Assessment 2: Brownfield Development - Requirements

Cohort 1 Team 2

Team Members:

Trace Chinelle
Vidhi Chohan
Apollo Cowan
Siyuan Liu
Aryaman Marathe
Charlie Mason

Introduction to Requirements - Greenfield Development

To begin eliciting the requirements we first analysed the product brief, which provided us with some initial fundamental requirements of the game.

We also noted down any remaining questions we had around the product brief, and then created a list of important questions to ask the stakeholder, ordered by priority. These were questions such as the role of sound effects in our game or guidance on the art style/theme we should choose.

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.

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. 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 shall, should or may - 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 is for things a system should do and non-functional is for qualities a system should have. 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 an ID.

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.

Introduction to Requirements - Brownfield Development

After inheriting this project we needed to create an up to date set of requirements that fit the new product brief. The first thing that we did was write some new User Requirements based on the new concepts that had been added to the product brief.

We added the new requirements to the User Requirements table, including the name, description and priority of each. The User Requirements we added also had a number of Functional Requirements associated with them that we, similarly added to the Functional Requirements table.

The next thing we did was assess each requirement in the existing requirements tables: User Requirements, Functional System Requirements, Non Functional System Requirements, and Constraint Requirements. We decided that some User Requirements and Functional System Requirements did not fit the new idea of the game which we had conceptualised based on the new product brief.

In the case of requirements that did not fit the new game idea at all, they were removed fully from the tables. In other cases, the requirements partially fit the new game idea. In these cases the description of the requirement was modified slightly, and in rare cases the title of the requirement was also changed.

As well as these changes, some User Requirements, that were initially set as low priority requirements that only ‘may’ be implemented, had their priorities increased so that they ‘shall’ be implemented.

User Requirements

ID	Description	Priority
UR_LEADERBOARD	The user shall be able to view a leaderboard that shows the names and scores of the highest 5 scores.	Shall
UR_ACHIEVEMENTS	The user shall be able to unlock achievements during gameplay that affect their final score.	Shall
UR_MONEY	The game shall include an in-game currency system that is used to purchase buildings. The income rate of this currency should increase accordingly with student satisfaction.	Shall
UR_STUDENT_SATISFACTION	The user shall be able to increase or decrease the satisfaction levels of the students in a variety of ways such as building quantity and proximity, and reacting to events appropriately.	Shall
UR_BUILDING_VARIETY	The user shall have a number of different building types to choose from. The variety of buildings should include at least: one place to sleep, one place to learn, one place to eat, and two for recreational activities.	Shall
UR_EVENTS	The user shall be able to react to events that occur regularly during the course of the game. These events should affect the user's satisfaction and/or currency levels.	Shall
UR_GAME_PROGRESS	The game shall simulate a time of around 5 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.	Shall
UR_IMMERSION	The game should be immersive for the player. Meaning that sounds and graphics should match the theme of the game.	Should
UR_TARGET_MARKET	The game should be suitable to be played by 16-20 year old students, or people who want to be students.	Should
UR_PERFORMANCE	The game shall perform well on the minimum spec machines and provide a pleasant user experience.	Shall
UR_INTUITION	The game should be intuitive to play for users with a range of gaming experience. It should be straightforward to select and place buildings, and understand and react to events. The game	Should

	controls should be intuitive and react to user inputs.	
UR_BUILDING_COUNTER	The game shall have a counter denoting how many of each type of building have been placed so far.	Shall
UR_CAMPUS_CREATION	The user shall be able to create their own university on a provided map which includes obstacles where buildings cannot be built.	Shall

Functional System Requirements

ID	Description	User Requirements
FR_TIME_LIMIT	When the time left on the game timer is less than or equal to 0 seconds the game must end, stopping the user from doing any more actions. The game shall then display the user's new achievements, final score and leaderboard.	UR_GAME_PROGRESS UR_ACHIEVEMENTS UR_LEADERBOARD
FR_MAP	The game shall provide a visual campus map for the user.	UR_CAMPUS_CREATION
FR_BUILDING	The game shall allow users to place buildings on the campus map. The building shall take a few seconds to be fully built.	UR_CAMPUS_CREATION UR_BUILDING_VARIETY
FR_BUILDING_TYPES	The game shall have at least one building of the types: Educational, Residential and Eatery. The game shall have at least two buildings of the type Recreational.	UR_BUILDING_VARIETY
FR_SATISFACTION_BUILDINGS	The number of each building type shall increase satisfaction. The closer each type of building is to another type of building should also increase satisfaction.	UR_STUDENT_SATISFACTION
FR_OBSTACLES	The map shall have preplaced obstacles that block the user from building on top of them.	UR_CAMPUS_CREATION
FR_EVENT_EFFECTS	Events shall be able to increase or decrease the user's satisfaction score. Events shall also be able to increase or decrease the user's currency level.	UR_EVENTS
FR_EVENT_VARIETY	One event shall occur halfway through every in-game year. The events shall be	UR_EVENTS

