Date: 13/01/25

UniTycoon

REQUIREMENTS

Group 6

TEAM6 Game Studios

Work Undertaken by:

Thomas Koukouris
Adam Khan
Oliver Herron
Sam Jordan
Nathan Hopper
Fergus Irvine

Single Statement of Need

"A fast-paced, enjoyable university simulator, where young adults can build and manage their own university campus, and react to the different events throughout the game"

Introduction

To elicit requirements for our project, we first needed to carefully engineer questions for our customer (stakeholder), such that their answers gave us enough detailed information to decide what user, functional and non-functional requirements we needed to meet both their expectations and desires for what the game will look like, and how it will play.

The first step in this process was to read the product brief together, discussing which aspects we thought were clear, and which aspects we thought needed clarification. We also spent time thinking about less obvious questions, stemming not from the brief, but from general context and our knowledge of apps and games, such as whether the customer wanted the game to be viewed as an isometric projection, a top-down view, or some combination of the two. We made sure we had questions about every aspect of the game.

We also used these questions to elicit the SSON we have above, as it's useful to have this clear and concise goal for the game. This allows us to have an overarching thought process to refer back to when making decisions, if a feature we're considering/implementing seems to conflict with this statement, we know it needs to be reworked/refined.

Once we had our initial questions planned out, we booked a meeting with our customer, and all went away thinking of any extra questions or changes to make. We met before the meeting to finalize our questions, and to ensure that we had everything covered. During the meeting, we chose not to record any audio, but instead to have one team member write the customer's response to each question in a document, and the rest of the team focused on writing down the extra details within their answers. Since many questions we asked didn't simply elicit a yes or no answer, we gathered a significant amount of extra details about other aspects related to the feature the question pertained to. Once we had all our notes from the meeting, we combined all our separate notes into one document.

This process ensured that we asked a wide variety of questions, encompassing the entire scope of the project, and ascertaining a complete description of what the customer wanted from the game. From this, we were easily able to turn a combination of the initial product brief, and the responses we got from the customer, into our user requirements below.

We now had our user requirements, each of which using an ID with the UR prefix and a name that relates clearly to the description. As well as a priority, Shall: it must be implemented, Should: we hope to implement it, but it may have some risk (such as the time it would take to implement) and May: it's subject to revisions due to time constraints or responses from play-testing. We then mapped our user requirements to functional and non-functional requirements (FR and NFR prefixes respectively), using the same naming conventions for both. We also assigned a "fit criteria" to the non-functional requirements, describing a minimum expectation for implementation to meet the requirement.

USER REQUIREMENTS → “What the user wants/expects”

ID	Description	Priority
UR_TIMING	The user shall be given 5 minutes of live gameplay, where they can allow events and campus life to take place. There shall be some relation between in-game time passing and real world years (for example, 1 minute per academic year).	Shall
UR_PAUSE	The user will be able to pause the timer and live game action, and when certain in-game things occur (events/ certain dates).	Shall
UR_YEARLY_REPORTS	At the end of each real world year, the player should be shown a report of the year’s statistics - this should have a graph showing the satisfaction over the course of the year and the campus’ ranking against other fictional campuses.	May
UR_END_OF_GAME_REPORT	The user shall be given a report of how they performed at the end of the game.	Should
UR_EVENTS	The user shall be given randomly occurring and planned events to make the game replayable. The user may or may not have to react to these to improve their score.	Shall
UR_MAP	The user shall be able to play on at least one preset map.	Shall
UR_TERRAIN	The user shall be restricted in where they can place buildings based on the map terrain types.	Shall
UR_ADD_BUILDINGS	The user can place buildings. This takes a variable amount of in-game time. A minimum number of each building type shall be offered to the player for placement: learning: 1, accommodation: 1, recreation: 2, eating: 1.	Shall
UR_REMOVE_BUILDINGS	The user may be able to remove/destroy buildings.	May
UR_JOINABLE_BUILDINGS	The user may be able to join buildings together, when placed next to each other, granting them combined benefits of both buildings and improving the satisfaction score.	May
UR_UPGRADES	The user can do certain tasks to gain access to upgrades that will make the experience more fun and improve their satisfaction score.	May
UR_REPLAY	The user shall be able to replay the game.	Shall
UR_BUILDING_COUNTERS	The user shall be able to see a count of how many of each type of building they have placed.	Shall
UR_SATISFACTION_SCORE	The player will be able to see their student satisfaction score in the game UI at all times. Student satisfaction shall be tracked and used as a score. The user’s reactions to events and building placement affects this	Shall

