Requirements

Group 12

Matt Smith

Daniel Agar

Charles Thompson

James Leney

Vanessa Ndiangang

Leyi Ye

Our requirements for Heslington Hustle were elicited by conducting client meetings and continuously iterating on the initial stakeholder documented specifications. The negotiation of the requirements was formed on our client's aspirations and considerations regarding the importance of user experience. Furthermore, the requirements were classified in three categories – "User Requirements", "Functional Requirements" and "Non-Functional Requirements" as per Sommerville chapter 4[1], to ensure a complete coordinated analysis. These three tables were indexed with unique IDs for establishing connections between different classes of requirements and a logical transition from user to system requirements. The user requirements are closest to the stakeholder's statements regarding the tasks that the users should be able to perform in the game. Our method for prioritising requirements involved labelling each requirement as "should", "shall" or "may". Furthermore, we have assigned a priority to each user requirement, indicating the importance of fulfilling this requirement. This further helps to outline what the architecture and implementation teams need to focus working on.

Moreover, the software requirements include a more detailed description of the technical functionality needs and are used to instruct the game's implementation. In our requirements engineering we focused on defining the main activities our avatar should be able to perform in the game – eating, studying, recreational and sleep. For instance, we needed to consider some contradicting requirements in the product brief, regarding whether the avatar's sleeping habits were to be managed by the player or built-in to the game. We needed to liaise with our client as we discovered that these requirements were mutually exclusive and clarify which implementation they preferred.

Furthermore, there were requirements in the project brief that needed to be clarified, such as whether time would pass automatically (1 in game minute/second) or only by doing activities, or omitted entirely. The initial product brief did not include sound preferences and specifics regarding the menu controls, settings, pause menu, saving game functionality, avatar customisation and an option to name the avatar. Important components of the game include an activity counter that should track how many of each activity was performed so far and an energy bar to display the remaining energy. Time representation had to be further clarified with our client to ensure a unified game core structure. In addition, the initial product brief did not include sound preferences and specifics regarding the menu controls, settings, pause menu, saving game functionality, avatar customisation and an option to name the avatar. One of our core requirements is to be able to move the avatar around a map and for it to interact with the surrounding environment, this requirement must be translated into programming code that moves the avatar and implements a method for the avatar to interact through player input (i.e. WASD or arrow keys). The teachings from Sommerville[1] on how different types of requirements may be written in different types of language and the form of the language influenced how our requirements were written, with descriptions of functional/Non-functional requirements taking a more structured/programmatic form.Our functional requirements refer to what the system should be able to do, whereas the non-functional requirements refer to what the system should be like, for instance we have outlined that the client needs the game to run on a desktop, consequently, a non-functional requirement would be that the game shouldn't crash when running on a desktop. As this is not a tangible requirement, we need to define a fit criterion that we can use to measure if we have met our client's standards - in this case, less than 10% of crashes from the time running on a desktop. We ensured regular contact with our client later in the project's architecture and implementation iteration cycles, to improve and further develop the original game's blueprint that was based on our requirements engineering process. In this way, we can easily outline what the architecture and implementation teams need to focus working on.

