Requirements

Oliver Dixon
George Cranton
Praj Dethekar
Denys Sova
Shivan Ramharry
Albara Shoukri
Rafael Duarte

1.1 Requirements Elicitation

The requirements presented in this document have been elicited and negotiated through a collaborative process involving the customer and the members of the development team.

In order to ensure that our game meets the needs of the client, we took an iterative approach to requirement elicitation in which the team held a weekly meeting to discuss and document existing and changing requirements. As the project progressed, we adapted the priority of certain requirements to guarantee that our deliverable meets the expectations of the stakeholder.

Alongside these regular discussions, we consulted the client - both in arranged interviews and weekly practical sessions - to determine their specific requirements and personal preferences for design and implementation choices.

From these sessions we developed our Single Statement of Need (SSON): a clear, concise statement that details the overall goal of the system. It is presented in section 2.1 of this document.

We explored and expanded on this statement to determine our user requirements. Some of these requirements, such as UR_TIME_SCALE and UR_PLAYER_SCORE were elicited from both our own research and discussions with the client. Negotiation played a crucial role in determining the priority of each requirement. By engaging the client in an open discussion about constraints and conditions of each user requirement, we were able to define exact scales, values and priorities.

Each user requirement was then broken down into a series of related functional system requirements, presented in the tables below. These describe the functions that the system must perform in order to meet the user requirements. The third set of requirements are non-functional, defining the necessary characteristics of the system and providing measurable criteria for assessing their attainment.

1.2 Requirements Presentation

The Single Statement of Need (SSON) for our project is presented first in section 2.1 of this document.

We have chosen to utilise three tables (2.2-2.4) for the presentation of our elicited requirements. This approach to requirements specification and presentation incorporates distinct and meaningful identifiers that denote the type of requirement (UR for user requirement, FR for functional requirement and NFR for non-functional requirement) along with a keyword indicating the relevant domain. The tables also include supplementary details about each requirement, such as its designated priority, or dependency on other requirements.

Throughout the project, we have consistently updated the requirements tables with new concepts that emerged from client and group meetings. In instances where an entry has been surpassed by a new requirement, the original entry is labelled “_SUPERCEEDED” to indicate that subsequently introduced requirements with the same name have a higher priority.

2.1 Single Statement of Need

"Develop an educational but entertaining single-player survival game that simulates the experience of a second-year computer science student navigating the final week before their exams begin, emphasising the importance of time management and well-being."

2.2. User Requirements

ID	Description	Priority
UR_MOVEMENT	The player is able to move the avatar around a 2D map.	Shall
UR_INTERACTION	The player is able to make the avatar interact with buildings.	Shall
UR_INTERFACE	The interface of the game is simple and displays all the necessary information clearly.	Shall
UR_STUDYING	There is a building that the avatar can interact with to study.	Shall
UR_RECREATION	There is a building that the avatar can interact with to recreate.	Shall
UR_SLEEPING	There is a building that the avatar can interact with to sleep.	Shall
UR_EATING	There is a building that the avatar can interact with to eat.	Shall
UR_CONTROLS	The game's controls are intuitive and are clearly presented to the player.	Shall
UR_ACCESSIBILITY	The game is for new players, so it is easy to understand and play with no prior experience.	Shall
UR_OBJECTIVE	The objective of the game is presented clearly.	Should
UR_GAME_LENGTH*	The game takes approximately 5-6 minutes to play.	Should
UR_GAME_OVER*	A 'game over' screen is displayed if the player performs poorly.	Should
UR_PLAYER_SCORE*	A player will have a score that is incremented each day based on the activities done on that day.	Should
UR_MAP_DESIGN	The map is a simplified version of the campus with space between each building.	Should
UR_CAMPUS_BUILDINGS	It is obvious which activity can be completed in each building.	Should
UR_NAME	The player's character should have a name	Should
UR_LEADERBOARD	The player should be able to see the leaderboard at the end of the game.	Should
UR_ACHIEVEMENTS	The player should be able to get different achievements by doing specific tasks.	Should
UR_PAUSE	The player should be able to pause the game and resume it later	Should
UR_TIME_SCALE	One minute of real time is equal to one day in-game.	Should
UR_BACKSTORY	Information about the context of the game is presented briefly to the	Could

	player at the start of the game.	
UR_CHARACTER	The player is able to change the look of their avatar at the start of the game.	Could
UR_TIME_SKIP	The time will skip when the player proceeds to an activity.	Could
UR_AUDIO	Music is relaxed and creates a study environment. There are sound effects for various actions in the game.	Could
UR_ENDING	A positive and constructive message is displayed at the end of the game, regardless of the player's score.	Could
UR_MINI_GAME	A quick and fun mini game for the user to feel like they are doing a task when studying, and recreating in the game.	Could

** These entries were discussed in our client meeting, but are not required in Assessment 1 according to the Product Brief.*

