Requirements

Team 15

Aous Abdulla
Byron Morris
Eddie Barun
Ethan Griffiths
Theo Toth
Weihan Zhu

2 (a)

We obtained these requirements by learning the meaning of requirements engineering from our lectures and the interview that we have attended to know some general ideas and questions for developing our game.

We met as a team to decide on suitable questions to be posed during the stakeholder interview based on our interpretation of the design brief. We found misunderstandings to create a coherent and accordant set of requirements, e.g. we cut out some questions and changed some questions to be more accurate for user and system demand.

Using these questions we went into the interview to gain a deeper insight into what the stakeholder required from the system, and adapted some questions during the interview based on the responses given. We created a transcript from a recording of the interview and used this to generate a set of requirements. Upon completing the interview our interpretation of the design brief changed somewhat, and so our requirements were simplified. We separated these requirements into three parts, they are functional system requirements, non-functional system requirements and user requirements.

Following the elicitation of requirements, we formatted them in a table, similar to the table presented in slides from week seven. We created the functional requirements based on how the system should function based on the user requirements.

"ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes -- Requirements engineering," in *ISO/IEC/IEEE 29148:2018(E)* , vol., no., pp.1-104, 30 Nov. 2018, doi: 10.1109/IEEESTD.2018.8559686.

2(b)

<u>User requirements</u>

ID	Description	Priority
UR_MOVEMENT /	Movement of characters in game, in response to input.	Shall
UR_INTERACTING /	Going up to a station, such as frying pan and interacting with it.	Shall
UR_SWITCHING /	Switch between cooks	Shall
UR_CARRY /	Being able to select an item and carry it.	shall
UR_WIN	Win the game	shall
UR_LOSE	Failing 3 tasks(losing all reputations points), losing the level.	should
UR_FAIL	Failing a task	should
UR_ORDER_SEND /	Sending out order to customer	shall
UR_ORDER_TAKE /	Receive an order from a customer	should
UR_POINTS	Earning points dependant on how good your performance in playing the game was.	should
UR_REPUTATION	Starting with 3 reputation points, decreasing with every failed order	shall
UR_SCOREBOARD	Display an arcade style scoreboard	maybe
UR_SATISFACTION	Tiered score system	maybe

<u>Functional requirements</u>

ID	Description	USER_REQUIREMENTS
FR_TAKE_ITEM /	Picking up item (plates/ingredients/pans)	UR_INTERACTING
FR_PUT_ITEM /	Putting an item down, on a kitchen side or in the pan/chopping board/plate.	UR_INTERACTING
FR_INTERACTION_CONFIRMATION	Confirmation that an action is	<u>UR_INTERACTING</u>

	currently being undertaken, such as a progress bar.	
FR_INTERACTION_COMPLETION	Completing an action is shown, there is a way of knowing when an item is cooked/chopped.	UR_INTERACTING
FR_INTERACTION_FAIL	Failing an interaction such as burning an item that the user is cooking.	UR_FAIL
FR_MOVEMENT /	Moving around the map.	UR_MOVEMENT
FR_CUSTOMER_INTERACTION	Interacting with a customer, taking/delivering a order.	UR_INTERACTING
FR_ORDER_COMPLETION	Order complete, recipe disappears	UR_ORDER_SEND
FR_SWITCHING /	Switching chefs.	UR_SWITCHING
FR_ORDER_CONFIRMATION	Take orders from customer	UR_ORDER_TAKE
FR_INGREDIENT_COMBINE /	Combine ingredients of food	UR_INTERACTING
FR_SCORING	Scoring points from completing orders.	UR_POINTS
FR_SCENARIO_MODE	Five customer mode	UR_SEND

\\\\\\\\\\\\\\

NON-FUNCTIONAL REQUIREMENTS TABLE

ID	Description	User Requirements	Fit Criteria
NFR_TIMING na	A game should have a timer set	UR_WIN UR_FAIL	A game should last 5-10 minutes
NFR_ACCESSIBILITY na	Should be accessible to most audiences	UR_INTERACTING	Should be easily accessible by all visually-abled audience
NFR_USABILITY /	Should be easy to use in most, if not all environments	UR_MOVEMENT UR_INTERACTING UR_SWITCHING UR_CARRY UR_ORDER_SEND UR_ORDER_TAKE	(in loud room) Should be playable by people with little experience of games
NFR_AESTHETICS na	Game should have aesthetically pleasing visuals and assets	UR_SATISFACTION	Has to appear smooth to the user
NFR_COMPATIBILITY /	Game will be compatible		Should run on windows and linux

	with most modern PCs		
NFR_OPERABILITY	Game should have clear keybinds	UR_MOVEMENT UR_INTERACTING UR_SWITCHING UR_CARRY UR_ORDER_SEND UR_ORDER_TAKE	Using keys for movement: For interacting:
NFR_ARCADE_MODE na	Game should have an arcade mode for demonstration when not being played	NA	Game will activate arcade mode after 10 seconds
NFR_SIMPLICITY /	Game must avoid complex or hard to understand concepts	UR_MOVEMENT UR_INTERACTING UR_SWITCHING UR_CARRY UR_ORDER_SEND UR_ORDER_TAKE	Game will have only 2 modes, both of which require no game experience and well known keybinds
NFR_SUITABILITY na	Must be suitable for all audiences	UR_INTERACTING	Should be family friendly in a university open day.
NFR_RELIABILITY/	Game should not crash		Game should be stable enough to operate in relatively modern computers
NFR_READABILITY/	Code should be easy to maintain and read		Other team members should be able to easily read and understand the code
NFR_REUSABILITY /	Should be able to reuse classes in different stages and locations in the game.		Code should be able to function in different stages throughout the game.
NFR_MODULARITY /	Code should be modular		Classes should be able to be individually changed without affecting the other classes.
NFR_SCALABILITY /	The classes can be made for the purpose of being east to edit, expand, etc		Additions in graphic detail, levels, etc should be easy to add
NFR_MAINTAINABILITY /	The game can be easily updated and maintained by the members of the group.	UR_EASY_TO_MAINTAIN	Game should be accessible by members of the group for updates or maintenance.