Within the context of your chosen design paradigm, describe the software architecture of your prototype.

As previously stated, our group has chosen to use the object-oriented design paradigm. Due to this paradigm, our architecture incorporates several different classes within our game. These classes are automatically integrated into the Unity game engine and stored as assets. Each class or asset has the ability to interact with one another. These assets are all objects that can have their own individual C# scripts added to them. Furthermore, each object can have private member variables stored as metadata and this metadata can be manipulated and accessed by other classes. Thankfully most of the software architecture is already built into the Unity game engine. This gives our group several advantages. First, our software program will be very robust. This means it will not be prone to crashing thanks to the object-oriented software architecture and design paradigm. Next, our program will be maintainable. Any time that there is a problem, it will be easy to pinpoint where it is coming from due to the individual classes. Next, our program will be scalable and reusable. Our code already contains a lot of reused code due to the nature of building games on a game engine, and this means that we can scale rapidly as we need to for the upcoming project 4. The actual architecture is a sort of variation of a 3-tier architecture where the display layer is the actual game that the player is seeing. The logic tier is everything going on in the game that the player can't see (or is hidden by the camera within the game engine). And lastly, the data tier is all the data structures that hold the player's data. For example, the arrays that store player metadata. Each level of the 3-tier architecture is integrated into different classes and they all function to create our prototype.