Note: This pdf is the longer version of the requirements deliverable. The shortened version submitted in the zip file has the top priority requirements with this one collating all of them.

Eliciting System Requirements

In order to create a streamlined process for implementation and testing, we needed to gather user, functional, and nonfunctional requirements. To do this, we first considered the stakeholder for the project and created a list of relevant interview questions so that we could get an understanding of their vision for the project. We decided on a one-on-one interview method to maintain simplicity and keep a specific idea of what our stakeholder wanted, at the cost of having a less varied range of ideas to work on with the project [1]. This was a vital aspect of the planning process that allowed us to gather the user requirements which would shape both the functional and nonfunctional requirements later [2].

Once we had the answered interview questions, we organised the key user requirements we gathered from them as either a “Must, Should or Could” requirement based on the increasingly popular MoSCoW Model [3], [4], so that we could make sure the implementation team had a focus on the most critical requirements in the project rather than getting side-tracked with extra, non-essential features. The user requirements we labelled as “must'' can be found below in this document as well as the corresponding functional and nonfunctional requirements. The other non-essential requirements can be found on the project’s website.

When creating the functional and non-functional requirements we tried to keep the testing process in mind, so that we could make sure that they were fulfilled fully. This means that every requirement has a clear goal that we can test for either during development or with users during playtesting.

We arranged our requirements in tables for a clear layout where everything was easily able to be found. Functional requirements clearly state which User Requirements(s) they correspond to and Non Functional Requirements state what type of ability the system is keeping in check with. Non Functional Requirements were also assigned a type relevant to the functions and capabilities the system should provide. [5] A tabular layout allows for information to be sorted in a way that makes logical sense and allows for more information to be stored on the page than a list of all requirements would [6]. We also tried to keep descriptions of each requirement to a minimum to avoid clutter and to avoid technical jargon in the descriptions to ensure that these requirements are communicated to whomever reads the document effectively [7].

Definition and Acronyms:

- UR: User Requirement
- FR: Functional Requirement
- NFR: Non-Functional Requirement

These acronyms will be used as the basis for any ID tags created for our requirements. Every UR, FR and NFR has a unique ID tag.

User Requirements

Top Priority (Must)

User Avatar (UR_AVATAR)	The user shall be able to interact with the game by controlling an avatar that can move around the map and interact with the various objects around them.
Game Menu (UR_MENU)	The user will be able to view a menu when starting or pausing the game which will allow them to access credits, avatar/settings customization, and a tutorial.
World Map (UR_MAP)	The game will have a map of Heslington East Campus that contains various spots for the user to engage in activities. The user will be able to navigate and understand this map well.
Energy (UR_ENERGY)	The player will only have a limited amount of energy and will use this energy up when they perform any activity.
Game Clock (UR_TIME)	The user will have 7 in-game days with 16 in-game hours a day to complete their activities. Performing any activity will use up a certain amount of time.
Activities (UR_ RECREATION)	The user shall be able to interact with different recreational activities around the map which will increase their overall score.
Studying (UR_STUDY)	The user will be able to study at one location on the map in order to increase their overall score.
Eating (UR_EAT)	The user will be able to feed their character at one location on the map, this will contribute to their well being score.
Resting (UR_REST)	The user will be able to sleep at the end of the day to refill energy. They will also be able to rest earlier in the day if the player wishes to.
Win Condition (UR_WIN)	The player will be able to win the game at the end of the 7 days by earning a minimum pass mark in their exams. This is achieved by the player studying at least once every day.
Catching Up on Studying (UR_CATCHUP)	The player can bypass the requirement of studying once per day by “catching up” the next day if they miss the study requirement once. This action can only be done once per game.
Overall Score (UR_OVERALL_SC ORE)	If the user meets the win conditions they will be able to see an overall score at the end of the game, which will be calculated based on the different activities they did during the game.

Medium Priority (Should)

Customisation (UR_CUSTOMISE)	The user should be able to customise their avatar in the game. This will be done through an option on the main menu which the user can easily navigate to at any time.
Wellbeing Score (UR_WELLBEING)	The player should be able to improve their well being score by taking care of their character. This can be done by eating and sleeping regularly as well as doing enough recreational activities during the week.
Study Score (UR_STUDYSCORE)	The player can increase their study score by studying lots throughout the week without overworking their character. This score will be combined with the wellbeing score to give the overall score.

Low Priority (May)

Real World Playtime (UR_PLAYTIME)	The game should take up about 5-10 minutes of real world time from beginning to end. This is so the user can complete the game in a single sitting.
Action Consequences (UR_CONSEQUENCES)	The actions of a user during one in-game day can affect the player or the game world the next day, this will be displayed clearly to the player in various ways.

Functional Requirements

FR_MOVE	The user pressing inputs on the keyboard/mouse will lead to the system allowing the avatar to move around the map and interact with locations and activities. This includes diagonal movements.	**UR_AVATAR**
FR_OPENING	The system shall display an opening screen upon loading into the game for the first time that will give a basic tutorial of the controls and the premise.	UR_MENU
FR_SETTINGS	The system shall have a settings menu which allows users to change various settings in the game, for example the volume of any in-game sound effects or music.	UR_MENU
FR_CREDITS	The game shall show a credits screen either in the main menu or upon completion of a playthrough that displays the names of the game's creators.	UR_MENU
FR_MAP	The system shall display a small version of the map on the screen so that the user can navigate around the game world.	UR_MAP
FR_MAP_MOVEMENT	The system shall provide an interaction with the map to move to different locations quickly via bus at the cost of energy.	UR_MAP
FR_DURATION	The system shall keep track of how many days have passed since the beginning of the game.	UR_TIME
FR_TIMEKEEPING	The system shall keep track of the remaining time in the current day and display this to the user.	UR_TIME
FR_NEXT_DAY	The system shall begin the next day after the user sleeps and at the location they fell asleep	UR_TIME
FR_ENERGY_LEVEL	The system shall keep track of and display how much energy the user has remaining in a given day.	UR_ENERGY
FR_ACTIVITY	The system shall check that the user has sufficient time and energy levels to partake in an activity.	UR_ENERGY
FR_DUCKS	The system shall allow the user to feed some campus ducks by interacting with a duck pond on the map.	UR_RECREATION
FR_STUDY_REQ	The system shall keep track of whether a student has studied at least once in a given day or not.	UR_STUDY

