make the game shorter and simpler to understand, UR_IMAGES reflects this. We also had the idea that if the player failed to take the food off the baking station it would eventually catch fire (and possibly burn down the kitchen ending the game) but the client didn't want any negative connotations to the game so this was removed and UR_NO_NEGATIVES was also added. As the game continues to develop, the requirements evolve as there are often differences between planning and making the game. This required communication between each member of the team so every relevant section could be updated.

We decided to use a tabular format when presenting our requirements as it makes it simpler to read and understand the different requirements. User requirements in particular must be written for a non-technical audience so that all of the stakeholders can understand it, in case they wish to make any changes [2]. Some user requirements aren't vital to the project or are subjective. As such, a priority is given to each requirement (Shall/Should/May) so the developers know what to focus on to ensure the game's success. Each requirement should also have a unique ID, with a meaningful name, so it can be easily referenced.

The functional requirements address how to satisfy the user requirements therefore we referenced the user requirements that they satisfy, to ensure that they are relevant. They describe with more technical language but don't use specific technology as this could change throughout the project [2]. Non-functional requirements create standards for how the game must operate. This standard is specified through the fit criteria, which will give a threshold to demonstrate that the description of the requirement has been met [2]. Each non-functional requirement refers to one or more user requirements, which are linked to each other using the requirement's ID. Constraints are listed separately as they affect the entire system compared to the specific user requirements of non-functional requirements.

Some requirements may conflict with each other, such as UR_LOSE_GAME may conflict with UR_NO_NEGATIVES. A compromise must be reached as the player must be able to lose the game but without it being a negative experience.

User Requirements:

ID	Description	Priority
UR_CONTROL_COOKS	Move cooks around the kitchen and switch between them.	Shall
UR_INTERACT_FOOD	Cook can interact with what's in front of them.	Shall
UR_CUSTOMER_WANTS	Customers ask for a recipe and have a time limit for it to be done.	Shall
UR_CUSTOMER_LEAVES	Customers leave once they have their food or their timer runs out.	Shall
UR_SCENARIO_MODE	A certain number of customers need to be served for the game to end.	Shall
UR_STANDBY_MODE	The game plays itself as a demonstration for the player.	Should
UR_SWITCH_MODE	Can switch between modes.	Shall
UR_FOOD_PREP	Must manually prepare the food and restart if they fail.	Shall
UR_RECIPE	Multi-step recipes, at different stations for at least: salad, burger.	Shall
UR_COOKING_STATIONS	Perform actions on the ingredients, each at a different station: cutting, baking, frying, and serving.	Shall
UR_INGREDIENT_STATIONS	The pantry will store the ingredients at various ingredient stations.	Shall
UR_REPUTATION	Start with 3 reputation points, lose a point when a customer isn't served within their time limit.	Shall
UR_LOSE_GAME	If your reputation points reduce to zero, you lose the game.	Shall
UR_WIN_GAME	You win if the last customer leaves and your reputation points are above zero.	Shall
UR_ENDLESS_SCORE	In endless mode, the amount of customers served before losing will be tracked.	Shall
UR_CUSTOMER_ARRIVAL	Customers will arrive by themselves, then in 2's or 3's at different times and wait by the counter.	Shall
UR_EARN_MONEY	Collect money every time a customer is successfully served.	Shall
UR_LEADERBOARD	Have a leaderboard of top scores.	May
UR_ARCADE_GAME	Should feel like an arcade game, though the colour palette and images used.	Should
UR_NO_NEGATIVES	No negative connotations or violence.	Shall
UR_IMAGES	Prioritise using images over text in the game.	Should

Functional Requirements:

