Requirements

ENG1 Team 9

Jacob Dicken
Bertie Cartwright
William Croft
James Dovener
Henry Chan

This document has been modified for Assessment 2 to reflect the changes made by ENG1 Team 6:

- Hussain Alhabib
- Ellen Matthews
- Minnie Poon
- Jason Ruan
- Daniel Smith
- Owen Smith

Introduction to Requirements

To begin eliciting the requirements we first analysed the product brief, which provided us with some initial fundamental requirements of the game. We noted down any remaining questions we had around the product brief, and created a list of important questions to ask the stakeholder, ordered by priority. Additionally we did some further research into sensible system requirements for the game (e.g system specs and supported OS's). From this, we found that devices with at least 2GB of RAM and 2 or more CPU cores should be sufficient for a simple 2D game.

Next we set up a meeting with the client to discuss the questions we had curated. In doing so we gathered and negotiated new requirements and cleared up any ambiguity in the requirements we had already gathered. The client also noted that we had a lot of freedom to decide how some of the lower level design elements of the game worked, along with the style/theme of the game. This reduced the number of requirements relating to the style and design of the game.

During the meeting we also worked with the client to come up with a SSON which will help the whole team understand the broad concept of the game.

Single Statement of Need (SSON)
The system will consist of a campus planning game, which allows the user to place buildings, and simulates an in-game time of 3 years within 5 minutes of real world time.

----------OUR EDITS----------
"We will design a game called UniSim that allows players to manage a university campus by placing and upgrading buildings, with success metrics for student satisfaction and university income while navigating building restrictions and random events within a 5-minute gameplay session."

After collecting the requirements we needed a way to present them in a way that was easy to edit and reference. To accomplish this we created tables to house the requirements. Firstly we made a user requirements table which holds all the requirements that were obtained from the meeting with the client. **User requirements describe the main actions and experiences that the users should be able to experience during gameplay.** These user requirements were written in a non-technical way which allows for anyone involved in the project to understand their meaning and have a corresponding priority of either **"Essential", "High", "Medium" and "Low"** - this provides a clear indication of the priority assigned to a given requirement.

Next we made 3 tables for system requirements: functional, non-functional and constraint. The functional and non-functional requirements were determined by examining the user requirements and then creating descriptions of how the system will satisfy these requirements. Functional **defines how the system will meet the users needs. These include detailed, often technical descriptions of the games functionality and services required for gameplay,** and non-functional **focuses on the game's attributes and qualities, and defines how well the game should perform regarding usability, reliability and performance**. The constraint

requirements were determined from the initial research we did into sensible system requirements before the meeting.

Every requirement in each table has a description of what the requirement is and also **a unique ID to ensure traceability and consistency across documentation.** The ID is a meaningful name (for example UR_BUILDING_COUNTER) which is prefixed with either: UR, FR, NFR, or CR which correspond to: User, Functional, Non-functional and Constraint requirement. This allows for easy identification of what type of requirement the ID corresponds to but also allows for easy referencing in other areas of the project. **A dedicated column in each table lists related requirement IDs, which further helps visualising how different requirements interact and impact one another.**

----------OUR EDITS----------

User Requirements (URs)

