Tiny PYRPG Lab Report

**Title:** Tiny PYRPG by Jeffrey Smith, Chris Adams, Duncan Forsythe.

**Summary:**

This project will use python with the GUI library PyQt5 to create a dungeon and dragons themed game. Clients will be able to connect to a server that is hosting a game and choose their class of character. Once all players are ready, the game is started. Each player will have a turn to complete actions before ending their turn, allowing the next player to do the same. The game ends when all players have lost their health and the last player performs an action that ends the game.

**Documentation:**

The client starts by running the start-client.py which will open the main menu. Here the user will input a username between 4 and 26 characters long. Next, they will input an ip address to connect to, connecting to the server if it exists or returning an error. The player will be loaded into the lobby with any other players that have connected

A pre-game lobby will display all the players currently connected. In order to ready, the players must choose a profession. Anytime a player changes his profession, or a player is connected, the server updates its internal library. Players will need to press “Refresh” in order to see new players. The GUI is also updated when the player changes their profession. The clients work by waiting for buttons to be pressed which add commands to a queue. Commands such as “GET UPDATE” is sent to the server which responds with a library of the current players and their relevant information. The client will then update their own GUI. Multiple clients can be seeing different information; however, it is only cosmetic and does not affect the functionality of the game. Once all players are ready, player one can start a game. Players will need to hit refresh in order to open the game GUI.

GAME Explanation time

Duncan’s Big Brain Time.

**Network Application Protocol:**

To connect players to the server, a TCP connection is established with an extra handshake function added in the client/server. We chose TCP because you recommended it.

The client and server use a word style command system to enact functionality. “GET ACTION”, “DO ACTION”, “ERROR”, and several others. DUNCAN HERE EXPLAIN THE JSON.

**Challenges:**

The first scrapped idea was for clients being able to host games based on their ip and supplied port number. It probably could be created later, however right now the server creates a single game for 6 people.

In the lobby menu, we found it much easier to update all the character details whenever an update needed to happen. Duncan used json libraries to format the data making it easier for the clients to receive the data and display it.