	randomly selected from a list of potential events.	
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. The game shall also display the in-game year of the user.	UR_GAME_PROGRESS
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_USER_INTERFACE	The UI should display only important information to the user so that it is not overly complex and overwhelming.	UR_INTUITION
FR_INTERACTIVE_ELEMENTS	Interactive elements should react when clicked or moved (for example when a button is pressed its colour changes or a sound is made).	UR_INTUITION
FR_MONEY	Players should start with a predetermined amount of money and receive an income at the start of each year proportional to the student satisfaction level.	UR_MONEY
FR_BUYING	The game should allow users to spend their in-game money to place buildings and never allow users to place a building if they do not have sufficient in-game money.	UR_MONEY
FR_LEADERBOARD_MENU	The user shall be able to access the leaderboard of the top five highest scores from the main menu. The user shall be able to reset the leaderboard from this screen.	UR_LEADERBOARD
FR_NEW_HIGH_SCORE	If the user achieves a score higher than the lowest score on the leaderboard, the user shall be able to enter their name to add their score to the leaderboard after the game has finished.	UR_LEADERBOARD
FR_ACHIEVEMENT_SCORE	When the timer runs out the game shall display any achievements that the user achieved during gameplay. Each achievement shall have a positive or negative associated score. The user's score should be amended to include these bonuses and penalties.	UR_GAME_PROGRESS UR_ACHIEVEMENTS
FR_ACHIEVEMENT_MENU	The user shall be able to access a list of	UR_ACHIEVEMENTS

	which achievements they have and have not unlocked from the main menu. When the user achieves an achievement in gameplay, this list should be updated. The user shall be able to reset this list so that none of the achievements are unlocked	

Non Functional System Requirements

ID	Description	User Requirements	Fit Criteria
NFR_ERROR_MESSAGES	Any errors or warnings generated by the game should be easy to understand and not overly technical	UR_MAINTAINABILITY	The error messages should be understood by users who have no knowledge of the code.
NFR_PERFORMANCE	The game should run smoothly, at a framerate around 60, without hitches at any point during gameplay. This should apply on minimum spec machines.	UR_PERFORMANCE	The 60-second average frame-rate should exceed 58. The lowest 1% of frame times should also be above 50.
NFR_INTERACTIVE_ELEMENTS_REACTION	Interactive elements (for example buttons) should react quickly to user use. Preferably with some signal to the user like a sound or colour change.	UR_INTUITION	Interactive elements must react within <1 second
NFR_OPERABILITY	The game should be easily playable and navigable by new players.	UR_INTUITION	A new player shall be comfortable with the game after around 2 minutes of use.
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	The documentation should be understood by users who have no prior knowledge of the code.
NFR_END_OF_GAME	The game should immediately end after the timer has expired.	UR_GAME_PROGRESS	The game should end and block any further user actions within <1 after the timer stops.
NFR_IMMERSION	The game should have sounds and graphics that fit the theme of the game.	UR_IMMERSION	Sounds and graphics shall fit the tone of the game and not take away from the immersion of the user or stick out against the rest

			of the game.
NFR_THEME	The theme of the game should appeal to a 16-20 year old student audience.	UR_TARGET_MARKET	The theme should be something a 16-20 year old will relate to and be interested in.
NFR_CODE_MODULARITY	The code should be modular and easily extendable, without making the program difficult to follow.	UR_MAINTAINABILITY	Classes should hold references to necessary related objects and no more.

Constraint Requirements

ID	Description
CR_SYSTEM_COMPATIBILITY	The game must successfully build for Windows, macOS and Linux.
CR_MINIMUM_SPEC	The game shall run on 64-bit desktop computers with a minimum of 2GB RAM and a dual core CPU.