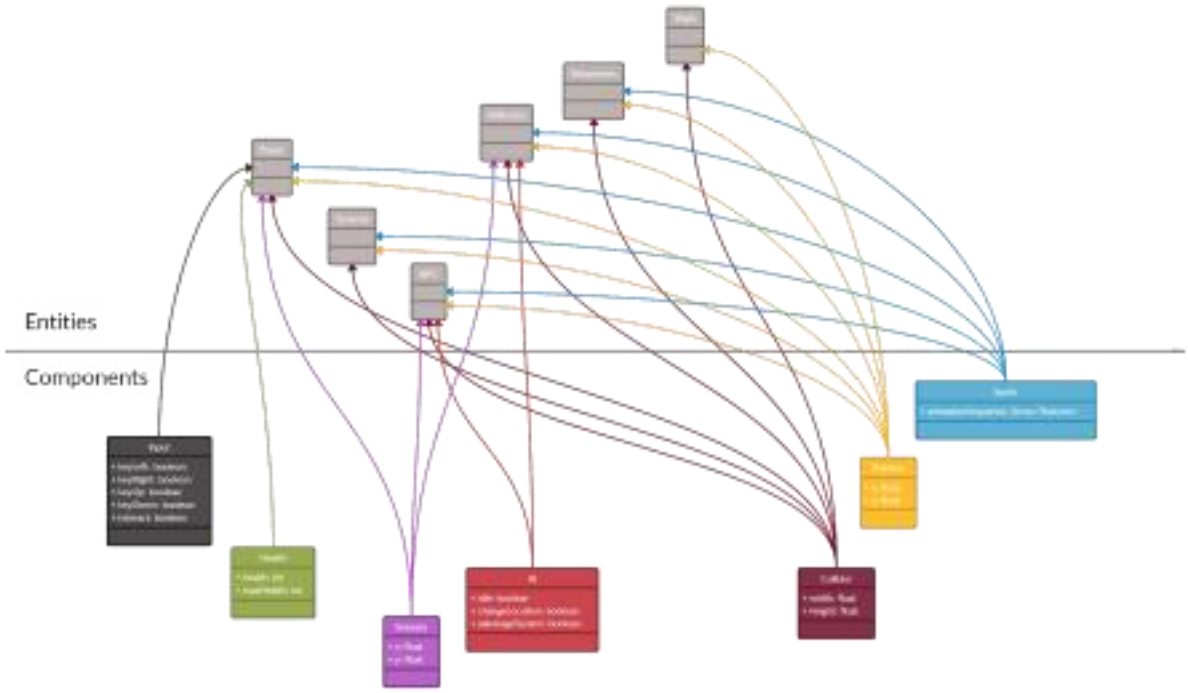
Module:	**ENG1/ASSESSMENT1**
	Architecture
	e: ENG1_Boolean_BobCats
Team members	**1. Adam Howard** **2. Lewis Mcshane** **3. Morgan Davis** **4. Muhidin Muhidin** **5. Roan Gibbons** **6. Zijun Zou**

