Requirements

Group Number: 10

Team Name: Decassociation

Group Member Names:
Mohammad Abdullah
Tom Broadbent
Poppy Fynes
Owen Lister
Michael Marples
Lucy Walsh

The 2D video game, Piazza Panic, shall allow players to manage staff around a kitchen, to prepare and cook dishes for customers who enter the Piazza restaurant and request orders for these dishes.

To ensure that we develop a game that accurately reflects what the customer envisions, we have elicited and detailed a set of requirements for the game in this document. These requirements were elicited through a rigorous process of breaking down the Piazza Panic specification from our customer, Antonio Garcia-Dominguez, and then speaking with them in an open interview to determine specific requirements that were not detailed within the brief. We came up with a set of questions that we as a team believed needed to be answered for us to be able to create the customer's vision, and these questions were answered. On top of this, the customer is representing the University of York Communications Officer, who may want to use our game for promotional activities such as open days, so we had to make sure we covered all grounds.

The requirements engineering process allows the customer to validate the end product we create, to ensure that it is what they actually wanted from us. Therefore the purpose of the requirements that we procured is to appropriately represent the needs of the customer and their vision for the product (as accurately as we can, which is challenging given that most requirements are interpreted in different ways by different people). Consequently, we have developed a set of requirements that is split into 3 categories: user requirements, functional requirements and non-functional requirements, where the functional and non-functional requirements are the system requirements. User requirements define the features of a system available to users, in general terms that any stakeholder can understand and the system requirements specify exactly how the system will meet the needs of the users, how it will provide features to the user, in a more technical format. Therefore, one user requirement may correspond to multiple functional requirements. We have split the system requirements into two groups because after research, we found that the two classes are significant for differentiating between requirements for responses to particular situations and defining behaviours for specific features of a system (functional), and those requirements which apply to a system as a whole (non-functional) [1], such as those which define loading times.

We have detailed each of the user, functional and non-functional requirements in three tables below. Each requirement is formatted in a specific way, however all of them have a unique identifier (the ID column), with each type of requirement having its own unique prefix, so that we can reference specific requirements in any context, without confusion or misinterpretation of which requirement(s) we are actually referring to. Building on our standard for identifying requirements, we also standardised keywords to specify the priority of user requirements, based on Ian Sommerville's recommendation [2]. The "Shall" keyword represents a requirement that must be satisfied. "Should" represents requirements that are desirable but not absolutely necessary and "May" is for requirements that would be nice to have but are by no means essential. Last on the structure of our requirements specification, we have a description of each requirement in the tables so it's easy to see our rationale behind the requirements, and we also have a 'fit criteria' for our non-functional requirements which specifies a way for us to measure our implementation of each of those requirements, otherwise it may be quite difficult to know how to meet each requirement and for the customer and stakeholders to actually verify whether we have met the requirements.

User requirements:

ID	Description	Priority
UR_GAME_TIME	Users should ideally not spend more than 10 minutes playing one scenario.	Should
UR_PAUSE	Should be able to pause the game and resume	Should
UR_MAIN_MENU	Main menu to start game from, possibly access settings, leaderboards and credits	Shall
UR_CREDITS	Users should be able to view names of group members and creators of assets used	Should
UR_COLOUR_BLIND	User should not have difficulty distinguishing 2 objects if they are colour blind	Shall
UR_CONTROL_CHANGE	Allow users to customise controls	Shall
UR_TUTORIAL	Have some sort of tutorial / controls / how to play screen	May
UR_MOBILE_SUPPORT	Allow game to be played on mobile devices as well as desktop	May
UR_SYSTEM_SPEC	Target platform should be low power - eg laptops	Shall
UR_SOUND	Should have sound effects and background music	Shall
UR_COOKS	There should be 2 cooks that are equal, and can do different parts of the same recipe or do different recipes.	Shall
UR_SCENARIO_MODE	A scenario-based mode with a fixed number of 5 customers to be served, who will arrive one by one, and will each wait indefinitely for their orders.	Shall
UR_RECIPES	Users will have different recipes to make (burger & salad at least, but not a pizza or jacket potato)	Shall
UR_COOKING_STATIONS	Users should have different stations at which to cook, prepare and interact with the different types of ingredients - cutting, baking, frying, serving.	Shall
UR_PANTRY	There should be a pantry area with ingredient stations for all recipes (burgers and salads).	Shall
UR_INGREDIENTS	There should be an ingredient for every component of every recipe in the game, with each ingredient able to be picked up by a cook at an ingredient station	Shall
UR_CARRY	Players should be able to carry a ‘stack’ of ingredients and should be able to carry complete recipes	Shall
UR_SERVE	There must be a counter where customers wait to be served.	Shall

