UNIVERSITY OF YORK

DEPARTMENT OF COMPUTER SCIENCE

ENG1
Group 9 - Kitchen Tossups

Report - Requirements

Group Members:

Liam Burnand

Jack Cameron

Cameron Fox

Adam Penny

Eoin O'Connor

Ben Brown

Before starting implementation, as a team we first read through the product brief to familiarise ourselves and ensure that we fully grasp what the stakeholders' product requirements are. From the product brief, we discussed and drew out a few basic user requirements that will be the foundation to help us form more requirements.

We then scheduled customer meetings to help clarify any questions and assumptions we had about the stakeholder's requirements and needs for the final product. This helped us confirm any vague requirements (such as what the limit is for the number of items a cook can hold) and allowed us to generate more requirements and also write our current requirements in more detail.

Our requirements consist of user requirements and system requirements. Beginning with user requirements we worked to create system requirements. Each system requirement satisfies or plays a part in satisfying a user requirement, occasionally with multiple system requirements satisfying one user requirement. System requirements are broken down into functional requirements (FRs) and non-functional requirements (NFRs). Each requirement has its own unique ID so we can easily refer to the requirement in a different context as well as linking which system requirements fulfil which user requirements. User requirements are essential so we understand what and how the user will need to interact with the system. FRs are the actions a system must do in order to provide useful functionality for its user while the NFRs are qualities a system must have. NFRs constrain FRs and are therefore often critical to a system's success. Each NFR has a fit criteria so we can qualitatively or quantitatively measure the success of the requirement.

We formed our Single Statement of Need (SSON) so we have an overall goal of the system to work towards and know how it will accomplish this goal. Our SSON is "A single-player game which requires the player to manage cooks who prepare various dishes and serve them to customers coming into the Piazza restaurant."

After an initial write up of our requirements, we started implementing the game and checking off our requirements. In the customer meeting we presented a demo version of the game. During this meeting we received feedback about whether the user requirements we had attempted to implement were met. Following the customer meeting, we modified our requirements to more closely fit the customer's requests and edit other requirements that we had now received clarity around certain aspects of the proposed product that we were unsure about.

In the Brownfield phase of the development, more requirements were elicited by the brief and customer. We held a secondary requirements meeting with our customer to elicit further requirements whilst also further specifying the finer details of pre-elicited requirements. This led to an extension of each requirement section (UR, FR, NFR) that we received from our chosen project. In the functional requirements section, two requirements were added to the section in order to show the two recipes that are to be added to the game. From the new brief, we were told that power-ups were required for the project, and therefore were subsequently listed in the table below.

User Requirements Table

ID	Description	Priority
UR_SWITCH_COOKS	Player shall be able to switch between the 3 cooks.	Shall
UR_ENDLESS	Player will be able to select the endless mode scenario	Shall
UR_CONTROL_COOK	Player shall be able to CONTROL the cooks.	Shall
UR_SERVE_DISH	User should be able to serve the finished dish to the customer (as long as it is in the correct order) at a counter	Shall
UR_FUN_GAME	The game should be enjoyable to play	Should
UR_RESTAURANT_DESIGN	The design of the restaurant in the game should be equivalent to interior design of Piazza	Should
UR_SCENARIO_MODE	The player shall be able to select and play the “scenario” mode from the menu screen	Shall
UR_RECIPE_PREP	Player can prepare/assemble each recipe one step at a time at different “cooking stations”	Shall
UR_SERVE_CUSTOMER	Customers demand to be served	Shall
UR_SMOOTH_GAME	Player should be able to play the game smoothly and without lag	Shall
UR_INGREDIENTS	Player can obtain an amount of different ingredients from different “ingredient stations”	Shall
UR_REP_POINTS	There shall be reputation points in both modes	Shall
UR_DIFFICULTY	Player shall be able to choose a difficulty setting	Shall
UR_POWER_UPS	Player shall be able to collect various power-ups	Shall
UR_SAVE_GAME	Player shall be able to save the game	Shall
UR_NEW_STATIONS	Players shall be able to open new stations	Shall

System: Functional Requirements Table

ID	Description	User Requirements
FR_CONTROL_ONE_COOK	User can only control one cook at a time	UR_SWITCH_COOKS
FR_MOVE_COOKS	User can move cooks around play area	UR_CONTROL_COOKS

FR_LOSE_REP_POINT	If a customer has not been served within the time limit, the player loses a reputation point.	UR_SERVE_DISH
FR_CARRY	Chefs should only be able to carry one ingredients.	UR_RECIPE_PREP
FR_INTERACT	Player should be able to interact with what's in front of the cook (chop / flip / grab an ingredient or dish / place what they are carrying).	UR_RECIPE_PREP
FR_GAME_OVER	Game ends when the player loses all 3 reputation points	UR_SCENARIO_MODE
FR_GAME_WIN	Player wins if all customers have been served	UR_SCENARIO_MODE
FR_SERVE	Player serves a variable amount of customers customers depending on difficulty selected	UR_SCENARIO_MODE
FR_LIMITED_TIME	Customers will not wait indefinitely	UR_SCENARIO_MODE, UR_ENDLESS_MODE
FR_TIME_TAKEN	Game records how long the player took to complete the scenario	UR_SCENARIO_MODE
FR_CUSTOMER_ARRIVALS	Customers will arrive one by one, then later in groups of 1, 2 or 3.	UR_SERVE_CUSTOMERS
FR_PATTY_FLIP	Player can only flip patty if the bottom side is fully cooked	UR_RECIPE_PREP
FR_BURGER	A burger can only be served if it consists of: cooked patty and toasted buns assembled together	UR_SERVE_DISH
FR_SALAD	A salad can only be served if it consists of: lettuce, tomatoes and onions which have been cut and assembled together	UR_SERVE_DISH
FR_PIZZA	A pizza can only be served if it consist of: cheese, pizza sauce and pizza base, all to be cooked together before serving	UR_SERVE_DISH
FR_POTATO	A jacket potato can only be served if it consists of: a baked potato and either cheese, beans or cheese and beans or plain.	UR_SERVE_DISH
FR_STATIONS	Players should be able to earn a way of unlocking stations	UR_NEW_STATIONS
FR_PICKUP_POWER_UP	Players will be able to collect the power-ups that have spawned	UR_POWER_UPS

System: Non-Functional Requirements Table

ID	Description	User Requirements	Fit Criteria
NFR_SCALE	Game should scale with window size and keep aspect ratio	UR_FUN_GAME	Can run on 1920 x 1080, 1366 x 768
NFR_PLAYTIME	Game playtime should not be too short nor too long.	UR_FUN_GAME	Game should last <=5 minutes
NFR_OS	Game should be compatible on all OS's	UR_SMOOTH_GAME	Should run on Windows, macOS
NFR_MEMORY_USAGE	Game should not need to use much RAM to run	UR_SMOOTH_GAME	Should run on devices with 4GB ram
NFR_FAST_INTERACT	Grabbing an ingredient or dish/place what they are carrying should be fast	UR_INGREDIENTS	Interaction done after <2 seconds
NFR_DIFFICULTY	Player should be able to choose the difficulty of the game	UR_FUN_GAME	There should be a difficulty selection menu.
NFR_CONTROL_SCREEN	There should be an instruction/tutorial menu	UR_CONTROL_COOK	Clear and easy access menu screen