Requirements

Group 17

Team Loading

Joel Warren
Charlie Wilson
Jake Keast
Hari Bhandari
Ishrit Thakur
Joshua Hills

Introduction to requirements

Interview

We elicited the requirements through interviews with the customer, as the customer could present their thoughts and opinions in an unrestricted way. It also allowed us to understand the customer's needs for the system as well as the core qualities that it should have. We conducted the interview by asking a set of predefined questions. This ensured that the information gathered was clear, concise and easy to analyse. However, we also used open ended follow up questions which provided us with more detailed responses, which helped us to better understand their perspective. At the end of each question we asked the customer to rate the importance of each requirement that we identified. We did this so that we could allocate our time appropriately and it also helped us in negotiating conflicting requirements.

Brainstorming

As a group we analysed the information gathered such as the overall context and the target demographic. We then used this information in conjunction with the product brief to produce the user requirements [fig.1]. By identifying the target demographic we ensured that the needs of the users were being satisfied. We then developed the system requirements by going through each user requirement and brainstorming ideas about what the system needs to do, in order for that requirement to be met. This method was used so that the system requirements were traceable, which ensured that we met the needs of the stakeholders and prevented any gaps or inconsistencies.

Negotiation

During the process of creating system requirements [imozwastaken.github.io/requirements.html#functional , fig.2] we identified some conflicts, caused by the differing needs of the stakeholders. In order to resolve these conflicts we re-evaluated each requirement by considering the project constraints such as time, technology and personnel and we also consulted the risks involved with each requirement. We then analysed the importance of each requirement to the respective stakeholder and then prioritised the requirements accordingly. Through group discussion we then identified the lower priority requirement and suggested ways to alter it so we could accommodate both requirements. However, if that wasn't possible then the lower priority requirement was removed.

Presentation

The requirements were documented in 3 separate tables for the user requirements [imozwastaken.github.io/requirements.html#user , fig.1], the functional requirements [imozwastaken.github.io/requirements.html#functional , fig.2] and non-functional requirements [imozwastaken.github.io/requirements.html#nonfunctional, fig.3]. The user requirements were also organised by importance, which allowed us to appropriately allocate our time during the implementation. The functional requirements were organised by the user requirement they were traced from and we included an index that references the corresponding user requirement. This ensured that the implementation met the needs of the

stakeholders, as we implemented the system directly from the system requirements. For the non functional requirements we included a fit-criteria so that the requirements are verifiable.

Inheriting requirements

Team ownership of the project has since changed, with our team (Team 17) assuming responsibility. The requirements introduction above describes the elicitation and negotiation carried out by the previous team (Team 21).

Since assuming responsibility, new requirements for assessment 2 were brought forward by the customer. Therefore, the requirements needed to be extended. To do this, we first analysed the existing requirements tables, to examine any missing requirements in relation to existing and new customer requirements.

An interview was not conducted because the team did not feel it necessary in extending the requirements tables. However, the brainstorming and negotiation methods described in the introduction above were both used to generate new requirements that sufficiently covers the customers' new requirements.

During our ownership of the project, new unexpected requirements were also brought forward by the customer, hence the aforementioned process needed to be repeated.

Lastly, the presentation of requirements was also kept the same. However, the indexing was updated to distinguish between existing and new requirements, to improve traceability and help focus implementation. This was done by distinguishing between assessment 1 and 2, through the prefixes of the index.

Requirements

User requirements

Index	ID	Description	Priority
1.01	UR_SWITCHING_COOKS	The user should be able to switch between cooks.	Shall
1.02	UR_MOVING_COOK	The user needs to be able to move the cook to the cutting, baking, frying and serving stations and the pantry.	Shall
1.03	UR_COOK_ACTION	The user needs to be able to interact with the stations their cooks are standing near.	Shall
1.04	UR_STATION_ACTION	While using the station the user needs to be able to complete an action (i.e. flipping a patty). If they fail this, they have to restart the step.	Shall
1.05	UR_COOK_STACK	The user needs to be able to stack ingredients from different stations to a cook - essentially a cook's inventory.	Needs
1.06	UR_CUSTOMER_VIEW	The user needs to be able to see the waiting customers, their orders and the recipes of the orders.	Shall
1.07	UR_TIME_CUSTOMERS	User needs to be able to see how much time has elapsed since the customer has placed their order.	Needs
1.08	UR_MAX_SERVE	Users need to be able to see their maximum number of customers they have served in endless mode.	Shall
1.09	UR_SCENARIO_TIME	In scenario mode the user should know how long it took them to complete the scenario.	Shall
1.10	UR_USER_EXPERIENCE	The user should be familiar with the design of the game, and it should be simple and intuitive.	Should
2.01	UR_REPUTATION	The user starts with 3 reputation points (essentially HP). These need to be clearly displayed and possibly can be increased.	Shall
2.02	UR_EARNINGS	Users need to be able to collect earnings. Current earnings shown clearly to the user.	Shall

