Requirements

Group 24

Joe Cambridge
Joss Davis
Emily Dennison
Louise Evenden
Amber Hemsley
Josh McWilliam
Lianyu Zhao

Introduction

User requirements were elicited from the assessment brief given to us which was then followed by a client meeting, in which we asked for specifics on details included in the assessment brief.

Within the assessment brief we were given a SSON (Single statement of need):
"You are to build a single-player game that requires managing the staff around a kitchen, who will be preparing various dishes requested by customers coming into the Piazza Restaurant". This paired with information given in the rest of the assessment brief and the client meeting allowed us to create a complete list of requirements which was split into user requirements and system requirements:

- User requirements are for non-technical people involved in the process and is a list of tasks that users should be capable of doing within the system.
- System requirements details the technical implementation including a description of how the system will deliver the needs of the users. (System requirements is further broken down into functional and non-functional requirements):
    - Functional requirements are things the system must do
    - Non-functional requirements are qualities the system must have

The notation we chose to represent requirements was Easy Approach to Requirements Syntax (EARS). (Alternate link / description: https://www.jamasoftware.com/requirements-managementguide/writing-requirements/adopting-the-ears-notation-to-improve-requirements-engineering) This was chosen as it reduces subjective language when reading through the requirements and it remains consistent and easy to read. It is also widely used in industry and in universities so it should be understood by most individuals.

The requirements are presented within three tables (User, Functional, Non-functional). This was decided as it is clearer and more succinct than any other presentation (e.g. textual). With this approach each requirement can be given a unique key (ID), which then can make documentation for architecture easier as requirements may be referenced by their ID's. Each User Requirement is given a priority:

- Shall - must be fully implemented (Highest priority)
- Should - should be fully implemented but isn't strictly required(Medium priority)
- May - an optional element that would be desirable but is not necessary (Low Priority)

The functional requirements table is partially or wholly explained by a user requirement, so each of these will link back to the first table. Along with this fit requirements have been added, highlighting the specific criteria for the non-functional requirements to be classes as fully implemented without issues.

User Requirements

ID	Description	Priority
UR_SYS_REQ	The game must be able to run on most computers	Shall
UR_USER_AGE	The game must be designed for anyone from age 5+	Shall
UR_GRAPHICS	The game should be designed so that different assets are very clearly distinguishable	Should
UR_GAME_SIMPLE	The game should be very simple to use and play so that almost anyone can immediately be capable of understanding it	Should
UR_EXHIBIT	The game is to be designed for a high flow of users	Should
UR_SINGLE	The game should be single player e.g. offline	Shall
UR_DOC	The game should be completely documented so that the team taking on this game will be able to continue to develop it	Should
UR_TOOLS	Everything used in the game needs to be open source, so that if this game is shared there are no issues with copyright	Shall
UR_GAME_PLAY	This should be a game in which you have to cook some food to then deliver to a customer	Shall
UR_TIMING	The game must time how long the user takes to serve customers	Shall

Functional Requirements

ID	Description	User Requirements
FR_CONTROLS	The controls shall be exclusively mouse and keyboard	UR_GAME_SIMPLE
FR_STATION_NUMBERS	The user shall have access to 2 cooking and ingredient stations for each cook (4 in total e.g. 2 chopping boards and two friers)	UR_GAME_SIMPLE
FR_GAME_MODES	The game shall include two game modes: scenario mode and endless mode	UR_GAME_PLAY
FR_SCENARIO_MODE	The scenario game mode shall have a configurable number of customers that will have to be served (this number will default to 5)	UR_GAME_PLAY
FR_ENDLESS_MODE	The endless game mode shall have customers arrive more and more often until the user runs out of reputation points	
FR_COMPLETION	When all orders are completed, the scenario shall end	UR_GAME_PLAY
FR_TIMING	The game shall take approx. 5 minutes, due to high flow of users	UR_EXHIBIT
FR_CUSTOMERS	While a customer has an order, they shall be visible somewhere on the screen	UR_GAME_PLAY
FR_RECIPES	There shall be 4 recipes: pizza, salad, jacket potato and burger	UR_GAME_PLAY

