Updated Requirements

Cohort 1, Group 9
Chris Sewell, Fedor Kurochkin, Matt Durham, Max Peterson,
Oladapo Olaniran, Wojtek Tomaszewski,
Yuqi Fu.

Requirement Elicitation Pre Handover

Initially we created a Single Statement of Need and collected and inferred requirements from the initial brief. While this meant our initial requirements were rough around the edges, we planned to interview the customer to get further details on the game. From there, we planned to change any requirements that needed changing, as well as adding some new ones if we deemed it necessary.

Single Statement of Need (SSON):

A game where the user escapes from a university-themed maze with a time limit while avoiding obstacles and acquiring power-up items

After the interview with the customer, in which we asked them a set of pre-planned questions to determine what they wanted the game to be like, we reviewed our initial requirements and found that we had covered most of the points that the customer wanted for the game. We added a few minor changes as well as some extra requirements to ensure the game was as close to the customer's description as possible.

To organise our requirements in a way that would make them easy for anyone unfamiliar with our game to understand, we decide to divide the requirements into 4 categories:

- User requirements - Features that the user would encounter or interact with during the game
- Functional requirements - What the system would need to fulfil the User requirements
- Non-functional requirements - Standards to measure the system's performance against such as reliability, timing and usability when fulfilling the Functional and requirements.
- Constraint requirements - Any constraints that we would need to follow during the development of the game

Each category of requirement would then get a table of requirements made up of 3 columns. The first column of every row was the requirement ID. Our requirement ID started with the table it was in, to indicate what category it was. Then we added a description of what its requirement was to the ID. The second column of the requirements table was a description of each requirement that the game would need to fulfil the brief. The third columns of the tables were different for some categories. The third column of the user and constraint requirements was the priority of each task, with the option of Shall, Should or May for each row while the third column of the Function and Non-functional tables was the User requirement that would be fulfilled if we fulfilled the requirement.Use case

To verify that our requirements were valid, we created a text-based use case of the user playing the game and assessed if the requirements would be satisfied in it. The use case was:

- Actor: User
- Precondition: The user has already loaded the game
- Trigger: The user starts the game
- Main success scenario:
    - The user moves around the maze
    - The user finds the way out the maze
    - The user escapes within the time limit
- Secondary scenarios:
    1) The user encounters positive events that help them
    2) The user encounters negative events that hinder them
    3) The user encounters hidden events that change the course of the game
    4) The user does not escape the maze within the time limit and the game automatically ends ● Success Postcondition The user receives a higher score at the end of the game
- Minimal Postcondition: The game ends and the user receive their score

User Requirements Table		
ID	Description	Priority
UR_ESCAPE_MAZE	The player shall be able to escape from a university themed maze.	Shall
UR_POSITIVE_EVENTS	The player shall encounter at least 3 visible events that provide beneficial effects.	Shall
UR_NEGATIVE_EVENTS	The player shall encounter at least 5 visible events that hinder their progress.	Shall
UR_HIDDEN_EVENTS	The player shall encounter at least 3 hidden events that alter the game in beneficial or detrimental ways.	Shall
UR_UI	At least 90% of players shall intuitively understand what all menu options mean and how to navigate them. All controls shall be displayed in-game so that players know how to use them.	Shall
UR_TIME_LIMIT	The player shall be able to complete the maze within a time limit of 5 minutes.	Shall
UR_PAUSE	The player shall be able to pause the game during play.	Shall
UR_SCORE	The player shall receive a score after completing the game.	Shall
UR_BOUNDARIES	There should not be areas of the map where it is unclear if they are explorable.	Shall
UR_THEME	The maze shall include clearly recognisable university-like features.	Shall
UR_DIFFICULTY	The game should allow for varying difficulties, including an easy mode that always permits pausing.	Should
UR_HIDDEN_EVENT_HINTS	The player could find hints that tell them the location of the hidden events, or clues as to what they are	May
UR_DEAN	There may be a feature based around a dean, or university staff member that shall inconvenience the player	May
UR_AUDIO	The User may hear music that enhances their play experience	May