ID	Description	User Requirements
FR_MOVE_COOKS	The system shall provide input keys to move the cooks	UR_CONTROL_COOKS
FR_SWITCH_COOKS	The system shall provide input keys to	UR_CONTROL_COOKS

	switch which cook is being controlled	
FR_MOVE_CONTROLLED_COOK	The system shall only move the cook currently being controlled	UR_CONTROL_COOKS
FR_INTERACT	The system shall provide input keys to interact with what is in front of them	UR_INTERACT_FOOD
FR_INTERACT_NOTIFY	The system should notify the user that they can interact with an item and which item they are interacting with	UR_INTERACT_FOOD
FR_RANDOM_CUSTOMER	The system shall randomly send customers at different intervals	UR_CUSTOMER_ARRIVAL
FR_CUSTOMER_WANTS	The system shall assign one of the available recipes to the customer	UR_CUSTOMER_WANTS
FR_CUSTOMER_SERVED	The customer shall leave after being served	UR_CUSTOMER_LEAVES
FR_SCENARIO_MODE	The system shall send a fixed number of customers to be served	UR_SCENARIO_MODE
FR_SCENARIO_WIN	The system shall award the player with a win if there are no customers left and rep points > 0	UR_SCENARIO_MODE UR_WIN_GAME
FR_FOOD_PREP	The system shall require that different ingredients have their own prep stations and must be prepped manually	UR_FOOD_PREP
FR_RECIPE_SALAD	The system shall require that a salad is made through multiple steps	UR_RECIPE
FR_RECIPE_BURGER	The system shall require that a burger is made through multiple steps	UR_RECIPE
FR_COOKING_STATIONS	The system shall provide a cooking station for cutting, baking, frying, serving	UR_COOKING_STATIONS
FR_INGREDIENT_STATIONS	The system shall provide ingredient stations for all the available ingredients inside the pantry – The ingredients will never run out	UR_INGREDIENT_STATIONS
FR_COOK_STACK	The system shall allow the user to put an item on top of their stack and place the top item of their stack down	UR_INTERACT_FOOD
FR_EARN_MONEY	The system shall allocate the user with money after a customer has been successfully served	UR_EARN_MONEY

Non-Functional Requirements:

ID	Description	User Requirements	Fit Criteria
NFR_SWITCH_COOKS	The system will be	UR_SWITCH_COO	Less than 1 second delay before

	responsive	KS	switching between cooks
NFR_MOVE_COOKS	The system will be precise	UR_MOVE_COOKS	The movement of the cooks is easy to control through a valid movement system
NFR_RANDOM_CUSTOMER	The system will be solvable	UR_RANDOM_CUSTOMER	The algorithm to generate customers will accommodate the time taken to prepare and serve the dishes the customers request
NFR_FOOD_PREP	The system will be accompanied by documentation	UR_FOOD_PREP	The player will have the ability to open a recipe book
NFR_GAME_OUTCOME	The system will be auditable	UR_LOSE_GAME UR_WIN_GAME	The system shall maintain a durable record of all events related to the reputation and customer system
NFR_UNLIMITED_INGREDIENTS	The system will be reliable	UR_UNLIMITED_INGREDIENTS	The system will always be available, never running out of ingredients
NFR_RECIPE	The system shall be usable	UR_RECIPE	The system will use icons or plain English avoiding technical jargon
NFR_CUSTOMER_LEAVES	The system shall be precise	UR_CUSTOMER_LEAVES	The system shall not fall out sync of in-game ticks by more than 1%, ensuring the system is representative of reality
NFR_SCENARIO_MODE	The system shall be maintainable	UR_SCENARIO_MODE	Support engineers shall be able to diagnose and repair errors in the system through logs of the game
NFR_ADD_INGREDIENT	The system shall be resilient	UR_INTERACT_FOOD	Adding an ingredient to the top of the stack will not impact the other ingredients in the stack

Constraint Requirements:

ID	Description
CON_PROGRAMMING_LANGUAGE	The system must be written in Java (version 11).
CON_OS	The system must be able to run on all desktop operating systems: Linux, Windows and MacOS.
CON_SYS_REQUIREMENTS	The game must be able to run on any modern desktop computer or laptop.
CON_RESOLUTION	The game must be able to adapt to any screen size

References:

[1] team17. Overcooked!, team17.com. [Online]. Available: Overcooked | Cooking Video Game | Team17 [Accessed: 17 January 2023].
[2] Visure Solutions. What is Requirements Specification: Definition, Best Tools & Techniques | Guide, visuresolutions.com. [Online]. Available: What is Requirements Specification: Definition, Best Tools & Techniques | Guide - Visure Solutions [Accessed: 17 January 2023].