Requirements

ENG1 Team 9
1. Jacob Dicken
2. Bertie Cartwright
3. William Croft
4. James Dovener
5. Henry Chan

This document outlines the procedure followed in acquiring and presenting the user and system requirements for the development of our project, UniSim, a single-player simulation game that allows the player to build their own university campus.

Requirements Elicitation & Negotiation

The requirements gathering process began with brainstorming sessions among team members, with the aim of aligning our creative visions with the goals outlined in the product brief. To facilitate this, we conducted an interview with our client, which was crucial for understanding the overall vision for the game, its key features, and user interactions.

We began the interview by asking the client for their **Single Statement of Need (SSON)** for the product, which helped us become acquainted with the overall goal of the project.

SSON: "I'm looking for a short, fun, jovial game for young adults to play, to enjoy seeing the positive aspects of working in computer science."

With this, we were able to note the intended audience and use of the game:

Target Audience: Young adults (16-21).

Intended Use: A fun, entertaining game with a focus on making computer science appealing to the player in a more casual setting.

The interview then involved our client providing valuable input on the game's style and perspective, the platform and performance, and the gameplay elements, allowing us to ensure that our requirements align with the expectations of our stakeholder. 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.
The negotiation phase involved us prioritising the requirements based on feasibility and importance, with the time constraint in mind. Our client's input was significant in shifting our focus towards creating a Minimum Viable Product (MVP) that delivered a simple yet engaging gameplay experience. For instance, while we were tempted by the idea of building multiple maps with varying difficulty levels, our client stressed the importance of starting with a single, well-designed map with predefined geography, to allow us to collect feedback before exploring more complex systems. Thus, this arrangement allowed us to effectively obtain and prioritise our requirements in a way that will guide us to build a game that will both meet the client's vision and be enjoyable for the players.

Requirements Presentation

We are presenting our user and system requirements in the form of structured tables that include the ID and description of each requirement. With this approach, we aim to create a clear roadmap for developing the game, allowing us to set goals and track our progress using this referencing system. The next steps will involve translating these requirements into the architecture and implementation phases.

User Requirements:

These requirements reflect the needs or expectations of the game's end-user. They focus on user interactions, outlining specific tasks that the players want to perform. The table below contains the IDs, descriptions and prioritisation of the user requirements of our game.

ID	Description	Priority
UR_MAP	The game should have a map with predefined characteristics including obstacles where buildings cannot be built.	High
UR_GAME_PERSPECTIVE	The game should have an isometric perspective which allows for greater visibility.	High
UR_GAME_ENJOYABILITY	The game should have a lighthearted, positive atmosphere and be engaging for first time and repeat users.	High
UR_BUILDING_PLACEMENT	The player must be able to place buildings with their being at least one of each type of building: teaching, accommodation, catering and at least two recreational buildings.	High
UR_BUILDING_MANAGER	The player should have access to a simple interface for managing buildings.	High
UR_GAME_CLOCK	The game should have a countdown timer displaying the remaining game duration which starts from 5 minutes.	High
UR_PAUSE_GAME	The player should be able to pause the game at any time during the 5-minute session.	High
UR_BUILDING_COUNTER	The game should display a simple building counter showing how many of each building have been placed so far.	Medium
UR_GAME_DIFFICULTY	The game should be simple enough for young adults to enjoy without needing advanced skills.	High
UR_MINIMUM_HARDWARE	The game should run comfortably on devices with the minimum specification to play the game which is: 2GB of RAM, and 2 or more CPU cores.	Medium
UR_EVENTS	The user shall be able to interact and react to events that occur during the course of the game there should be at least one positive, negative and neutral event.	High
UR_LEADERBOARD	The users should be able to see a leaderboard that displays the name and score of the top 5 players.	High
UR_NAME_INPUT	The user should be able to input their name to be seen on the leaderboard at the end of the game.	High
UR_SATISFACTION	The game should have a satisfaction score that the user should be able to increase and decrease in multiple ways including: placing more buildings; having certain types of buildings located near each other; and reacting to events accordingly.	High

ID	Description	Priority
UR_ACHIEVEMENTS	The user should be able to earn achievements over the course of the game by filling certain requirements. These achievements will positively or negatively affect the player's score at the end of the game.	High
UR_USABILITY	The game should be easy to control, use and understand for a first time user and user actions should be streamlined to avoid repetition.	Medium
UR_EXTENDABILITY	The game may be extendable which allows for future additions to the game like new building types.	Low

Functional Requirements:

These requirements describe what the game must do, detailing the functions and features that it must support. The table below contains the IDs and descriptions of the functional requirements of our game, as well as their link to the user requirements.