ID	Description	Priority
UR_BASIC_BUILDINGS	The user shall have a number of different building types to choose from; a place to learn, a place to sleep, a place to eat, and a reactionary activity. Users may also be able to upgrade and demolish their buildings.	Essential
UR_TIME	The game should simulate a time of around 3 years within a time frame of 5 minutes. And have a counter on the screen that displays how much time is left. When the timer stops the game should end.	Essential
UR_BUILDING_ COUNTER	The game should track and display the number of each building type (e.g. sleep, eat, learn, recreational) placed by the player.	Essential
UR_SCORE	Users should have a way to measure their success in the game via various metrics such as satisfaction and university income. The user can increase or decrease the satisfaction levels of the students in a variety of ways (for example having entertainment and food buildings near accommodation, and reacting to events appropriately).	High
UR_EASE_OF_USE	The game should be intuitive to play at the fundamental level, for example, it should be straightforward to select/place buildings and deal with events. The game should stay intuitive for users with no gaming experience. Also the game should react to the users inputs.	High
UR_BUILDING_LIMITS	The player should be restricted from placing buildings in certain areas of the map (e.g. over a lake, road, or other buildings).	High
UR_EVENTS	The game will include at least three core events that affect the user's experience. The user shall be able to interact and react to events that occur during the course of the game, these depend on the difficulty setting chosen by the user.	High
UR_DEPLOYMENT	The game should be accessible and run smoothly on standard desktops and laptops across all major operating systems. The game should perform well and provide a pleasant user experience.	High
UR_TIPS	The game should include tips and guidance to help players understand how to play the game, such as tutorials or hints.	Medium
UR_SETTINGS	The game should include settings to allow the user to adjust in-game sound levels if sound assets are	Medium

See the Change Report (Change2) for a full list of changes.

	implemented.	
UR_DIFFICULTY_ SETTINGS	The game may contain multiple difficulty settings that the user can choose from. These settings should affect different parts of the game (for example increase the frequency and difficulty of events and change the amount of money you start with). The default setting should be the easiest which helps with accessibility for new users.	Low
UR_MONEY	The game may include an in-game currency system that is used to purchase buildings. The income rate of this currency should increase accordingly with the student population.	Medium
UR_MAINTAINABILITY	The game should be easily modified and maintained by future developers.	High
UR_LEADERBOARD	The game should display a leaderboard that tracks the top 5 scores of players within the same local game session/instance.	High
UR_ACHIEVEMENTS	The game should feature achievements, such as maintaining high satisfaction levels. They should modify the final score and be displayed on the Game Over screen if unlocked.	High

System Requirements

Functional Requirements (FRs)

ID	Description	Link to URs
FR_BUILDINGS	The game shall allow users to place/build and possibly upgrade buildings on the campus map. Players should be able to demolish buildings. The game shall have at least one building of every type namely: Educational, Recreational, Residential and Restaurant.	UR_BASIC_BUILDINGS
FR_TIMER	The game should have a timer that is displayed at all times during gameplay, showing the remaining time from the original 5 minutes.	UR_TIME
FR_BUILDING_ COUNTER	The game should display a counter at all times during gameplay that shows how many of each building type have been placed in the world.	UR_BUILDING_ COUNTER
FR_EVENTS	Events should occur throughout the game - these should come in 3 different types: positive (which benefits the player), negative (which hinders the player) and neutral (which does not affect the	UR_EVENTS, UR_ DIFFICULTY_SETTINGS

See the Change Report (Change2) for a full list of changes.

	game). These events will be set and occur at set times on the lowest difficulty but be randomised on higher difficulties.	
FR_OBSTACLES	The map shall have preplaced obstacles that block the user from building on top of them.	UR_BUILDING_LIMITS
FR_USER_INTERFACE	The UI should display only important information to the user so that it is not overly complex and overwhelming. Satisfaction metrics should be visible to players at all times during gameplay. Interactive elements should react when clicked or moved (for example when a button is pressed its colour changes or a sound is made).	UR_SCORE, UR_EASE_ OF_USE
FR_BUILDING_ EFFECTS	Buildings should have different impacts on player satisfaction based on their type and position in relation to other buildings.	UR_SCORE
FR_BACKGROUND	The game should include background elements (e.g. student walking) to make the map appear more dynamic and engaging.	UR_BASIC_BUILDINGS
FR_SETTINGS	The game should include sound settings for adjusting in-game sound levels, if such assets are included.	UR_SETTINGS
FR_DIFFICULTY_ SELECTION	The game shall allow the user to select the difficulty of a new game. The default should be the easiest setting.	UR_DIFFICULTY_ SETTINGS, UR_EASE_OF _USE
FR_DIFFICULTY _EFFECTS	The difficulty selected should affect the variety and frequency of events along with the amount of money players receive at the start.	UR_DIFFICULTY_ SETTINGS
FR_MONEY	Players should start with a predetermined amount of money based on difficulty and receive an income which increases proportionally to the student population level.	UR_MONEY, UR_ DIFFICULTY_SETTINGS
FR_BUYING	The game should allow users to spend their in-game money on purchasing and upgrading buildings and never allow users to purchase or upgrade a building if they do not have enough in-game money.	UR_MONEY
FR_LEADERBOARD	The game should maintain a session-specific leaderboard with the names and scores of the top 5 players during that instance. Players should be given the option to input their name at the end of their game to appear on the leaderboard.	UR_LEADERBOARD, UR_SCORE