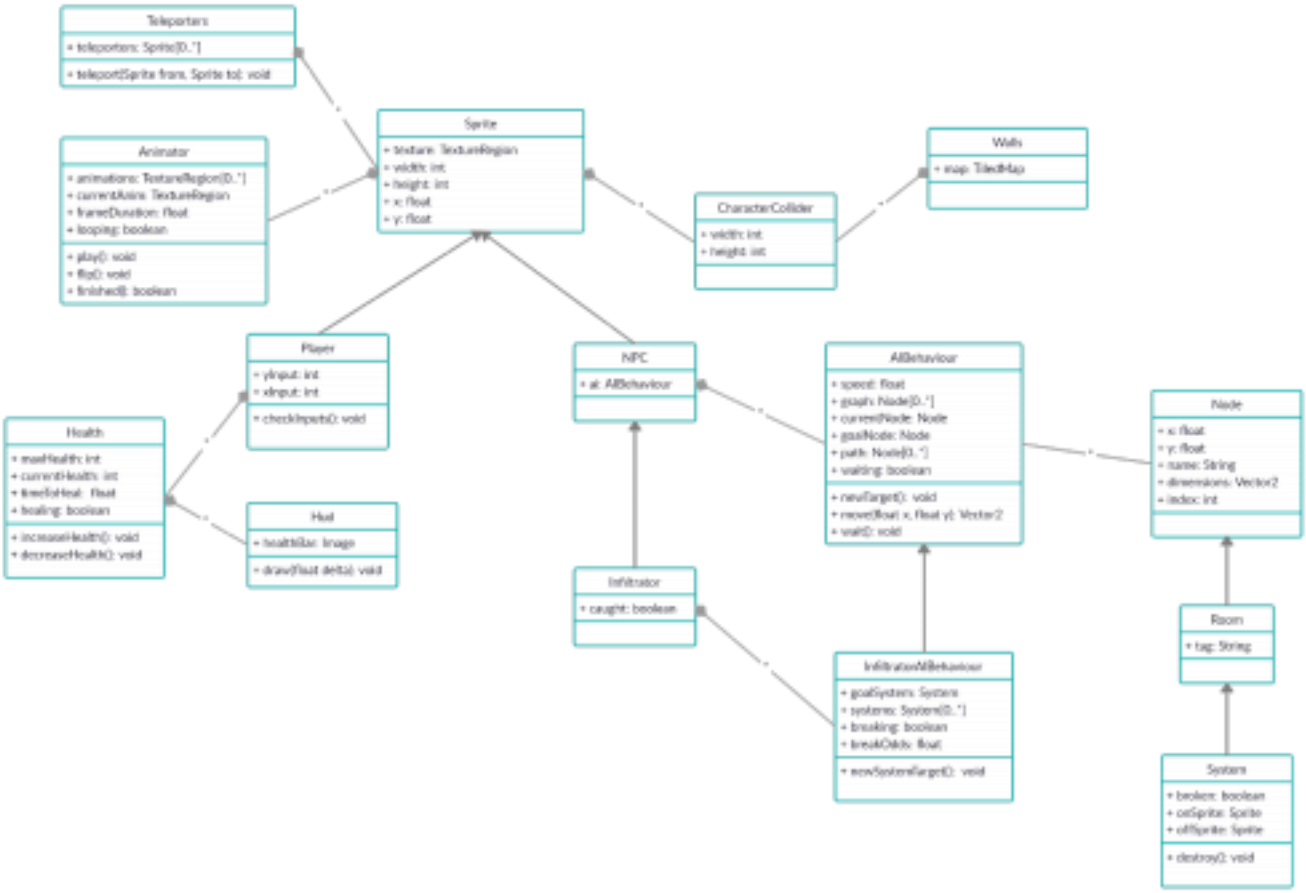
The abstract architecture is an Entity Component System we used to help define what components the game consisted of and where they have common elements. This helps to translate into the more concrete architecture which is a class diagram which attributes components to specific classes and creates associations where needed so all entities have a connection to the relevant components.

Both diagrams were made with Creately (https://app.creately.com/)

Abstract Architecture: ECS



Concrete Architecture: Class Diagram



The abstract diagram focuses mainly on entities within the game which the player will have interaction with, for example the NPC's and Infiltrators that make up the core gameplay.

Within the diagram we can see there are some entities with unique or rarer components, input is naturally unique to the player and therefore is handled by the player class itself in the class diagram. Health is also unique to the player entity but in considering an infiltrator power could be to also possess health we decided to separate health from the player class, health would also need to be visible to the player on the hud and so in the class diagram it has an association to the hud class.

As the infiltrators require unique skills the AI component has different requirements for the different types of NPCs. Differing AI configurations can be facilitated with inherited classes and then associations with the relevant sprite types.
Everything requires a position to allow the game engine to draw the relevant textures and compute the possible collisions. To facilitate this requirement the core of most entities, the Sprite class, contains the grid position reference which is then inherited by all the other entity classes that need one.

Also inherited along with the sprite class is the association to the CharacterCollider class which is used to establish the collision boundary around an entity's position

such that the full dimensions of the texture are encompassed to ensure everything collides as it should.

The one exception to this is the Walls class which also has its own association to the CharacterCollider but is instead comprised of a TiledMap, an inbuilt class of the libgdx engine, allowing us to not worry so much about the specific implementation of boundaries ensuring that the player safely remains within the play area.

The concrete architecture also diverges slightly from the abstract one with some small implementation details.

We decided to use a node system to direct the AI around the map which has its own class with two inheriting classes. The first inheriting class is the Room class which is used to distinguish between normal nodes and nodes in rooms on which it would be more normal for the AI to exhibit its
waiting behaviour rather than randomly stopping and gathering around invisible points in corridors. The second inheriting class once again inherits from the Room class to further distinguish nodes which are in front of the key systems of the ship, the main focus around which the game is played as the player is required to protect them from the infiltrators. This allows infiltrators to decide upon random targets anywhere on the map and approach them to exhibit their sabotaging behaviour, which may also be punctuated by waiting for a short period to attempt to emulate the normal NPC's.

The other divergence is the Teleporters class. Rather than being a part of the systems entity and thus being vulnerable to sabotage by infiltrators they are their own class, all of them are represented in the one class however as they are all linked to one another and so need access to the other instances. This decision was made due to the
importance of the teleporters to the gameplay and player's mobility the removal of which would result in a less engaging and satisfying user experience where the player feels punished for not being able to be in multiple parts of the map at the same time.

The classes of saving and loading have to be able to access all elements and save the attributes of all classes. This could cause potential scope problems otherwise, and certain values may be inaccessible if approached the wrong way.

The Auber's powers can be approached in the same way as the infiltrators' powers, and at the base, they are the same, with tweaks in their methods which is how the powers are performed at a game level.

The difficulty must be able to access the AI attributes so that their performance

reflects the difficulty chosen by the player. The change will occur in the attributes of their speed, chance to break items and their abilities.