2.03	UR_ENDLESS_MODE	The user should be able to play an “endless” mode of the game.	Shall
2.04	UR_MOVEMENT_WASD	The user should be able to move each cook with WASD and interact with stations depending on their location.	May
2.05	UR_PURCHASE_COOK	The user should be able to spend their in game money on a third functional cook.	Shall
2.06	UR_POWERUP	The user should be able to buy a powerup that changes how the chef or restaurant acts, (e.g. speed boost, faster cook times).	Shall
2.07	UR_DIFFICULTY_SELECT	The user should be able to select the difficulty for their game.	Shall
2.08	UR_SAVE_GAME	The user should be able to save their game and reload in the future.	Shall
2.09	UR_RECIPES	The user should be able to make salads, burgers, pizzas, and jacket potatoes with the necessary ingredients.	Shall

[fig.1]

Functional requirements

Index	ID	Description	User Requirement
1.11	FR_DIFFERENT_COOKS	The system should be able to differentiate between different cooks.	UR_SWITCHING_COOKS
1.12	FR_CURRENT_COOKS	The system should be able to have a current cook that is being controlled by the user. Current cook is highlighted.	UR_SWITCHING_COOKS
1.13	FR_SWITCH_COOKS	The system should allow the user to switch between cooks by clicking on their respective sprites.	UR_SWITCHING_COOK
1.14	FR_MOVE_COOK	The system should allow the user to move their selected cook by clicking on the stations.	UR_MOVING_COOK
1.15	FR_DESTINATIONS	The system should have a set of different coordinates that the cooks can move to.	UR_MOVING_COOK
1.16	FR_MOVING_ROUTES	The system should have a set of routes that cooks can take between the different stations.	UR_MOVING_COOK
1.17	FR_MOVING_GRAPHICS	The system should have graphics for cooks moving around the kitchen.	UR_MOVING_COOK
1.18	FR_COOK_COLLISIONS	The system should ensure that two cooks moving on the same path won't affect their movement.	UR_MOVING_COOK
1.19	FR_USE_STATION	After the cook has arrived at the serving station or pantry, a menu opens up. After the cook has arrived at a cooking station, the user must click on the station to use it/drop their item.	UR_COOK_ACTION
1.20	FR_DROP_RESTRICTION	The system should not allow the user to drop an ingredient on an incorrect station.	UR_COOK_ACTION
1.21	FR_TAKE_PREPPED_INGREDIENT	Once the preparation step has been completed the system should allow the user to add the	UR_COOK_ACTION

		prepared ingredient to their stack by clicking on a button.	
1.22	FR_COOK_RESTRICTIONS	The system should prevent the user from being able to control the cook during a preparation step.	UR_COOK_ACTION
1.23	FR_VIEW_PANTRY	When a user clicks on the pantry a window should pop up with various ingredients; the top row of order tickets(with the recipes) should still be visible, as well as the stack on the right. There should also be a bin icon that removes the top item from the stack.	UR_COOK_ACTION
1.24	FR_EXIT_PANTRY	There should be an exit button to leave the pantry window.	UR_COOK_ACTION
1.25	FR_SERVING_STATION	When the user clicks on the serving station a menu should pop up with images of the dishes.	UR_COOK_ACTION
1.26	FR_EXIT_SERVING_STATION	The system should provide an exit button to leave the serving station.	UR_COOK_ACTION
1.27	FR_SERVE_DISH	If the user has the correct ingredients on their stack (in any order) the system should allow them to click on the image of the dish to serve it, it is automatically served to the customer that waited to longested and the order ticket is removed from the top row.	UR_COOK_ACTION
1.28	FR_INCOMPLETE	If the user tries to serve an incomplete dish the system.	UR_COOK_ACTION
2.10	FR_BURGER_BURN	The burgers should burn if not flipped in time.	UR_COOK_ACTION
2.11	FR_CUSTOMER_GROUPS	Customers should start arriving in groups after x time, depending on difficulty.	UR_CUSTOMER_VIEW
2.12	FR_CUSTOMER_LEAVE	Customers should leave if not served on time.	UR_TIME_CUSTOMERS
2.13	FR_REPUTATION	Reputation should change based on customer satisfaction - Lose 1 rep on dissatisfied customer /	UR_REPUTATION