ID	Description	User Requirement(s)
FR_MAP	The game will provide a visual map for the user to traverse.	UR_MAP
FR_OBSTACLES	The map shall have preplaced obstacles that block the user from building on top of them	UR_MAP
FR_ISOMPETRIC_CAMERA	The game should provide a camera that is viewing the map from an isometric perspective	UR_GAME_PERSPECTIVE
FR_CAMERA_CONTROL	The camera should be controlled via the user's mouse with click and drag to move around the map and the scroll wheel being used to move in and out.	UR_USABILITY
FR_BUILDING_VARIETY	The game shall have at least one building of every type namely: teaching, accommodation, catering and at least two recreational buildings	UR_BUILDING_PLACEMENT
FR_BUILDING_SELECT	The game will allow the users to select buildings from the building manager menu.	UR_BUILDING_PLACEMENT UR_BUILDING_MANAGER
FR_BUILDING_PLACEMENT	The game will allow a user to place the selected building only in valid places e.g inside the map and not on an obstacle or building.	UR_BUILDING_PLACEMENT
FR_BUILDING_MANAGER	The game shall provide UI and controls to manage buildings.	UR_BUILDING_MANAGER
FR_GAME_PAUSE	The game will be pausable on the player's request.	UR_GAME_PAUSE
FR_GAME_PAUSE_EFFECT	While the game is paused the timer will no longer decrement.	UR_GAME_PAUSE
FR_TIMER_COUNTDOWN	The timer will decrement at a rate of 1 second for every real world second	UR_GAME_END
FR_TIMER_START	The timer will start at a time of 5 minutes and begin decrementing as soon as the (play button is pressed/the game begins).	UR_GAME_END
FR_GAME_END	The game shall terminate and end the game when the 5 minute timer is up.	UR_GAME_END
FR_BUILDING_COUNTER	Display a counter to show the player the number of each type of building placed.	UR_BUILDING_COUNTER
FR_SATISFACTION_COUNTER	The game should display the satisfaction as a percentage to the user.	UR_SATISFACTION
FR_SATISFACTION_BUILDING_DISTANCE	The satisfaction number should go up and down depending on how close buildings of certain types are together. E.g the	UR_SATISFACTION

ID	Description	User Requirement(s)
	satisfaction number will go up if an accommodation building is nearer to a catering building.	
FR_BUILDING_RATIOS	The satisfaction number should go up and down depending on the ratio of different building types. E.g the satisfaction will go down if there are too many accommodation buildings and not enough catering ones.	UR_SATISFACTION
FR_EVENT_TYPES	Events should come in 3 different types: positive(Which benefits the player), negative(Which hinders the player) and neutral(Which does not affect the player).	UR_EVENTS
FR_EVENT_FREQUENCY	Over the course of the game at least 3 events should occur.	UR_EVENTS
FR_LEADERBOARD	A leaderboard containing the names and scores of the top five players should be visible to players.	UR_LEADERBOARD
FR_ADD_TO_BOARD	If the current player reaches a score that is higher than the fifth highest player on the leaderboard, their name and score will instead be featured on the leaderboard and the current fifth highest player will be removed.	UR_LEADERBOARD
FR_NAME_INPUT	At the end of the game the user will be requested to input their name to be shown on the leaderboard. This will be done using character input.	UR_NAME_INPUT UR_LEADERBOARD
FR_ACHIEVEMENT_EARN	The player will be able to earn achievements throughout the game.	UR_ACHIEVEMENTS
FR_ACHIEVEMENT_CRITERIA	The game will check each achievement's criteria and if the criteria for a specific achievement is met the player will earn that achievement.	UR_ACHIEVEMENTS
FR_ACHIEVEMENT_NOTIFICATION	The game will notify the player when they earn a specific achievement.	UR_ACHIEVEMENTS
FR_ACHIEVEMENT_REWARDS	Achievements should affect the players final score in a positive way.	UR_ACHIEVEMENTS
FR_USER_INTERFACE	The UI should display only important information to the user so that it is not overly complex and overwhelming.	UR_USABILITY
FR_INTERACTIVE_ELEMENTS	All Interactive elements will react when clicked or moved (for example when a button is pressed its colour changes or a sound is made).	UR_USABILITY

Non-Functional Requirements:

These requirements describe how the game performs a task rather than what it should do. They ensure that the game meets certain standards of performance, usability, and reliability. The table below contains the IDs, descriptions, links to the user requirements and the fit criteria of the non-functional requirements of our game.

ID	Description	User Requirement(s)	Fit Criteria
NFR_GAME_ENJOYABILITY	The Game should maintain a positive, lighthearted atmosphere.	UR_GAME_ENJOYABILITY	Response to the game's atmosphere should be at least 70% positive.
NFR_GAME_ENGAGEMENT	The game should be engaging to first time and repeat users.	UR_GAME_ENJOYABILITY	Response from first time and repeat users on their engagement with the game should be at least 70% positive.
NFR_GAME_DIFFICULTY	The game should be accessible and straightforward, without requiring advanced skills.	UR_GAME_DIFFICULTY UR_USABILITY	The average satisfaction score of a new player when they end the game should be above 40%.
NFR_MINIMUM_HARDWARE	On machines with the minimum requirements the game should run smoothly, at a framerate of around 60, without hitches at any point during gameplay.	UR_MINUMUM_HARDWARE	During performance testing the 60-second average frame-rate should exceed 58. The lowest 1% of frame times should also be above 50.
NFR_INTERACTIVE_ELEMENTS	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_USABILITY	Interactive elements must react within <1 second of being used.
NFR_OPERABILITY	The game should be easily playable and navigable by new players.	UR_USABILITY	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_EXTENDABILITY	The documentation should be understood by users who have no prior knowledge of the code.
NFR_CODE_MODULARITY	The code should be modular and easily extendable, without making the program difficult to follow.	UR_EXTENDABILITY	Classes should hold references to necessary related objects and no more.