Functional Requirements Table		
ID	Description	Corresponding User Requirements
FR_MAZE	The system shall generate a university-themed maze that the player must escape from.	UR_ESCAPE_MAZE
FR_TIMER	The system shall provide a timer that tracks playtime and controls the time limit.	UR_TIME_LIMIT
FR_END_GAME	The system shall end the game when the player escapes, is caught by the Dean, or the timer expires.	UR_ESCAPE_MAZE
FR_PAUSE	The system shall fully pause all gameplay when the player pauses the game.	UR_PAUSE
FR_PAUSE_MENU	The system shall display a pause menu that provides appropriate options.	UR_UI, UR_PAUSE
FR_POSITIVE_EVENTS	The system shall include visible events that provide beneficial effects.	UR_POSITIVE_EVENTS
FR_NEGATIVE_EVENTS	The system shall include visible events that hinder progress or score.	UR_NEGATIVE_EVENTS
FR_HIDDEN_EVENTS	The system shall include hidden events that may help or hinder the user.	UR_HIDDEN_EVENTS
FR_DEAN	The system shall display and control Dean character that negatively impacts the player.	UR_DEAN
FR_WAY_OUT	The system shall provide an exit point to escape the maze.	UR_ESCAPE_MAZE
FR_SCORE	The system shall calculate the player's score, based on time to escape and interactions with events.	UR_SCORE UR_POSITIVE_EVENTS UR_NEGATIVE_EVENTS UR_HIDDEN_EVENTS
FR_BOUNDARIES	The system shall enforce boundaries that the player cannot pass.	UR_BOUNDARIES
FR_EVENT_AMOUNTS	The system shall include a minimum of 5 negative event types, 3 positive event types, and 3 hidden event types.	UR_EVENT_AMOUNTS
FR_THEME	The system shall simulate a university-like environment featuring appropriate scenery and aesthetics.	UR_THEME
FR_EVENT_TRACKER	The system shall include counters that track the total number of each event type that the player has triggered.	UR_EVENT_AMOUNTS
FR_DIFFICULTY	The system shall provide at least a base difficulty where pausing is always enabled.	UR_DIFFICULTY

Non-Functional Requirements Table			
ID	Description	User Requirements	Fit Criteria
NFR_PERFORMANCE	The game should run smoothly on an average computer	UR_UI	0 crashes in test runs under normal conditions
NFR_EVENT	The system shall trigger positive, negative or hidden events immediately upon event activation.	UR_POSITIVE_EVENTS, UR_NEGATIVE_ EVENTS, UR_HIDDEN_ EVENTS	<1 second delay after an event is initiated
NFR_SCORE	The system shall update the player's score in real time	UR_SCORE	<1 second after score changes
NFR_PRESERVE_GAMESTATE	The system shall preserve and subsequently restore the current gamestate upon pause.	UR_PAUSE	Gamestate is successfully preserved.

Constraint Requirements Table		
ID	Description	Priority
CR_PLATFORM	The game shall run on Desktop only, on any Operating System without needing specialist hardware	Shall
CR_COPYRIGHT	All 3rd party assets and resources shall be open source, free of copyright, or otherwise morally and legally acceptable for use	Shall
CR_SPEND	The project should not exceed £50 in total development cost.	Should
CR_ENGINE	The game shall be implemented using an open source engine and open source tools.	May

Requirements Elicitation Post Handover

The following updated list of requirements was established based on the new version of the brief as well as user evaluation tests conducted by the group. This results in a limited set of requirements added to the existing requirements record. Through team meetings focused on the analysis of the brief, all changes were recorded and later formed into complete requirements. By default, demands found directly in the brief were considered of the highest priority. This was due to these requirements having a significant impact on the game's core. In addition, given the version of the game that was completed for the previous set of requirements, some features were not included. Consequently, these demands from the client remain as requirements so that they can be implemented in the final prototype. For example, even though the pause menu was included in the requirements table, it was not delivered.

Following the project handover procedure, the team conducted multiple user evaluation tests. These tests presented the team with additional information and feedback on the mechanics implemented in the current system. This set of data was also converted from usability problems into requirements through extra group meetings. The evaluation ultimately revealed requests of further improvement of pre-existing functionality. Therefore, the requirements coming from user evaluation typically will be assigned lower priority as these are smaller changes overall. However, a shall priority level will be assigned to these requirements in case the issue prevents intended gameplay. One of the examples is a negative event that the users did not consider impactful enough. This resulted in a new requirement being added to the list in order to increase the negative effect for future users and make the event seem more punishing.

