Requirements

Team 2 - Billy’s Amazing Team

Member Names: Mitchel Bekink, Alyx Bruno-Bamford, Ashley Bryan, Rajul Chawla, Bosco Lau, Kacey Van Der Walt, Chris Grzywacz

Introduction

Stakeholder Identification:
As this is a small project with a small number of people involved, we only had one stakeholder with whom we could interact during the project: our customer (Kostas Barmpis).

Eliciting the Requirements:
Customer interviews were used to elicit the requirements of our project and show us what the customer was looking for. We first created questions about the parts we found most ambiguous in the brief that was initially provided to us. In the interview, we recorded the meeting to ensure we wouldn't lose requirement accuracy from note-taking. We then transcribed the recording and used this transcript, with the original brief, as a statement of needs, which we could then use to produce our requirements.

Requirements research:
Our research indicates that there are multiple methods used to specify and present requirements such as:

- SRS (software requirements specification) documents
    - A description of the functionality that typically contains the purpose, scope, system behaviour, requirements (functional and non-functional), assumptions and constraints.
- User stories
    - Simple stories structured as follows: As a [user role], I want to [task] so that I can [goal].
- Use case diagrams
    - A visual or textual description containing: Actors, scenarios and branching paths.

Using user stories and SRS's formatting style, we decided to use a table (grouped into 3 categories) to represent our system's requirements.

Deciding on the Requirements Specifications and Presentation:
We have three categories of requirements: UR (user requirements), FR (functional requirements) and NFR (non-functional requirements). The UR informs the FR and NFR.

To present these requirements we created three separate tables, one for each type of requirement. Each requirement would have a row of the table with information provided for each, to allow us to easily view the data and refer to it during this project:

- Every requirement is assigned a unique ID that identifies the type of requirement and what it is. It also has a brief description explaining it.
- User requirements also have a priority level: shall, should or may. This allows us to prioritise certain requirements over others.
- FRs and NFRs link to at least one user requirement as the UR informs the FR/NFR.
- Finally, the non-functional requirements also have fit criteria, allowing us to measure whether they have been successfully implemented.

User Requirements

ID	Description	Priority
UR_MAP	The user can view the whole campus and see the slots for the buildings	Shall
UR_LAND_TRAITS	Empty plots (such as lakes) give a boost to satisfaction for buildings in plots nearby	Shall
UR_PLACE_BUILDING	The user should be able to place buildings in the lots available, provided they have enough money. The buildings should affect student satisfaction based on their proximity to other buildings and their types.	Shall
UR_GHOST_BUILDING	The user can view what it would look like to place a building in that slot (maybe working with the land traits). Turns red if it is in an invalid slot	Should
UR_AUDIO_FEEDBACK	The user can audibly hear that they have placed a building or spent money, etc, as well as alerts such as when an event is occurring, or they are attempting an action they cannot perform	May
UR_CONSTRUCTION_TIMER	The user can view how long a building will take to build	May
UR_BUILDING_COUNTER	The user can see the count of each type building of building place	Shall
UR_EVENTS	A series of events must occur after a period that affects the way your campus functions (possibly random)	Shall
UR_ALERTS	The user may receive alerts from the system regarding making mistakes or if something is going wrong. (could be audio feedback too)	Should
UR_FIVE_MINUTES	The user can see the timer (how much time is left, excluding paused time)	Shall
UR_PAUSE_GAME	Ability to pause and resume the game, freeze gameplay, exit the current game to the title screen	Shall
UR_PAUSE_TIME	The user can pause the game but still interact with the map, view statistics, and place buildings. However, no construction will be paused.	May
UR_TUTORIAL	Indicates to the user how to interact and win the game	Should
UR_SATISFACTION	The game should calculate student satisfaction based on various factors, this should then be shown not only at all times but have a score calculated from it shown at the end of the game	Shall
UR_PC	It has to function on a typical laptop or PC	Shall
UR_DIFFICULTY	The user should be able to select what difficulty they want, with harder difficulties bringing a higher frequency of negative events	Should
UR_NO_NEGATIVE_FEEDBACK	The feedback the game gives you should be positive - the game should never end prematurely if you do poorly, it should only be reflected in your score when the game ends in 5 minutes	Shall
UR_MONEY	The user should be able to see their balance at all times and be able to check a summary of their income and expenditures	Should
UR_TITLE_SCREEN	On startup, the user interacts with a title screen to begin the game (maybe select difficulty) and exit the application	Should
UR_CAMERA	The camera is a fixed angle that allows the user to view the whole map and zoom in	Shall

