Architecture

Group 6 - M6

Members:

- Mir Baydemir
- Adam Fraulo
- Azib Hj Abu Akmar
- Agata Pittarello
- Esther Scanlon
- Jazz Stubbins
- Sonia Szetela

-ilities:

In keeping with our requirements, we have prioritised various different design functions, such as extensibility and modularity. These are important aspects of the game, because of UR_DEVELOPMENT, and we will make sure to apply these in our project by writing clear code with good commenting, and organising the code logically by using functions, procedures and classes. These measures will also greatly improve the code's testability, and allow the project to be easily added to in the future.

Because of UR_DEVELOPMENT and UR_GAME_GRAPHICS, we have decided not to prioritise performance. This is because the game has 2D graphics and will require limited-resource features, so will not require any high-performance aspects. The performance of the game still needs to be good enough to not impact usability, but this is likely to be easily achievable without actively focusing on performance.

Due to the nature of the game as a locally hosted game that does not save any data, we have chosen not to prioritise security or availability. Because there is no sensitive data that is being stored, there doesn't need to be any dedicated security for it, and, with the game being locally hosted, uptime is also not a priority.

As specified in UR_PLAYABILITY, usability is a high priority attribute, as well as reliability so that players do not get frustrated and struggle to play the game. These design functions will be achieved by making simple and intuitive controls with a short tutorial at the beginning of the game to improve usability, as well as hard coding certain pieces of data so that they cannot be edited accidentally, which will make the game reliable.

We have not prioritised scalability, as the game is specified as being single-player, and when interviewing our customer, they did not expect to have to increase the number of players, or do anything that would require the game to be easily scalable. Similarly, due to the simplicity of the game, we do not expect any of its components to stop working, and so fault tolerance has not been prioritised.

In the requirement NFR_SYSTEM_AVAILABILITY, we have said that the game should be downloadable on standard PC operating systems. This means that deployability is a priority for us, and we have ensured the ease of deployability of the game by testing it to ensure that it opens and runs, and making sure that the permissions allow for external access.

Required and a Priority:

-ility	why	how
Reliability	Ensures there are less problems to solve and less complaints from stakeholders. Makes sure the system can perform as intended	Ensure that set variables are hard coded so cannot be changed by anything and disrupt functionality.
Extensibility	In the interview, it was said that the game should be a quick project that can be built upon if it performs well. Also we know that we need	Doing good commenting and organising code in a logical way so it is easy to add to

	to extend the project in the second half of the module.	
Usability	Should be family friendly and easy for anyone to use.	Making controls that are simple and intuitive, and including a tutorial at the start of the game
Modularity	Important because it makes it easier for others to work on the project and collaborate	Functions, procedures, and classes.
Deployability	Needs to be usable. The client needs to be able to run it and use it as intended.	Testing to ensure that it opens and runs, make sure that the permissions allow for external/client access (and for licensing ensure they have access to any source code they are entitled to)
Testability	Helps to check maintainability and usability. Checks everything works as intended. Very useful for any project.	Create clear code with blank spacing, commenting, and sectioning.

<u>Not a priority:</u>

-ility	why	how
Security	There is no data being saved by the game, so there is no need to secure anything	N/A
Maintainability	From our interview, it was said that the game is going to be a quick iteration and isn't likely to be revisited, so long term maintainability isn't a priority, but can be useful in case of any goal changes.	Commenting.

<u>Required but not a Priority:</u>

-ility	why	how

Availability	Uptime is not a priority because the game is to be hosted locally, but needs to be available locally.	Allow access permissions for the stakeholders involved.
Fault Tolerance	Components are unlikely to break, as the game is hosted locally and isn't storing any data.	Make sure the system is thoroughly tested.
Scalability	Not likely to have many people involved, single player, and is not required by the company.	Create at least 1 version that runs as intended and has the correct access.
Performance	The system needs to perform as intended but does not require high performance features, so is not a priority.	Test the system to ensure it works, and check that the time it takes to complete its functionality is a reasonable length of time - as in it shouldn't take 2 minutes to load a new scene. Check what data is loading when and the size of that data and compress where it may be too large.

Diagrams and Comparisons:

The maze game/system is being written in Java (company requirement) using LibGDX[1] (highly recommended game engine for Java). We used Visual Studio Code [2] to create the UML class diagram[3]. Then we used draw.io[4] to create the general plan/relationship diagram versions 1[5] and 2[6]. To create the UML component diagram[7] we drew it by hand.

When creating the relationship diagrams, the idea was to start from the top and work down. This meant that more complex options could be thought about as we got to them, and it made it easier to approach the creation, design and layout of the diagram. We started with the overall game itself at the top because that was the largest umbrella category. From there we thought about the main categories to consider when creating the game, such as layout, functionality, and playability. From there, it was then easier to think about what we wanted in the game to fulfill those categories, such as how a character is needed for the player to control, so they can actually interact with and immerse themselves in the game. The requirements for the project needed there to be different events and obstacles within the game, so we thought about different visible and invisible challenges to add, such as an enemy or enemies and not only including a timer, but events that can increase and decrease the amount of time left.

Between the two relationship diagrams, we realised that the game should have an event tracker to meet requirements better. We also thought about the link between categories more and having a 'How to play' menu to help users understand how to play the game and what the controls are. We thought about how to apply the different events to meet the

required target of at least 3, and that's when the idea of having the timer change based on in-game decisions would be a good advantage or obstacle. We also then decided how to allow the user to move and interact with different parts of the game and decided that for simplicity and ease of understanding, using wasd for movement like many other games would be a good idea, as well as f for interactions.

We researched the game ‘Pix the Cat’[8] because it had similar features to our game and we wanted to see what features would be good to also use and which not to. Those that met our requirements, such as being a maze(-like) game, having different events occur based on different interactions and obstacles, having settings and menus, as well as having a top down view. Some things we did not want to keep were the smaller screen size (not fullscreen game map), item trailing (not required for our game style), and different difficulty levels/different gamemodes (as this is not a requirement and it is better to focus on quality over quantity, especially for a small game).

As seen in the final class diagram[3], the architecture of the game has changed compared to our initial plans shown in earlier diagrams. In the project brief, it was stated in the premise that the player is running from the dean, so we have added enemies to the game, as this is one of our three required events, and created more interest and interactability in line with the brief.

References:

[1] LibGDX. (v.1.13.5), LibGDX (Mario Zechner). Accessed: Nov. 10, 2025. [Online]. Available: https://libgdx.com/
[2] Visual Studio Code. (v.1.105.1), Microsoft. Accessed Nov. 09, 2025. [App]. Available: https://code.visualstudio.com/
[3] M6. “Class Diagram”, 2025[digital image]. Unnamed Maze Game. https://frlo14.github.io/group6/
[4] GA. Alder, “app.diagrams.net.”, draw.io. https://app.diagrams.net/
[5] M6. “Relationship Diagram V1”, 2025[digital image]. Unnamed Maze Game. https://frlo14.github.io/group6/
[7] M6. “Component Diagram”, 2025[digital image]. Unnamed Maze Game. https://frlo14.github.io/group6/
[8] Pix the Cat. (v.1.0), Pastagames. Accessed: Nov. 10, 2025. [Online]. Available: https://www.pastagames.com/pix-the-cat/