Amyleila Mejia

**Conceptual Design:**

A variety of conceptual designs should be discussed. Particularly, the way each possible design satisfies or does not satisfy the problem requirements should be detailed. This portion of the report includes all information from the Conceptual Design Report and should conclude with the choice of a single design to pursue. This portion of the report includes all information from the Conceptual Design Report.

-------------------------------------------------------------------------------------------------------------------------------

Originally there was a Python Transmission Control Protocol, TCP, that was to be used for game rendering in Unity. After having issues with the weight file conversions the team did not follow through with this original plan and instead progressed towards the preparation of two demonstration models instead. This would allow the team to showcase the working parts of the project. Not implementing the use of the Python TCP did not satisfy the following requirements:

* Implemented using a HoloLens
* Total system latency will be under two seconds
* Improve runtime and battery life
* Perform the app rendering and processing on a host computer

We ended up not using the Python TCP Server since it added complexity that was not strictly necessary with the project. We found a solution simply by using the command prompt to start the game. An emulator was used to model what the Hololens was to render. This was used in the demonstration involving the Hololens.

The conceptual design proposed the use of the YOLO algorithm version 2, YOLOv2. The final version that was used for the project was YOLOv4 due to software compatibility issues. This led to a change in the neural network implementation software to the use of Barracuda which was not originally in the specifications but it was the only software that allowed the implementation of the Darknet neural network and was compatible with Unity.

Committing to the use of Unity into the finalizing phase of this project allowed for the development of a proper special effects and game logic demonstration as well as a demonstration for the proper use of the algorithm. The main reason for committing to Unity was for the compatibility that YOLO had with Unity.

The game logic class was created within Unity methods start and update, modeling classic Black Jack game rules. The game logic design was implemented using C# as originally stated in the requirements. A python script called “blackjack.py” was used to calculate the probabilities of winning when the player chooses “hit” or a “stay” for each round of blackjack. The Python script would write to a text file with the probabilities, and the main GameLogic.cs C# script would read the text file and then update the UI. The “GameLogic.cs” script would also launch the python script using the command prompt. Objects in Unity could be rendered as graphics with their own script components. The “GameLogic” object has the “GameLogic.cs” script attached as a component.