See the Change Report (Change2) for a full list of changes.

FR_ACHIEVEMENTS_ SYSTEM	The game should track players progress and inform them when they earn an achievement.	UR_ACHIEVEMENTS
FR_ACHIEVEMENTS_ EFFECT	Achievements should modify the final score based on their nature i.e. positively or negatively. If unlocked, they should also be listed on the Game Over screen.	UR_ACHIEVEMENTS, UR_SCORE

Non-Functional Requirements (NFRs)

ID	Description	Link to URs	Fit Criteria
NFR_PERFORMANCE	The game should run smoothly, without hitches at any point during gameplay. This should apply on minimum spec machines.	UR_DEPLOYMENT	The game should run 99% of the time at at least 30FPS.
NFR_INTERACTIVE _ELEMENTS_ REACTION	Interactive elements (for example buttons) should react quickly and smoothly to user use. Preferably with some signal to the user like a sound or colour change.	UR_DEPLOYMENT	99% of interactive elements should respond within 1s of interaction.
NFR_OPERABILITY	The game should be easily playable and navigable by new players and accessible to as wide an audience as possible, accommodating players with different needs.	UR_EASE_OF_USE	95% of players should be able to play the game for the first time, with no more than 1 min of tutorials.
NFR_IMMERSION	The game should have sounds and graphics that fit the theme of the game.	UR_EASE_OF_USE	80% of gameplay should have sound assets. 85% of players should agree assets enhance the game.
NFR_LICENSE	The game should use appropriately licensed sounds and music assets to create an enjoyable in-game experience.	UR_SETTINGS	100% of assets should have verified licenses.
NFR_DEPLOYMENT	The game should run smoothly on desktops and laptops across all major operating systems optimised for various hardware.	UR_DEPLOYMENT	The game should maintain a frame rate of at least 30 FPS on 95% of systems that use the game.

See the Change Report (Change2) for a full list of changes.

NFR_ERROR_ MESSAGES	Any errors or warnings generated by the game should be easy to understand and not overly technical.	UR_MAINTAINABILITY	During feedback and gameplay, 90% of players should find error messages clear and easy to understand.
NFR_ DOCUMENTATION	The game should come with clear documentation that explains what each part of the code does and how to modify it.	UR_MAINTAINABILITY	98% of the code's functionalities should be detailed in the doc.
NFR_END_OF_GAME	The game should immediately end after the timer has expired.	UR_TIME, UR_SCORE	A “Game Over” message should appear within 1 second of the timer ending.
NFR_CODE_ MODULARITY	The code should be modular and easily extendable, without making the program difficult to follow.	UR_MAINTAINABILITY	90% of functions should remain under 50 lines long.
NFR_LEADERBOARD_ VISIBILITY	The leaderboard should be easy to understand, and able to be accessed before or after their game.	UR_LEADERBOARD	95% of players should locate and understand the leaderboard within 5 seconds.
NFR_ACHIEVEMENTS _NOTIFICATION	Achievement notifications should be clear, easy to understand, and appear promptly once unlocked.	UR_ACHIEVEMENTS	Achievement notifications should appear within 2 seconds of being unlocked.

Constraint Requirements (CRs)

ID	Description
CR_LOW_SPEC	The game should be optimised to run on low-spec devices, ensuring it is accessible to all players.

See the Change Report (Change2) for a full list of changes.