User Requirements

UR_ACCESSIBILITY	The avatar and buildings in the game should be distinguishable by shape and colour. The text throughout the game should be legible.	Shall
UR_CONTROL	The player shall be able to move the avatar on the map and interact with different objects in the game.	Shall
UR_MAP_LOCATIONS	The map will represent at least some of the Heslington East campus with one location each for sleeping, eating, studying, relaxing.	Shall
UR_SLEEP_FEATURE_TASK	The avatar must sleep at the end of each day.	Shall
UR_ENERGY_TIME_FEATURE	In a progress bar the user will be able to see how each activity consumes energy and time from the initial energy and time capacity.	Shall
UR_RECREATIONAL_ACTIVITY_TASK	At least one recreational activity must be available for interaction in the map, this cannot be sleeping.	Shall
UR_STUDY_TASK	The avatar must study at least once per day, with the option to study twice in one day if they missed studying on another day. Studying twice is allowed only once per game.	Shall
UR_GAME_ART	The components and background of the game should be bright, colourful, welcoming.	Should
UR_PLATFORM_COMPATIBILITY	The game should run on desktop.	Shall
UR_SOUND_CONTROL	The player should be able to always control the sound in the game. The sound controls should be visible and easy to navigate.	Should
UR_COMPLETE_GAME_SCREEN	The player shall automatically return to the main menu once they finish the game.	Should
UR_PAUSE_MENU_VISIBILITY	The user should be able to access the pause menu throughout the game.	Should
UR_DEMOGRAPHIC_ENGAGEMENT	The game is for young adults.	Should
UR_CUSTOMISE_AVATAR	The player should be able to name and customise their avatar.	Should
UR_CHOOSE_TASKS	The user shall be able to perform different tasks, like studying, eating, relaxing, that would take different amounts of energy and time. With sleeping as the only available action when the avatar runs out of energy or to progress to the next day.	Shall
UR_MAIN_MENU_NAVIGATION	The user shall be able to access the main menu throughout the game and will be able to adjust the volume of the game, start, pause game, view credits, select avatar.	Shall
UR_ACTIVITY_COUNTE	This counter shall display the number of times the avatar has performed each	Shall

R	activity so far and display the final count at the end of the game.	
UR_GAME_DURATION	The game lasts for 7 days with each day ending when the avatar sleeps. The game is over when the 7th day is over. The real time playthrough for the game should last for about 5 -10 min.	Shall
UR_RECEIVE_FEEDBACK	The player should receive text feedback after completing some activities.	Should
UR_LEADERBOARD	The player must be able to access a leaderboard of the highest scores recorded in the game.	Shall
UR_SCORE	At the end of the game the score the player has achieved must be shown to the player	Shall
UR_STREAKS	At the end of the game the player must be able to see any streaks they have earnt	Shall
UR_STREAKS_ACHIEVEABLE	The player must be able to earn streaks throughout the playthrough	Shall

Functional Requirements

FR_MAIN_MENU	The main menu of the game allows adjustment of sound, credits, asset sources and the ability to start the game	UR_MAIN_MENU_NAVIGATION
FR_ANIMATIONS	The game must have some level of animation	UR_GAME_ART, UR_ACCESSIBILITY
FR_CONTROL	The game avatar should move in response to the player using the arrow keys	UR_CONTROL
FR_SOUND	The game has sound	UR_SOUND_CONTROL, UR_MAIN_MENU_NAVIGATION
FR_PAUSE_MENU	The game has a pause menu	UR_PAUSE_MENU_VISIBILITY
FR_FINISH	Upon ending the game returns to the main menu	UR_COMPLETE_GAME_SCREEN
FR_UNIVERSITY_TASKS	The game facilitates studying, eating, sleeping and recreational activities by interacting with the relevant buildings	UR_CHOOSE_TASKS, UR_MAP_LOCATIONS, UR_SLEEP_FEATURE, UR_STUDY_TASK
FR_FINISH_CREDITS	The credits are shown when finishing the game	UR_COMPLETE_GAME_SCREEN, UR_MAIN_MENU_NAVIGATION
FR_REPRESENTATIVE SPRITES	The sprites must accurately represent the thing they represent (e.g. water tiles look like water)	UR_GAME_ART, UR_ACCESSIBILITY
FR_INTRODUCTION	The user is introduced to the game upon starting	UR_RECEIVE_FEEDBACK, UR_GAME_PROGRESSION
FR_RANDOM_EVENTS	When doing something, a random event with a positive/negative effect can occur	UR_RECEIVE_FEEDBACK, UR_GAME_PROGRESSION

