Requirements

Group 9

Group name: KOWLAAB

Aiden Sayer
Lurvïsh Polodoo
Ben Salkield
Oliver Rogers
Alex Sharman
William Roebuck
Kiyan Eiles

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 finalise 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

ID	Description	Priority
UR_SCORE	Student satisfaction shall be tracked and used as a score. The user's reactions to events and building placement affects this score.	Shall
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).	Should
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.	Should
UR_END_OF_GAME_REPORT	The user shall be given a report of how they performed over the course of the game, showing how this compares to other fictional UniSims, and/or their previous attempts.	Shall
UR_EVENTS	The user shall be given randomly occurring events to make the game replayable. The user may or may not have to react to these to improve their score.	Shall
UR_DEPENDENT_EVENTS	Events should be dependent on the environment factors. i.e. upgrading/closing a building.	Should
UR_PERFORMANCE	The user shall be able to run the game on an average desktop or laptop.	Shall
UR_PLAYABILITY	The user shall be able to play the game in an accessible way, using a mouse.	Shall
UR_ISOMETRIC	The user's perspective should be in an isometric style. Especially with regard to the placeable buildings.	Should
UR_MAP	The user shall be able to play on at least one preset map, using a tileable grid system, that has map features that act as restrictions for the user.	Shall
UR_BUILDINGS	The user can place and remove buildings. This takes a variable amount of in-game time	Shall
UR_BUILDING_LOCATION	Placing buildings in different locations affects their effectiveness. For example, placing a social area near a lake will result in higher satisfaction.	Shall
UR_BUILDING_VARIANTS	The user should be able to place buildings with the same purpose (e.g. "accommodation"), with different statistics (e.g. size, building time, quality, design, number of rooms).	Should

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_JOVIAL	The user shall be able to have fun whilst playing the game, and easily be able to replay it and pick it back up again.	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

Functional Requirements

ID	Description	User Requirements
FR_PAUSE	The system should allow all the game mechanics to be accessed while paused(although construction is also paused). A pause will have no time limit. At the beginning, the game will start paused. The game pauses at the end of each semester or year (decided which plays better through later testing). There will be a limited number of pauses, outside of set pauses, tracked by the system.	UR_PAUSE
FR_TIMING	The system will allow the user to be able play live gameplay for a maximum of 5 minutes.	UR_TIMING
FR_REALW_TIME	There shall be some relation between in-game time passing and real world years (for example, 1 minute per academic year).	UR_TIMING
FR_INPUT	The game shall be played using a mouse.	UR_PLAYABILITY
FR_ACCESSIBILITY	The game shall be designed in an accessible way, and should therefore not rely on colour or small text.	UR_PLAYABILITY
FR_TILEABLE	The system shall allow the user to place buildings on a tileable grid system.	UR_MAP
FR_MAP_FEATURES	The map should have different terrain features, such as lakes and hills that restrict building.	UR_MAP
FR_CONSTRUCTION	Buildings can be built by the user, however restricted based on terrain features and student satisfaction.	UR_BUILDINGS
FR_DESTRUCTION	Buildings can be removed, taking a varying amount of in-game time.	UR_BUILDINGS
FR_BUILDING_TY	The system should have at least: a lecture building,	UR_BUILDING_

PES	accommodation building, two recreational buildings, cafeteria, leisure building, fitness centre, reception building.	VARIANTS
FR_BUILDING_RELATIONSHIP	The system should determine student satisfaction based on the relationship of nearby buildings and features.	UR_JOINABLE_BUILDINGS
FR_SATISFACTION_ALGORITHM	The distance between buildings has an effect on student satisfaction. The magnitude of this is dependent on the building type.	UR_BUILDING_LOCATION
FR_STUDENT_VISUAL	Animated students may be displayed on the map to indicate score.	UR_SCORE
FR_SCORE	The system should keep a record of all actions that will impact the user's score.	UR_SCORE
FR_PLANNED_EVENTS	The system should activate events that happen at the same in-game time, every game.	UR_EVENTS
FR_DEPENDENT_EVENTS	The system should activate events based on environmental factors and the user's actions.	UR_DEPENDENT_EVENTS
FR_MUSIC	The system should provide various soundtracks driven by in-game events.	UR_JOVIAL
FR_UPGRADES	The system should provide tasks that the user can complete to obtain user specific upgrades.	UR_UPGRADES
FR_BUILDING_COUNTERS	The system should display on screen the number of each type of building placed.	UR_BUILDING_COUNTERS

Non-Functional Requirements

ID	**Description**	**User Requirements**	**Fit Criteria**
NFR_DISPLAY	The system shall be designed for a typical laptop display.	UR_PLAYABILITY	Be playable on at least a 1080p, 16:9 screen.
NFR_ISOMETRIC	The system should be designed in an isometric style.	UR_ISOMETRIC	Placeable buildings in isometric style.
NFR_RELIABILITY	The game should not crash during normal gameplay and be reliable.	UR_PERFORMANCE	The game should not crash during at least 3 play tests.
NFR_INTUITIVE	The game shall be designed in an intuitive and easy to understand way.	UR_JOVIAL	Play testers are not overwhelmed on their first playthrough and are able to complete the playthrough.