		gain 0.2 rep on satisfied customer.	
2.14	FR_MULTIPLE_CUSTOMERS	Multiple customers should be served at once - all orders should be visible at the same time.	UR_CUSTOMER_VIEW
2.15	FR_DIFFICULTY	Must be an Easy, Regular, and Hard mode. (Faster customers, more reputation decrease, etc.).	UR_DIFFICULTY_SELECT
2.16	FR_SAVE_STATES	The project must be able to produce and read save files so that the game can be reloaded.	UR_SAVE_GAME
2.17	FR_POWERUP	Powerups must be introduced that can change the behaviour of the game in some way.	UR_POWERUP

[fig.2]

Non-functional requirements

Index	ID	Description	User requirement	Fit Criteria
1.29	NFR_PORTABILITY	It should be able to run on different operating systems.	UR_USER_EXPERIENCE	It should be able to operate on at least 3 different operating systems
1.30	NFR_VISUAL_CUE	When the user needs to perform a station action, the system should provide the user with a visual cue that shouldn't rely on text or colour.	UR_STATION_ACTION	80% of testers were able to identify and respond to each visual cue
1.31	NFR_TIMING	The scenario mode should take 5-10 minutes, endless mode is indefinite.	UR_USER_EXPERIENCE	5<= meantime <= 10 (For scenario mode)
1.32	NFR_USABILITY	The system should be easy to understand and use.	UR_USER_EXPERIENCE	80% of testers would agree that the system was easy to use
1.33	NFR_PERFORMANCE	The system should be able to run for an entire day, without any major drops in performance.	UR_USER_EXPERIENCE	The system can run for at least 8 hours
1.34	NFR_FAILURE_RATE	The system should not crash or freeze on a standard computer.	UR_USER_EXPERIENCE	The system should crash no more than 1% of the time
1.35	NFR_LOAD_TIME	The system should load quickly.	UR_USER_EXPERIENCE	Maximum time to load should be 20 seconds
1.36	NFR_USER_ENGAGEMENT	The system should be fun and engaging to prospective students.	UR_USER_EXPERIENCE	80% of testers would describe the system as fun and engaging

1.37	NFR_OPERABILITY	Ensure new users can play the game without a tutorial or a rules page.	UR_USER_EXPERIENCE	80% of testers completed the game without assistance
1.38	NFR_FRAME_RATE	The frame rate should not drop during expected usage.	UR_USER_EXPERIENCE	Minimum frame rate of 20fps
1.39	NFR_VISIBILITY	The system should use a colour scheme that maximises visibility.	UR_USER_EXPERIENCE	80% of testers found that the colour scheme didn't hinder visibility
1.40	NFR_ACTION_TIME	The cooking actions shouldn't take too long so the user doesn't become unengaged.	UR_COOK_ACTION	Maximum time for a cooking action is 30 seconds
1.41	NFR_SERVING_TIME	The majority of users should be able to serve the dish in time.	UR_COOK_ACTION	80% of testers were able to serve the order in time
1.42	NFR_MOVING_TIME	The time it takes for a cook to move to a station should be reasonable.	UR_MOVING_COOK	Maximum time for a cook to reach a station should be 5 seconds
1.43	NFR_TEXT_SIZE	Ensure that any text is of the appropriate font size.	UR_USER_EXPERIENCE	Minimum font size 7
2.18	NFR_NO_VIOLENCE	The game should not have any violent themes or content.	UR_USER_EXPERIENCE	Game must adhere to a "3" PEGI age rating
2.19	NFR_JOYFUL	Game must be joyful and colourful	UR_USER_EXPERIENCE	No dark colours or themes

[fig.3]