Requirements 1

Cohort 1, Group 6

Group Members:

Hussain Alhabib

Ellen Matthews

Minnie Poon

Jason Ruan

Daniel Smith

Owen Smith

Eliciting our requirements

To facilitate the implementation of our project, we elicited requirements directly from our client via a 20-minute in-person interview. The interview, which was recorded and conducted by two group members, followed an interview schedule that had been collaboratively worked on by all team members.

This structured approach to the first client interview allowed us to generate a comprehensive document containing the client's answers to over 20 questions, which helped assist the team later on when producing the statement of requirements and respective referencing system.

(**Assessment 2**) - To further elicit requirements we held a group meeting where we reviewed the project we were picking up and compared it with the updated client brief to identify where we'd need to add further functionality. We did not feel that another client meeting would be beneficial at this stage as it was unlikely for us to gain any further insight towards what we should do next.

Presenting our requirements

In the following group meeting, we began to interpret the client's responses and produce our requirement referencing system. After researching and discussing best practices in requirements engineering as a group, particularly in Ian Summerville's *Software Engineering 10th Edition*, we decided to present the requirements in a tabular format, splitting them into four tables based on the previously established categories. Each table was formatted with columns for the requirement ID, description, priority, requirement link, and current status.

We considered several other formats for the referencing system, including Natural Language, Structured, and Use Case (UML) specifications. However, we found a Natural Language specification lacked the clarity and detail needed for our primary stakeholders, despite being accessible for wider less-technical audiences. The Structured specification was too complex given our project's scope, and while a UML specification seemed useful especially due to its more visual aspects it seemed redundant as our architecture section would already include graphical elements. Ultimately, we felt as a team that the tabular format with an Agile-inspired approach, that prioritised requirements based on importance, best suited our needs, enabling us to collectively determine the order and focus of requirement implementation.

(**Assessment 2**) - We found it convenient to continue with the previously used method of presenting requirements for the sake of convenience and saving time.

Negotiating our requirements

To ensure the project aligned with the client's vision, we shared drafts of the requirements document to encourage feedback and revisions. This approach enabled us to refine the requirements with the client's input and approval, which ensured that our understanding and implementation of the game matched the client's expectations. After acting on feedback, we emailed the client our final draft of the requirements, which they approved before we proceeded with designing our requirements referencing system.

Our Single Statement of Need

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 **as well as achievements which can be earned for meeting certain requirements** while navigating building restrictions and random events within a 5-minute gameplay session, **with the best scores being presented as a top 5 leaderboard once the game finishes**.

An Introduction to Our Requirement Categories

We defined our initial client questions and their respective requirements according to the following four categories:

1. **User Requirements** describe the main actions and experiences that users must be able to experience during gameplay. These must avoid using technical jargon and be accessible to everyone regardless of their technical skills.
2. **System Requirements** define how the system will meet the user's needs. These include detailed, often technical, descriptions of the games functionality and services required for gameplay. Multiple system requirements can be written to fulfil a single user requirement.
3. **Non-Functional Requirements** focus on the game's attributes and qualities, and define how well the game must perform regarding usability, reliability, and performance. 4. **Constraint Requirements** address limitations or restrictions that may be imposed on the game, such as hardware limitations and any relevant legal regulations we must adhere to.

Our Requirements Referencing System

Each requirement is assigned a unique ID to ensure traceability and consistency across our documentation. Initially, we used numerical IDs, but switched to more meaningful names to improve readability and clarity, making it easier to reference requirements in other documents. This ID system also helps us link related requirements, such as ensuring each system requirement is tied to a corresponding user requirement. A dedicated column in each table lists related requirement IDs, which further helps in visualising how different requirements interact and impact one another.

User Requirements (URs)

ID	**Description**	**Priority**	**SR Link**	**Status**
UR_BASIC_ BUILDINGS	The player must be able to place at least one type of each building type; a place to learn, a place to sleep, a place to eat, and a recreational activity.	Essential	SR_PLACE_ BUILDINGS	APPROVED
UR_TIME	The game must last for a maximum of 5 real-world minutes.	Essential	SR_TIME	APPROVED
UR_BUILDING_COUNTE R	The game must track and display the number of each building type (e.g. sleep, eat, learn, recreational) placed by the player.	Essential	SR_BUILDING_ COUNTER	APPROVED
UR_ LEADERBOARD	The user must be able to view a leaderboard of the top 5 scores and the names of the people with those scores. If the user scores a top 5 score, they must be asked to input their name to be added to the leaderboard.	Essential	SR_LEADERBOARD	APPROVED
UR_ ACHIEVEMENTS	The user must be able to earn achievements whilst playing the game if they meet certain conditions. These achievements could influence the final score of the player, either positively or negatively.	Essential	SR_ACHIEVEMENTS	APPROVED
UR_SCORE	Players must have a way to measure their success in the game via various metrics such as satisfaction and environmental impact.	High	SR_METRICS SR_SATISFACTION	APPROVED
UR_EASE_OF_USE	There must be an intuitive interface with visual indicators for performance (e.g. student satisfaction, building usage).	High	SR_METRICS	APPROVED
UR_BUILDING_LIMITS	The player must be restricted from placing buildings in certain areas of the map (e.g. over a lake, road, or other buildings).	High	SR_BUILDING_ RESTRICTIONS	APPROVED