FR_COOKING_STATIONS	The cooking stations shall enable preparation of ingredients - for example cutting board for lettuce, or grill for patty	UR_GAME_PLAY
FR_PREP_FAIL	Preparation steps can be failed resulting in the step having to be repeated	UR_GAME_PLAY
FR_COOKS	The player shall have access to 3 cooks which the player can switch between and which the player can move	UR_GAME_SIMPLE
FR_INGREDIENT_STATIONS	The user shall have access to unlimited amounts of the specified ingredient from the ingredient station	UR_GAME_PLAY
FR_CUSTOMER_FLOW	1 to 3 customers shall arrive at different intervals, and when a customer has been served the next customer shall become the first in line.	UR_GAME_PLAY
FR_COMPLETION_TIME	The game shall return the time taken to serve all 5 customers	UR_TIMING
FR_MENU	The game shall have a simple start menu	UR_GAME_SIMPLE
FR_REPUTATION	The game shall display the user's number of reputation points, initially 3	UR_GAME_PLAY
FR_COMPLETION_TIME_LIMIT	When a customer has not been served within a given time limit, the user shall lose 1 reputation point	UR_TIMING
FR_GAME_OVER	When the user's reputation points drop to 0, the game shall end and display a game over screen	UR_GAME_PLAY
FR_COLLISIONS	The entities in the game shall not overlap, (e.g. basic physics should apply to the game)	UR_GAME_SIMPLE
FR_RECIPE_BOOK	The game shall have some way to see the recipe that the customers have asked for at all times	UR_GAME_SIMPLE
FR_COUNTER	When a dish is complete, the dish shall be able to be delivered by being placed on the counter	UR_GAME_PLAY
FR_INVESTMENT	The game shall allow the user to invest their earnings to unlock more cooks and cooking stations	UR_GAME_PLAY
FR_SAVE_FILES	The game shall allow the user to save the state of a game to be resumed later	UR_GAME_PLAY
FR_DIFFICULTY	The game shall include multiple levels of difficulty for the user to choose between	UR_GAME_PLAY
FR_POWERUPS	The game shall include a number of power-ups, which make the game easier or more fun in specific ways	UR_GAME_PLAY

Non Functional Requirements

ID	Description	User Requirements	Fit Criteria
NFR_SINGLE_PLAYER	The game shall be offline and single player	UR_SINGLE	The game should have no capability whatsoever of connecting to the internet
NFR_GAME_SIMPLE	The game shall be very easy to "pick up and play" so almost all new players should immediately be capable of playing	UR_GAME_SIMPLE	The game should be immediately understandable, intuitive and by at least 95% of new players (they should all understand its concepts and the games goals)
NFR_SYS_REQ	The game shall be designed for a standard computer, without any special hardware.	UR_SYS_REQ	The game should be able to run on any computer with 4gb of ram a minimum of an i3 (7th gen or equivalent) and at least 5 gb of free storage
NFR_USER_AGE	The game shall have no reference to swearing, violence or any graphic content	UR_USER_AGE	This game should aim to not offend a minimum of 99.5% of individuals playing this game and should be playable by anyone of any age (including children)
NFR_CONTRAST	The games assets shall be clearly discernible	UR_GRAPHICS	The game should still be playable by those who are colour blind
NFR_APROACHABLE	The game shall be appealing to a wide demographic of users	UR_USER_AGE UR_GRAPHICS	95% of Individuals aged 5-80 should be able to look at the game and agree that it is appealing and have a desire to play it
NFR_DOCUMENTATION	The game shall be documented completely, with comments explaining lines of code and java docs for classes	UR_DOC	The game should be documented well enough so that other teams would be capable of choosing this code base to continue to develop into assessment 2 and not need to use the internet to a significant degree to understand the code
NFR_SCREEN_RES	The game shall run on different resolution screens	UR_SYS_REQ	The game should be capable of running on a screen resolution of a minimum of 640*480 pixels up to any resolution
NFR_DOC_MAINTANABILITY	The documented code shall be easily maintainable in case of further developments	UR_DOC	Code should aim to be as "modular" as possible so that new features can be quickly implemented and designed
NFR_TOOLS	All libraries, assets and code shall be open source	UR_TOOLS	The entire game should be open source as it will need to be shared and distributed. Therefore anything that is licensed must be able to be shared and distributed by the university and its students.