Requirements

Cohort 1 Group 10

Cai Hughes<cabh500@york.ac.uk>
Ben Slater<bs1463@york.ac.uk>
Adeola Adeniji<aa3098@york.ac.uk>
Mathew Riedy<mr1723@york.ac.uk>
Riad Kasmi<rmk526@york.ac.uk>
Simon Konieczny<sk2144@york.ac.uk>

Requirements Elicitation, Negotiation, and Planning

Requirements were gathered through a combination of the product brief and primary research. The product brief specified basic requirements for the product, detailing the game and necessary elements of mechanics and objectives. These were very straightforward bullet points that provided an excellent jumping board for further enquiry during the first Client-Team meeting.

The team collaborated on a list of questions to elicit a comprehensive set of user requirements and a Single Statement of Need to guide the creation of a comprehensive set of system requirements and constraints. In preparation for the first Client-Team meeting the team came up with questions spanning game mechanics, features, preferred styles, as well as finding out which aspects of the game are the most critical to the client and what are nice-to-haves, so that tasks could be assigned appropriate priority. The interview was audio recorded (with the consent of all parties present) to facilitate better analysis and make it flow more naturally, as to maximize the client's comfort, with less pressure on noting answers down. The client explained their vision for the products, including an example scenario of the game being played and their motivations and goals for the development of the game. The meeting covered more general user requirements and elicited a Single Statement of Need from the client: "The idea is to have a short game that is entertaining and that reflects the exam period at the university that is playable by people at/coming to the university".

Following the client meeting, a formal set of requirements could be established. Using a convention of numbering the user requirements using numbers beginning with 'UR' and assigning system requirements to user requirements enabled accurate tracking of project-critical requirements while making sure nothing was redundant. These requirements were structured into 3 tables (aligning with principles of Requirements Engineering) of User Requirements, Functional Requirements ('FR'), and Non-Functional Requirements ('NFR'). This scheme enabled easy retrieval and modification of specifications as they changed and evolved. Additionally it provides a clear snapshot for any stakeholder to understand where the team is focusing energy and identifies the clear areas of focus, aligning with current understanding of the Client's wishes.

2 Use Case Scenarios were developed to contextualize the major user interactions with the game:

Scenario 1: End Game

Primary Actor: Player
Secondary Actor(s): None (single-player game)
Precondition: Player has made it to the last day of the game
Trigger: Time and/or energy run out on the last day
Main Success Scenario: Game stops, final score is shown to the Player on the screen and stored on the machine.
Success Postcondition: Player is asked to play again or close the program

Scenario 2: End of In-Game Day

Primary Actor: Player
Secondary Actor(s): None (single-player game)
Precondition: Game is ongoing, Player is going through one of the days
Trigger: Player has exhausted time and/or energy for the day
Main Success Scenario: The Player is forced to sleep to move onto a new day
Success Postcondition: New in-game day starts and Player continues the game

Requirements Presentation

Table 1: User Requirements

ID	Description	Priority
UR1_Game_Interface	The game interface should be approachable, easy to understand utilizing established conventions to minimize any learning curve.	Shall
UR2_Game_Controls	The game should utilize standard control conventions to maximize approachability.	Shall
UR3_Target_Audience	The game should appeal to individuals who are about to or have recently started attending university. The game should not have any mention of violence, substance abuse or any other explicit themes.	Shall
UR4_Game_Time	The game should last between 7-10 real life minutes, it is an offline game so should be pausable.	Shall
UR5_Game_Objective	The game objective should be clearly explained to the player at the start of the game, this includes an introduction into the game mechanics and scoring system. The two main game resources: *time* and *energy* should be clearly explained.	Shall
UR6_Game_Security	The game data should be stored locally on the machine.	Shall
UR7_Game_Scoring	The game scoring system should be transparent, explained to the user, and engage the player to try and improve.	Should
UR8_Game_Difficulty	The game should be approachable to all players, therefore reasonably easy to start with possible options to increase difficulty.	Should
UR9_Score_Tracking	Scores should be tracked on the local machine, engaging the player to beat their own score.	Should
UR10_Platform_Compatibility	The game needs to be playable on all major desktop and laptop platforms running moderate hardware for devices that students would use.	Shall
UR11_Accessibility	The game could contain accessibility features such as subtitles or color-blind modes to be more approachable.	Could

Table 2: Functional System Requirements

ID	Description	User Requirement	Context
FR1	Simulate 7 in-game days with a constant game clock.	UR4_Game_Time	Ensures the game progresses over a defined period, reflecting the passage of days
FR2	Implement a lightweight local database or file system for saving game states and high scores.	UR6_Game_Security	Allows for secure, local storage of game progress and scores.
FR3	Support basic control schemes (WASD, arrow keys) for navigation and interaction.	UR2_Game_Controls	Facilitates an intuitive gameplay experience
FR4	Provide a tutorial or help system for new players.	UR7_Game_Scoring	Offers guidance on game mechanics and strategies to improve player understanding.
FR5	Feature dynamic difficulty adjustment based on player performance.	UR8_Game_Difficulty	Keeps the game challenging yet accessible, enhancing replay value.
FR6	Implement an energy and time management system for player activities.	UR5_Game_Objective	Critical for gameplay balance, ensuring players manage resources effectively.

Table 3: Non-Functional System Requirements

ID	Description	User Requirement	Fit Criteria	Context
NFR1	The game should run on laptops and desktops with reasonable hardware specifications.	UR10_Platform_Compatibility	Smooth performance on targeted hardware.	Guarantees a fluid gameplay experience across common student devices.
NFR2	Visual design ranging from realistic to stylized for clear representation of locations.	UR1_Game_Interface	Clear, identifiable game elements.	Accommodates diverse player preferences and improves accessibility.
NFR3	Include accessibility features such as colorblind modes and scalable text.	UR11_Accessibility	Accessibility for all players.	Expands the game's reach and inclusivity.

NFR4	Adaptive UI to different screen sizes and resolutions.	UR1_Game_Interface	Consistent UX across devices.	Ensures the game is visually appealing and functional on any screen.
NFR5	Secure storage for player data and scores.	UR6_Game_Security	Encryption and compliance with data protection regulations.	Protects player information, enhancing trust and safety.
NFR6	Minimize game download and installation size.	UR10_Platform_Compatibility	Efficient use of device storage resources.	Facilitates easier distribution and access, particularly for users with limited bandwidth or storage.

This refined approach to outlining the game's requirements ensures clarity and specificity, enabling effective tracking and implementation throughout the development process. The addition of context for functional and non-functional requirements helps stakeholders understand the rationale behind each requirement, ensuring alignment with the game's overall vision and objectives.

Constraints

Content Appropriateness:

All game content should be appropriate for a broad audience, avoiding sensitive or potentially offensive material.

Development Time:

The project timeline is constrained, requiring efficient management and prioritization of development tasks to meet deadlines.

Hardware Limitations:

The game's design and functionality must account for the lower end of the specified hardware spectrum, ensuring it runs smoothly on less powerful machines

Feedback and Community Engagement:

Mechanisms for collecting and responding to user feedback need to be in place, requiring resources for community management and regular updates based on user input.

Balancing Game Difficulty:

Ensuring the game is challenging enough to keep players engaged but not so difficult that it becomes frustrating, especially in managing the balance between studying, resting, and engaging in recreational activities.