	score.	
UR_SOUND	The player may have the option to listen to background music while playing the game. Also, different sound	May
UR_UX	The player shall expect to play the game with no bugs or crashes. The game shall provide a pleasant experience. The game shall be intuitive to play, ensuring that the player can easily understand and navigate the interface with minimal explanations.	Shall
UR_LEADERBOARD	The user shall be able to see a leaderboard with the name and score of the top 5 scores.	Shall
UR_ACHIEVEMENTS	The user shall be shown achievements for milestones they have reached in the game. Some examples are: - maintaining a student satisfaction of over 80% for 3 straight minutes (I heart uni), - placing a total of 5 buildings during the game (minimalist), - for placing the maximum number of buildings possible over the duration of the game (jam packed).	Shall

SYSTEM REQUIREMENTS

- **FUNCTIONAL REQUIREMENTS** → "What the system does to meet what the user wants"

ID	Description	User Requirements
FR_PAUSE	The game will start paused. While the game is paused, all game mechanics shall also be paused.	UR_PAUSE
FR_TIMING	The system shall initiate a 5-minute timer when a game starts. It should also pause when the game is paused.	UR_TIMING
FR_TILEABLE_MAP	The system shall allow the user to place buildings on a tileable grid system.	UR_MAP
FR_TERRAIN	The system shall implement distinctly different map terrains such as lakes and hills.	UR_TERRAIN
FR_CONSTRUCTION	The system shall permit/forbid building placement based on map terrains.	UR_ADD_BUILDINGS
FR_DESTRUCTION	The system may be able to remove buildings, updating the map, altering the satisfaction growth rate proportionally to the removed building's effect and taking a varying amount of in-game time.	UR_REMOVE_BUILDINGS
FR_STUDENT_SATISFACTION_BUILDINGS	The system should alter student satisfaction score based on proximity of buildings, building types and amount, building upgrades and	UR_SATISFACTION_SCORE

	maintenance.	
FR_STUDENT_SATISFACTION_EVENTS	The system should alter student satisfaction based on user event handling.	UR_SATISFACTION_SCORE
FR_STUDENT_VISUAL	Animated students may be displayed on the map to indicate score.	UR_SATISFACTION_SCORE
FR_SCORE	Student satisfaction shall be tracked and used as a score. The system should keep a record of all actions that will impact the user's score.	UR_SCORE
FR_RANDOM_EVENTS	The system should activate random events that happen at the same in-game time, every game.	UR_EVENTS
FR_PLANNED_EVENTS	The system should activate planned events based on environmental factors and the user's actions to previous events.	UR_EVENTS
FR_SOUND	The system may provide background music options and various soundtracks driven by in-game events.	UR_SOUND
FR_BUILDING_COUNTERS	The system should display on screen the number of each type of building placed.	UR_BUILDING_COUNTERS
FR_LEADERBOARD	The system shall save the player's name and score and compute and display the top 5 satisfaction scores achieved in the game along with the player name that achieved it	UR_LEADERBOARD
FR_ACHIEVEMENTS	The system shall have a list of achievements that are checked constantly during the game and if true (meaning they have been met) the system shall display feedback to the user in some specific format.	UR_ACHIEVEMENTS

- **NON-FUNCTIONAL REQUIREMENTS** → "How the system does it"

ID	DESCRIPTION	USER REQUIREMENTS	FIT CRITERIA
NFR_DISPLAY	The system shall be designed for a typical laptop/PC display.	UR_UX	Be playable on at least a 1080p, 16:9 screen.
NFR_ISOMETRIC	The system should be designed in an isometric style.	UR_UX	Placeable buildings in isometric style.
NFR_RELIABILITY	The game should not crash during normal gameplay and be reliable.	UR_UX	The game should not crash during at least 3 play tests.

NFR_TUTORIAL	The game may provide a short tutorial or hover over explanations to attempt at explaining more complicated aspects of the game.	UR_UX	The player shall have the option to check the game mechanic explanations as many times as they want.
NFR_ACCESSIBILITY	The game shall be designed in an accessible way.	UR_UX	Compatible with mouse and keyboard. Not rely on colour. Text font shall be large enough.
NFR_COMPATIBILITY	The game shall be compatible with the latest operating systems.	UR_UX	Windows 10+ and macOS Ventura+
NFR_SYS_REPONSE _SPEED	The system shall provide instant feedback on button presses.	UR_UX	In <0.1s from time of input.