Game Design:

I choose to make a simple Multiplayer game where each player spawns randomly in a small arena & tries to shoot at each other. The movement is a simple WASD for moving in your forward/backward direction and strafing as well as QE for rotating the player around the Y Axis. Left mouse button is used for shooting the bullet in the direction the player is facing(indicated by a grey barrel).

Network Implementation:

For the movement I decided to make it so that each client changes a NetworkVariable with 2 movement float values, m_moveX & m_moveZ. These two values detect the input from the player based on the GetAxisRaw() function meaning that the values will be either -1, 0, or 1. The 2 movement is a struct derived from INetworkSerializable which has a function that detects if the values input by the player has changed, if so we then tell the server that this player wants to move and move the player on the server side applying the different directions based of m_moveX & m_moveZ inputs. Rotation is handled exactly the same where instead it's just a float that goes from -1, 0, or 1.

For Bullets I choose to make it so that each client requests that it wants to shoot a bullet called by the function ShootBulletServerRpc(). ShootBulletServerRpc Instantiates the bullet at a specified postion and rotation based on the player and then we call NetworkObject.Spawn so that all active clients receive a bullet on the screen. Each bullet is then handled by the server.

Player to Bullet collision detection is handled by the server where it then Destroys & despawns it and calls the function TakeDamageClientRpc to damage the player. I decided to make a client rpc function for this but I will say that it is not the most ideal thing to do. TakeDamageClientRpc just lowers the currentHealth of the Player and when it reaches zero it sets player is alive to false, resets the respawn timer & calls the function KillPlayerServerRpc that tells the server that this player is dead so we move it out of the arena and spawn a particle effect at its location to tell the other players that the this player has died.

The chat is handled by each client where a client sends a string of the message to the server that then gets distributed to each connected client with the displaying message. Nothing complex at all.

Challenges:

The challenges I faced during this assignment was mostly just wrapping my head around how everything will be set up, what should and should not be done by the server or clients. What type of data should be read only by the clients and edited by the server etc. Once I started to understand how things and how it should run, I didn't face any big problems as it was mostly just trial and error and reading up a lot on the forums and documentation.

All in all, it was very interesting to design systems and gameplay around networking as it requires you to actually think about the data that is being used and how to write and read it while also synchronising with the other clients and keeping things smooth.