UR_TIMER	The player should be timed on how long it takes to complete the scenario (serve all customers) as customers wait indefinitely for their orders.	Shall

Functional requirements:

ID	Description	User requirements
FR_SWITCH_COOK	Players should be able to switch between 2 cooks	UR_COOKS
FR_MOVE_COOK	Should be able to move all cooks	UR_COOKS
FR_COOK_INTERACT	Players should be able to have the cook they are currently using, interact with what is in front of them (including serving customers)	UR_COOKS, UR_SERVE
FR_DROP_INGREDIENTS	Cooks should be able to interact with a cooking station to drop the top ingredient of the stack they're carrying onto the station	UR_COOKING_STATIONS
FR_COOKING_STATIONS	Cooks should be able to use multiple cooking stations to complete different parts of a recipe (e.g. to flip a burger at one and then make the burger up at a separate station)	UR_RECIPES, UR_COOKING_STATIONS,
FR_INGREDIENT_STATIONS	Cooks should be able to interact with an ingredient station in the pantry to add an ingredient to the top of the stack of ingredients they are carrying.	UR_RECIPES, UR_PANTRY, UR_CARRY
FR_UNLIMITED_INGREDIENTS	Ingredients at the ingredient stations in the pantry should never run out	UR_PANTRY
FR_CREDITS_SCREEN	Display names of group members and credit any assets used	UR_CREDITS
FR_PAUSE_SCREEN	Allow user to press a button to pause the game, and then allow user to later resume the game from this screen	UR_PAUSE
FR_MAIN_MENU	Game should have main menu from which the user can start playing, possibly access settings and credits	UR_MAIN_MENU
FR_CONTROL_CHANGE	Have a menu where users can customise the controls of the game, possibly also a console as this allows for far better customisation e.g. bind one key to multiple actions, one action to multiple keys, completely unbind action etc	UR_CONTROL_CHANGE
FR_TIMER	The game should only time and display how long it takes to complete	UR_TIMER

	the scenario (serve all customers).	
FR_TUTORIAL	The game may provide a screen which teaches the user how to play, accessible from the main menu	UR_TUTORIAL
FR_TUTORIAL_CONTROL_CHANGE	The tutorial screen should adapt to the customised controls	UR_TUTORIAL, UR_CONTROL_CHANGE
FR_MOBILE	Have touch-screen friendly controls and buttons/joysticks appear on screen when played on a mobile device	UR_MOBILE_SUPPORT

Non-Functional requirements:

ID	Description	User requirements	Fit criteria
NFR_COLOUR_BLIND	Keep colour blindness in mind, ensure objects can be distinguished by more than just colour	UR_COLOUR_BLIND	90% of colour blind users should not suffer from any significant difficulties while playing the game
NFR_QUICK_COOK	A recipe should take around 2 minutes to make, so each ingredient should take around 30 seconds to prepare	UR_GAME_TIME	75% of users should complete a 5 customer scenario in under 10 minutes
NFR_LOW_SPEC	The game should perform acceptably on low end machines like laptops by making use of things like simple assets and efficient algorithms	UR_SYSTEM_SPEC	The game should run at more than 30 frames per second on a laptop with an Intel core i3 and 4GB of ram
NFR_EASE_OF_USE	Users should be able to play the game even if they don’t have experience in playing video games, so the controls must be intuitive and basic.	UR_TUTORIAL, UR_CONTROL_CHANGE	At least 90% of users should be able to play the game/feel confident in playing the game after completing the tutorial and seeing the controls.

References:

[1] I. Sommerville, *Software engineering*. Essex: Pearson Education, 2015, pp.107
[2] I. Sommerville, *Software engineering*. Essex: Pearson Education, 2015, pp.122