Updated Requirements

Group 28

Piazza Panic

By OuseWorks (from devCharles, Group 26)

Ben Harris
Joshua Gill
Niamh Hanratty
Amy Raymond
Matthew Czyzewski
Matt Rohatynskyj

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 the Easy Approach to Requirements Syntax (EARS) [1]. 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.

We found that following this format reduced errors and also helped us make sure we didn't make duplicates of requirements. On reflection, we think that this format was the right choice as it made logical sense to order the requirements this way and therefore is easier to read.

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_DIFFICULTY	The player shall be able to play the game on easy, medium or hard difficulty.	Shall
UR_ENDLESS	The player shall be able to select an endless mode.	Shall
UR_EXHIBIT	The game is to be designed for a high flow of users	Should
UR_GAME_PLAY	This should be an easy to play open source game, where the user has to cook some food to then deliver to a customer	Shall
UR_GRAPHICS	The game should be designed so that different assets are very clearly distinguishable	Should
UR_SAVE	The player should be able to save their gameplay and return to it at another time whilst the game is running	Shall
UR_SCENARIO	The player shall be able to select a scenario mode, in which they can toggle how many customers they will have to serve.	Shall
UR_SINGLE	The game should be single player e.g. offline	Shall
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

Functional Requirements

ID	Description	User Requirements
FR_TIMING	The default scenario gamemode shall take approx. 5 minutes, due to high flow of users	UR_EXHIBIT
FR_TOGGLE_CUSTOMERS	The scenario mode should let the player toggle the number of customers.	UR_SCENARIO
FR_COMPLETION_TIME	The game shall display the time taken to serve all customers	UR_GRAPHICS
FR_COMPLETION_TIME_LIMIT	When the customer has not been served within a given time limit, the user shall lose 1 reputation point (from a maximum of 3)	UR_GAME_PLAY
FR_HELP	There will be a tutorial explaining gameplay	UR_GAME_PLAY
FR_COMPLETION	When all orders are completed, the scenario shall end	UR_GAME_PLAY
FR_DIFFICULTY	When selecting the difficulty, the game shall offer support for different levels of difficulty in the game (e.g. easy, normal, hard)	UR_ENDLESS
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, salad, burger, pizzas, and jacket potatoes	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_INGREDIENT_STATIONS	The user shall have access to unlimited amounts of the specified ingredient from the ingredient station	UR_GAME_PLAY
FR_CUSTOMER_FLOW	Customers shall begin arriving one at a time initially, and then will soon arrive in groups of two or three. They shall form a queue and be served one at a time.	UR_GAME_PLAY
FR_CONTROLS	The controls shall be exclusively mouse and keyboard	UR_GAME_PLAY
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_PLAY
FR_CUSTOMER_TIPS	Customers shall have a chance of giving a monetary tip once served in endless mode.	UR_ENDLESS
FR_MONEY	While in endless mode, the player shall have a monetary balance.	UR_ENDLESS
FR_DISH_PRICES	While in endless mode, dishes shall have a monetary value and the customer shall pay the player for their dish when it is served.	UR_ENDLESS

ID	Description	User Requirements
FR_COOKS	The player shall start with access to 2 cooks which the player can switch between and which the player can move	UR_GAME_PLAY
FR_MORE_COOKS	While in endless mode, kitchen staff can be called back from leave when there are enough reputation points and earnings.	UR_ENDLESS
FR_MENU	The game shall have a start menu where the scenario type and game mode are selectable.	UR_GAME_PLAY
FR_POWER_UP	While in endless mode, the user will be able to unlock 5 different power ups for the chef.	UR_ENDLESS
FR_UNLOCK_STATIONS	While in endless mode, the game shall allow users to unlock more stations and ingredient stations.	UR_ENDLESS
FR_COLLISIONS	The entities in the game shall not overlap, (e.g. basic physics should apply to the game)	UR_GAME_PLAY
FR_RECIPE_BOOK	The game shall have some way to see the recipe that the customers have asked for at all times.	UR_GAME_PLAY
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_SAVE_GAME	While in endless mode, the player shall be able to save the game and resume at a later date.	UR_ENDLESS
FR_PREP_FAIL	While in endless mode, if the player cooks the food for too long, then the food becomes inedible and the customer does not accept the order.	UR_ENDLESS

Non Functional Requirements

ID	Description	User Requirements	Fit Criteria
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 team members would be capable of continuing to develop it.
NFR_DOC_MAINTAINABILITY	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, documented and designed
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 80% of new players (they should all understand its concepts and the games goals)
NFR_CONTRAST	The games assets shall be clearly discernible	UR_GRAPHICS	The game should still be playable by those who have colour vision deficiency
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_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_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_TOOLS	All libraries, assets and code shall be open source	UR_GAME_PLAY	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.
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)

ID	Description	User Requirements	Fit Criteria
NFR_APROACHABLE	The game shall be appealing to a wide demographic of users	UR_USER_AGE UR_GRAPHICS	Over 60% 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

References:

[1] Easy Approach to Requirements Syntax (EARS)
(Alternate link / description:
https://www.jamasoftware.com/requirements-management-guide/writing-requirements/adopting-the-ears-notation-to-improve-requirements-engineering)