UR_EVENTS	The game will include at least three core events that affect the player’s experience and require player interaction.	High	SR_EVENTS	APPROVED
UR_DEPLOYMENT	The game must include tips and guidance to help players understand how to play the game, such as tutorials or hints.	Medium	SR_TIPS NFR_ACCESSIBLE	APPROVED
UR_TIPS	The game must include tips and guidance to help players understand how to play the game, such as tutorials or hints.	Medium	SR_TIPS NFR_ACCESSIBLE	APPROVED
UR_SETTINGS	The game must include settings to allow the user to adjust in-game sound levels if sound assets are implemented.	Medium	SR_SETTINGS	APPROVED

System Requirements (SRs)

ID	**Description**	**Priority**	**SR Link**	**Status**
SR_PLACE_BUILDINGS	Players must be able to place, upgrade, and demolish buildings within the game.	Essential	UR_BASIC_BUILDINGS	APPROVED
SR_TIME	Time must be tracked and shown within gameplay, lasting for a maximum of 5 real-world minutes.	Essential	UR_TIME	APPROVED
SR_BUILDING_COUNTER	The game must count and display the number of each building type placed by the player.	Essential	UR_BUILDING_COUNTER	APPROVED
SR_LEADERBOARD	The game must include a leaderboard, which stores and displays the top 5 scores with a name connected to each score. If a user’s final score places within the top 5 they should be asked to input their name to be added to the leaderboard.	Essential	UR_LEADERBOARD	APPROVED
SR_ACHIEVEMENTS	The game must feature achievements which can be earned in a playthrough with each achievement having a criteria to unlock. The game must keep track of unlocked achievements to influence score and to ensure the same achievement is not earned multiple times.	Essential	UR_ACHIEVEMENTS	APPROVED

SR_EVENTS	The game must include core events (e.g. strikes or fires) that require player action and occur randomly during the game.	High	UR_EVENTS	APPROVED
SR_BUILDING_ RESTRICTIONS	The map must restrict building placement based on rules (e.g. no buildings over lakes/rivers or on existing paths).	High	UR_BUILDING_ LIMITS	APPROVED
SR_METRICS	Satisfaction metrics must be visible to players at all times during gameplay.	High	UR_SCORE UR_EASE_OF_USE	APPROVED
SR_DEPLOYMENT	The game must run smoothly on desktops and laptops across all major operating systems optimised for various hardware.	High	UR_DEPLOYMENT	APPROVED
SR_SATISFACTION	Building proximity and events must affect pla yer satisfaction, which must be visible in-game.	Medium	UR_SCORE	APPROVED
SR_BUILDIN G_EFFECTS	Buildings must have different impacts on player satisfaction based on their type and position in relation to other buildings.	Medium	UR_SCORE	APPROVED
SR_SETTING S	The game must include sound settings for adjusting in-game sound levels, if such assets are included.	Medium	UR_SETTINGS	APPROVED
SR_DIFFICULTY	The game must offer various difficulty levels to accommodate a broad audience but must include a baseline level for everyone.	Low	UR_TIPS	APPROVED

Non-Functional Requirements (NFRs)

ID	Description	Priority	SR Link	Status
NFR_RUNS_WELL	The game must run smoothly on all laptops and desktops on major operating systems.	High	UR_DEPLOYMENT	APPROVED
NFR_NO_DELAY	The gameplay experience must be smooth, with minimal delays or lag during player interactions.	High	UR_DEPLOYMENT	APPROVED
NFR_ACCESSIBLE	The game must be accessible to as wide an audience as possible, accommodating players with different needs.	High	UR_TIPS	APPROVED

NFR_GRAPHICS	The graphics must reflect the selected environment, maintaining visual cohesion throughout the game.	High	UR_BASIC_ BUILDINGS	APPROVED
NFR_LICENSES	The game must use appropriately licensed sounds and music assets to create an enjoyable in–game experience.	Medium	UR_SETTINGS	APPROVED
NFR_BACKGROUND	The game must include background elements (e.g. students walking) to make the map appear more dynamic and engaging.	Low	UR_BASIC_ BUILDINGS	APPROVED

Constraint Requirements (CRs)

ID	**Description**	**Priority**	**SR Link**	**Status**
CR_LOCAL	The game must run locally on a device without needing an internet connection	High	UR_DEPLOYMENT	APPROVED
CR_LOW_SPEC	The game must be optimised to run on low-spec devices, ensuring it is accessible to all players.	High	UR_DEPLOYMENT	APPROVED
CR_LEGAL	The game must be legally compliant, using appropriately licensed and attributed assets.	High	UR_SETTINGS UR_ BASIC_ BUILDINGS	APPROVED