In order to keep a clear record of all requirements in a structured way, the team decided to use four separate tables. The first table is used to keep track of all user goals that come from either the two versions of the brief or the user evaluation (user requirements). The second table is keeping track of all user requirements translated into the system's functionality (functional requirements). The third table consists of records listing quality properties of functional requirements such as efficiency, security, reliability and so on (non-functional requirements). The final table is used to record all boundaries and rules limitations which must be followed for the process of development (constraint requirements). Regarding the format, the tables will consist of the same two fields followed by additional ones exclusive to the table as follows.

Common fields:
- Requirement ID - Unique identifier of a requirement
- Description - Explanation of the requirement

User Requirements:
- Priority - Measurement of how important a requirement is (Highest → Lowest: Shall, Should, May)

Functional Requirements:
- User Requirements - IDs of related user requirements

Non-Functional Requirements:
- User Requirements - IDs of related user requirements
- Fit Criteria - Measurable conditions that prove a system meets quality standards

Constraint Requirements:
- Priority - Measurement of how important a requirement is (Highest → Lowest: Shall, Should, May)

User requirements

ID	Description	Priority
UR_PAUSE	The player shall be able to pause the game at any time during play	Shall
UR_LEADERBOARD	The game shall include a leaderboard with the name and score of the top 5 scores	Shall
UR_ACHIEVEMENTS	The game shall notify the user when an achievement is unlocked	Shall
UR_DIFFICULTY	The game may include a difficulty setting that makes the game more or less challenging	May
UR_GEESE_IMPACT	The game event should reduce the score in an impactful way	Should
UR_BOB_CONDITION	The player shall only receive the reward if all steps in the event are completed successfully	Shall
UR_PRESSURE_PLATE	The player shall be able to clearly see and understand that the pressure plate is intractable	Should
UR_ATTENDANCE_IMPACT	The event reward should make a bigger contribution towards the final score	Should

Functional Requirements

ID	Description	User Requirements
FR_PAUSE	When pausing the game, the system shall stop all real-time processes except for UI and be able to continue them when user desires	UR_PAUSE
FR_PAUSE_MENU	When pausing the game, the system shall display a menu of options that the user can select from	UR_PAUSE
FR_LEADERBOARD_STORAGE	The game shall first compare new scores and then store the top five linked to corresponding user names	UR_LEADERBOARD
FR_LEADERBOARD_DISPLAY	The game shall display the top five best results connected to usernames that have achieved them	UR_LEADERBOARD
FR_ACHIEVEMENTS_SYSTEM	The game shall include a system for displaying an acknowledgement message when the user performs a certain sequence of actions	UR_ACHIEVEMENTS

Functional Requirements

FR_DIFFICULTY_SETTING	The game shall include a setting that allows the player to select how punishing the environment is	UR_DIFFICULTY
FR_GEESE_IMPACT	The amount of points reduced due to geese contacts shall be increased. This number shall affect the score in a meaningful way	UR_GEESE_IMPACT
FR_BOB_CONDITION_FIX	The game shall not reward the user with points if glasses are simply picked up by the character. Instead they need to be delivered to bob for the vent to fully count	UR_BOB_CONDITION
FR_PRESSURE_PLATE_CHANGE	The hidden pressure plate shall have an outline when coming within a certain distance	UR_PRESSURE_PLATE
FR_ATTENDANCE_IMPACT	The bonus from the positive attendance event shall be larger. Successful completion of the event shall grant the user a noticeable increase in score	UR_ATTENDANCE_IMPACT

Non-Functional Requirements

ID	Description	User requirements	Fit Criteria
NFR_LEADERBOARD_ PERSISTENCE	When storing the top five scores, data shall remain unchanged	UR_LEADERBOARD	After restarting the game 5 times, data is consistent
NFR_PAUSE_MENU_NAVIGATION	The game shall present with a pause menu that is easy and intuitive to navigate	UR_PAUSE	The pause menu does not contain dead ends. From pausing the game, the user is able to resume in one action
NFR_ACHIEVEMENTS_VALIDATION	The game shall check for achievements in a consistently reliable and accurate way only	UR_ACHIEVEMENTS	All achievement unit tests pass without false positives
NFR_IMPACT_VALIDATION	The game shall increase/decrease the score in a proportionally impactful way	UR_GEESE_IMPACT UR_ATTENDANCE_ IMPACT	All modified events shall increase/ decrease the final score by at least ten percent each run