2.3. Functional Software Requirements

ID	Description	User Requirements
FR_START_GAME	When the game is first opened, the player is presented with a screen detailing the objective, controls and context. The player can press a START GAME button that begins the game.	UR_OBJECTIVE UR_CONTROLS UR_BACKSTORY
FR_FULLSCREEN	The game has a full-screen display on any window size.	UR_INTERFACE
FR_START_SCREEN	The starting screen has options to view the controls, adjust settings, or start the game.	UR_INTERFACE
FR_SETTINGS_SCREEN	The settings screen allows the user to adjust the music volume and choose a character.	UR_INTERFACE UR_AUDIO UR_CHARACTER
FR_CONTROLS_SCREEN	The controls screen contains instructions for how to play the game, and some backstory about the context of the game.	UR_INTERFACE UR_BACKSTORY UR_ACCESSIBILITY
FR_MOVEMENT_KEYS	The player uses the WASD keys to move the avatar around the map.	UR_MOVEMENT UR_CONTROLS
FR_MOVEMENT_ARROWS	The player uses arrow keys to move the avatar around the map.	UR_MOVEMENT UR_CONTROLS
FR_INTERACTION_TRIGGER	When the avatar collides with a building's door/entrance area, an interaction menu appears on the screen.	UR_INTERACTION UR_CONTROLS
FR_INTERACTION_MENU	The interaction menu tells the player what actions can be completed in that building. It can be navigated with the arrow keys or mouse.	UR_INTERACTION UR_CONTROLS
FR_COMPLETE_ACTION	From an interaction menu, the user can select an activity to complete. In assessment 1, this is displayed in a simple counter.	UR_STUDYING UR_RECREATION UR_SLEEPING

	When this happens, energy is lost and time progresses.	UR_EATING
FR_DISPLAY_ENERGY	The amount of energy remaining is displayed clearly on the screen as a coloured bar.	UR_INTERFACE
FR_DISPLAY_TIME	The day (out of 7) and time is displayed clearly on the screen in the format HH:MM.	UR_INTERFACE
FR_TIME	Every 10 seconds, the in-game clock will be incremented by 30 minutes.	UR_GAME_LENGTH UR_TIME_SKIP
FR_SAVE_PROGRESS	The game is not currently required to have a ‘save progress’ feature.	UR_OBJECTIVE
FR_SCORING*	The score is incremented by a base value of 15 after each day. Depending on the amount of times an activity is done, a multiplier is used on the score to reward or punish the player.	UR_PLAYER_SCORE
FR_STUDY_MULTIPLIER	The study multiplier is applied to the score at the end of the day. The multiplier is increased by 0.5 for every time the player studies in one day. It resets every day back to 1.	UR_PLAYER_SCORE
FR_REC_MULTIPLIER	The rec multiplier is applied to the score at the end of the day. The multiplier is increased by 0.5 for every time the player studies in one day. It resets every day back to 1.	UR_PLAYER_SCORE
FR_LEADERBOARD	The leaderboard is visible to the player at the end of the game and it shows the top 10 scores with the names of the characters that achieved those scores.	UR_PLAYER_SCORE UR_LEADERBOARD UR_NAME
FR_ACHIEVEMENTS	The achievements are hidden from the player. When they finish the game they will be able to see which achievements they got. The achievements gained must impact the final score of the player.	UR_PLAYER_SCORE UR_ACHIEVEMENTS
FR_PAUSE_ACTION	At any time, while walking around campus, the player is able to pause the game by clicking ESC. This will take them back to the main menu.	UR_PAUSE UR_INTERFACE
FR_END	The game ends when 7 in-game days have passed. In assessment 1, this is communicated to the player using an ‘end game’ screen.	UR_GAME_OVER
FR_LOSE_CONDITION	A ‘game over’ screen is displayed if the player performs poorly.	All user requirements
FR_WIN_CONDITION	The game is won when the user has reached their quota	All user requirements

2.4. Non-Functional Software Requirements

ID	Description	Fit Criteria
NFR_JAVA_VERSION	System is programmed using Java version	The system is fully programmed in Java

	11.	version 11,
NFR_SYSTEM_SIZE	The size of the end system is manageable such that it can be run on various devices.	The program executes in under 5 seconds and does not experience any memory-related crashes.
NFR_COMPATIBILITY	The system runs on Linux, Windows and MacOS.	The system has full functionality on both Linux and Windows operating systems. The system has a minimum of 75% functionality on MacOS systems.
NFR_PERFORMANCE	The system appears to respond immediately to user input.	Keypresses are registered and acted upon within 10 milliseconds of input.
NFR_SCALABILITY	The system supports further development by another team.	Each class is properly encapsulated with minimal dependencies on other classes.
NFR_MAINTAINABILITY	The system is well-documented following relevant coding standards.	All classes and functions are accompanied by docstrings and necessary documentation.
NFR_RELIABILITY	The system does not suffer from failures that affect player experience.	The mean time between failures is at least 50 hours.
NFR_COMPLIANCE	The system conforms to the current industry regulated standards relevant to its domain.	Regulatory information is documented.
NFR_EFFICIENCY	The system minimises CPU and resource use while running to optimise performance.	The average response time for all user interactions is less than 10 milliseconds under typical operating conditions.