FR_LIBRARY	The system shall allow the user to study at the library in order to fulfil the study requirements for the game.	UR_STUDY
FR_PIAZZA	The system shall allow the user to eat at the piazza restaurant on campus.	UR_EAT
FR_ACCOMODATION	The system shall allow the user to end the day by sleeping in their accommodation.	UR_REST
FR_SLEEP	The player will fall asleep after 16 hours, no matter where they are.	UR_REST
FR_WIN_CONDITION	The system shall display that a user has won the game if by the end of the week they fulfil study requirements.	UR_WIN
FR_CATCH_UP	The system shall allow the user to catch up on their studies if they missed the opportunity the previous day.	UR_CATCHUP
FR_SCORE	The system shall keep track of points accumulated from completing activities and display these at the end.	UR_OVERALL_SCORE
FR_SPRITES	The game shall give the user a group of preset sprite options to choose from as part of the game's customisation.	UR_CUSTOMISE
FR_REPEAT_ACTIVITY	If a user tries to complete the same activity twice in one day, they will receive less wellbeing points than the first time they completed the activity.	UR_WELLBEING
FR_WARNING	The system shall alert the user if they have low energy/hunger/recreation and will advise them to complete an activity that replenishes the respective resource.	UR_WELLBEING
FR_MORE_DUCKS	The system shall increase the number of ducks on campus/around the duck pond if the player chooses to feed them the day before.	UR_CONSEQUENCES
FR_OUTSIDE_SLEEP	If the player sleeps anywhere besides their accommodation (e.g. the Library), the system shall penalise the player via an energy loss the following day.	UR_CONSEQUENCES, UR_REST
FR_EATING_PENALTY	If the player goes too long without a meal or does not eat at the correct times each day, their wellbeing score will be penalised and activities will drain more energy throughout the day.	UR_CONSEQUENCES, UR_ENERGY, UR_EAT

Non-Functional Requirements

NFR_DATA	Performance	Variables stored by the game (e.g energy, food levels) must be correctly updated after a relevant event occurs within 0.05 seconds of the event occurring.
NFR_CONTROL_RESPONSE	Performance	The character should move in under 0.05 seconds of the user controlling the user to move.
NFR_MENU_RESPONSE	Performance	Menus and windows should be displayed within 0.05 seconds of the user prompting to open them. Buttons should also perform the correct action with 0.05 seconds of being pressed.
NFR_UPDATING	Scalability	The system should be capable of correctly updating multiple[α] variables at the same time without any variables becoming corrupted, or any degradation in response time.
NFR_OS_PERFORMANCE	Portability	The system should run on Windows 10 and 11 operating systems without significant changes in performance.
NFR_OS	Compatibility	The system should be able to run on Windows 10 and 11 OS.
NFR_SYSTEM_FAILURE	Maintainability	If the system fails, it should be able to restore itself within a time span of 10 minutes on average.
NFR_FRAME_RATE	Maintainability	The game should cap the framerate at a reasonable level, depending on the user's hardware specifications.
NFR_HARDWARE	Availability	90% of users should be able to run the game smoothly, regardless of their hardware specifications.
NFR_SECURITY	Security	In all use cases, player data stored by the system must remain securely stored and insusceptible to being accessed by intruders.
NFR_EASE	Usability	90% of users will include in their feedback that they felt the controls and UI of the game felt intuitive and easy to understand.

References

[1] M. Eid, “Requirement Gathering Methods,” Umsl.edu, Nov. 2015. Available: https://www.umsl.edu/~sauterv/analysis/F2015/Requirement%20Gathering%20Methods.html.htm

[2] Isotoma, “The Importance of Well-Defined User Requirements”, 2019, June 19 [Online] Available: https://isotoma.com/blog/2019/06/19/the-importance-of-well-defined-user-requirements/

[3] A. Jarzębowicz and N. Sitko, “Agile Requirements Prioritization in Practice: Results of an Industrial Survey,” Procedia Computer Science, vol. 176, pp. 3446–3455, 2020, doi: https://doi.org/10.1016/j.procs.2020.09.052

[4] Product Plan, “What Is MoSCoW Prioritization? | Overview of the MoSCoW Method,” Productplan.com, 2018. Available: https://www.productplan.com/glossary/moscow-prioritization/

[5] “Nonfunctional Requirements in Software Engineering: Examples, Types, Best Practices”, altexsoft, 2023, Dec. 30. Available: https://www.altexsoft.com/blog/non-functional-requirements/

[6] “For the Love of Tables: Building Success in English Language,” codexterous, 2022, Feb. 13. Available: https://codexterous.home.blog/2022/02/13/for-the-love-of-tables-building-success-in-english-language (accessed Mar. 12, 2024)

[7] INCOSE, *Why use the Textual Form of Communication* in “Guide to Writing Requirements,” pp. 11-15, May 2022. Available: https://moodle.insa-toulouse.fr/pluginfile.php/121422/mod_folder/content/0/INCOSE_RWG_Guide_to_Writing_Requirements_V3.1_041822.pdf