Functional Requirements

ID	Description	User Requirements
FR_MAP	A map to place buildings on, view the campus with established roads/paths (maybe established bus routes)	UR_PLACE_BUILDING, UR_MAP
FR_PLOTS	Plots are locations where buildings can be built	UR_PLACE_BUILDING, UR_MAP, UR_LAND_TRAITS
FR_BUILDINGS	At least one place to sleep, one place to learn, one place to eat, one recreational activity	UR_PLACE_BUILDING
FR_TIMER	Up to 5 real-world minutes	UR_FIVE_MINUTES
FR_BUILDING_COUNTER	How many of each building has been placed	UR_PLACE_BUILDING, UR_BUILDING_COUNTER
FR_PAUSE_GAME	Allows the timer and game to be paused	UR_FIVE_MINUTES, UR PAUSE_GAME
FR_PAUSE_TIMEFLOW	Allow only the timer to be paused	UR_PAUSE_TIME, UR_FIVE_MINUTES
FR_MONEY	Keeps track of how much money the user has, also how much has been spent and received for end-game statistics	UR_MONEY
FR_PC	The game should be able to run and display on PCs	UR_PC
FR_AUDIO	The system can play sounds through the speakers at call	UR_AUDIO_FEEDBACK
FR_TUTORIAL	The system should display a tutorial on the first playthrough of the game	UR_TUTORIAL
FR_EVENTS	The system must have a store of potential events and then make these events occur at random based on the difficulty	UR_EVENTS, UR_DIFFICULTY
FR_FEEDBACK	The system should display feedback to the player at the end of the game, this feedback is based on their score	UR_NO_NEGATIVE_FEEDBACK, UR_SATISFACTION
FR_WARNING	The system will warn the user if they are about to make a mistake like constructing a building that will not be completed within the remaining time	UR_ALERTS, UR_AUDIO_FEEDBACK
FR_TITLE_SCREEN	Upon loading up the game the first screen displayed is the title screen	UR_TITLE_SCREEN
FR_CONSTRUCTION_TIMER	The user can view how long it will take to build, counting down to 0	UR_CONSTRUCTION_TIMER
FR_CAMERA	The camera is movable (not at full zoom out), zoomable but not able to change the angle	UR_CAMERA

Non-functional Requirements

ID	Description	User Requirements	Fit Criteria
NFR_AVAILABILITY	The game should be able to run without a network connection	UR_NETWORK	The game runs without a network connection
NFR_FRAMERATE	The gameplay should run the same regardless of PC performance, e.g game lasts 5 minutes regardless of lag	UR_PC, UR_TIMER	The game lasts 300 seconds regardless of lag/game stress/resources
NFR_NO_CRASHES	The game should not crash	UR_PC	The game does not crash 99% of the time
NFR_DIFFICULTY	The game should be winnable yet challenging for all	UR_DIFFICULTY	The game allows for all users to enjoy the game no matter their skill level
NFR_LOADING_TIME	The game should load quickly	UR_PC	Less than one second to load
NFR_LATENCY	The game should be nearly instant to respond to user interactions	UR_PC	50ms for user interactions to be resolved
NFR_USABILITY	The game's UI should be intuitive to use		Users can place a building, pause the game, etc within 3 clicks or less
NFR_PLATFORMS	The game should run on a variety of platforms	UR_PC	Should run on Windows, Mac, Linux
NFR_RESOLUTION	The game should scale the screen for different resolutions	UR_PC	The lowest resolution is 720p, Max resolution is 4K
NFR_CODE_MODULARITY	The codebase should be modular		Allows for new features such as buildings to be added with minimal impact

Bibliography

[1] QAT Global, “Guide to User Requirements,” QAT Global, 2024. [Online]. Available: https://qat.com/guide-user-requirements/.