Requirements

Introduction
In our project we are designing a maze-like game with a single obstacle where a character at university must escape from the dean. The game must run on all desktop operating systems and be lightweight and casual. To elicit the user requirements from the customer we organised an interview where we discussed subjects including player data, map layout, gameplay, and checkpoints, amongst other things. Whilst discussing with the customer we talked about more advanced features like procedurally generated terrain and a point based score system. Furthermore, we explored various requirements highlighted by many potential users from our target audience as soon as the foundations were made clear by the customer. This led to negotiation on expectations whereby we discussed the scope of the project at this stage and what would be the most effective use of resources.

User Requirements		
ID	**Description**	**Priority**
UR_MAZE	The game must consist of one or more mazes.	HIGH
UR_MAIN_OBJECTIVE	The player must escape from the maze before the Deen catches them to win.	HIGH
UR_EVENTS	The game must contain at least one event of each of the three types: positive, visible negative, and hidden negative.	HIGH
UR_TIME_LIMIT	The game must last for no more than 5 minutes and must contain a countdown of the time remaining.	HIGH
UR_FAMILY_FRIENDLY	All content in the game must be suitable for audiences of any age.	HIGH
UR_STUDENT_AUDIENCE	The maze must look like a university and the events for UR_EVENT must be related to university life.	HIGH
UR_DIFFICULTY_SCALE	Difficulty must be easy with the potential to scale for future iterations.	HIGH
UR_BONUS_OBJECTIVES	The game may contain bonus optional	LOW

User Requirements		
	objectives for the player to find.	
UR_COMBAT	Any combat used must be of a non violent nature and be puzzle based.	LOW
UR_PLAYER_MOTIVE	The player must care about any objectives set.	MED
UR_TOP_DOWN	The game must use 2D graphics with a top-down view.	HIGH
UR_CHECKPOINT	The game may contain a checkpoint halfway through.	MED
UR_LEADERBOARDS	The game may have a scoreboard.	LOW
UR_SCALABLE_CODE	The code must be scalable for future teams.	HIGH
UR_RANDOMISED_MAZE	The maze may be randomised each playthrough for replayability.	LOW
UR_PAUSE_BUTTON	The user must be able to pause at any time	HIGH
UR_REWARDS	The user may find rewards that increase the time left or provide a similar advantage.	LOW
UR_LICENCING	The game assets must be available under a free commercial licence.	HIGH
********UR_RESTART**********	TO ASK	
UR_WIREFRAME_DESIGN	The project must not go beyond a wireframe for this stage in development.	HIGH

Functional Requirements		
ID	**Description**	**User Requirements**
FR_GAME_PAUSED	The game must pause when the user presses the pause button. The	UR_PAUSE_BUTTON

Functional Requirements		
	countdown timer and all simulation must stop.	
FR_GAME_END	The game must end when either the player has “escaped” the maze or the timer has run out.	UR_TIME_LIMIT
FR_MOVEMENT	The player must be able to move using arrow keys.	UR_2D_TOP_DOWN_VIEW
FR_CAMERA	The camera must follow the character and where they move to.	UR_2D_TOP_DOWN_VIEW
FR_REWARDS	The player should get rewards from completing certain objectives.	UR_PLAYER_MOTIVE
FR_GAME_COMPLETION	The player should only be allowed to complete the game when all the required objectives have been completed.	UR_MAIN_OBJECTIVE
FR_TIMER_VISIBLE	The timer must be shown at all times to the player.	UR_TIME_LIMIT

Non-Functional Requirements			
ID	**Description**	**User Requirements**	**Fit Criteria**
NFR_SCALABLE_CODE	The code must be scalable to allow the game to be worked upon further after development.	UR_SCALABLE_CODE	Good use of proper code management, for example breaking down into reusable functions and classes.
NFR_DOCUMENTATION	The code must be easily understandable to new developers.	UR_SCALABLE_CODE	Clear use of comments and documentation to

Non-Functional Requirements			
			explain the purpose of each part of the code and how it behaves.
NFR_PROJECT_SCOPE	The code must not go beyond the requirements for this stage in development.	UR_WIREFRAME_DESIGN	Look at implementation of features and assess them for time investment

Use Case 1

Name: Player pauses the game
Primary Actor: Player
Trigger: Player taps on the pause button to stop the game temporarily
Main Success Scenario:

- The player presses the pause button
- The game stops and temporarily presents a pause menu to the player
- The player has the options to select different outcomes (e.g Resume, Restart etc)

Secondary Success Scenario:

- The player resumes the game:
    - Player selects the resume button
    - The game is resumed from the moment it was paused
- The player restarts the game:
    - Player selects the restart button
    - The game and the timer are reset and the player goes back to the start

Success Postcondition:

- The game is successfully paused
- All game activities (The player, Dean etc) are halted
- Pause menu shows up

Use Case 2

Name: Player runs out of time
Primary Actor: Player
Trigger: Player fails to escape the maze within the specified time limit
Main success scenario:

- The game ends and the player is redirected to the start menu.
- The player is presented with options to select outcomes(e.g Start, Quit).

Second success scenario:

- Player restarts the game:
    - The player gets to choose if they want to continue progress from the checkpoint(if reached), where the timer starts again in accordance with the checkpoint's timestamp or restart the game from the start point.
    - The player selects the restart button.
    - The game restarts and the timer starts again.
- Player quits the game:
    - The player selects the exit button.
    - The game closes and the user is redirected to the desktop.

Success Postcondition:

- The game successfully ends and the start menu appears.
- All game activities (Events, Character, Dean etc) are halted.
- The start menu shows up.