FR_STATIC_TIME	Time only increments when performing actions	UR_CHOOSE_TASKS, UR_ENERGY_TIME_FEATURE
FR_QUICK_TIME_ACTIONS	Actions should take little time such that a user can perform at least 2-3 per day	UR_CHOOSE_TASKS, UR_ENERGY_TIME_FEATURE
FR_PERSPECTIVE	The user must be able to view the game from a top down perspective	UR_ACCESSIBILITY
FR_RESOLUTION	The game should be in 1080p	UR_PLATFORM_COMPATIBILITY, UR_ACCESSIBILITY
FR_NPCS	The game has NPCs that can be interacted with to make the game more lively	UR_ACCESSIBILITY, UR_GAME_ART, UR_CONTROL
FR_ENERGY_TIME_MANAGEMENT	The time passes/energy depletes by performing activities. If an avatar has insufficient energy or it is too late in the day to do an activity then they can't do that activity and must sleep.	UR_SLEEP_FEATURE_TASK, UR_ENERGY_TIME_FEATURE
FR_STUDYING_RESTRICTIONS	The game avatar can study exactly once per day, except for once where they can study twice if they haven't studied on a prior day	UR_STUDY_TASK
FR_MAIN_MENU_CUSTOMISATION	The appearance of the game avatar can be customised in the main menu	UR_CUSTOMISE_AVATAR
FR_ACTIVITY_COUNTER	There must be a visible counter that tracks how often a user has performed an activity.	UR_ACTIVITY_COUNTER
FR_LEADERBOARD_DISPLAY	There must be a leaderboard screen the user can access from the main menu.	UR_LEADERBOARD
FR_LEADERBOARD_STORE	The leaderboard must be stored in a file within the game so its state can be preserved for the next playthrough.	UR_LEADERBOARD
FR_ACTION_LOGS	The game must log the location and type for every activity completed by the player and additionally log the time of day for eating activities.	UR_SCORE UR_STREAKS
FR_SCORE	Score must be calculated at the end of the game, this should be based on the number of each activity completed as well as the locations and time of day if applicable.	UR_SCORE
FR_STREAKS	Streaks must be calculated at the end of the game based off of the information gathered in the logging of activities	UR_STREAKS
FR_STREAKS_DISPLAY	Achieved streaks must be displayed in the game over screen. A Small badge & Name of the streak should be visible to the player for this.	UR_STREAKS
FR_STREAKS_ACHIEVEABLE	The user shouldn't have to do something every day of the week to be able to achieve a streak.	UR_STREAKS_ACHIEVEABLE
FR_RECREATIONAL_ACTIVITY_TASK	Map must contain at least one place the user can interact with to complete a recreational activity.	UR_RECREATIONAL_ACTIVITY_TASK

Non-Functional Requirements

NFR_ACCESSIBILITY	The system should be operable by users with red-green colorblindness	UR_ACCESSIBILITY	90% of all users including those with red-green colourblindness can correctly identify differentiate sprites
NFR_RELIABILITY	The game shall be stable	UR_CHOOSE_TASKS	Average crash rate of <1% for every hour the game is played
NFR_PERFORMANCE	The game shall run smoothly with minimal frame drops	UR_CONTROL	The game runs with at least 30 FPS, 90% of the time for all systems meeting minimum requirements mentioned in FR_PLATFORM
NFR_USABILITY	The GUI should be understandable and intuitive	UR_MAIN_MENU_NAVIGATION	>90% of players will understand the game mechanics and what to do after their first run through of the game
NFR_MAINTAINABILITY	The code must be planned out, well documented, and modular for easier updates	UR_GAME_DURATION	Feature updates are deployed within a week. At least 2 teams pick our project due to our highly maintainable code
NFR_COMPATIBILITY	The game shall be compatible with the given operating systems	UR_PLATFORM_COMPATIBILITY	The game successfully loads at least 95% of the time on compatible systems
NFR_PLATFORM	The game shall be compatible with Windows 10 and above, MacOS Sierra and above with x86_64 architecture and Linux (Ubuntu 18.04 and above, Fedora 28 and above or similar distributions).	UR_PLATFORM_COMPATIBILITY	The game is able to run on the indicated platforms for at least the duration of one game without crashing.

References

[1] I. Sommerville, *Software Engineering*. 10th ed. Essex: Pearsons Education Ltd, 2016.